Seazone Data Science Challenge – Solution by Guilherme Fulop

1 INTRODUCTION

The objective of this work is to answer the questions made by Seazone and report the step by step to achieve them. The modules that were used are listed below:

pandas == 1.3.4
numpy == 1.19.5
seaborn == 0.11.2
matplotlib == 3.4.3
geopy == 2.2.0
Python == 3.9.7

2 EXPLAINING FUNCTIONS

In this section the functions present in the file ‘functions.py’ will be explained.

- date_prepare(data): this function receives a dataframe and creates a new column named date by merging the columns ‘ano’, ‘mes’ and ‘dia’ with a ‘-‘, resulting in the date format ‘yyyy-mm-dd’ and returns a dataframe with this new column. It also returns the min and max value of the date.
- get_location(data): this function receives a dataframe and gets the columns 'ad_id', 'latitude', 'longitude' in a new dataframe. Then, it joins the columns 'latitude', 'longitude' in a new column. Using the ‘geopy’ module in this column, is possible to get the address of the location. But we were interested just in the neighborhood, so, at the end, a we only got this information. It returns a dataframe that links each ‘ad_id’ to the neighborhood it belongs.
- get_price_date(data, max_date, min_date): it receives a dataframe and the min and max date value given by date_prepare() function. This function was created to select a date interval and decrease the size of the price dataframe.

- most_wanted_properties(data1, data2): this function receives two data frames, one containing the details of the Airbnb apartments and the other is the price dataframe generated by the function get_price_date(). From the price dataframe, it was selected 'ad_id', 'date', 'av_for_checkin' columns, merging it with the details datagrame that was grouped by. The merge was done on 'ad_id', 'date', that way, we will get the date which the house will be avaible or not. A new column called ‘occupied’ was created based on the colums 'av_for_checkin', it received 0 if it was True and 1 if it was False. A data frame called ‘occupation’ was created by grouping on ‘ad_id’ and counting it, that way we will get the total number of days for each property. Another dataframe called ‘tx’ was created grouping on ‘ad_id’ but summing it, that way, the column ‘occupied’ will give us the days that the property was occupied. This column was sent to the ‘occupation’ dataframe. A new column ‘rate’ was created dividing the ‘tx’ and the ‘occupied’ columns giving us the occupation rate of each property. Finally, a dataframe called ‘most_wanted’ was created with the properties with occupation rate higher than 90% and the houses with more than 9 days in the dataset. The function returns this dataframe.
- create_details_dackup(data): this function creates a dataframe by grouping the columns 'ad_id', 'suburb', 'price', 'number_of_bathrooms', 'number_of_bedrooms' and 'number_of_beds' and returns it.
- duplicate_eraser_price_selector(data, index): this function erases the duplicated values and selects the properties presents in the ‘most_wanted’ dataframe created by the most_wanted_properties() function. The houses with daily rental price higher than R$ 30000,00 were erased. Some properties had prices such as ‘300350’. I verified the values of some of these properties by entering in the link available and the value was about R$ 300,00. So, this leads us to believe that for some reason it was collected a price interval from R$ 300,00 to R$350,00. So, these values were erased from our dataframe.
- most_wanted_characteristics(data): this function receives the dataframe with the most wanted properties and makes three processes:
  1) creates a dataframe using groupby on 'number_of_bedrooms', 'number_of_bathrooms' and counting it. This gives the number of houses

with X bathrooms and Y bedrooms. The values were sorted by the highest count. Then, a columns called ‘percent’ was created dividing the quantity of properties with a specific characteristic by the total number of properties. It returns the percentage of the main characteristics and the most common number of bathroom and bedroom.

2) Creates a dataframe using groupby on suburb and count it, this gives us the number of houses in each neighborhood. It returns the neighborhood with more houses and its percentage.
3) Creates a dataframe using groupby on 'suburb','number_of_bedrooms', 'number_of_bathrooms' counting it. This gives us the quantity of houses with a given characteristic. It returns the percentage of houses with a given characteristic.

