

**School of InfoComm Technology**

**Machine Learning**

Diploma in Data Science (DS)

Diploma in Information Technology (IT)

October 2023 Semester

**INDIVIDUAL ASSIGNMENT 1**

(30% of Machine Learning Module)

**Deadline for Submission:**

**10th Dec 2023 (Sunday), 2359 Hours**

|  |  |  |
| --- | --- | --- |
| Student Name | : |  |
| Student Number | : |  |
| Video Presentation Link | : |  |

**Penalty for late submission:**

10% of the marks will be deducted every day after the deadline.

**NO** submission will be accepted after 17th Dec 2023, 23:59.

Contents

[Overview 4](#_Toc153144690)

[Dataset 1 (HR Analytics) 5](#_Toc153144691)

[Problem Understanding 5](#_Toc153144692)

[Data Exploration 5](#_Toc153144693)

[Data Cleansing 9](#_Toc153144694)

[Missing values 10](#_Toc153144695)

[Outlier Handling 12](#_Toc153144696)

[Data Transformation 14](#_Toc153144697)

[Numerical Transformation 14](#_Toc153144698)

[Categorical Encoding 16](#_Toc153144699)

[Scaling 18](#_Toc153144700)

[Correlation Analysis 19](#_Toc153144701)

[Dropping columns 19](#_Toc153144702)

[New feature(s) 19](#_Toc153144703)

[Polynomial Expansion 20](#_Toc153144704)

[Others 21](#_Toc153144705)

[Model Improvement 21](#_Toc153144706)

[PCA 21](#_Toc153144707)

[Down sampling 22](#_Toc153144708)

[Dataset 2 (Airbnb listings) 23](#_Toc153144709)

[Problem Understanding 23](#_Toc153144710)

[Data Exploration 23](#_Toc153144711)

[Univariate Analysis 24](#_Toc153144712)

[Categorical Analysis 24](#_Toc153144713)

[Numerical Analysis 27](#_Toc153144714)

[Data Cleansing 28](#_Toc153144715)

[Missing Values 28](#_Toc153144716)

[Outlier Handling 30](#_Toc153144717)

[Data Transformation 35](#_Toc153144718)

[Numerical Transformation 35](#_Toc153144719)

[Categorical Encoding 38](#_Toc153144720)

[Scaling 42](#_Toc153144721)

[Variable Discretization 43](#_Toc153144722)

[Correlation Analysis 44](#_Toc153144723)

[Creating new features from name column 44](#_Toc153144724)

[Dropping columns 45](#_Toc153144725)

[Polynomial Expansion 46](#_Toc153144726)

[Summary and Further Improvements 47](#_Toc153144727)

[Further improvements 48](#_Toc153144728)

# Overview

The assignment delves into two datasets, with the task being to perform data exploration, analysis and preparation. Data visualization techniques and statistical approaches are involved in this process and are important in understanding the given datasets thoroughly. The primary objective is to prepare the data for a machine learning model in the next assignment. In this report, I will document the entire analysis process along with its findings.

The report for each dataset will be split into the following sections:

* Problem Understanding
* Data Exploration
* Cleansing
* Transformation
* Correlation Analysis
* Others

Ultimately the assignment lays the bricks for the subsequent machine learning modelling task in the next assignment, ensuring that the datasets are appropriately explored, processed, and structured for the application of machine learning algorithms.

# Dataset 1 (HR Analytics)

## Problem Understanding

Efficient decision-making is hampered by human resources' traditional manual data collection and analysis methods. Utilization of machine intelligence brings to the table a chance to forecast employee promotions. ML models can identify high-potential employees by using past HR data, such as KPIs and training scores. More intelligent, effective, and equitable promotion decisions are anticipated as a result of the switch from manual to automated predictive analytics. As a result, HR departments can maximize talent management and promote employee happiness and organizational success by working with machine learning.

## Data Exploration

I performed surface level exploration of the data with basic functions like df.info(), df.describe() and df.head(). I also referred to the metadata provided in the assignment brief. This allowed me to gain an understanding of what the columns represent, as well as the nature of their values. For example, some numerical columns are binary columns. I then renamed the columns so that they would be simpler for me to type throughout the assignment.

An interesting find I had was the distribution of values in the target variable.

A blue circle with a orange triangle in the center

Description automatically generated

Referring to the pie chart above, we can see that the amount of 0 values overwhelms that of the 1 values. This is not good for the model as an extremely uneven dataset will lead to an extreme bias of the model. I did not perform any sampling techniques at this stage yet however, as I want to be able to capture all the data when performing things like imputation.

Renaming columns

|  |  |
| --- | --- |
| **Old column names** | **New column names** |
| no\_of\_trainings | trainings |
| previous\_year\_rating | prevRating |
| length\_of\_service | serviceLength |
| awards\_won? | awarded |
| avg\_training\_score | avgTScore |
| is\_promoted | isPromoted |

Univariate Analysis

In univariate analysis, I compared the relationship between features and the target variable.

Categorical Analysis

A graph of different colored bars

Description automatically generated

Here we see the percentage of employees that were promoted in each department, with technology and procurement being promoted most often.

A graph of different colored lines

Description automatically generated with medium confidence

Here we see the percentage of employees that were promoted in each region. The percentages are very varied with some regions having 4 times less promotion rate than others.

A group of rectangular objects

Description automatically generated with medium confidence

Here we see the promotion percentages per education level. Surprisingly even though one would expect employees with a bachelor's degree to have a higher rate of promotion than those without, it is not the case. Employees below secondary education level actually have a slightly higher rate of promotion than those with a bachelor's degree.

A blue and pink rectangles

Description automatically generated

Females have a higher promotion rate

A graph of different colored rectangles

Description automatically generated

Employees from the sourcing and other recruitment channels have similar promotion rates, but those who were referred in are nearly 1.5x more likely to be promoted.

A graph of different colored squares

Description automatically generated

Here we can see that the higher the rating of an employee, the more likely they are to be promoted.

