

**ANL252**

**Python for Data Analytics**

# **Group-Based Assignment**

**July 2023 Presentation**

**Submitted by:**

| **Name** | **PI No.** |
| --- | --- |
| **DON LEE YANG** | **W2270423** |
| **ISAAC CHRISTOPHER ONG WEN XUAN** | **Q2180896** |
| **ADAM ABDULLAH ANG** | **H2210484** |
| **NUR HAMIZAH BINTE RIDWAN** | **M2311185** |

**Tutorial Group: T03**

**Instructor’s Name: Dr. Munish Kumar**

**Submission Date: 13/10/2023**

**Question 1a**

#Import Panda

import pandas as pd

#Read the dataset

df = pd.read\_csv('HDB\_GBA.csv')

#Print output

print(f"The dataset has {df.shape[0]} rows and {df.shape[1]} columns.")

We would use Pandas to import and read the CSV file. Pandas dataframes are size-mutable and can easily identify missing data with isnull() and notnull(). This gives coders the freedom to customize the data output (geeksforgeeks, 2023).

Secondly, we would use the ‘pd.read\_csv()’ function to read the file. This function imports the CSV file and identifies it as a dataframe. This allows us to relate further instructions back to the dataframe.

Next, we would use the function ‘df.shape’ to identify the number of rows and columns in the dataset. This function is a tuple of array dimensions which presents the row and column of the dataframe (Akim, 2023).

As Shapes prints the data as (rows, columns), we can use the number 0 to indicate the first number (rows) and 1 to indicate the second number (column). To make it comprehensible to readers who are unfamiliar with Python, we have coded the output as a string, clearly indicating the number of rows and columns. Which gives us the output: ‘The dataset has 1250 rows and 11 columns’.

(200 Words)

**Question 1b**

#Find and count missing values in columns

df.isnull().sum()

Output:

month 0

town 0

flat\_type 40

block 0

street\_name 1

storey\_range 0

floor\_area\_sqm 0  
flat\_model 0

lease\_commence\_date 0

remaining\_lease 0

resale\_price 134

dtype: int64

By not handling the missing values it will lead to inaccuracies in the data set. For this data set the columns with missing values are “flat\_type”, “street\_name” and “resale\_price”. With the data missing we might have concluded a different outcome and made an inaccurate statistical judgement about the data set. Compromising the integrity of the data resulting in a biased outcome. It also hinders machine learning algorithms as it cannot handle missing data. Hence handling missing values would improve the quality of the data which would meet the assumption of the statistical tests and models which assumes the data is complete and would give a better data visualisation when doing up charts.

(111 Words)

**Question 1c**

#Treat missing data (method 1)

df.dropna()

In this dataset, the number of NaN values is about 10% of the whole dataset which is 100+ rows out of 1200+ rows. The sample size is reasonable enough to carry on with the analysis even after dropping the rows with the missing values. If the dataset were to have 50% of the data missing, it will churn out an outcome that might be skewed and not usable for plotting.

#Treat missing data (method 2)  
#Flat type

df[‘flat\_type’].fillna(“Unknown”, inplace = True)

Besides dropping the rows with the missing data, we can fill it and give it some meaning.  
For above ‘flat\_type’, the name ‘Unknown’ is filled in for better visibility if there’s a need to plot it in a graph, it will be clearer.

#Treat missing data (method 3)

from sklearn.impute import SimpleImputer

import numpy as np

imputer = SimpleImputer(missing\_values=np.NaN, strategy='mean')

df.resale\_price = imputer.fit\_transform(df['resale\_price'].values.reshape(-1,1))

#(Lucy, 2022)

With the SimpleImputer library from sick-it learn, we can replace the missing values with a statistical “mean”. The mean would vary for different rows, as compared to using df['resale\_price’'] = df['resale\_price'].fillna(df[‘resale\_price’].mean())

It will give the same mean for all the rows regardless.

