4/30/2025

Alvine Gadeu Nee Djomatchie Touko

IU AKADEMIE

**Tidying up the dataset “California Housing Prices”**

Assignments for the course: Data quality and data wrangling

**Content**

Introduction

1. Data Analysis
   1. Understand the data
   2. Remove duplicates
   3. Handle Missing Data
   4. Outliers and bad data detections
   5. Standardize Data formats
   6. Rename columns for clarity
   7. Correct data type
   8. Remove outliers
   9. Reshape data
   10. Save the cleaned data
2. Data visualization

Conclusion

Appendix

References

**Introduction**

Data analysis is especially important in today’s world, where we are constantly surrounded by vast amounts of data. However, data alone is not valuable unless it can be understood and utilized effectively and that’s where data analysis becomes essential. The first step in working with data is data wrangling, which involves cleaning and preparing the data for analysis. In the analysis phase, the data is inspected, transformed, and modeled to extract meaningful insights. Once this is done, the information can be used to drive progress and support data-driven decision-making.

Python has become the leading programming language for data analysis and is widely used in data science due to its simplicity and powerful libraries such as Pandas and Numpy for data extraction and Matplotlib, Seaborn for the visualization and much more. In this assignment, we will use Jupyter notebook to write the code and analyze our dataset effectively.

As part of the course *Data Quality and Data Wrangling*, I will practice analyzing a dataset, summarizing its main characteristics, and using visualization techniques to uncover patterns, relationships, and insights. For this purpose, I have chosen the regression dataset **"** california-housing-prices **"** from Kaggle, which explores factors influencing the house prices in california.

For this assignment, I will first examine the dataset to ensure it is clean and ready for analysis. Then, I will carry out the analysis and include visualizations to identify trends and explore the relationships between life expectancy and the various immunization, mortality, economic, and social factors included in the dataset.

1. **Data Analysis**
   1. **Understand the data**

To understand the dataset, it was loaded and its structure inspected using df.head(), df.dtypes, and df.info(). The dataset consists of 22 columns and a total of 20,640 rows. The data types include 9 float features and object-type features.

The dataset was checked for data formats, data types, and column names, all of which were found to be correct. The column names are self-explanatory, and the following are the features included in the dataset [1]:

* **Longitude**: a measure of how far west a house is; a higher value is farther west
* **latitude**: A measure of how far north a house is; a higher value is farther north
* **housing Median Age**: Median age of a house within a block; a lower number is a newer building
* **total Rooms**: Total number of rooms within a block
* **total Bedrooms**: Total number of bedrooms within a block
* **population**: Total number of people residing within a block
* **households**: Total number of households, a group of people residing within a home unit, for a block
* **median Income**: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
* **median House Value**: Median house value for households within a block (measured in US Dollars)
* **ocean Proximity**: Location of the house with respect to ocean/sea.
  1. **Location and variability**

For the numeric features descriptive statistics can easily be obtained with the following code df.describe(). With this, we have the descriptive statistics that summarize the central tendency, dispersion of a dataset distribution, excluding NaN values. For our data, the result are summarized in the picture below.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Table 1: descriptive statistics of the dataset*

The table give us information about the count of values, the mean, variance, min and max values and the percentiles of your data: 25%, 50%, 75% by default.

Comment?

* 1. **Remove duplicates**

The dataset has been checked in duplicates with the following code:

***Code***

*print(df.drop\_duplicates())*

And no row has been dropped from the data, so there were no duplicates in it.

* 1. **Handle Missing Data**

I used the codes *print(df.dropna()), print(df.dropna(how='all')) and print(df.dropna(thresh=2))* to investigate missing values in the data frame. There were 207 missing values in the columns "total\_bedrooms". I decided to replace these missing values by the mean using the code *df1["total\_bedrooms"] = df["total\_bedrooms"].fillna(df["total\_bedrooms"].mean())* in a copy df1 of the original dataset df.

* 1. **Outliers and bad data detections**

Knowing how to identify and handle outliers is an important part of data cleaning phase. There are various methods available for this task. For the visual inspection I decided to use box plots and histograms, which provide a comprehensive understanding of data distribution and ensure robust outliers detection.

The box plot with the following code:

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*plt.figure(figsize=(15, 10))*

*for i, col in enumerate(df.columns):*

*plt.subplot(5, 5, i + 1)*

*sns.boxplot(y=df1[col], data = df1)*

*plt.title(col)*

*plt.tight\_layout()*

*plt.show()*

*A diagram of a bar graph

AI-generated content may be incorrect.*

***Figure 1:*** *boxplots*

These plots identified the features containing outliers. To quantify them, the code below was used. First, I wrote the code to calculate the first (Q1) and third (Q3) quartiles. The interquartile range (IQR) is then determined as the difference between Q3 and Q1 (IQR = Q3 - Q1). Next, I calculated the upper and lower fences, and all values outside these fences are considered outliers.