A white and blue rectangles

Description automatically generated

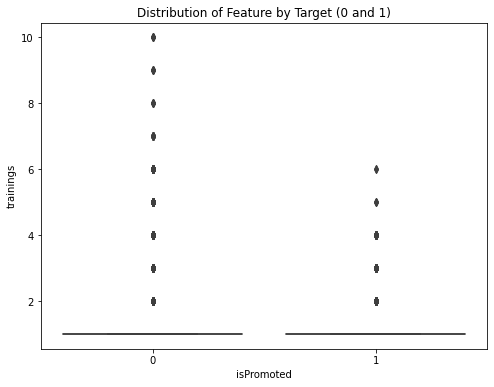
Employees who meet their KPI are a lot more likely to be promoted.

A white and blue rectangles

Description automatically generated

Employees who receive awards are also a lot more likely to be promoted.

Numerical Analysis

A blue and orange rectangular objects

Description automatically generated

A diagram of different colored objects

Description automatically generated with medium confidence

The distributions of numerical features when it comes to promoted and non-promoted employees are quite similar.

## Data Cleansing

### Missing values

Handling missing values was my first step in data cleaning. There were only columns with missing values – education and **prevRating**. Referring to the chart below, we can see that 4% of the dataset had missing **education** values, while 7% had missing **prevRating** values.

A black and blue rectangle with lines

Description automatically generated

#### Education

I plotted a bar chart that showed me the count of values in the **education** column.

A blue and white graph

Description automatically generated

From this chart, I noticed that there was an inconsistency in the unique values of the column. The column accounts for the education levels of below secondary, bachelor’s and master’s but it does not account for the education levels in between – diploma.

A question mark and a question mark

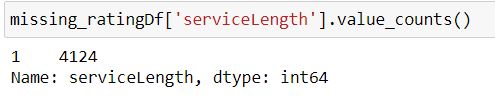
Description automatically generated

*This is a simple illustration of mine to visualize what I am trying to say*

The education level that is between “Below Secondary” and a “Bachelor’s” would be “Diploma”, which is what I will impute the missing values with.

#### prevRating

Just like Education, I checked for similarities in records with missing **prevRating** values. Contrary to Education, I did end up finding a similarity. All records with missing **prevRating** values have a value of 1 for the **serviceLength** column, as shown in the below image.



This most likely meant that the corresponding employees have not yet served a full year, or they have not made it to the next yearly rating check yet. Using this finding, I decided to treat missing records like they belonged to new employees.

When thinking of a new employee’s rating, it’s safer to say that their performance would be more akin to that of a regular employee’s rating, and not as an extremely skilled or unskilled person. With that in mind, I decided to use mode imputation to impute the missing values in **prevRating**, which was 3.0.

A blue and white bar graph

Description automatically generated

I did not use mean imputation, as the **prevRating** column is numerical and categorical in nature. Performing mean imputation would cause the creation of a new category within the column with no logic, reason, or pattern behind it. This would cause the model’s performance to drop as less meaning would be extracted from the data by the model.

### Outlier Handling

I began by using df.describe() to see which of my columns could contain non-binary numerical values. I then checked my non-binary numerical columns for outliers. All of the columns had outliers except for avgTScore.

A graph with lines and dots

Description automatically generated with medium confidence

*Trainings*

A blue line with red line

Description automatically generated

*age*

A blue and red line

Description automatically generated

*serviceLength*

A blue line with red line

Description automatically generated

*avgTScore*

However, I did not handle any of the outliers as my numerical transformations later on would also handle outliers without loss of data. Below are the additional reasons as to why outliers were not handled at this stage.

#### Trainings

This column represents the number of trainings taken by the employee in the previous year on soft / technical skills. My assumption here is that the trainings are not compulsory and that the employee participates in the training of their own free will. When an employee trains themselves to improve their skills, they are more likely to be promoted due to their improved contributions / effort. This means that the more training an employee takes there would be a bigger likelihood of promotion.

#### Age

This column represents the age of the employee. an older employee in many cases comes with more experience or expertise in their job role or field. this means that they are more skilled and their value as an employee is higher, meaning a higher chance of promotion.

#### serviceLength

This column represents the number of years the employee has worked with the company. just like age in many cases an employee who has served the company longer has a higher chance of promotion. this likely means that the outliers in the lower end and upper end of the spectrum will have a lower and higher likelihood of promotion.

## Data Transformation

### Numerical Transformation

Out of 4 non-binary numerical columns, only 3 were continuous – age, serviceLength and avgTScore.

#### Age

Original distribution:

A blue line and red line

Description automatically generated

Post-transformation:

A graph with a red line

Description automatically generated

I chose the reciprocal transformation for the age column as not only did it gave me what I believed was the most normal distribution of what was once skewed data. It was also able to handle almost all my outliers without removing them. This means that the possible insights the machine learning model can derive from outliers has been preserved.

#### serviceLength

Original distribution:

A blue and red line

Description automatically generated

Post transformation:

A red and blue line

Description automatically generated

I chose the cube root transformation for the serviceLength column as it also gave me the most normal distribution of data. The red line in the probability plot shows a perfect normal distribution, and I find that the cube root transformation handled the normalization of value the best. The distribution of values in other transformations like boxcox and yeo-johnson tend to fall off at the higher ends of the range. Additionally, cube root transformation helped me to curb my outliers.

#### avgTScore

Original distribution:

A blue line with red line

Description automatically generated

I chose not to transform this column at all because none of the transformations changed the distribution significantly. After all the transformations, the distribution remains similar to pre-transformation.

### Categorical Encoding

Using df.head(), I identified the categorical columns that weren’t in numerical form yet.

#### Department

This column states the department the employee works in, so I will use ordered ordinal encoding as some departments have a higher chance of promotion than others. For example, a chef would have less chances of being promoted than an employee in marketing since there are lesser positions to promote to in the career path of a chef. Additionally, the column has a semi-high cardinality which makes one-hot encoding a less viable option as it increases dimensionality of data and increases risk of overfitting.

A close-up of a computer screen

Description automatically generated

After skimming through sites online, most sites including LinkedIn stated that things concerning sales, marketing and finance tend to be most likely to get a promotion. According to [this LinkeAndIn newsletter](https://www.linkedin.com/pulse/top-10-jobs-get-ahead-andrew-seaman/) by Andrew Seaman, Senior Managing Editor at LinkedIn News, the ranking is as follows: *Marketing > HR > Finance > Sales > Procurement*

I couldn't find sites that talk about the other departments, so I fed the rest of the values into ChatGPT and asked it to rank them. This resulted in contradictory rankings, which I resolved by prioritizing rankings from actual information from actual sites like LinkedIn and arrived at the following ranking:

*Sales & Marketing > HR > Technology > Finance > Operations > Analytics > Procurement > R&D > Legal*

#### Region

The values in this column tell me absolutely nothing about the geographical location of the employee, meaning there is no data that I can search for online for this. The column is extremely high in cardinality, having 34 different unique values as shown below.