(198 words)

**Question 1d**

#Creating a Bar Chart

#To return objects of unique values. (Without this function, we will not be able to do construct the horizontal bar chart below as this function counts all the unique value)

town\_dataset = hdb\_dataset ["town"].value\_counts()

#Printing of Table

print (town\_dataset)

#(GeeksforGeeks, 2021)

#Accessing town\_dataset in sequence ()

town\_name = town\_dataset.index

fig, ax = plt.subplots(figsize =(16, 9))

##Creating a bar plot (Matplotlib Bars, n.d.)

towns = ax.barh(town\_name, town\_dataset)

#Label barcharts

ax.bar\_label(ax.containers[0])

#Setting x\_label as the number of flats in each town

ax.set\_xlabel('Number of Resale HDB Flats')

#Setting y\_label as the town name

ax.set\_ylabel('Town')

ax.set\_title('Resale HDB Flats in Each Town')

ax.grid (axis = 'x')

plt.show()

JURONG WEST 104

TAMPINES 91

BEDOK 89

WOODLANDS 84

SENGKANG 75

YISHUN 70

HOUGANG 62

BUKIT BATOK 61

ANG MO KIO 59

CHOA CHU KANG 58

PASIR RIS 51

BUKIT PANJANG 50

BUKIT MERAH 49

PUNGGOL 48

KALLANG/WHAMPOA 42

SERANGOON 38

GEYLANG 35

QUEENSTOWN 34

TOA PAYOH 33

JURONG EAST 26

BISHAN 25

SEMBAWANG 24

CLEMENTI 22

CENTRAL AREA 15

MARINE PARADE 3

BUKIT TIMAH 2

Table 1

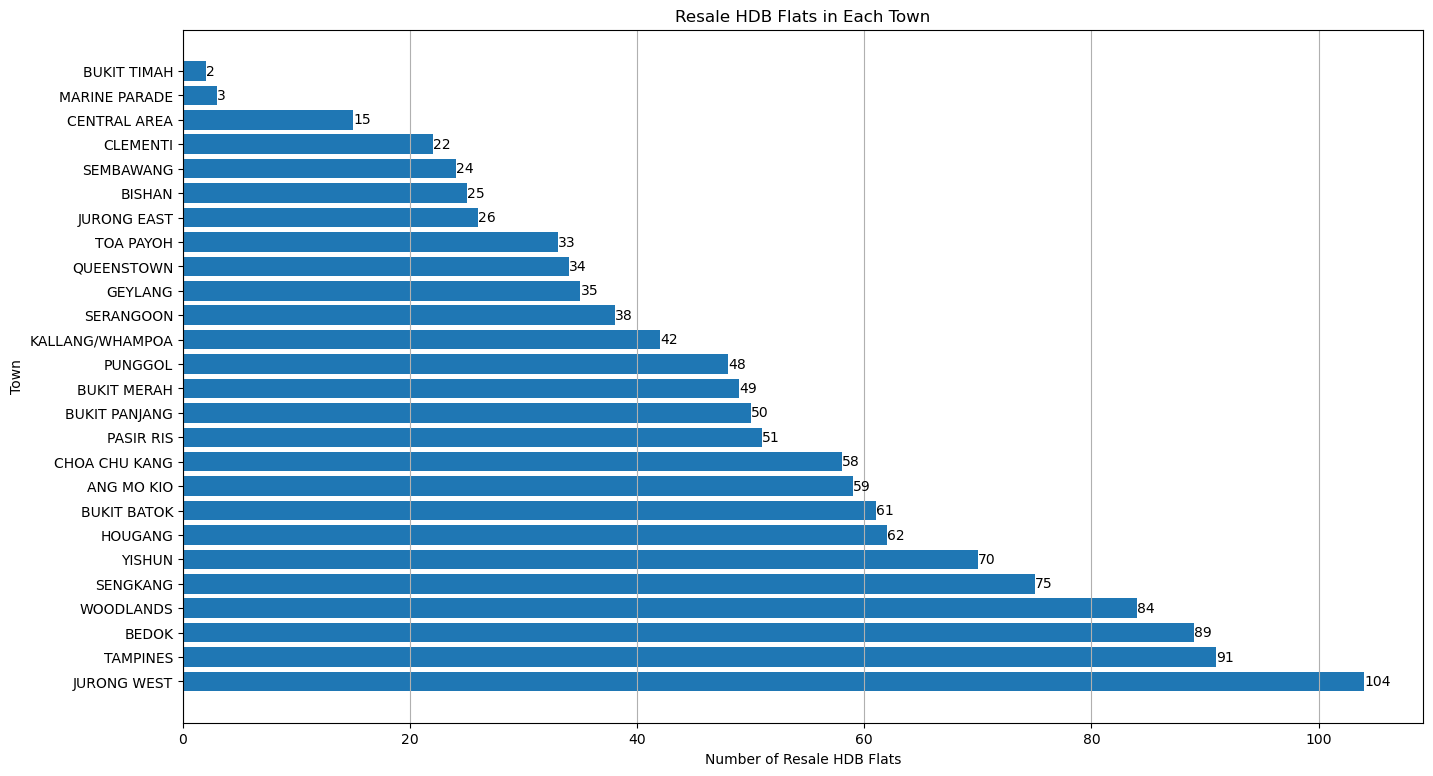


Figure 1

By constructing the horizontal bar chart in Figure 1, we can observe that the highest number of resale flats are in Jurong West, and the least are in Bukit Timah. Furthermore, we observe the highest number of resale flats falls between 20 and 40, and the lowest number falls in the range of more than 100.

Given that Bukit Timah has the fewest resale apartments in Figure 1, we can assume that the apartments there may be exclusive and prime. Thus, there are few resale apartments in that town. However, Jurong West has the highest percentage of resale flats, which may suggest that either the prices are affordable or that there are a larger volume of apartments in that town.

#Creating a histogram

#Importing of libraries

import matplotlib.pyplot as plt

import pandas as pd

#Reading HDV csv files, but specifically the resale prices

hdb\_resale\_prices = pd.read\_csv ("GBA\_HDB.CSV", usecols = ["resale\_price"])

#Printing of Table

print(hdb\_resale\_prices)

#Plotting of histogram

plt.title ("Resale Prices Histogram")

plt.hist (hdb\_resale\_prices, bins = None , range = None , align = "mid", orientation = "vertical", rwidth = None , color = None)

# Labelling the chart

plt.xlabel("Resale Price")

plt.ylabel("Number of Units Sold")

# Display the plot

plt.show()

resale\_price

0 255000.0

1 275000.0

2 285000.0

3 290000.0

4 290000.0

... ...