- house_distribution(data, column): receives a dataframe and a column name and returns a countplot figure saved in png format.
- neighborhood_adjustment(data): corrects some neighborhood names present in Viva Real dataframe.
- adjusting_iptu_and_monthly_fee(data): there are several houses in Viva Real dataframe without iptu or monthly condo fee. This function corrects it.
  1) First it selects the neighborhoods with more than 100 houses to give us a more accurate statistical value.
  2) The properties without sale price value were erased and the NaN values of ‘yearly_iptu’ and ‘monthly_condo_fee’ with the number 1.
  3) For the neighborhoods with more than 100 houses, it was selected the index for a given address neighborhood and splitting for the unity type APARTMENT or HOME. For these properties, those with yearly_iptu’ and/or ‘monthly_condo_fee’ in range 0 to 5 had these values changed by their mean value for each neighborhood.
  4) For the neighborhoods with less than 100 houses, the values of ‘yearly_iptu’ and ‘monthly_condo_fee’ in the range of 0 to 5 had this values changed by the mean value of the total dataframe, not by each neighborhood.
  5) Returns the dataframe with the corrected values

- boxplot_plotter(data, column): receives a dataframe and a column name and returns a boxplot figure saved in png format.
- outlier_detector(data, column): it receives a dataframe and a column name and finds the outliers using quartiles (Tukey Method). Returns a dataframe without outliers value.
- maintain_houses(data, column): returns a dataframe with the neighborhoods with more than 50 houses. This was done to avoid neighborhoods with a few houses and disturb and skew the results in the future analysis.
- get_number_of_houses(data1, data2, data3): this function creates two dataframes: one with the number of houses for the Airbnb dataset and the other with the number of houses for the Viva Real dataset.
- year_income(data): this function returns a dataframe created by summing the price values for each neighborhood.
- year_outcome(data): this function returns a dataframe created by summing the iptu and monthly condo fee for each neighborhood.
- highest_revenue(airbnb_year_income, airbnb_houses, viva_year_cost, viva_houses): this function does some steps:
    1) merges the airbnb_year_income dataframe and the airbnb_houses, a new column called ‘year_income’ is created multiplying the daily rental price for 365 (all days of the year) and consider an occupation tax of 80%. Another column called ‘year_income_by_house’ is created by dividing ‘year_income’ by the total number of houses for each neighborhood. This gives us the mean income value per house for each neighborhood.
    2) Merges the viva_year_cost and the viva_houses dataframes, a new column called ‘year_cost’ summing the columns ‘yearly_iptu’ and ‘monthly_condo_fee’ (this last column is multiplied for 12, to gives us the total fee for the whole year). Two new columns called ‘year_cost_per_house’ and ‘sale_price_per_house’ were created by dividing, respectively, the ‘year_cost’ column by the number of houses for each neighborhood and the ‘sale_price’ column also by the number of

houses for each neighborhood. This gives us the mean outcome and houses sales values per house for each neighborhood.

3) Changes the name of the column ‘address_neighborhood’ by ‘suburb’ and merges the airbnb_year_income and viva_year_cost dataframes on ‘suburb’ column. To calculate the revenue, first subtract the ‘year_cost_per_house’ column for the ‘year_cost_per_house’ column and divides by the ‘sale_price_per_house’ column. We sorted by the descending value of revenue. The function returns this dataframe.

- drop_condo(data): this function was created to select the allotment lands available that are not located in a condominium, since we are looking for a land to build a building with 50 apartments.
- cheaper_square_meter(data): this function follows some steps:
  1) An auxiliar dataframe is created selecting the lands in unity type in 'ALLOTMENT_LAND', 'RESIDENTIAL_BUILDING', 'RESIDENTIAL_ ALLOTMENT_LAND' and neighborhood in 'Morretes', 'Casa Branca', 'Meia Praia' (the three neighborhoods with the highest revenue values).
  2) Uses the drop_condo() function to drop the properties inside a condominium.
  3) Fills the NaN values of the columns 'usable_area' and 'total_area' with 0. Creates two auxilixar columns called 'usable_area_c' and 'total_area_c'. There are some empty values of usable area and/or total area. The 'usable_area_c' equals 0 receives the value of 'total_area' and the 'total_area_c' equals 0 receives the value of 'usable_area'. The ‘final area’ is given by the mean value of 'usable_area_c' and 'total_area_c'.
  4) Divides the sale price by the total area of the property. Sort the values by the lowest m² value.
  5) Returns the dataframe.
- cheaper_total_cost(data, dict_name): sums the m² value with the total cost per neighborhood with the year_cost_per_house.
- get_parking_spaces(data): returns the most common number of parking spaces and its percentage in the Viva Real dataset.

