Awarding Body: Arden University
Programme Name: Data Science
Module Name (and Part if applicable): Programming for Data Analytics
Assessment Title: A Dataset Analysis of Housing Information in Berlin
Student Number: STU221742
Tutor Name: Mohammad Amin Mohammadi Banadaki
Word Count: Please refer to the Word Count Policy on your Module Page for guidance


ARDEN
UNIVERSITY

Introduction

In this analysis, I have focused on Python, for processing data and using statistical models and exploratory data analysis ([1]EDA) to analyze the Berlin housing dataset and study the characteristics affecting housing prices in this metropolitan. Factors, such as Physical and qualitative conditions, time construction, and renovation conditions of the houses are investigated. In addition, important factors such as the total square foot, which has been claimed to have a straight impact on price are surveyed. Meanwhile, the correlation between living area and living square feet is studied. The purpose of these analyses is to gain the ability to predict housing prices by finding the affective relationships between price and other factors.

Data Description:

The data set has the Price variable as The most important dependent variable and its relationships are supposed to be examined here. Bedrooms, Bathrooms, floors, condition, grade, built, and renovated are independent variables. Sqft_living, sqft_area_living, and sqft_totall are the continuous factors.

Step 1: Upload and Preliminary review

1.1 Libraries:

They are modules that contain a set of different functions, variables, and classes that allow programmer to achieve their goals without writing long code and just by calling libraries and their subset functions. It should be noted that the biggest reason for the popularity and power of Python is for these libraries.(VanderPlac,2016)

```
#liebraries

import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

Figure 1- python codes for importing libraries

Pandas:" provides high-level data structures and functions designed to make working with structured or tabular data fast, easy, and expressive. Since its emergence in 2010, it has helped make Python a powerful and productive data analysis environment" (McKinney,2017, p. 4).

[1] The first step is to analyze the data, which leads to understanding the data structure, discovering the relationships between variables and identifying patterns.

Seaborn: It offers tools for visualizing designed data. Drawing advanced and correlation graphs, coordination with Panda, and the ability to perform styling tasks.

Statsmodel.api: A library that investigates probabilities in hypothetical tests and statistical analyses. Meanwhile, analyze relationships between variables.

Matplotlib.pyplot: This module allows users to draw linear and columnar charts, and allows the user to control and edit the appearance of the diagrams, as well as interact with the data.

1.2 Upload File:

```
data = pd.read_csv('BerlinHousing4049.csv') # Uploading the csv file
data.info()
data.describe()
```

Figure2- Python code for uploading and accessing the file

Firstly, we access the Data Frame where we upload the dataset from the root of Jupiter platform file

Note: in this method, abnormal values are not considered.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9997 entries, 0 to 9996
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             9988 non-null   float64
 1   bedrooms          9986 non-null   float64
 2   bathrooms         9988 non-null   float64
 3   sqft_living       9990 non-null   float64
 4   sqft_total        9992 non-null   float64
 5   floors            9997 non-null   float64
 6   condition         9996 non-null   float64
 7   grade             9997 non-null   int64
 8   built             9994 non-null   float64
 9   renovated         9997 non-null   int64
 10  living_area_sqft  9986 non-null   float64
 11  build             9994 non-null   object
 12  renovate          458 non-null    object
dtypes: float64(9), int64(2), object(2)
memory usage: 1015.4+ KB
```

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft
count	9.988000e+03	9986.000000	9988.000000	9990.000000	9.992000e+03	9997.000000	9996.000000	9997.000000	9994.00000	9997.000000	9986.000000
mean	5.333549e+05	3.364711	2.063301	2053.169870	1.604302e+04	1.431930	3.446379	7.594278	1967.26636	91.462639	1976.015021
std	3.769158e+05	0.910943	0.765345	911.453717	4.505838e+04	0.511857	0.666369	1.165926	27.98690	417.444121	672.388027
min	7.500000e+04	1.000000	0.500000	380.000000	5.720000e+02	1.000000	1.000000	3.000000	1900.00000	0.000000	620.000000
25%	3.150000e+05	3.000000	1.500000	1410.000000	5.426500e+03	1.000000	3.000000	7.000000	1950.00000	0.000000	1490.000000
50%	4.458190e+05	3.000000	2.000000	1890.000000	7.916500e+03	1.000000	3.000000	7.000000	1969.00000	0.000000	1830.000000
75%	6.399250e+05	4.000000	2.500000	2500.000000	1.118000e+04	2.000000	4.000000	8.000000	1990.00000	0.000000	2340.000000
max	7.700000e+06	11.000000	8.000000	12050.000000	1.651359e+06	3.500000	5.000000	13.000000	2015.00000	2015.000000	5790.000000

Figure3- Preview of Data Frame

Step2: Preprocessing:

Based on these data previews, the data framework should be modified with standardization, data type modification, and integration processes.

The **ITU-T** process is a data preparation and standardization process that is significant in data analysis Science, because it causes increases the quality of data and leads to standard modeling and analyzing.

2.1 Type Conversion and Standardization

The values entered in the Build and Renovation fields refer to the year of construction and renovation of the houses. According to the type of input data, which is Object, the datetime data type is more logical for these factors.

```
#Type Conversion & Correction
data['build'] = pd.to_datetime(data['built'], format='%Y', errors='coerce')
data['renovate'] = pd.to_datetime(data['renovated'], format='%Y', errors='coerce')
data[['build','renovate']].info(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9997 entries, 0 to 9996
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   build     9994 non-null   datetime64[ns]
 1   renovate  458 non-null    datetime64[ns]
dtypes: datetime64[ns](2)
memory usage: 156.3 KB
```

Figure 4- Python code for converting Data types

Note:

- **format='%Y':** This argument indicates the annual format entries. It means, the input is only one year, which does not include the month and day. If the year 1990 is entered, considered as 01.01.1990.
- **format= '%Y/%m/%d':** If the input date includes month and day.