*#column to analyse for outlier*

*column\_to\_analyse = "total\_rooms"*

*#column\_to\_analyse = df["total rooms"]*

*#calculate Q1 and Q3*

*Q1 = df[column\_to\_analyse].quantile(0.25)*

*Q2 = df[column\_to\_analyse].quantile(0.5)*

*Q3 = df[column\_to\_analyse].quantile(0.75)*

*# Calculation of the interquartile range*

*IQR = Q3 - Q1*

*#calculate the upper and lower bound for outliers*

*lower\_bound = Q1 - 1.5\*IQR*

*upper\_bound = Q3 + 1.5\*IQR*

*#detect outliers*

*outliers=df[(df[column\_to\_analyse]<lower\_bound)|(df[column\_to\_analyse]>upper\_bound)]*

*#print the number of outliers*

*print(outliers.shape[0])*

The following is the list of columns with outliers and the corresponding number of outlier values:

* total\_rooms: 1287 outliers
* total\_bedrooms: 1306 outliers
* population: 1196 outliers
* households: 1220 outliers
* median\_income: 681outliers
* median\_house\_value: 1071 outliers

This results in a total of 6761 outliers in the dataset. Since removing all rows with outliers would lead to significant data loss, I will investigate each column individually. To handle the outliers, I will apply the winsorisation technique using the code below. This approach replaces outlier values with the nearest acceptable limits (lower and upper bounds).

*#winsorization as outliers treatment on the "total\_bedrooms" column*

*print(lower\_bound, upper\_bound)*

*df1["total\_bedrooms"] = df["total\_bedrooms"].clip(lower=lower\_bound, upper=upper\_bound)*

After the transformation, the outliers have been calculated again, to make sure that the transformation was effektiv.

*#control of the number of outliers after treatment*

*outliers=df1[(df1["total\_bedrooms"]<lower\_bound)|(df1["total\_bedrooms"]>upper\_bound)]*

*print(outliers.shape[0])*

Also a visual inspection of the data after the transformation of the first columns "total\_rooms” has been done by plotting a histogram and a boxplot, the no outlier left in the column.

*#visual inspection of the data after treatment of the column "total\_rooms" with outliers to see if the outliers have been removed*

*fig, axes = plt.subplots(1, 2)*

*sns.histplot(x = "total\_rooms", data = df1, bins=40, ax=axes[0])*

*axes[0].set\_title("Histogramm of total\_rooms")*

*sns.boxplot(x = "total\_rooms", data = df1, ax=axes[1])*

*axes[1].set\_title("Boxplot of total\_rooms")*

*plt.show()*

*A comparison of a graph

AI-generated content may be incorrect.*

***Figure 2****: Histogram and boxplot of the column "total\_rooms" after the winsorisation*

* 1. **Save the cleaned data**

**The cleaned data has been saved**

1. **Data visualization**
   1. **Histogram**

Histogram illustrates the distribution and frequency of values, showing how often certain values occur. This allows characteristics of the distribution to be identified. Patterns may also become apparent—for example, a particular value might occur especially often, or there may be recurring peaks. However, it is important to ensure that the number of bars (bins) in the histogram is appropriately adjusted. It must correspond to the data; otherwise, important information may be lost

* 1. **Correlation between the features using heat map**

For the visualizations of correlation between the features, I used seaborn heatmap.

*selected\_cols = df1[["longitude","latitude","housing\_median\_age","total\_rooms","total\_bedrooms","population","households","median\_income","median\_house\_value"]]*

*# Calculate the correlation matrix*

*corr\_matrix = selected\_cols.corr()*

*# Plot the heatmap*

*plt.figure(figsize=(8, 6))*

*sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm", fmt=".2f", square=True)*

*plt.title("Correlation Heatmap of Housing Data")*

*plt.show()*

The heatmap shows a strong correlation between the features total rooms, total bedrooms, population, and households. This is logical, as more rooms generally mean more bedrooms, which in turn implies that more people are likely to live in a housing unit (household). Consequently, this leads to a higher population within the block.

* 1. **Scatterplots**

*Scatterplots*

*As shown below, in scatter-plots variables can not only be assigned to the x- and y-coordinate, but also to the color and the size of the markers. Moreover, it is possible to include a regression-line.*

*Code:*

*import plotly.express as px*

*df1=df.copy()*

*fig = px.scatter(df1, x="population", y="median\_income", color="median\_house\_value", size="total\_rooms", hover\_data=["ocean\_proximity"], title="Population over median income", trendline="ols")*

*fig.show()*

* 1. **Mapping of the median\_house\_value with the location**

*import plotly.express as px*

*# Create a scatter mapbox plot*

*fig = px.scatter\_mapbox(df1, lat="latitude", lon="longitude", color="median\_house\_value", size="median\_house\_value", size\_max=10,*

*zoom=5, mapbox\_style="open-street-map", hover\_data=["median\_house\_value"])*

*fig.show()*

**A map of the coast

AI-generated content may be incorrect.**

**Conclusion**

**References**

<https://www.kaggle.com/datasets/camnugent/california-housing-prices>

Ben Bolker. Exploratory data analysis and graphics. Princeton University Press, 2007

Francis John Anscombe. „Graphs in statistical analysis“. The American Statistician 27.1 (1973), S. 17–21

<https://maucher.pages.mi.hdm-stuttgart.de/python4datascience/09MachineLearningInaNutshell.html>

https://seaborn.pydata.org/generated/seaborn.heatmap.html