- total_occupation (data1, data2, data3, neighborhood_name): this function was created to find the occupation rate for the land with the lowest m² value.
- get_price(data): this function gets the mean rental daily value in the neighborhood where the land is located.
- income(mean_price, final_occupation_rate, year, data): receives the mean_price calculated by the function get_price(), the final_occupation_rate calculated by total_occupation(), the year which we want to calculate and the data_generated by cheaper_total_cost().
    1) To calculate the income for Airbnb in the year of 2023 we made: mean_price * final_occupation_rate * 365 (days in a year)* 50 (number of apartments) * 0.85 (tax that Airbnb charges) * 0.6 (assuming that 60% of 50 apartment will be used for Airbnb rent).
    2) To calculate the income due to apartment rent: 0.4 * 50 (assuming that 40% of the 50 apartment will be used for rent* 2500 (rent value)* 12 (months of the year).
    3) The outcome due to building is given by: data['sale_price'][0] (terrain value) + 5 (number of flors)* 1300 (m² for floor, 10 apartments of 100 m² per floor and 300 m² for extra things as hall, and so on)* 2000 (m² value to build).
    4) Calculating the outcome fee: data['cost'][0] * 0.6 * 50.
    5) For years after 2023, a 10% raise in income and outcome values is add due to annual readjustment.

3 RESOLUTION

The following questions needed to be answered:

- What is the best property profile to invest in the city?
- Which is the best location in the city in terms of revenue?
- What are the characteristics and reasons for the best revenues in the city?
- We would like to build a building of 50 apartments in the city, where should we build it

and how should the apartments be designed in order to be a great investment?
- How much will be the return on investment of this building in the years 2024, 2025 and 2026?

In this section, it will be explained the order followed by the ‘main.py’ archieve.

3.1 First Step

The dataframes are created and the csv files are read. The dataframe called details receives the 'Details_Data.csv'; viva receives the 'VivaReal_Itapema.csv'; price receives the 'Price_AV_Itapema-001.csv' and mesh receives the 'Mesh_Ids_Data_Itapema.csv'.

The column 'airbnb_listing_id' from mesh and price dataframes are changed to ‘ad_id’, since they have the same information of the column ‘ad_id’ from the details dataframe.

The column ‘date’ from price dataframe is changed to datetime type.

3.2 Getting Neighborhoods

A new dataframe is created using the function get_location() and inputing mesh dataframe. As explained before, a dataframe with two columns (‘ad_id’ and ‘suburb’). At first, it was used the details dataframe, but it was taking too long and lots of redundant work would be done, because the same property appears more than once in the dataset. The mesh dataframe has one record to each property, so, using this dataframe is faster and free of redundant work. It takes about 13 minutes to get the neighborhood name, using the details dataframe could last more than two hours.

3.3 Getting date and merging prices

Using the function date_prepare(details) we generate the max_date and min_date with the maximum and minimum date values from the details dataframe. It also creates a column called ‘date’ in the details dataframe.

Now, the details and neighborhood dataframe wer merged on ‘ad_id’. Each ‘ad_id’ now has its respective neighborhood.

Using the whole price dataframe was being difficult because it raised ‘MemoryError’ due to its size (about 4,5 Gb and 43 millions lines). To solve this problem, the dataframe was

filtered by the dates present in the details dataframe. That is why max_date and min_date were taken. After that, no more MemoryError was raised.

A dataframe called price_backup was created only with the columns 'ad_id', 'date', 'price'. A new details dataframe was created merging details with price_backup. Now, every property has a price for a given date.

The dataframe details_backup was created to group each property with the data, price and number of bedrooms and bathrooms.

3.4 Getting the 10% most wanted houses

To get the most_wanted dataframe, which contains the 10% most wanted houses, we used most_wanted_properties() function, selecting the houses with more than 90% occupation rate. This function was explained in the section 2. most_wanted_ad_id array gets the index of the most_wanted properties.

Then, the duplicated values are dropped and inly houses with price lower than R$30000,00. As explained before, some properties had prices such as ‘300350’. I verified the values of some of these properties by entering in the link available and the value was about R$ 300,00. So, this leads us to believe that for some reason it was collected a price interval from R$ 300,00 to R$350,00.

Using the function most_wanted_characteristics() in the dataframe created, six variables could be taken:

- most_wanted_suburb: gives the neighborhood with the highest number of hoses in the 10% most wanted;
- percent_sub: gives the percentage of houses in the most wanted neighborhood;
- percent_sub_with_char: gives the percentage of houses in the neighborhood with the main characteristics;
- most_wanted_bedrooms: gives the number of bedrooms that are most wanted;
- most_wanted_bathrooms: gives the number of bathrooms that are most wanted
- percent_char: gives the percentage of houses that have the main characteristics.

3.5 Answer to question one

The best property profile to invest profile to invest is with 3.0 bedrooms and 2.0 bathrooms.

