**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | *Statistics for Data Analytics*  *Data Preparation & Visualisation*  *Programming for Data Analytics*  *Machine Learning for Data Analysis* |
| **Assessment Title:** | *MSc\_Data\_Analytics\_CA1* |
| **Lecturer Name:** | *John O’Sullivan*  *David McQuaid*  *Sam Weiss*  *Muhammad Iqbal* |
| **Student Full Name:** | *Mahmut Zahid Malkoc* |
| **Student Number:** | *2023034* |
| **Assessment Due Date:** | *14/04/2023* |
| **Date of Submission:** | *14/04/2023* |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

**Abstract:**

Due to the housing crisis in Ireland in the last two decades, such a study has been carried out on the excessive increase in rent and house prices. The main objective of this assignment aims to the estimation of house prices in Dublin that the capital city of Ireland. The main reason for choosing Dublin in this study is that many factors that took place in Dublin were proportionally close to Ireland. In this assignment, Ireland's population, number of immigrants, inflation rate, per capita income, unemployment rate, number of refugees granted asylum, and home construction cost index have been chosen as some of the most fundamental factors affecting new house prices in Dublin.

**Housing in Ireland: Dublin New House Price Estimation**

**Introduction:**

Overall, this assessment was carried out on the housing problems in Ireland, provided that at least one housing data was selected [data.gov.ie](https://data.gov.ie/), Ireland's open data portal. Two different data sets were selected from this website. The first of this dataset from 1975 to 2016 is the new house prices of Dublin that one of the 5 largest cities and capital of Ireland and national new house prices. Another data set, from 1994 to 2015, based on the year 1991 as 100, is the national house construction cost index and the rate of the national house construction cost index increased on the previous year.

In addition to these datasets, other datasets that will affect the new house price estimation are taken from the [macrotrends.net](https://www.macrotrends.net/) website. The Ireland and Dublin populations from the [macrotrends.net](https://www.macrotrends.net/) website provide us with data from 1950 to 2023. Similar to those datasets, the urban population of Ireland from 1960 to 2021 was obtained from the same website. Another dataset from 1960 to 2015 in a year interval close to these datasets illustrates the migrant population of Ireland.

The inflation rate in Ireland by [data.worldbank.org](https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG?locations=IE) and the rate of change in the annual inflation rate, which are two different datasets from 1960 to 2021, were obtained as a percentage.

Moreover, these datasets on Gross Domestic Product (GDP) by [data.worldbank.org](https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?locations=IE) and Gross Domestic Product per Capita (GDP per Capita) by [data.worldbank.org](https://data.worldbank.org/indicator/NY.GDP.PCAP.CD?locations=IE) in Ireland from 1960 to 2021 were collected from The World Bank website. Gross Domestic Product (GDP) is provided in billions of dollars, while Gross Domestic Product per Capita (GDP per Capita) is defined in dollars.

The annual growth rate of the Gross Domestic Product (GDP) in Ireland from 1971 to 2021 is given by [data.worldbank.org](https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG?locations=IE). The ratio of the total labor force to unemployment and also an annual change in the unemployment rate from 1971 to 2021 was obtained by [data.worldbank.org](https://data.worldbank.org/indicator/SL.UEM.TOTL.ZS).

From 1984 to 2021, Refugees granted asylum, and the annual change rate of refugees granted asylum were obtained by [macrotrends.net](https://www.macrotrends.net/countries/IRL/ireland/refugee-statistics).

**Statistics, Exploratory Data Analysis, Preparation and Visualization:**

As we know, exploratory data analysis (EDA) is used to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

First of All, in order to analyze the data, some data-cleaning processes must be applied. And also, the data set should be checked for missing data. Moreover, columns that are not required for analysis should be removed. Finally, the data should be made sense by applying data discovery analysis and possible outliers or unusual values should be checked.

The following steps can be followed to clear the data:

* Checking for missing data
* Removing unnecessary columns
* Checking for outliers and unusual values

After the data is cleaned, some data discovery analysis should be applied to understand the data and check for patterns or trends. Visualizations should be created to support the analysis. In the final stage, the outputs should be summarized and some conclusions should be drawn based on the data.

**Data Preparation and Programming:**

The excel file imported into this assessment was brought together as a preliminary inquiry. However, while selecting this scenario, examining hundreds of resources during the research and investigation phase, a valuable dataset has been obtained. This dataset has taken its first form by using the advantages of this situation. This dataset is called OtherDatasets.

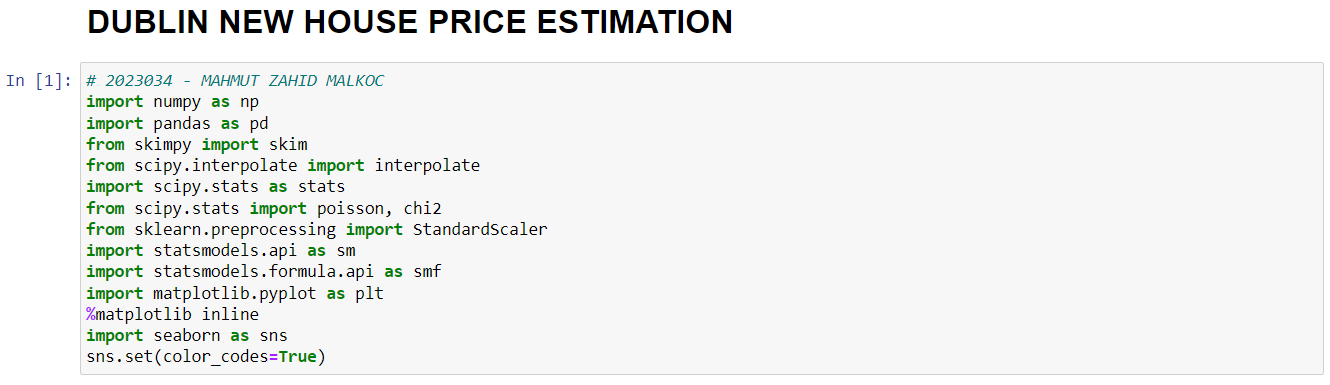
This dataset contains 13 different attributes on a yearly basis. In fact, it has 14 different attributes in total, including the year attribute. These attributes are explained group by group.