2.2 Integration

```
#Removing the Duplicates values
dupli_count = data.duplicated().sum() # collecting the duplicated rows
data_cle = data.drop_duplicates() # Removing
print(f"\nNumber of Duplicate Rows Removed: {dupli_count}")
data_cle.to_csv("BerlinHousing4049.csv", index=False) # apply the changes to file
```

Figure 5- Python code for exploratory and eliminating duplicate rows

Integration greatly increases the quality and speed of data analysis and also reduces the percentage of analysis errors.

2.3 Normalization

" Normalization is defined as a process that aims to reduce data redundancy and improve data integrity by organizing tables and relationships in a relational database" .(C.J.Date,2016)

The scale difference between the data can be seen, especially in important variables, such as price and Total square. While the scale of price is between thousands and millions scale, the Total square's scale is between ten and hundred square meters, this difference in scale affects the performance of the machine learning and reduces the influence of other variables on price.

Note: StandardScaler method: use of Mean and Deviation for standardization

```
from sklearn.preprocessing import StandardScaler#Enter class
scaler = StandardScaler() # an object of the StandardScaler class named scaler is created

data[['price','sqft_total','sqft_living', 'living_area_sqft']] = scaler.fit_transform(data[['price','sqft_total','sqft_living',
                                                                                            'living_area_sqft']])#Apply standardization
data.head(5)
```

Figure 6 - Python code for Normalization data

Note: These 4 variables scaled a range of between 3- and 3, with a mean of 0 and a Standard deviation of 1.

	price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	grade	built	renovated	living_area_sqft	build	renovate
0	-0.826359	3.0	1.0	-0.958009	-0.230639	1.0	3.0	7	1955.0	0	-0.945876	1955-01-01	NaT
1	0.012394	3.0	NaN	0.567156	-0.195302	2.0	3.0	7	1951.0	1991	-0.425329	1951-01-01	1991-01-01
2	-0.937538	2.0	1.0	-1.407878	-0.134083	1.0	3.0	6	1933.0	0	1.106565	1933-01-01	NaT
3	0.187521	4.0	3.0	-0.102161	-0.245066	1.0	5.0	7	1965.0	0	-0.916130	1965-01-01	NaT
4	-0.061903	3.0	2.0	-0.409388	-0.176701	1.0	3.0	8	1987.0	0	NaN	1987-01-01	NaT

Figure 7 - Preview of normalized Data Frame

2.4 Handling outliers

Outliers values: Outlier data are abnormal data that are at an unusual distance from the rest of the data in a set, Usually more than 3 standard deviations from the mean. They are the values that are clearly defined in the pattern or graph of the set. They are not random data, but a part of real data that is significantly different from the rest of the data.

Why we should remove outliers:

- They can affect the result of modeling, Mean, Media, and prediction, and cause deviation in the results
- Increase the complexity and sensitivity of models
- Missing reliability of analysis outputs.

Initially, we have an overview of the general situation of the Outliers, Medians, Box, and [2]Whiskers in the data frame.

[2] Whiskers: It refers to the tentacles on both sides of the data box

2.4.1 Visual Finding the Outliers in the Data Frame:

```
# Boxplot for 'price'

plt.figure(figsize=(8, 3)) #Adjust the size of the chart frame

# Price Boxplot
plt.subplot(1, 2, 1) #Division of chart display space
sns.boxplot(data['price']) #making a box diagram of the price factor
plt.title('Boxplot of Price')
plt.xlabel('Price')
plt.tight_layout() #Chart appearance settings
plt.show()
```

Figure 8- Python code for searching outlier values in price and total square foot

First, a frame with a height of 8 and a width of 3 is created. Then, using the function [3]subplot (), we determine that it can show two charts in one row and two columns. Finally, after identifying the title and horizontal label for diagrams, using the boxplot () function the distribution diagrams of the price and total square foot variables are drawn.

Boxplot (): A tool for graphical display of data, which is used to draw distribution charts and visualize data, especially outliers.

[3] plt.subplot(nrows, ncols, index)