1245 460000.0

1246 500000.0

1247 525888.0

1248 538000.0

1249 550000.0

Table 2

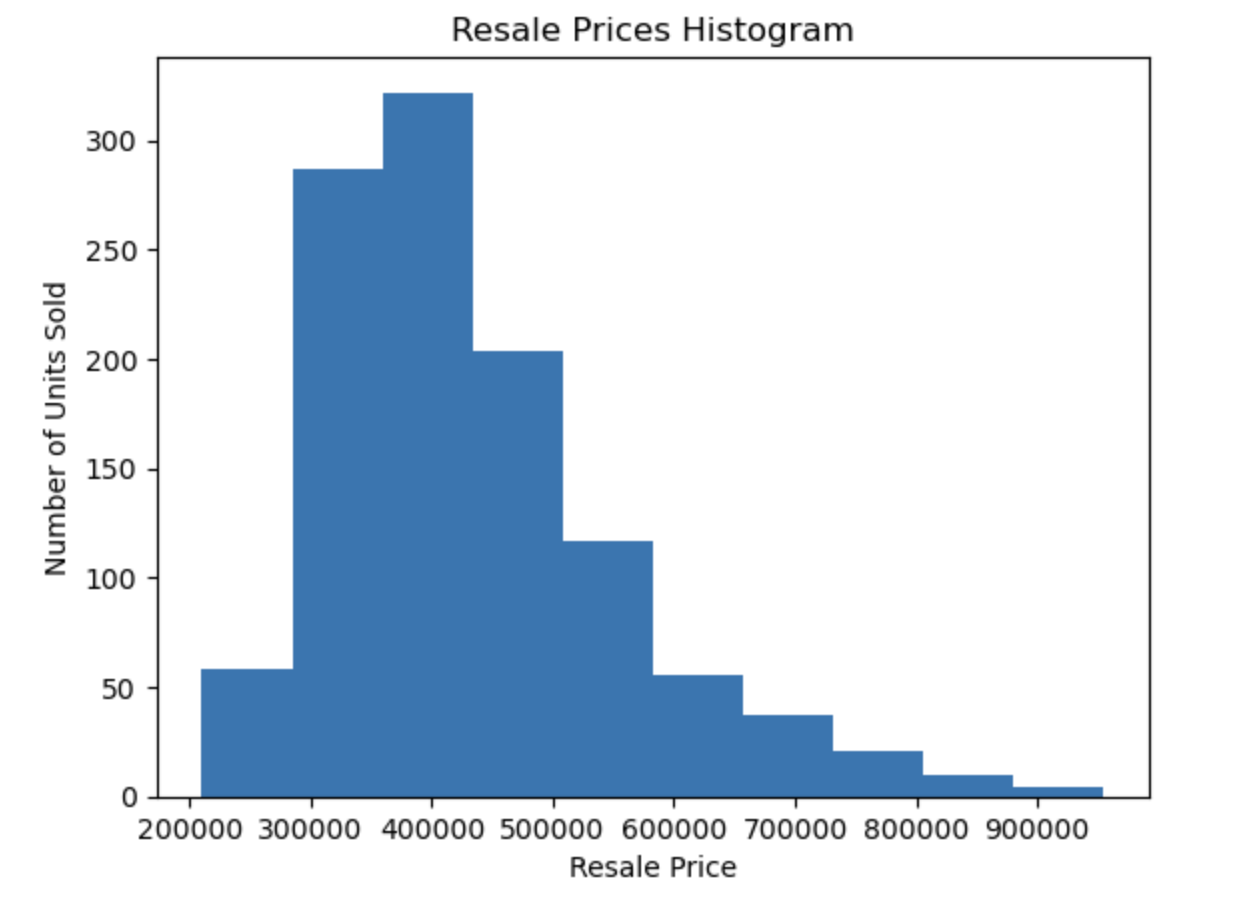


Figure 2

The histogram above is all the 1250 resale prices of HDB flats imported from the CSV file "GBA\_HDB." The histogram in Figure 2 is slightly skewed to the right. This indicates that the mode is less than the median price, while the median price is less than the mean price. We observe in the histogram that most resale prices are within the range of $300,000 to $500,000. We have also noted very few resale prices between $700,000 to $900,000. This suggests that units priced lower are more popular now.

The team also notes that we may use this histogram to compare with the horizontal bar chart in Figure 1 and determine whether our presumption that particular HDBs are exclusive and prime is accurate. For instance, we can contrast the sparse data in the $700,000 to $900,000 price range with the fewest resale HDBs. We can infer that our assumption is correct if the list of HDBs falls inside the range.

#Creating a Scatter plot

#Importing of libraries

import matplotlib.pyplot as plt

import pandas as pd

#Plotting a scatter plot

hdb\_resale\_prices = pd.read\_csv ("GBA\_HDB.CSV", usecols = ["resale\_price"])

hdb\_size = pd.read\_csv ("GBA\_HDB.CSV", usecols = ["floor\_area\_sqm"])

plt.title ("Price vs Size of House")

plt.xlabel("Area in ft")

plt.ylabel("Price of House in thousands")

plt.scatter (hdb\_size , hdb\_resale\_prices , color = None , marker = None , linewidths = 3 , edgecolors = None)



Figure 3