A close-up of a computer code

Description automatically generated

This means that one hot encoding is ineffective here as it will generate many new columns that will end up increasing the complexity of the dataset. There is also no hierarchy in the values, so ordinal encoding will make some values have a higher value and my model may incorrectly interpret bigger numbers as something significant. However, I have to choose the lesser of the 2 evils and in this case I choose ordinal encoding as it is better suited to handle higher cardinality columns.

#### Education

A close-up of a computer screen

Description automatically generated

Here we see that the education column is a low cardinality column. One-hot encoding would work here, but I have reason to believe that this column has a hierarchy and so I did not perform one-hot encoding.

While there weren’t any sites that stated education levels directly affect promotion chances, a report by Susana Santi, “The Positive Impact of Education, Training and Work Experience to Influencing Employee Performance”, states on page 4 that “every 1 unit increase in performance is partially influenced by the educational level factor of 0.217 units”. This means that the better the education level of an employee, the better the improvement in employee performance. Better performance from an employee will lead to higher chances of promotion. Therefore, I encoded this column using ordered ordinal encoding.

#### Gender

Since the column only contains 2 unique values and has no hierarchy, the result is the same whether I encode the gender column using one-hot or ordinal encoding, as the column ends up becoming a binary column.

#### Recruitment\_channel

The way an individual is recruited only affects the chances of the said individual to become an employee in the company. For example, you are much more likely to get the job if you are headhunted rather than applying for a job listing, as being headhunted means the company wants you. However, that is where the benefits of different recruitment channels stop. Promotion chances are affected by how much an employee contributes, not by how much the company wants the employee. To get to the point, this means that there is no hierarchy in the recruitment\_channel column.

A close-up of a computer screen

Description automatically generated

The image above shows that the column has extremely low cardinality. Since the column has low cardinality and no hierarchy, I performed one-hot encoding.

### Scaling

A graph of a number of data

Description automatically generated with medium confidence

To scale my data, I performed standardization. After numerical transformation, my continuous numerical columns are on different scales with little to no overlap. This may lead to the model interpreting some features as more important than others, so I performed standardization to ensure that all my continuous numerical columns area in the same range.

A graph of a line graph

Description automatically generated

## Correlation Analysis

### Dropping columns

For this section I first dropped the employee\_id column. The dataset contains thousands of records and every value in the column is unique hence I decided to drop it due to its extremely high cardinality which machine learning models do not handle well. Moreover, in the real-world context, your employee ID has no say in your chances of promotion just like how my student ID doesn’t guarantee a perfect GPA.

### New feature(s)

A black and red text

Description automatically generated

Next, I created a created a new column named “skilled”. It is derived from the Boolean values of “KPIs\_met > 80%” and “awarded”, as shown in the image above. This means that when the employee has fulfilled most of their KPIs and has won award(s) in the last year, the employee is labelled as a skilled worker or is regarded more proficient as a job. Better skilled workers tend to contribute more to the company and promotion likelihood is heavily affected by your contributions to the company.

### Polynomial Expansion

I started off by plotting a correlation heatmap to see the correlation coefficients of all the features.

A screenshot of a computer

Description automatically generated

From the heatmap I chose the following columns to use for expansion: department, avgTScore, awarded, KPIs\_met >80%, and prevRating. I chose “department” as it had a strong negative correlation with avgTScore. The rest of the columns were chosen as they had the highest correlation coefficients with the target variable. Since I set the degree parameter to 4 in the constructor, this resulted in over 100 columns being generated.

## Others

### Model Improvement

I tried to generate a summary of the model to look at the p-value for each coefficient to see the confidence levels of the model. From that, I was planning to drop appropriate features to improve my model. However, I was met with an error that I was unable to fix as seen in the image below.

A screenshot of a computer program

Description automatically generated

### PCA

After performing polynomial expansion, the number of features in my dataset increased from 15 to 138, severely increasing the dimensionality of my data. As a result, my model ended up being slightly overfitted so I resorted to performing PCA.

PCA allows for the reduction of data dimensionality by preserving the most critical features that explain most of the variance in data. To simplify, PCA reduces the number of columns while keeping the model’s score as high as possible.

I trial and errored until the difference between my training and testing set accuracy was within 0.01 of each other.

### Down sampling

As mentioned in the Data Exploration section, the distribution of values in the target variable is very uneven. Below is a chart representing the distribution, to refresh your memory.

A blue circle with a orange triangle in the center

Description automatically generated

To solve this issue, I decided to perform down sampling. Down sampling was performed after I had done all my transformations and encoding, as if I did it before I might miss out on some values because of limited representation. For example, a column might have 8 unique values but due to down sampling the 7th unique value may not appear in my train set. By down sampling after my data transformation, I could capture all the values in the dataset.

A screenshot of a computer code

Description automatically generated

From the image above, we can see that the values are present in a roughly 10:1 ratio. I aim to bring the ratio down to 2:1. I chose the ratio of 2:1 as I wanted to preserve as much data as I could while limiting potential model bias. The dataset contains over 50 thousand records but bringing the ratio down to 1:1 will cut the amount of data by just under 10 thousand. This means about 80% of the data and possible insights the model could extract will become unavailable. Losing so much data could lead to overfitting, due to having limited representativeness. With a 2:1 ratio, the dataset has over 10 thousand records which means more data is preserved. Hence, I chose a 2:1 ratio instead of 1:1.

# Dataset 2 (Airbnb listings)

## Problem Understanding

Having a machine learning model that can accurately predict Airbnb prices is important for several reasons. Firstly, it provides valuable insights for both hosts and guests. Hosts can use the predicted prices to set competitive rates and optimize their earnings, while guests can make informed decisions about their accommodation options based on price and value. Secondly, accurate price predictions can contribute to a more transparent and fairer marketplace, helping to prevent overpricing or underpricing of listings. Lastly, it can assist in market analysis and planning for various stakeholders, including investors, real estate agents, and policymakers, by providing valuable information on supply and demand dynamics in the short-term rental market.