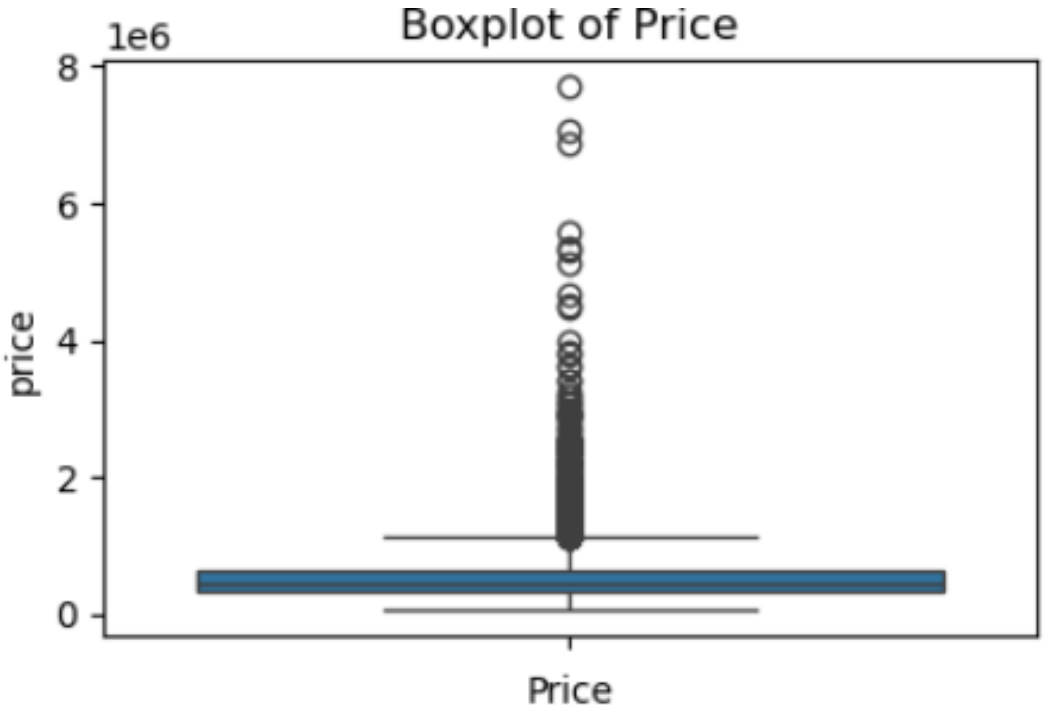

Diagram1- a preview of outliers in price and total square feet

The distribution of values in the price variable is skewed to the right, which means that a large percentage of the data is compact and located in the lower range of the graph. outliers are mostly seen in the faraway points above the box. There are few houses with very high prices.

2.4.2 Searching and Removing Outliers IQR:

```
Q1_price = data['price'].quantile(0.25)  # Assign 25% of  price column data to Q1_price
Q3_price = data['price'].quantile(0.75) #Assign 75% of price column data to Q3_price
IQR_price = Q3_price - Q1_price         #Calculate the distance between Q1 and Q2 to get the distance between these two quadrants
lower_bound_price = Q1_price - 1.5 * IQR_price  #Calculation of  Low quadrants
upper_bound_price = Q3_price + 1.5 * IQR_price   #Calculation of High  quadrants
data = data[(data['price'] >= lower_bound_price) & (data['price'] <= upper_bound_price)] # Remove the price column outliers
print("Number of rows after removing outliers:", len(data)) # show the number of removed rows
plt.figure(figsize=(8, 3)) #Adjust the size of the chart frame
plt.subplot(1, 2, 1)
sns.boxplot(data['price'])
plt.title('Boxplot of Price')
plt.xlabel('Price')
plt.tight_layout() #Chart appearance settings
plt.show()
```

Figure 9- Python code for Removing outlier values in price and total square feet

R Language Programming: "R is a language and environment for statistical computing and graphics, similar to the language originally developed at Bell Labs. It's an open source solution to data analysis that's supported by a large and active worldwide research community". (Kabacoff, 2011)

IQR: one of the most practical strategies for finding and removing outliers. First, we specify the lowest (25%) and highest (75%) data ranges using the quantile () function, this function is used in R language to calculate quartiles and other quarters. Then by subtracting them from each other, we get the distance between the two limits(IQR_sqft).

In the next step, using the formula Q1_sqft- 1.5 * IQR, identify the outlier in the left side, and by calculating Q3_sqft- 1.5 * IQR, we find the outliers in the right of the bound.

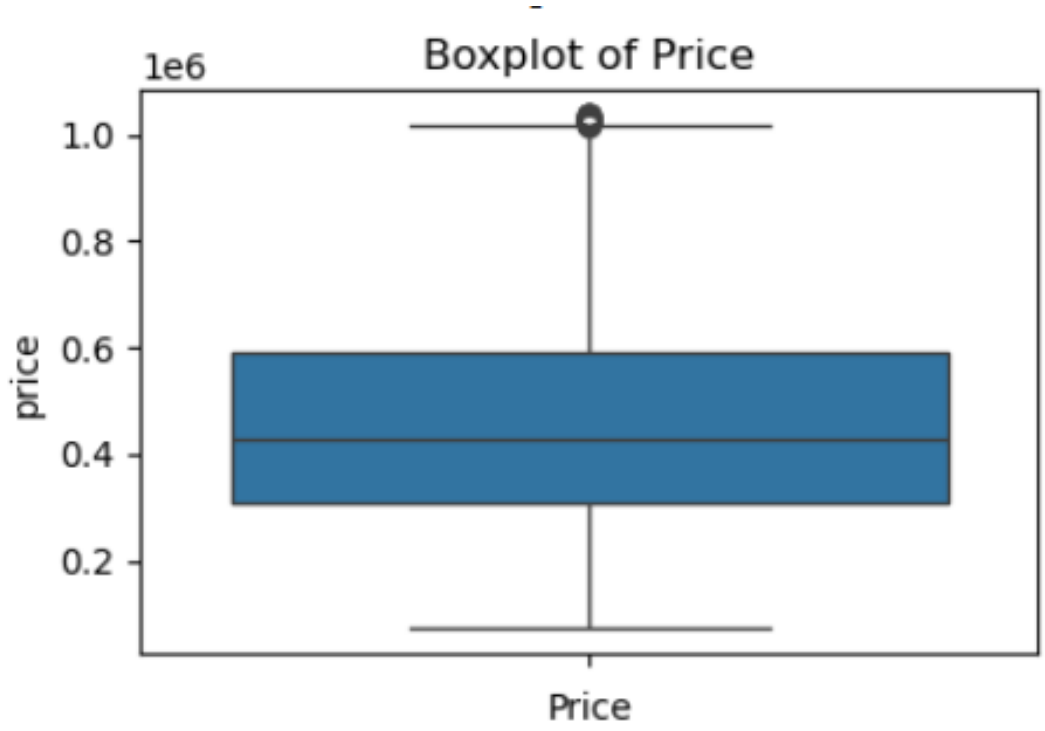

Diagram 2- a preview of Data Frame after removing the outliers

Step3: Descriptive statistics

Description

Analytical report of the data frame after pre-processing and preparation for analysis .By performing standardization on the data, the values are displayed on a standard scale, but the distribution graph is still preserved

```
#Descriptive statistics
```

```
# Statistic Report
print("Statistical report of numerical data:")
print(data.describe(include='all'))# bcause of using all, All numeric properties display all numeric and non-numeric columns
```