Considering the properties with more than 90% of occupation rate between 2022-06-06 and 2022-12-05 there are 25.61% houses with these characteristics.

The neighborhood with the highest amount of houses is Meia Praia, there are 55.86%, and this neighborhood has the highest amount of properties with the best profile, about 17.98%.

The images below show the results used for the question one.

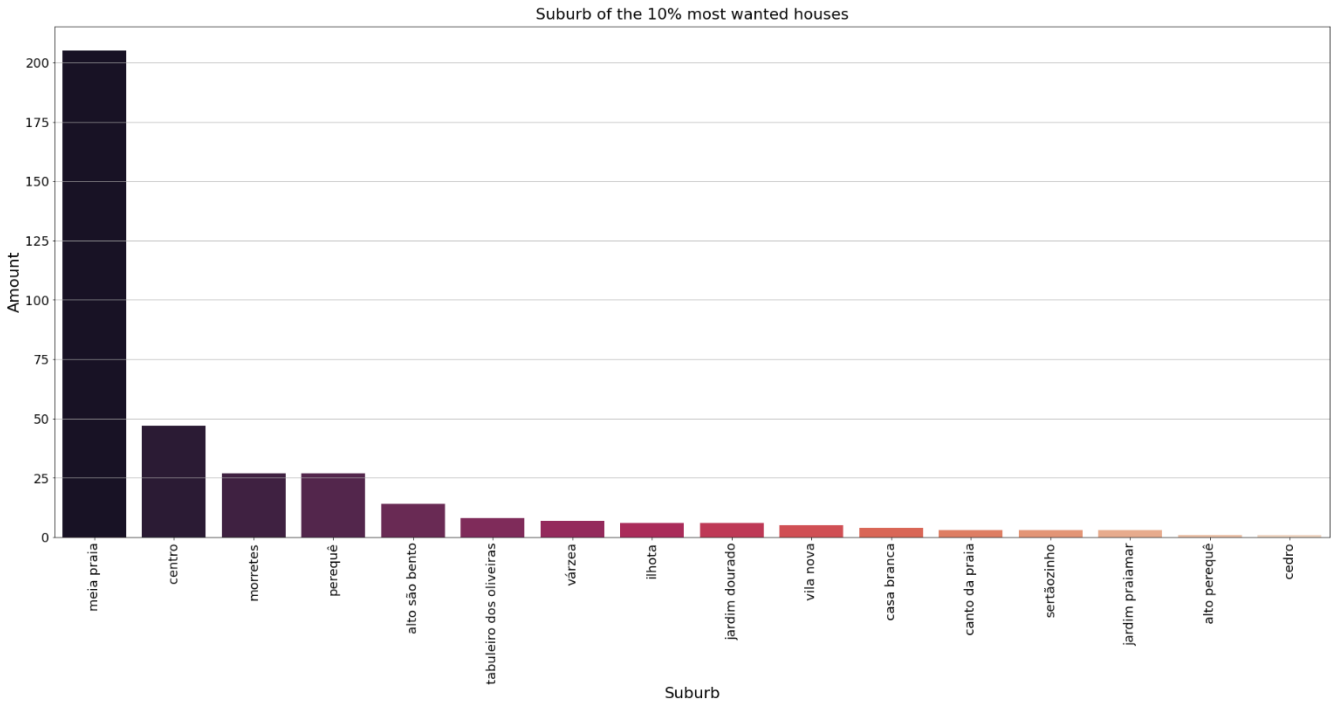

Figure 1 – Amount of house per neighborhood of the 10% most wanted houses.

Figure 1 shows the distribution of houses per neighborhood. As we can see, Meia Praia has the highest amout of houses, followed by Centro and Morretes.

Figure 2 shows the price distribution. It is possible to see that most properties are between R$ 400,00 and R$ 500,00 the daily.

Figure 3 shows the number of bathrooms distribution and it shows us that most of houses have 2 bathrooms.

Figure 4 shows the number of bedrooms distribution and it shows that most houses searched have 3 bedrooms, but 2 bedrooms also has a considerable amount.