## Data Exploration

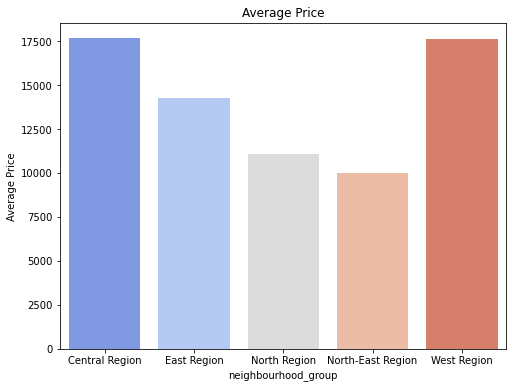
Just like the previous dataset, I started by using the info(), head() and describe() functions to gain a basic understanding of my data such as column names and what kind of values will appear.

Renaming columns

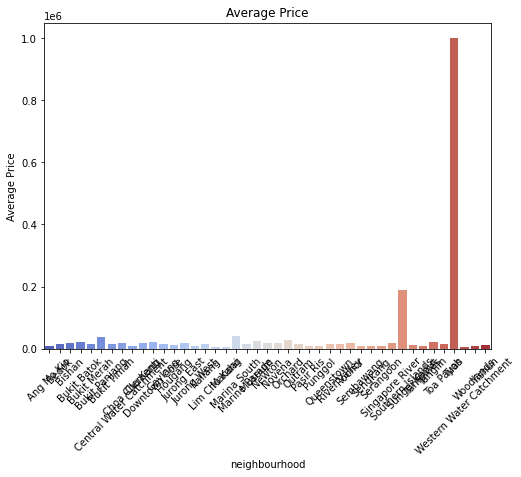
|  |  |
| --- | --- |
| Old column names | New column names |
| minimum\_nights | minNights |
| number\_of\_reviews | nReviews |
| reviews\_per\_month | monthlyReviews |
| calculated\_host\_listings\_count | cHostListCount |
| availability\_365 | avail365 |

## Univariate Analysis

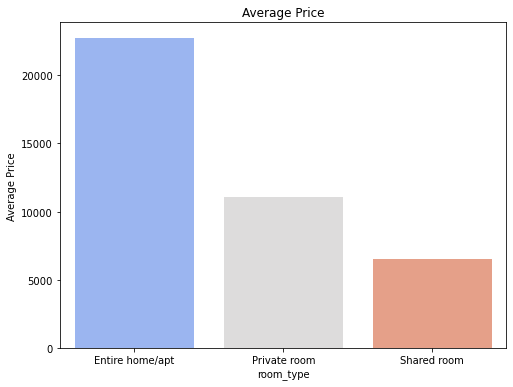
### Categorical Analysis



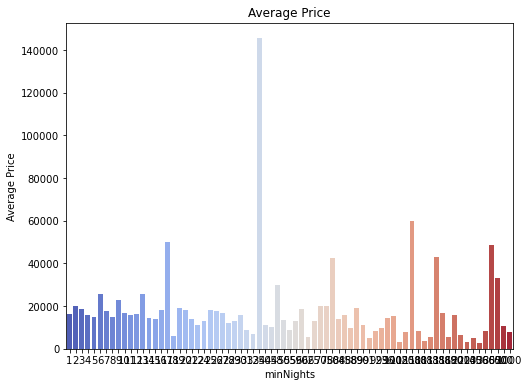
We can see that the average price in regions are in decreasing order: Central, West, East, North, North-East. As a Singaporean it doesn’t make sense to me that listings in the West can cost as much as the Central region, so this tells me that there either extreme outlier records in the West region, or there is a pattern in the data concerning records in the west.



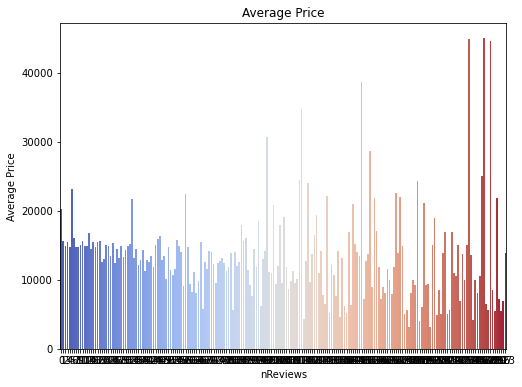
We can see that there are extreme outliers in the neighbourhood column. However due to the high cardinality of the data the label is unreadable even after rotating, so this can be investigated into further during the outlier handling section.



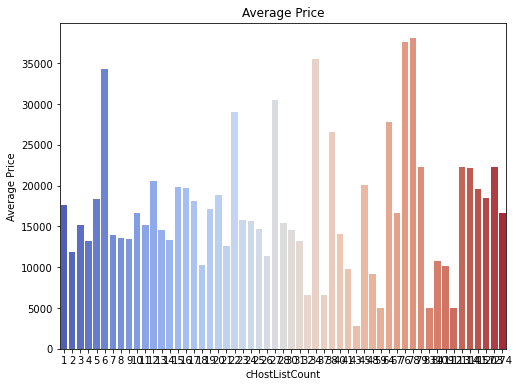
This shows us that the average price for a listing for each room\_type decreases as the quality of room\_type decreases. For me, quality is defined by living space and accessible amenities.



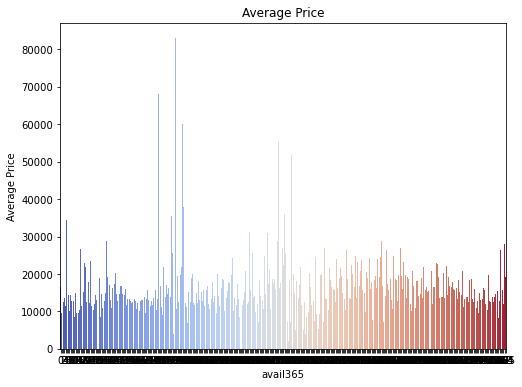
From this chat, we can see that there is a sudden spike for one of the minNights counts. This could belong to the same extreme outlier that is affecting the other columns.