Figure 10- Python code of data frame descriptions

```
Statistical report of numerical data:
              price     bedrooms    bathrooms  sqft_living   sqft_total  \
count  9490.000000  9480.000000  9482.000000  9483.000000  9485.000000
mean     -0.161558     3.326899     2.003691    -0.108026    -0.010630
min      -1.216150     1.000000     0.500000    -1.835802    -0.343353
25%      -0.595344     3.000000     1.500000    -0.727588    -0.237031
50%      -0.267014     3.000000     2.000000    -0.233830    -0.182916
75%       0.176907     4.000000     2.500000     0.380625    -0.116326
max       1.572617    11.000000     7.500000     5.954608    36.298504
std       0.553746     0.896356     0.708559     0.844799     0.965668

             floors    condition        grade        built    renovated  \
count  9490.000000  9489.000000  9490.000000  9487.000000  9490.000000
mean      1.412908     3.443145     7.473551  1967.123643    80.125290
min       1.000000     1.000000     3.000000  1900.000000     0.000000
25%       1.000000     3.000000     7.000000  1950.000000     0.000000
50%       1.000000     3.000000     7.000000  1969.000000     0.000000
75%       2.000000     4.000000     8.000000  1990.000000     0.000000
max       3.500000     5.000000    12.000000  2015.000000  2015.000000
std       0.505661     0.663711     1.030255    27.736870   391.814908

         living_area_sqft                        build  \
count       9480.000000                         9487
mean          -0.094466  1967-02-15 06:49:13.009381248
min           -2.016715            1900-01-01 00:00:00
25%           -0.752530            1950-01-01 00:00:00
50%           -0.261729            1969-01-01 00:00:00
75%            0.422418            1990-01-01 00:00:00
max            5.062719            2015-01-01 00:00:00
std            0.895215                            NaN

                            renovate
count                            381
mean   1995-10-10 00:22:40.629921280
min              1940-01-01 00:00:00
25%              1986-01-01 00:00:00
50%              2000-01-01 00:00:00
75%              2010-01-01 00:00:00
max              2015-01-01 00:00:00
std                              NaN
```

Figure 11- A preview of Normalized data frame descriptions

Step4: Visualization

4.1 categorization

- Find relationships between other features and price
- Finding the strategy and influence of other fields with price
- Help to simplify the analysis

```
#Data grouping and average Catigurization
grouped_data = data.groupby(['price']).mean() #Grouping and average
plt.figure(figsize=(8, 2))
print(grouped_data.head())
data['price'].value_counts().plot(kind='bar', figsize=(6,4), color='green') # Create a column chart of frequently
plt.title('Price grouping column chart')
plt.ylabel('frequently')
plt.show()
```

Figure 12- Python code for categorization

price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition	\
-1.216150	1.0	NaN	-1.517602	0.606772	1.0	3.0	
-1.202882	1.0	0.75	-1.780939	-0.243957	1.0	2.0	
-1.197575	3.0	1.00	-1.309126	-0.124628	1.0	3.0	
-1.196249	2.0	1.00	-1.682188	0.139689	1.0	2.0	
-1.192269	2.0	1.00	-1.484684	0.090768	1.0	3.0	

price	grade	built	renovated	living_area_sqft	build	renovate
-1.216150	3.0	1966.0	0.0	-1.213586	1966-01-01	NaT
-1.202882	4.0	1912.0	0.0	-1.154095	1912-01-01	NaT
-1.197575	6.0	1954.0	0.0	-1.243331	1954-01-01	NaT
-1.196249	5.0	1951.0	0.0	-0.600828	1951-01-01	NaT
-1.192269	6.0	1949.0	0.0	-0.722785	1949-01-01	NaT

Figure 13- normalize price column categurization

price	bedrooms	bathrooms	sqft_living	sqft_total	floors	condition
80000.0	1.0	0.75	430.0	5050.0	1.0	2.0
82000.0	3.0	1.00	860.0	10426.0	1.0	3.0
89000.0	3.0	1.00	900.0	4750.0	1.0	4.0
89950.0	1.0	1.00	570.0	4080.0	1.0	3.0
90000.0	1.0	1.00	780.0	4000.0	1.0	3.0

price	grade	built	renovated	living_area_sqft	build	renovate
80000.0	4.0	1912.0	0.0	1200.0	1912-01-01	NaT
82000.0	6.0	1954.0	0.0	1140.0	1954-01-01	NaT
89000.0	6.0	1969.0	0.0	900.0	1969-01-01	NaT
89950.0	5.0	1942.0	0.0	890.0	1942-01-01	NaT
90000.0	5.0	1905.0	0.0	1150.0	1905-01-01	NaT

Figure 14- unnormalized price column categorization

Obviously, Square living factor has a significant impact on price changes. Meanwhile, factors such as build, renovation, and condition have a relative impact on housing market changes. On the other hand, the remaining features number of bedrooms and bathrooms do not have a notable effect on the price.