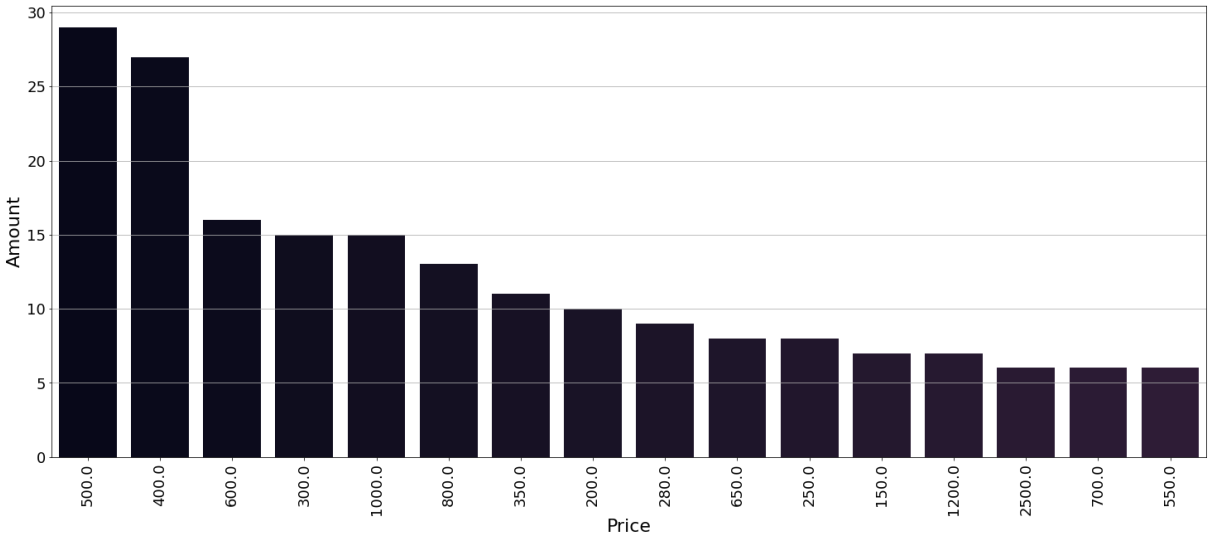

Figure 2- Price distribution for the 10% most wanted houses.

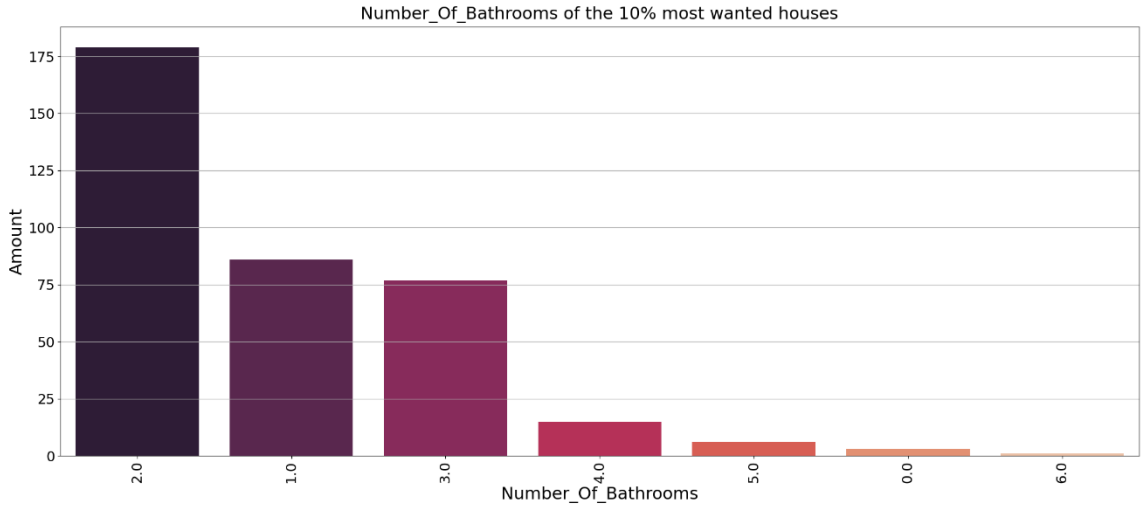

Figure 3- Distribution of number of bathrooms for the 10% most wanted houses.

Figure 4 - Distribution of number of bathrooms for the 10% most wanted houses.

3.6 Working with Viva dataset

3.6.1 Adjusting Neighborhoods Names

When checking the neighborhoods names from viva dataframe, there were some names that wasn't present in the internet [1], so, it was needed to check one by one and several names were 'wrong' (the names exist, but it is a local nomenclature), so, we corrected it using neighborhood_adjustment().

3.6.2 Visualizing and treating outliers

Using boxplot_ploter() Figures 5 and 6 were created.