From here we can see that the total number of reviews for a listing don’t affect the price in any discernable way until the upper ends of the spectrum, where the listings with the top percentile amount of reviews are priced a lot higher. However, there is also the chance that this could be due to the same extreme outlier(s) affecting other columns.

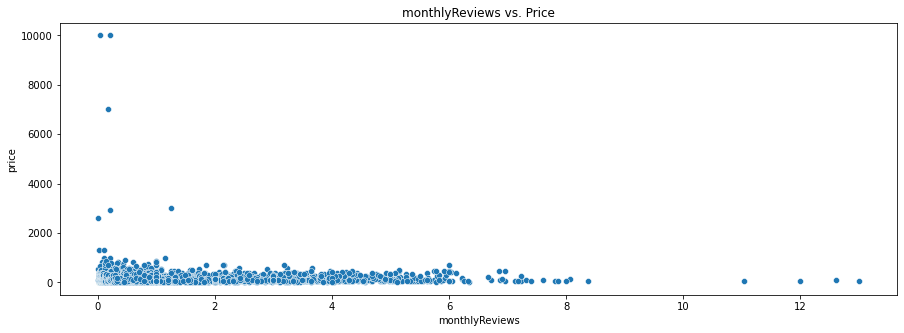


No pattern can be found here



No pattern can be found here. Gigantic spike could be due to extreme outlier.

### Numerical Analysis



From this, we can see that there is no relationship between monthlyReviews and price in this dataset.

## Data Cleansing

### Missing Values

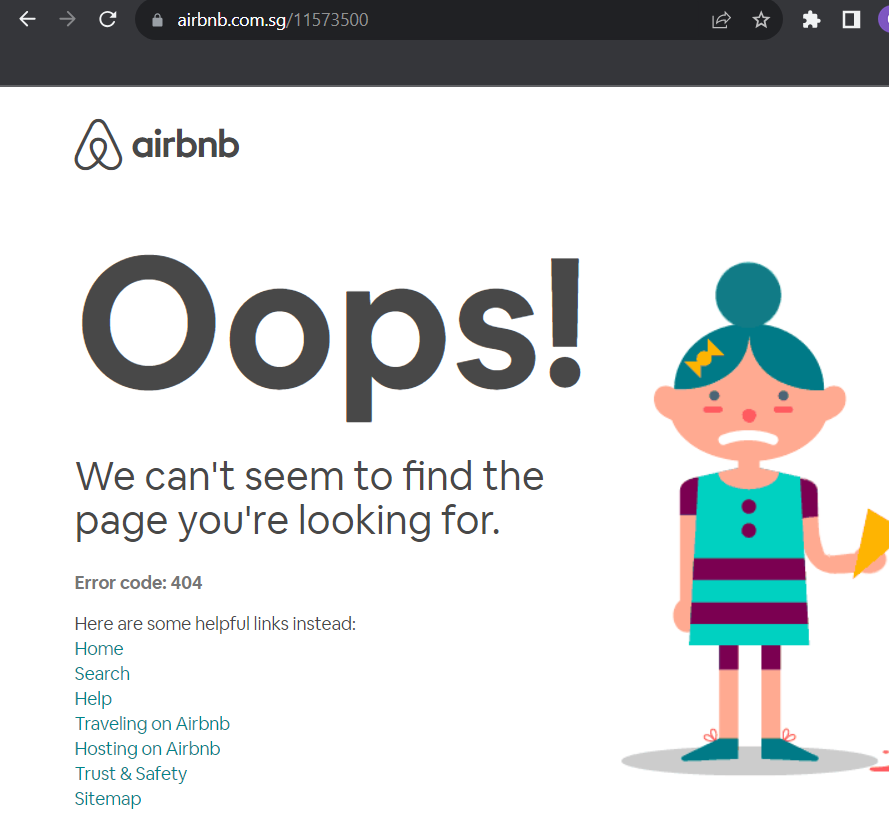
#### Name

First I viewed all the missing columns in the name column and tried to find similarities in the names of records which share the same features as those with missing values. For example, for records with missing names in the Central Region, I would check records in the Central Region for any similarities in their names. There were only 2 records with missing names.

A screenshot of a computer

Description automatically generated

Unfortunately, I was unable to find any similarities in the records, so my last strategy was to search the Airbnb website using the ID of the records with missing names. However, it came up with nothing as the site did not exist.



In the end, I decided to drop the two records as they are an extremely miniscule portion of the entire dataset, which consists of over 7000 records, therefore the loss of data would be insignificant towards the performance of my model.

#### Last\_review

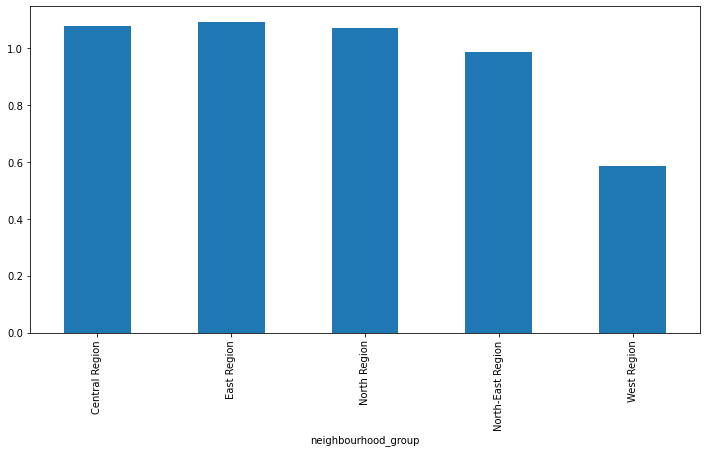
I dropped this column as I couldn’t see how the date of a listing’s last review affected its price. Same may reason that the date of the last review can reflect how often the listing is booked, and if the listing is not booked often the host would lower the price. However that is not necessarily the case. For example, the last review of a listing could have been a year ago, but multiple guests could have booked the listing between the date of the last review and the current date. In fact, a quick search on google reveals that it is not mandatory for a guest to leave a review on the listing. In conclusion, the date of the last review of a listing may not be a good indicator of how often a listing is booked, and thus not a good indicator of price. With all that in mind, I decided to drop the column to handle the missing values.