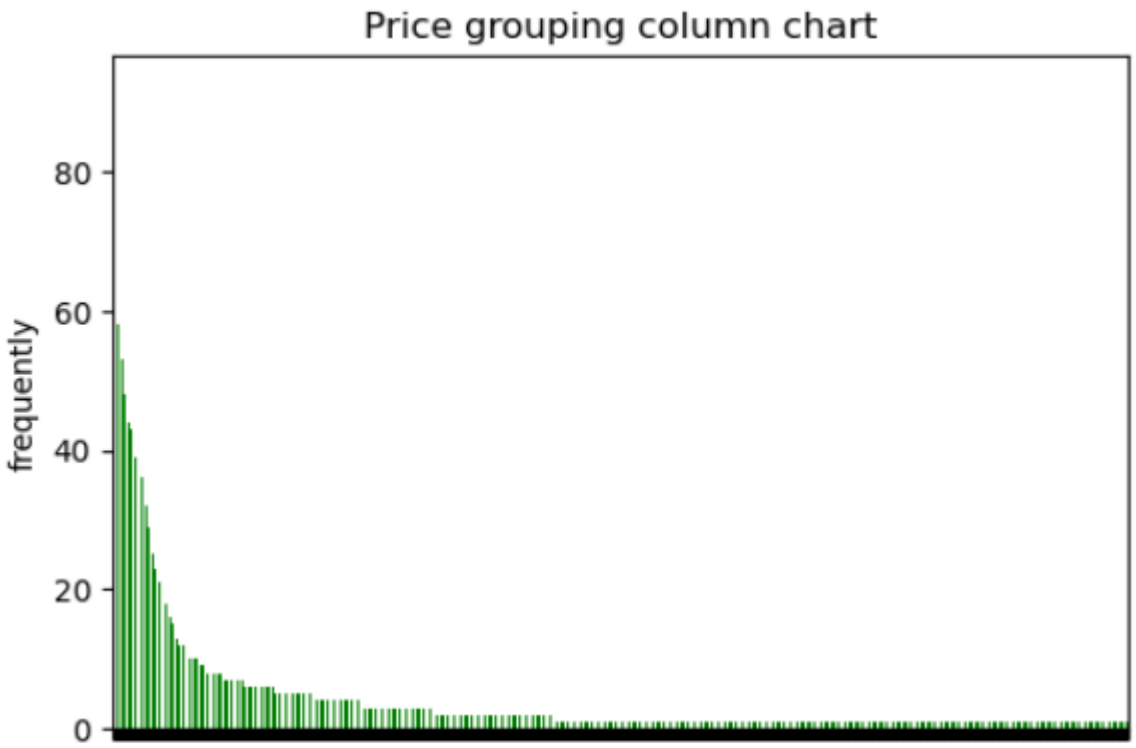

Diagram 2- a preview of the most 10 repeated values price grouping chart

4.2 Distribution Diagram of price

```
# Distribution of 'price'

plt.figure(figsize=(10, 3))   # Create a frame with dimensions of 3x8
plt.subplot(1, 1, 1)  #drawing a chart in 1 row,1 column and index=1
sns.histplot(np.log(data['price']), kde=True, bins=100, color="skyblue", edgecolor="black")# Drawing the price histogram

plt.title('Price Distribution Diagram')

plt.tight_layout()# adjusting chart appereance
plt.show()
```

Figure15- Python code for calculating the correlation

Initially, the data distribution has a slight deviation to the right, which indicates that most of the prices are distributed in the lower range of the graph, 0 to -1. Then we apply the logarithm function on data to symmetrize the distribution histogram. Therefore, the histogram has logarithmic values between 0 and -7. Most of the data is concentrated between the 0 and -2 regions. Meanwhile, the graph has outliers in distant negative points, which shows that house prices in these areas are very low.

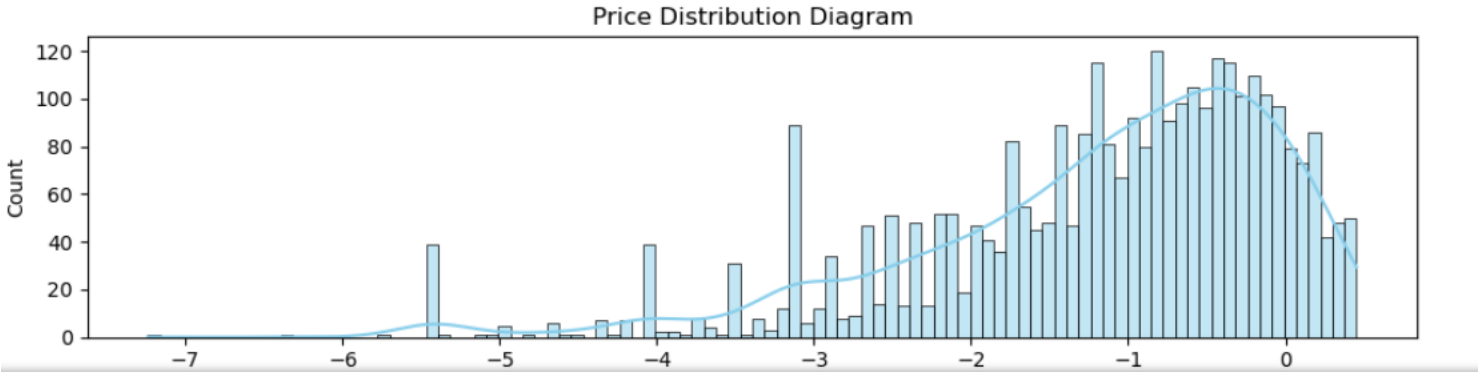

Diagram4- Distribution Histogram

4.3 Price and Square total foot analysis:

```
#Price and Square total foot analysis
# Scatter plot for 'price' vs 'sqft_total'
plt.figure(figsize=(8, 6)) #creating a frame with specific dimension
sns.scatterplot(x='sqft_total', y='price', data=data) # Drawing the scatter plot of two variables
plt.title('Price vs Total Square Footage') #Identify the title
plt.xlabel('Total Square Footage')# The title of the horizontal axis
plt.ylabel('Price')#The title of the vertical axis
plt.show()
```