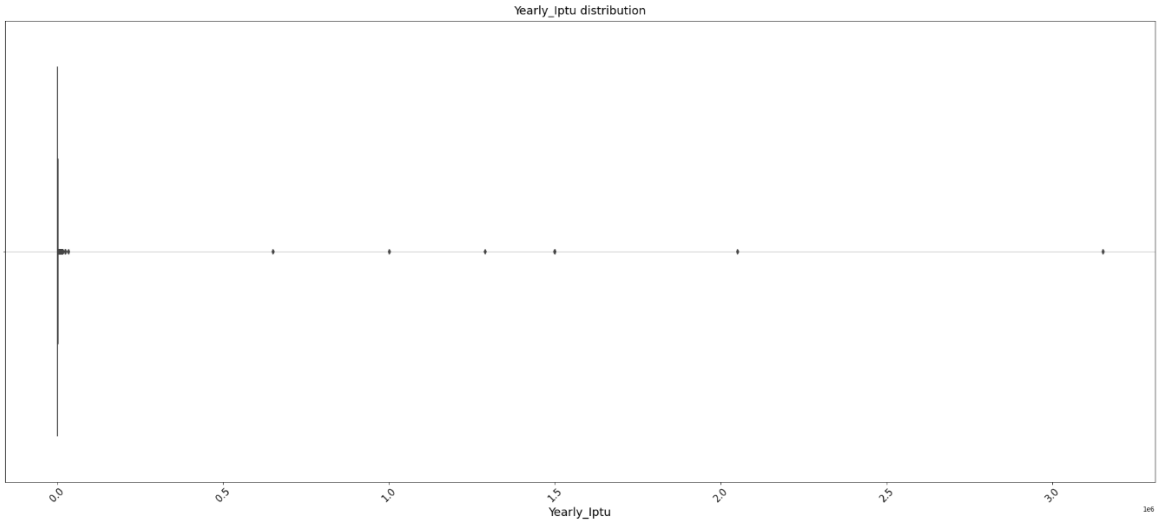

Figure 5 – Boxplot visualization for yearly iptu.

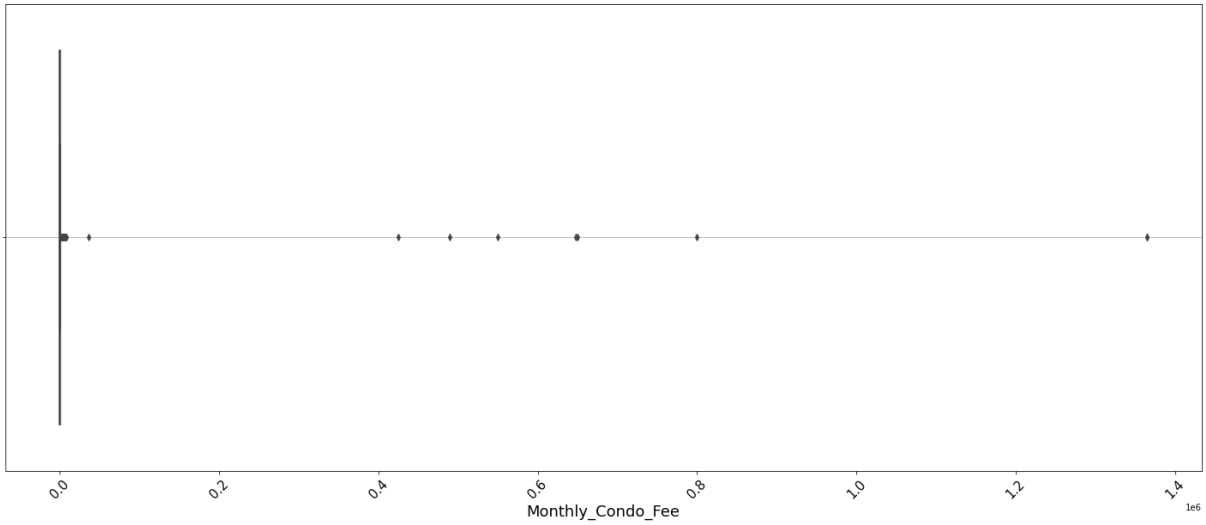

Figure 6 – Boxplot visualization for monthly cond fee.

Both Figures 5 and 6 shows that there are outliers values and some of them are absurd, such as monthy condo fee of R$ 1.4 million.

Using the outlier_detector() function, we identify the outliers values and delete them by the Tukey's Method, i.e., identifying the 1st (Q1) and 3th (Q3) quartiles, calculating the interquartile range (IQR). The values below Q1 – 1.5 * IQR and above Q3 + 1.5 * IQR are considered outliers, and these values are excluded, returning the dataframe without the outliers values. The dataframe corrected can be seen in Figures 7 and 8.