#### monthlyReviews

I decided not to drop this column to deal with its missing values, as I felt that the amount of reviews a listing receives monthly is able to reflect its price. For example, a listing that receives more reviews a month would mean that more people are able to afford the Airbnb, meaning it would have a lower price per night as compared to the other listings with similar features.

I displayed the records with missing monthlyReviews values and found out that every record with missing values in monthlyReviews also had a value of 0 for nReviews. This explains the reasoning behind null values for monthlyReviews as a listing with no reviews at all cannot be aggregated to show the amount of reviews a listing receives monthly on average.

To handle the missing values, I first checked for similarities between other records and the ones with missing values, or for any patterns that relate to reviews in other features. Through my investigation, I found that the average value of monthlyReviews between all neighbourhood\_groups was roughly about 1.0, except for the West Region which was around 0.6, as shown below.



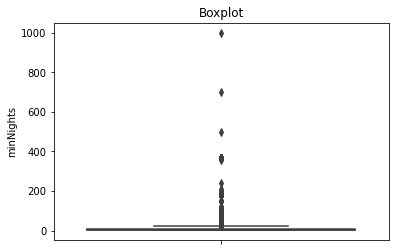
I chose to perform mean imputation as not only did I want to maintain the distribution of the values, but also because the other options did not seem applicable to me. For example, mode imputation didn’t make sense as a value being the mode of a column doesn’t necessarily mean that most of the values in that column are the mode.

When performing mean imputation, I decided that I shouldn’t include the records in the West Region, as their presence as an outlier would interfere with the mean value of the other regions. Thus, I decided to perform mean imputation separately – imputing the missing values of regions excluding the west, then imputing the missing values of the West Region.

### Outlier Handling

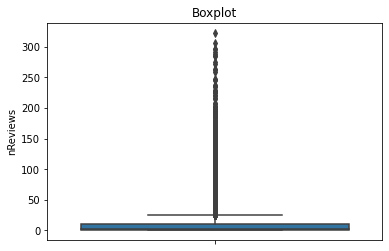
I charted box plots for my numerical columns to find out which ones had outliers.

#### minNights



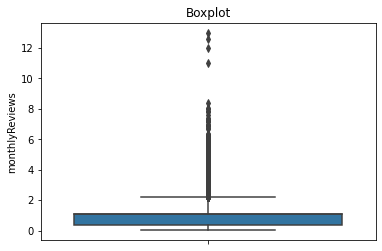
We can see that some hosts have set the minimum night requirement absurdly high, for example, 1000 nights. In a real-world context, hosts may set the requirement to be longer as they only want to attract long-term guests. However, in this dataset there is the presence of requirements above 200 nights (more than half a year) which I believe is quite rare in Airbnb listings and so I capped the values in this column to 180 (half a year) max.

#### nReviews



We can see that the outliers are distributed almost evenly across the range of around 50 to just over 300. Since there are so many outliers available, a trend or pattern could be discerned by the model, and it may improve performance. As mentioned before, the number of reviews a listing has can be a good indicator of a listing's price, as more reviews may mean that the listing is affordable by more people. However, there are many extreme outliers present and those may affect my model greatly by introducing biases in mean aggregations and skew relationships. Since I want to target only the extreme outliers, I will perform winsorization.

#### monthlyReviews



Once again, we see there are extreme values. I investigate by first displaying all the records with monthlyReviews over 10.

A screenshot of a computer

Description automatically generated

We can see that half the extreme outlier listings are hosted by “Fang”. I decided to check “Fang” out by displaying the listings that they are hosting.

A screenshot of a computer

Description automatically generated

From the above table we can see that "Fang" is listing multiple rooms in the same house / area, as the values in `neighbourhood` and `room\_type` are the same. To top it off, the `latitude` and `longitude` values are very close to each other. Both the outlier listings from “Fang” have the same value for both nReviews and monthlyReviews, which means that it is still the listing’s first month on Airbnb. However, other Airbnb listings by “Fang” have very similar names and features. With the information the dataset has right now, it is impossible to find out why those listings have the amount of reviews they have as compared to his other listings, which are all very similar in terms of features. Hence, I concluded that those two listings are not beneficial for the model as they will introduce redundant noise and skew relationships between nReviews and other features.

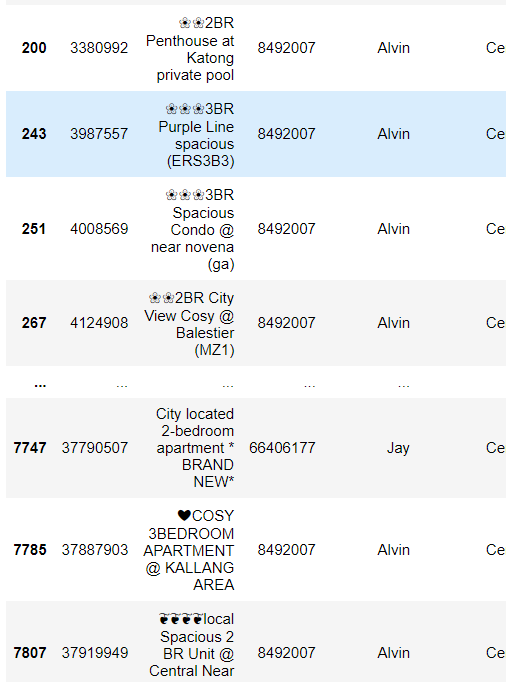
I decided to drop those two outliers, then perform winsorization to curb the other outliers.

#### cHostListCount

A diagram of a diagram

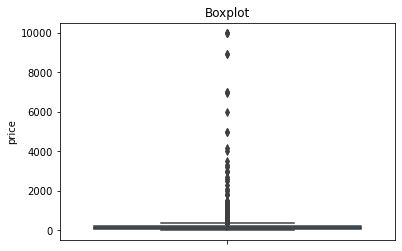
Description automatically generated with medium confidence

I started off by displaying records with a cHostListCount value over 170.



From the display we can see that unlike "Fang", there are two hosts "Alvin" and "Jay" that list residences that vary in features, meaning they contribute to useful data representation in the dataset and that hosts with multiple listings does not mean their listings are not meaningful. Therefore, I will not handle outliers in `cHostListCount` as it merely reflects the amount of listings the host has.