Figure16- Python code for drawing scatter Diagram of price and sqft_total

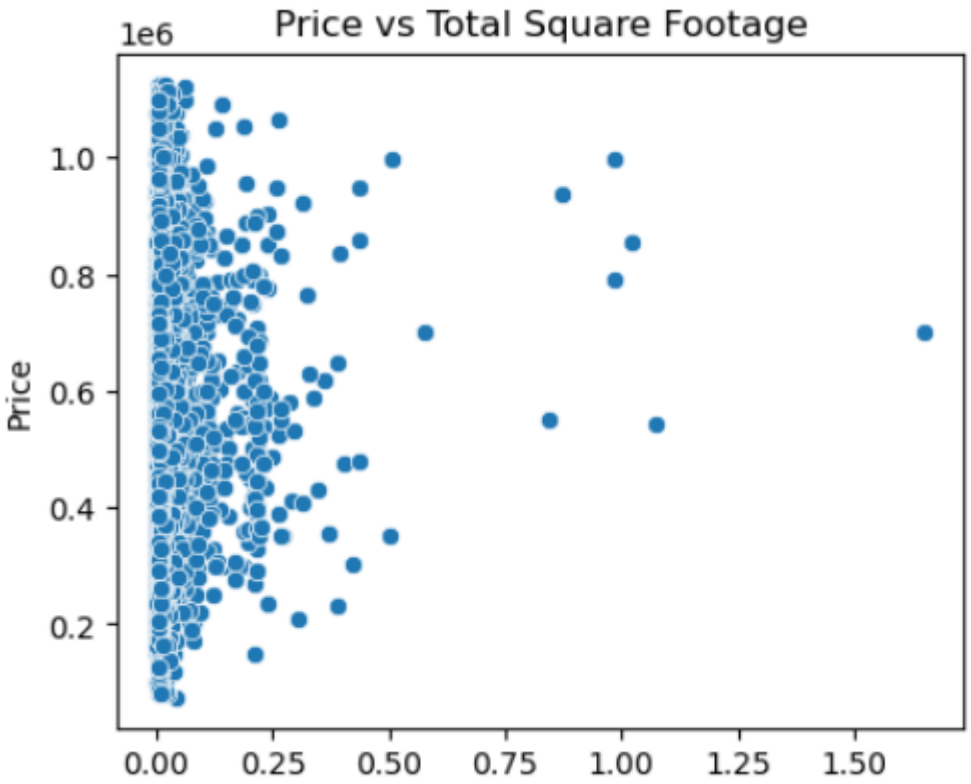

Diagram5- Scatter Histogram of price and Total square

The frequency of data is mostly between 0 and 0.25 points. It means that most of the houses have low square footage. According to the diagram, there is not even a fixed price pattern for small square footage houses and the prices are variable. While there is no linear correlation can be seen between these two variables, perhaps there are some non-linear relationships with the influence of other factors such as build, grade, and living area.

4.4 square living and Square living area analysis:

```
#Square living and Square living area analysis
#Scatter plot for 'square living' vs 'Square living area'
plt.figure(figsize=(5, 4))
sns.scatterplot(x='sqft_living', y='living_area_sqft', data=data)# Drawing the scatter plot of two variables
plt.title('sqft_living vs living_area_sqft')
plt.xlabel('Square living')
plt.ylabel('square living area')
plt.show()
```

Figure 17- python code for drawing scatter diagram square living and Square living area

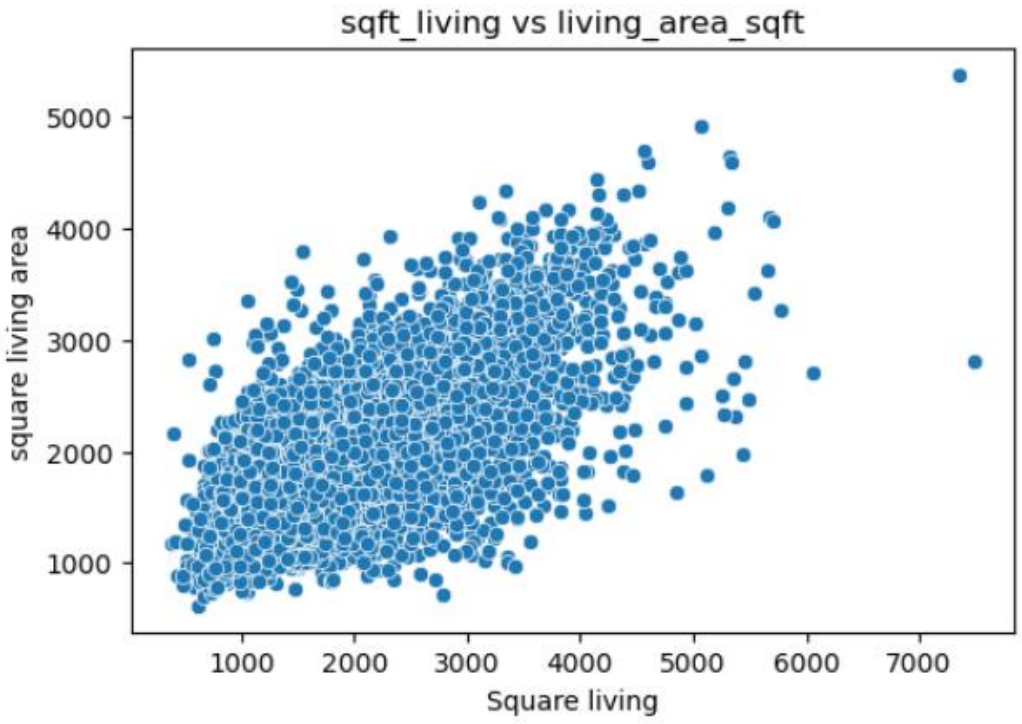

Diagram6- Scatter Histogram of square living and square living area

According to the histogram, the scale of square living is between 500 and 7000, and the square living area is between 1000 and 3500 square feet. Most of the data is scattered between 1000 and 4000 in both variables. Meanwhile, the distribution of houses is more in the middle of the graph, which shows that most houses have an area between 2000 and 3000 square feet. On the other hand, outliers are also seen, which represent houses with big and unusual living spaces (luxury houses). According to the graph, there is a linear relationship between these two variables.

