INTERNACIONALNI
UNIVERZITET U SARAJEVU

Fundamentals of Data Science

Term project

Professor:

Fahir Kanlić

Student:

Asja Bašović

IUS Spring 2024.

Table of contents

a) Briefly explain the data set. 3
c) Where did you find the dataset, and what format is it in? 4
d) How many observations and variables are there in the dataset? 4
e) Develop the code in Python and provide answers to the questions given below: 4

a) Briefly explain the data set.

The housing price dataset provides details about property listings in the United States of America. It includes houses built from 1900 to 2015, and it describes what areas each house has (how many floors, bathrooms, if it has a waterfront, etc.) and ranks each house in a grade ranging from 1 being the lowest to 13 being the highest. The dataset provides additional information such as the zip code, coordinates, date of listing, price, etc. It is very comprehensive as it has over 20 thousand observations and no null/missing values. The dataset uses qualitative data (price, bedrooms, etc.), however, there are also binary qualitative data (waterfront) expressed as integers (0 for no, 1 for yes), and there is ordinal qualitative data for, for instance, the condition of each house where it is ranked from 1 to 5. Additionally, nominal data such as the ID and the zip code, and datetime data such as the year a house was built are included.

The dataset includes the following variables:

- id -Unique identifier for each property
- date - Date of property listing
- price - property price in US dollars
- bedrooms - Number of bedrooms
- bathrooms -Number of bathrooms
- sqft_living - Living area size in square feet
- sqft_lot - Total lot size in square feet
- floors - Number of floors
- waterfront - Indicates if property has waterfront view (0 for no, 1 for yes).
- view – Quality level of property view (0 to 4)
- condition - overall condition rating (1 to 5)
- grade - Overall grade rating (1 to 13)
- sqft_above - Living area above ground level in square feet
- sqft_basement - Basement area in square feet
- yr_built - Year property was built
- yr_renovated - Year property was last renovated (0 if never)
- zipcode - Property location zip code

- lat - Latitude coordinate of property location
- long - Longitude coordinate of property location
- sqft_living15 - Living area size of 15 nearest properties in square feet
- sqft_lot15 - Lot size of 15 nearest properties in square feet

c) Where did you find the dataset, and what format is it in?

The dataset was found on Kaggle and it was in the .csv format.

d) How many observations and variables are there in the dataset?

There are 21613 observations and 21 variables in the dataset.

e) Develop the code in Python and provide answers to the questions given below:

```
import pandas as pd

import statsmodels.api as sm

import numpy as np

import statistics

import matplotlib.pyplot as plt

import seaborn as sns


df=pd.read_csv(r"C:\Users\Asja\Desktop\Housing.csv")

df

df.describe() #observing the data


#drop extra/unwanted columns

df=df.drop(['id','lat', 'long', 'yr_renovated', 'zipcode', 'date'],axis=1)

df.head(30)
```

```
#check the data type for each variable contained in the data set
print(df.dtypes)
df['price']=df['price'].astype(int)
df['yr_built']=pd.to_datetime(df['yr_built'])
print("****************************************")
print(df.dtypes)

#check if the null values exist in the data set
df.isnull().sum()

#check the number of unique values for each column in the data set, if they exist
df.nunique(dropna=False)

#explore whether the data set contains outliers for each variable
outliers = {}
for col in df.columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers[col] = df[(df[col] < lower_bound) | (df[col] > upper_bound)][col]
print("Outliers detected using IQR method:\n", outliers)
```

```
#if outliers are detected in the previous step, remove them from the data set for the purpose of making further analysis

#I decided to only remove the outliers of the 'price' column but i could have used

#numeric_df = df.select_dtypes(include=[float, int]) to include all relevant columns

Q1 = df['price'].quantile(0.25)

Q3 = df['price'].quantile(0.75)

IQR = Q3 - Q1

threshold = 1.5

outliers = df[(df['price'] < Q1 - threshold * IQR) | (df['price'] > Q3 + threshold * IQR)]

df = df.drop(outliers.index)
```


```
#if outliers are detected in the previous step, remove them from the data set for the purpose of making further analysis

#I decided to only remove the outliers of the 'price' column but i could have used

#numeric_df = df.select_dtypes(include=[float, int]) to include all relevant columns

Q1 = df['price'].quantile(0.25)

Q3 = df['price'].quantile(0.75)

IQR = Q3 - Q1

threshold = 1.5

outliers = df[(df['price'] < Q1 - threshold * IQR) | (df['price'] > Q3 + threshold * IQR)]
```

```
df = df.drop(outliers.index)
df


#f) Visualize at least two variables from the data
sns.countplot(data=df, x='grade')
plt.title('The total number of houses of each grade ',  fontsize = 20)
plt.xlabel('The grade of house (1=lowest grade)', fontsize = 20)
plt.ylabel('The number of houses', fontsize = 20)


sns.scatterplot(data=df, x='sqft_living', y='price')
plt.title('Living area sizes at different price points ',  fontsize = 20)
plt.xlabel('The living area size (square feet)', fontsize = 20)
plt.ylabel('The price of houses (millions of $)', fontsize = 15)
sns.set_theme(style="darkgrid")



#g) Calculate basic descriptive statistics measures of central tendency and dispersion. Explain the obtained results.
print("The mean (average) price: %1.2f" % df['price'].mean())
print("The median (middle price) price: %1.2f" % statistics.median(df['price']))
print("The mode (most frequent) price: %1.2f" % statistics.mode(df['price']))


rangeOfPrice=df['price'].max()-df['price'].min()
```

```
print("The range of prices (the difference between the minimum and maximum price): %1.2f" %
(df['price'].max()-df['price'].min()))

print("The variance (variability from the average/mean) of the price: %1.2f" %
statistics.variance(df['price']))

print("The standard deviation (the square root of variance) of the price: %1.2f" %
statistics.stdev(df['price']))


Q1_stats = df.quantile(0.25)

Q3_stats = df.quantile(0.75)

IQR_stats = Q3_stats - Q1_stats

print("The interquartile range (the distance between the first quartile and the third quartile) of the
price: " + str(IQR_stats))


#h) Develop a linear regression model by choosing one dependent and one independent variable
from the analyzed data set.

x=sm.add_constant(df['sqft_living'])

y=df['price']

results=sm.OLS(y,x).fit()

results.summary()


'"i) Explain the slope and intercept from the developed regression model.

slope: 167.3596

The slope represents the rate at which the price of the property changes with respect to the living
area size.
```

For every additional 1 square foot of living area, the price of the property increases by approximately $167.36.

Therefore, larger living areas cost more.

intercept: 1.464e+05 or 146400

the intercept represents the estimated price of the property when the living area size is zero square feet.

It represents the baseline value from which the effect of the living area size (as determined by the slope) starts.

'''

#j) Predict the future value of used dependent variable Y by using some value of independent variable X not contained in the data set.

slope= 167.3596

intercept= 146400

x1=df['sqft_living']

yfunc=slope*x1+intercept

sns.scatterplot(data=df, x='sqft_living', y='price')

fig = plt.plot(x1,yfunc, c='orange', label = 'regression line')

plt.title('Living area sizes at different price points ', fontsize = 20)

plt.xlabel('The living area size (square feet)', fontsize = 20)

plt.ylabel('The price of houses (millions of $)', fontsize = 15)

```
#checking if value is in the dataframe

result = df[df['sqft_living'] == 6000]

print(result)


new_x = 6000

predicted_y = slope*new_x+intercept


print(f"Predicted price for a lot size of {new_x} sqft_living: {predicted_y}")


df.describe() #checking if the predicted y is reasonable and it seems like it is
```