#### Price



From the box plot, we can see that many listings cost more than a thousand per night. Since I have no knowledge in exactly what makes an airbnb listing have the price it has, I decide to perform winsorization to target only the extreme outliers and preserve as much data as I can in the original dataset.

For all my winsorizations, I only winsorized the right tails as that was the only end where the outliers were. It didn’t make sense to winsorize the 0 to 5th percentile of records when they were not outliers at all.

## Data Transformation

### Numerical Transformation

For my numerical transformation, I only transformed minNights as it was the only column that I felt was continuous. I also performed numerical transformation as firstly, it transforms data to achieve linearity which the machine learning model may be able to interpret better. Secondly, numerical transformations may be able to aid in handling outliers and preserve their value without discarding them, which makes the data more robust to extreme values.

Original distribution:

A blue line with red line

Description automatically generated

The initial distribution of minNights is an extreme right skew.

Logarithmic

A graph with a red line

Description automatically generated

Reciprocal

A blue and red line on a white background

Description automatically generated

Square cube root

A blue and red line graph

Description automatically generated

Power

A blue and red line graph

Description automatically generated

Boxcox

A blue line with red line

Description automatically generated

Yeo JohnsonA blue line with red line

Description automatically generated

Between Boxcox and Yeo Johnson, I chose Yeo Johnson as while they provided similar distributions, the Yeo Johnson transformation handled more outliers.

Boxcox

A blue line with red line

Description automatically generated

Yeo Johnson

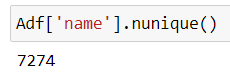
A blue line with red line

Description automatically generated

### Categorical Encoding

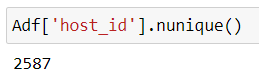
The categorical columns I have to encode are name, host\_id, host\_name, neighbourhood\_group, neighbourhood, and room\_type. For each column, I first check the number of unique values that the column has.

#### Name

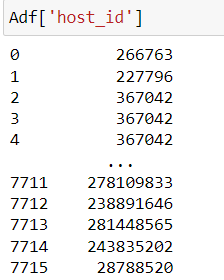


This column has an extremely high cardinality since almost every listing name is unique. Therefore, one-hot encoding and ordinal encoding will be detrimental to the performance of my model as it will severely increase the dimensionality of my data and may cause overfitting of my model. Thus, I will have to drop the column. However, before I drop the column there are features that I want to create based off this column so I will leave it alone for now.

#### Host\_id



This column also has an extremely high cardinality as it is the unique identifier of hosts. I then checked the values in the column.



The values for host\_id are not sequential in order and are very different from each other. Thus, I perform ordinal encoding for simplicity and interpretability. Ordinal encoding simplifies the data representation, making it easier to interpret for both humans and the machine learning model.

#### Host\_name

A close-up of a computer code

Description automatically generated

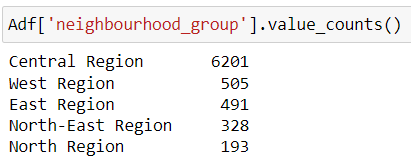
The fact that the number of unique values in host\_name is lesser than that in host\_id tells me that there are hosts with the same name but different IDs. This means that host\_id is the unique identifier for hosts. If I was to ordinally encode the host\_name column, I would be grouping different hosts and their listings together, which is not what I want. Therefore, I will drop the host\_name colummn as the host\_id column can explain more data to the model than the host\_name column.

#### Neighbourhood\_group

A close-up of a computer screen

Description automatically generated

Since the cardinality of the column is low, I check the unique values of the column.



Now that I had determined the cardinality of the columns, it was time for me to determine if there was a hierarchy in the values. To do this, I performed secondary research and came across [this site](https://www.pilotoasia.com/guide/cost-of-living-in-singapore#:~:text=Central%20Region%20%2D%20The%20Most%20Expensive,shopping%20malls%20and%20international%20schools), and from it I gathered that the Central region had the higher rent prices as you are closer to major shopping malls. From the website, this is the ranking of cost of living in each region from lowest to highest: North, East, West, North-East, Central.

Another method of encoding would be target-mean encoding, however I did not perform that as I was scared of data from my target variable leaking into my dataset.

Since there is a hierarchy, I will perform ordered ordinal encoding.

#### Neighbourhood



We can see that the column has high cardinality, so I will not perform one-hot encoding.

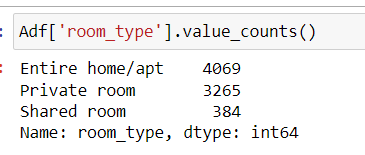
From the previous encoding of neighbourhood\_group, I have already ranked the different regions. Using that ranking, I will encode the neighbourhoods by the order of the region rankings. For example, neighbourhoods in the North region will be first in sequence when encoding, followed by East and so on.

#### Room\_type

A close-up of a computer code

Description automatically generated

We can see that the column has low cardinality. I then checked the unique values of the column

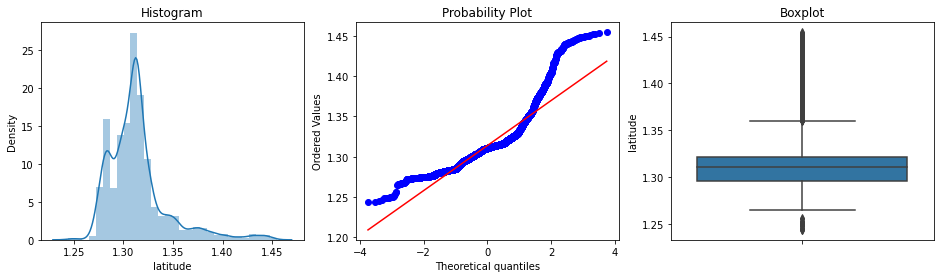


I decided to use ordered ordinal encoding to handle this column as firstly, the cardinality Is not too high. Secondly, the column has a hierarchy. The unique values for the room\_type reflect how much space they are getting and what are the amenities that the host has prepared for them e.g. the Entire home/apt will come with more space and may include other facilities like a pool. There is also the level of privacy. “Entire home” suggests complete privacy as guests have exclusive access to the entire property. “Private room” suggests a dedicated private space within a property which offers a higher level of privacy as compared to “Shared room” where guests have to share their space with others. When it comes to purchasing anything in general, the customer will have to pay more to get more, and that is why I performed ordered ordinal encoding.