4.5 price with square living & Square living area analysis:

```
# Price vs square living and Square living area
plt.figure(figsize=(5, 4)) #creating a frame with specific dimension
sns.scatterplot(x='sqft_living', y='price', data=data) # Drawing the scatter plot of two variables
plt.title('Price vs sqft_living') #Identify the title
plt.xlabel('sqft_living')# The title of the horizontal axis
plt.ylabel('Price')#The title of the vertical axis
plt.show()

plt.figure(figsize=(5, 4)) #creating a frame with specific dimension
sns.scatterplot(x='living_area_sqft', y='price', data=data) # Drawing the scatter plot of two variables
plt.title('Price vs living_area_sqft') #Identify the title
plt.xlabel('living_area_sqft')# The title of the horizontal axis
plt.ylabel('Price')#The title of the vertical axis
plt.show()
```

Figure 18- python code for drawing scatter diagram of price & square living and price & Square living area

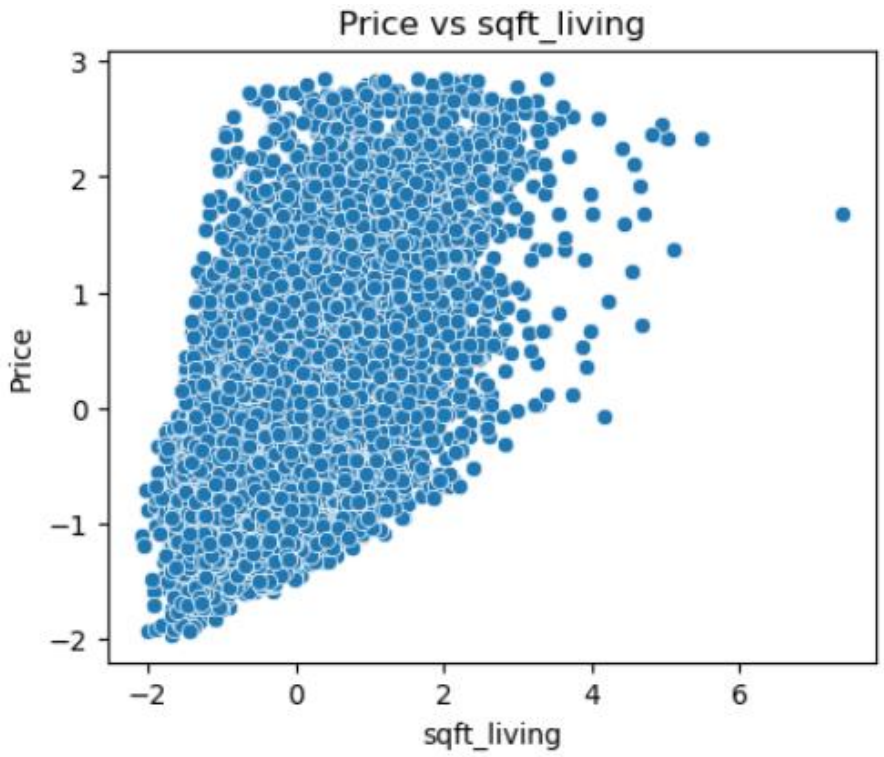

Diagram7- Scatter Histogram of square living and price

According to the graph, there is a positive relationship that the price increases as the living space increases. The extreme scatter in the data suggests that other factors also influence the price. In this chart, the outliers on the right represent large, luxurious homes with unpredictable prices. An outlier can be seen at the farthest point on the right side of the graph, which represents a house with very large dimensions and an unusual price.

Diagram8- Scatter Histogram of price and square living area

According to the graph, there is a positive but very weak correlation between these two variables, which has a lot of noise. As the living space square increases, the price of houses increases, but due to the large dispersion that is observed, the number of outlier data also increases.

Step5: Correlation

The statistical relationship and mutual effect of two factors is called correlation. So that the changes of one factor depend on the changes of another factor. Correlation is usually displayed with three numbers 0, 1, -1.

- 0: absence of correlation between two factors.
- 1: Whenever one factor increases with the increase of another factor and vice versa.
- -1: When two factors have an inverse relationship.