The above figure is a scatter plot of the price of a house versus the size of the house. The purpose of plotting a scatter plot is to observe if there is any correlation between the house's cost and the house's size.

Observing the scatter plot, we can see a positive correlation between the price of the resale flats and the flat area as the scatter plots go upward. There are a few data within the 40 sqm range, which only cost around $200,000 to $300,000, while a few data within the 180 sqm area cost upwards of $900,000. Thus, we can assume that the bigger the house, the more expensive it is.

Additionally, there are a sizable number of outliers such as a unit larger than 60sqm, costing less than a 40 sqm unit. While a 120 sqm unit and a 180 sqm unit cost upwards of $900,000. This suggests that aside from the size, there are other deciding factors such as location which affects the price.

(452 words)

**References**

Akim, A. (2023). *How to return the shape of a DataFrame in Pandas*. Retrieved from educative:

<https://www.educative.io/answers/how-to-return-the-shape-of-a-dataframe-in-pandas>

GeeksforGeeks. (2021). Python Pandas Index.value counts. *GeeksforGeeks*.

<https://www.geeksforgeeks.org/python-pandas-index-value_counts/>

geeksforgeeks. (2023, March 22). *Python Pandas DataFrame*. Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/python-pandas-dataframe/#Basics3>

*Matplotlib bars*. (n.d.).

<https://www.w3schools.com/python/matplotlib_bars.asp>

Lucy (2022) *How to handle missing data using SimpleImputer of Scikit-learn.* Retrieved from Code Underscored:

<https://www.codeunderscored.com/how-to-handle-missing-data-using-simpleimputer-of-scikit-learn/>