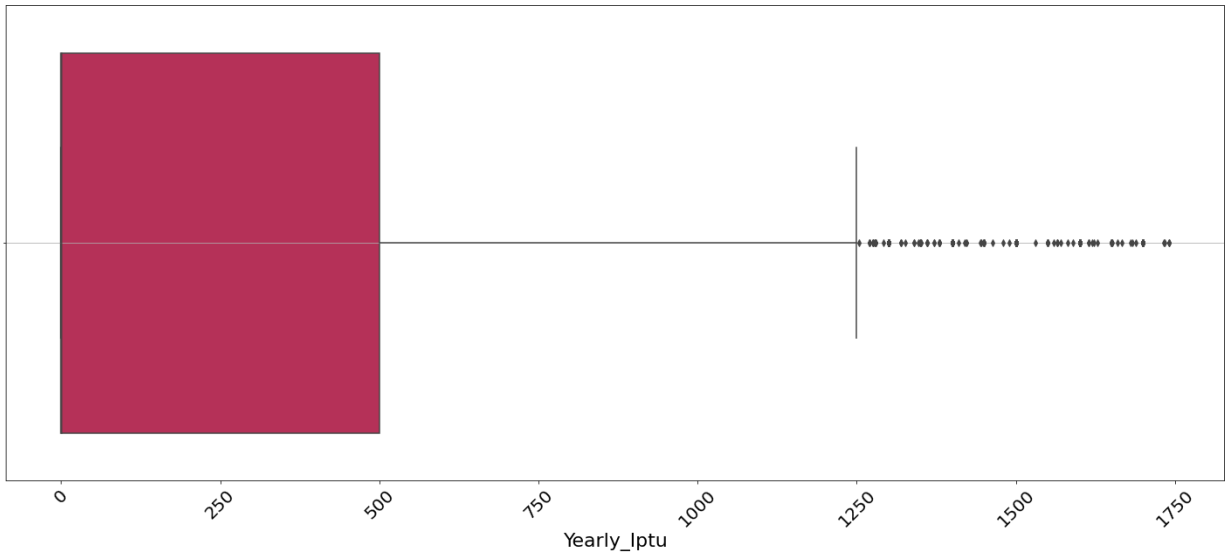

Figure 7 – Boxplot visualization for yearly iptu corrected.

Figure 8 – Boxplot visualization for monthly cond fee corrected.

3.7 Finding the best location in terms of revenue

With the viva dataframe cleaned, it is possible to find the best location in terms of revenue. Using the get_number_of_houses() function we input the mesh, neighborhood and viva dataframes. And it is possible to get the numbers of house in each neighborhood in Airbnb dataset and Viva Real dataset. This function was explained in section 2.

To correct the missing values of viva dataframe we use the adjusting_iptu_and_monthly_fee() function, that fill the missing values for the properties available. This function was also explained in section 2, but a reminder of how it works; for neighborhoods with more than 100 properties, to give a most accurate statistical measure for each neighborhood. The neighborhood with less than 100 properties, the missing values are filled up with the mean value of the total dataset, i.e., the whole city.

Using the function maintain_houses() we maintain only the neighborhoods with more than 50 houses. This was done to avoid data bias for the neighborhoods with a few houses.

The year_income() function calculates the price values for each neighborhood in a year. And year_outcome() function calculates the outcome values for each neighborhood in a year.

Finally, a dataframe called revenue is created using highest_revenue() and the best location in terms of revenue is Casa Branca.

3.8 Answer to question two

The best location in terms of revenue is Casa Branca. Followed by Morretes and Meia Praia.

3.9 Answer to question three

Despite of Meia Praia be the most wanted neighborhood, Casa Branca and Morretes have higher revenue values than Meia Praia. This happens because the sale price for each house tends to be 36.0% lower than for 'Meia Praia', even though the anual income is similar.and, also, the cost per year is highr for Meia Praia.

Morretes has a cost and sale price similar, but the income is 161.0% higher for Casa Branca, this makes the revenue be 1.7 times greater than the revenue for Morretes.

So, the reasons that make Casa Branca have the highest revenue value is because it has the highest income per house and the lowest sales price per house and also it has the second lowest cost values.