### Scaling

For scaling, I chose the floating point columns like latitude, longitude and monthlyReviews as their values were on different scales.

#### Latitude



#### Longitude

A blue and red line

Description automatically generated

#### monthlyReviews

A blue line with red line

Description automatically generated

From the charts above we can see that the columns all contain outliers, so I decided to use robust scaling instead of standardization. Robust scaling utilizes the median while standardization uses the mean, which is affected more by outliers. As a result, robust scaling is more resilient against outliers than standardization.

A screenshot of a graph

Description automatically generated

After scaling, we can see that the values are all now in the same ranges.

### Variable Discretization

I perform variable discretization to bin my categorical columns which have a very wide spread of values. For example, minNights, nReviews, and avail365. Binning such columns will lower the cardinality of my data alot and may help the model interpret the data better.

For all the columns, I used equal width discretization. This is because I want to capture their outliers as their own group. If I was to use equal frequency discretization, I may end up grouping the outlier records and normal records together. This will interfere with my model as it will blur the line between outliers and regular values.

## Correlation Analysis

### Creating new features from name column

First, I performed text cleaning to simplify the words in the name column as well as extract only the important words as I want to use bag-of-words to create new features. There are a lot of different words in the column, so if I do not clean the text, the bag-of-words transformation will create a lot of new features.

A close up of a computer screen

Description automatically generated

*Before text cleaning*

*A close up of words

Description automatically generated*

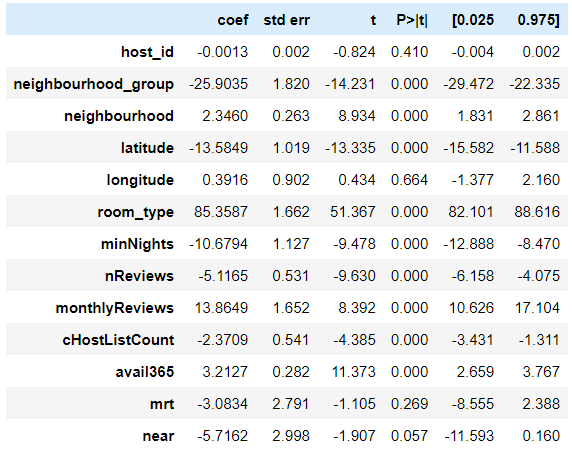
*After text cleaning*

After cleaning the text, the bag-of-words transformation generated a few new binary columns that represented whether the column name was in the name of the listing, two of which I chose to append to my current dataframe – “near” and “mrt”. This means that columns with a 1 value in “near” and “mrt” contain those words in the listing name. I chose to append “near” as when an Airbnb is close to something notable, hosts will definitely want to capitalize on it and they will include the phrase “near”, such as “near mrt”. Such listings would most likely cost more than other listings with similar features. I chose “mrt” as when an Airbnb is close to the mrt the host would also capitalize on it and state it in the listing name, which will give the host a reason to drive prices up due to the convenience the location of the listing brings.

### Dropping columns

I dropped name and id as I had no use for them anymore, as well as their high cardinality would be bad for my model’s performance should I choose to encode them.

I then checked my model’s summary for values that have a high p-value, which means that we have very low confidence on the coefficients of these features and therefore the features may not be the good indicators to predict the target.



From the above summary, we can see that `host\_id`, `longitude`, `near`, and `mrt` have extremely high p-values. This means that these features may not be good indicators on predicting the `price` of listings. However, I choose not to drop `near` and `mrt`, as I believe they can still be good indicators for reasons mentioned above.

### Polynomial Expansion

A feature engineering technique, polynomial expansion creates additional features by generating combinations of existing features. These new features may be insightful for the model. I first plotted a correlation heatmap to view the correlation coefficients of the features with each other.

A screenshot of a computer

Description automatically generatedUsing the heatmap, I chose to expand the following columns: room\_type, cHostListCount, avail365, near, mrt. After polynomial expansion, the number of features I had in the dataframe rose from 11 ( excluding price ) to 66.

# Summary and Further Improvements

**Dataset 1**

* The technology, procurement, analytics and operations departments have the highest rate of promotion.
* Promotion rates can vary differently from region to region.
* Employees with bachelor’s promote less frequently than those below secondary education.
* Referred employees have the highest chances of being promoted.
* The higher the employee’s rating, the better their chances of promotion.
* Employees who meet their KPI and receive awards have a lot higher chances of promotion than those without.
* The “education” column does not account for diploma holders.
  + Imputed with arbtitrary value “Diploma”
* All records with null values for prevRating have a serviceLength of 1.
  + Imputed with 3.0
* Ratio of binary values in target variable was very imbalanced
  + Downsampling was performed

The data in dataset 1 was in general quite clean already. I did not have to delve past my surface-level investigations when figuring out which imputation or encoding technique to use when dealing with them.

**Dataset 2**

* Some categorical columns have a very high cardinality and can easily be confused for continuous columns when looking at the distribution.
* There are extreme outliers that affect the dataset and add a lot of unnecessary noise and bias when trying to perform statistical approaches or visualizations.
* There are multiple listings with similar features like name, room\_type and neighbourhood however some of such listings may have an “outlier column” which contains vastly different values from its peers. To top it off, such values are unexplainable by the other features in the dataset which severely decreases model performance.

The data in dataset 2 was definitely harder to navigate through than in dataset 1. The presence of outliers and unexplainable values made the data preparation process harder. However, to me this is an opportunity to hone my data exploration and analysis skills. Additionally, due to the increased complexity of dataset 2, it presented more options for navigating through data.

### Further improvements

For dataset 1, an improvement I could make would be to get the function for model summary working. The model summary provides a detailed table with values that could help me determine the importance of a column. I feel like if I was able to know which columns were a bad indicator of my target variable, my model performance would definitely improve.

For dataset 2, the first improvement I believe I could’ve made while preparing data would be to format my code blocks properly, as well as performing my transformations on dataframe copies, as when I accidentally transformed my dataset in way I didn’t like I had to restart my whole code, which lost me a lot of time.

The second improvement would be to perform further data analysis. The complexity of this dataset made me feel as if my investigations did not go deep enough. I felt as if there were more opportunities to dig deeper into the data that I just didn’t see.