The price variable is a dependent variable on other factors, which has the highest correlation with sqft_total variable which refers to the total square area of the house's infrastructure. In other words, the biggest change in house price is affected by the square of the house.

```
# correlation matrix
correlation_matrix = data[['price','sqft_total','sqft_living', 'living_area_sqft','bedrooms', 'bathrooms','floors',
                           'condition', 'grade', 'built', 'renovated']].corr() #calculating the correlation matrix

# Visualize correlation matrix
plt.figure(figsize=(7,7)) #Adjust the size of the chart frame
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')#Drawing a correlation diagram as a heat map
plt.title('Correlation Matrix')# adjust title for diagram
plt.show()# depicting diagram
```

Figure 20- Python code for calculating the correlation

In this correlation chart between price and other variables, the highest correlation is with the four factors, sqft_living (61%), grade (62), living_area_sqft (56%), and Bathrooms (45%), respectively. On the other hand, condition (6%) and renovate (9%) and Build (4%) have the least impact on housing prices. As mentioned before, two variables, living_area_sqft and sqft_living have a strong relationship. interestingly, the grade variable has two negative correlations with build and condition, which is reasonable. As the years of construction increase, quality decreases. Meanwhile high- quality house requires more maintenance

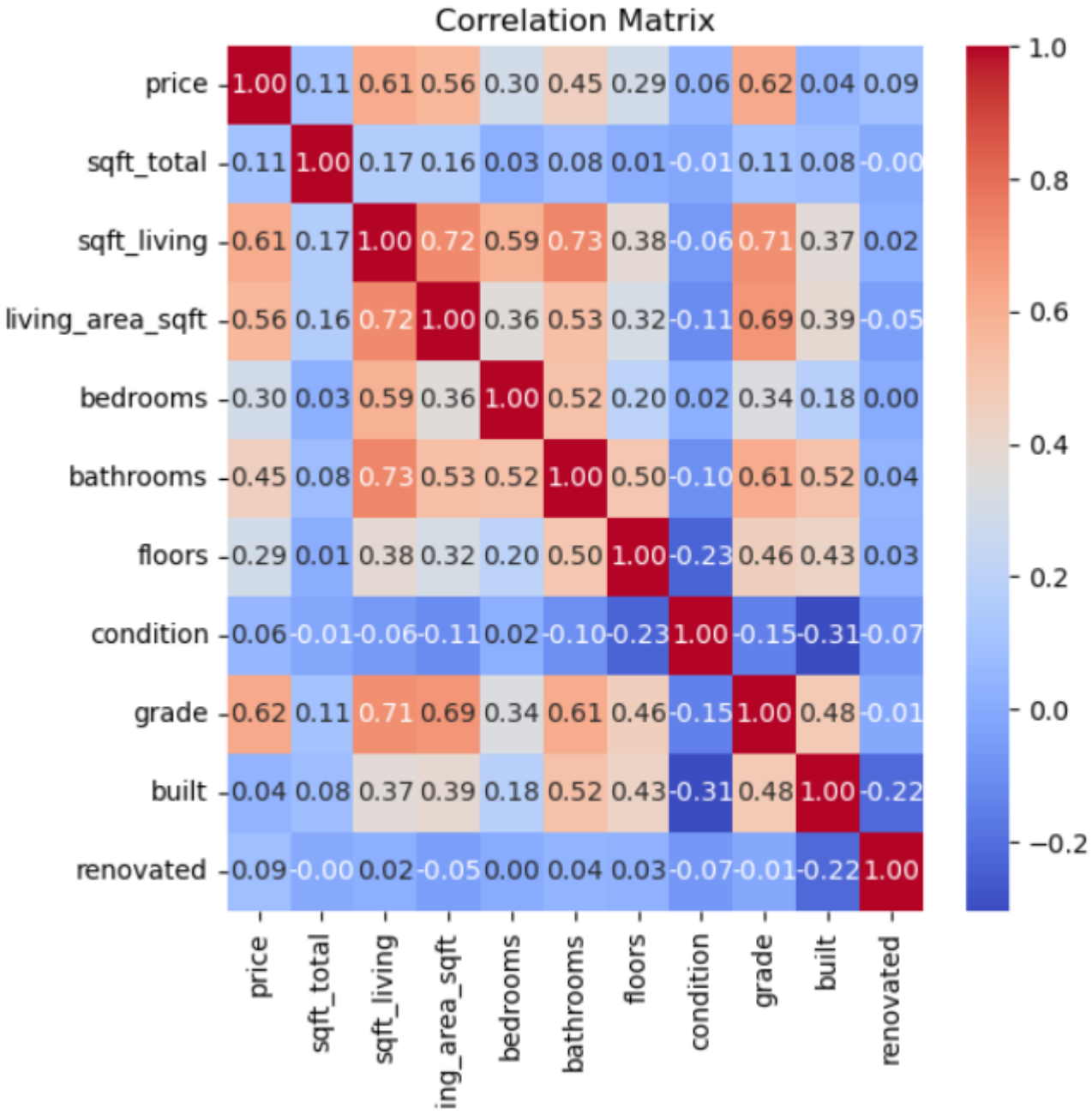

Diagram9- Heatmap diagram of the correlation all variables

Report

To conclude, analyzing Berlin housing data determines a considerable correlation and relationship between property value and other factors, such as living Area, Bedrooms, and grade which shows the feeling of peace and pleasantness of the house. Meanwhile, a mutual and positive relevance has been detected between price and Living space, which represents the highlighted effect of The vastness of space on house price. Although three factors Build, Condition, and Renovation, have the least impact on the price of the house, buyers pay attention to these items when buying. Contrary to expectations, the Total Square factor does not affect the price of the house, because the appearance and livable space are much more important for customers than the infrastructure of the house. Future analyses can exert more factors such as location, facilities, and amenities to accurately predict the real estate market.

Implications and Recommendations

- More focus on the appearance and physical condition of properties for more accurate housing price prediction
- Examining geographic data such as the level of access to the downtown, Public Transportation, recreational and educational places, markets, etc.
- Taking advantage of Machine Learning Algorithms for accurate and comprehensive forecasting. They examine a wider range of factors in comparison with traditional statistical analysis.

References:

McKinney, Wes., (2017). Python for Data Analysis[online]. 2nd Edition. O'Reilly Media, USA.

C.J.Date, (2003). An Introduction to Database Systems. eighth Edition. Pierson Education, USA.

KABACOFF, ROBERT I., (2011). R IN ACTION. Manning, NY