3.10 Fiding building location

Now, let's prepare for answer the question four. First, a dictionary was created linking the neighborhood to the respective cost value. cheaper_square_meter() receives viva dataframe and returns a dataframe with the columns 'listing_id', 'listing_desc', 'sale_price', 'address_neighborhood', 'unit_type', 'usable_area', 'total_area' and 'm2_value'. This function also drops the properties that have the word 'condomínio' in 'listing_desc'. Since we are looking for a land to build a building with 50 apartments, we are not interested in lands located in a condominium.

cheaper_total_cost() sums the cost per neighborhood to the m² value, and, as the dataframe is orderly by the lowest value, we have the land with the lowest value, that is the best land to build the building.

get_parking_spaces() is used to find the main parking spaces in viva dataframe, so, we can define some characteristics for the apartments.

3.11 Answer to question four

There are some allotment lands to sell in Viva Real dataset. The best place to build it is the lot with listing_id = 2567555072 because it has the lowest square meter value, even if we consider the the total cost. The apartments should be designed as it was found in question one: apartments with 3.0 bedrooms and 2.0 bathrooms. By the informations in the dataset, the ideal parking space is for 2.0 cars, 36.72% of all the dataset has this size for garage.

Although 'Casa Branca' is the best neighborhood in terms of revenue, the lot avaiable are way more expensive than for this land.

3.12 Calculating the return of the building

total_occupation() gives the mean occupation in Morretes neighborhood, about 60% and get_price() gives the mean daily rental price in Morretes.

The return and profit of the building is given by the function income(). The building considered is a 5 floor building with 10 apartments of 100 m² per floor and that 60% will be

destinated to Airbnb rent and 40% will be normal rent. The function is described in the section 2.

Sincerely, I was not able to create a function to put the results in a dataframe, so, the final sequence was created:

```
year_interval = [2024, 2025, 2026]
dic = {}
for i in year_interval:
    year = i
    data = income(mean_price, final_occupation_rate, i, data_final)
    dic[year] = data

    df = pd.DataFrame(list(dic.items()),
                      columns=['year', 'name'])
    df[['return', 'profit']] = pd.DataFrame(df['name'].tolist(), index=df.index)
    df.drop('name', axis = 1, inplace = True)
```

This code walks through the list with the years 2024, 2025 and 2026. The income() function calculates the return and proft for each year and saves it in a dictionary. Using pd.DataFrame() we can convert the dict into a dataframe.

The value of the m² built was considered R$ 2000,00 [2] and the normal rent was considered R$ 2500,00 [3].

3.13 Answer to question five

The dataframe below shows the return and the proft for 2024, 2025 and 2026.

year	return	profit
2024	9869508.53	455304.63
2025	20407596.67	14845684.81
2026	35183206.05	33858815.43

We can see that in 2024, the return is R$9869508.53, but the proft is R$455304.63, that is, a negative value. This happens because the total value invested in buying the lot and constructing the building is not totally payed.

But in 2025, the return is R$20407596.67 and the profit is R$14845684.81, that shows that the building it is generating a positive profit.

4 CONCLUSION

This work was focused in solving the problems given by Seazone.

The best profile properties to invest are those with 3 bedrooms and 2 bathrooms, but those with 2 bedrooms are also a good choice. We found that most of these properties are located in Meia Praia, but this neighborhood has lots of houses in the whole dataset, which can explain the presence of more houses.

The best locaton in terms of revenue is Casa Branca, the second is Morretes and the third is Meia Praia. This happens because, Casa Branca has the highest income value and the lowest income value of the dataset. Also, it has the second lowest cost value (iptu + monthly condominium fee).

To build a new building with 50 apartments, the best location is in Morretes, the allotment land with id 2567555072. It has the lowest m² value and can comport a huge building.

Considering the division of 10 apartments per floor and each one with 100 m² and spending R$ 2000,00 for each m² (luxury level apartments), the building would be very profitable, covering construction costs and land purchase in one year.

5 IMPROVEMENTS

Some improvements that can be made are related to data extraction. There are lots of properties that the values were wrongly collected, like ‘500550’, ‘300350’. This would give more valid data, giving more accurate analysis.

Related to code structure, it is needed to improve class and function creation, separate each class in a .py file and each class can be responsible to deal with a specific dataframe. Also, create less dataframes, which can save some memory and increase efficiency.

6 REFERENCES

[1] https://pt.wikipedia.org/wiki/Lista_de_bairros_de_Itapema, accessed in 19/12/2022.

[2] https://www.vivendobauru.com.br/qual-o-custo-para-construir-um-predio-de-5-andares/, acceded in 26/12/2022.

[3] https://www.imovelweb.com.br/imoveis-aluguel-morretes-itapema.html, accessed in 28/12/2022.