The first group is the population group and that is explained each number is one person. The population numbers are included as Ireland's population (IEPopulation), Dublin's population (DublinPopulation), Ireland's urban population (IEUrbanPopulation), Ireland's migrant population (IEMigrantPopulation), and Ireland refugees granted asylum population (IERefugeesGrantedAsylum).

The second group is the percentage group. It is included in this group Ireland's inflation rate (IEInflationRate), Ireland's annual rate of change in inflation (IEInflationAnnualChangeRate), Ireland's annual growth rate according to Gross Domestic Product (IEGDPGrowthRate), Ireland's unemployment rate (IEUnemploymentRate), Ireland's annual change of unemployment rate (IEUnemploymentRateAnnualChangeRate), Ireland's annual change of refugees granted asylum population (IERefugeesGrantedAsylumAnnualChangeRate).

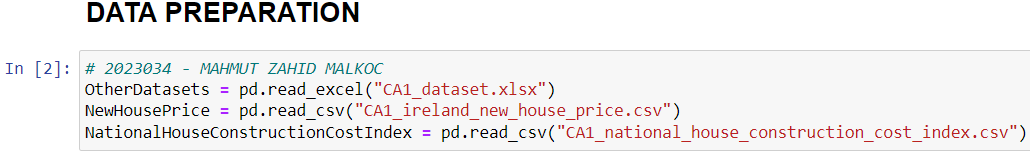
The third group is the gross domestic product group. In this group, Ireland's Gross Domestic Product in Billion Dollars (IEGDP$B), Ireland's Gross Domestic Product per capita in Dollars (IEGDPPerCapita$). Therefore, this dataset in three fundamental groups was gathered from different sources.

The scenario is associated with these collected datasets. On the other hand, the attribute to be taken as the basis is determined. This attribute is taken into account in this assessment as its main objective is to estimate new home prices in Dublin. In addition, the dimensions of the Dublin new home prices attribute should be looked at. As you can see, there are new house prices in Dublin from 1975 to 2016. Considering this situation, all other datasets were arranged from 1975 to 2016 as a basis.



*Fig 1 - Code*

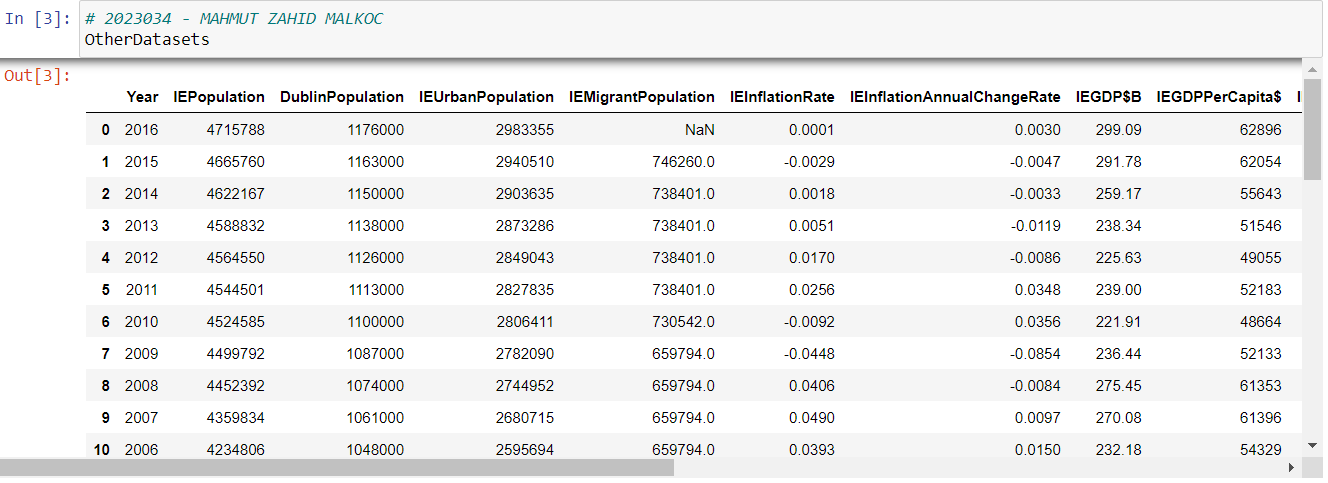
After opening the Jupyter notebook, a new notebook is created. The libraries needed in this assessment are imported into the first cell. At the same time, where and when these libraries are needed will be re-imported and explained.



*Fig 2 - Code*

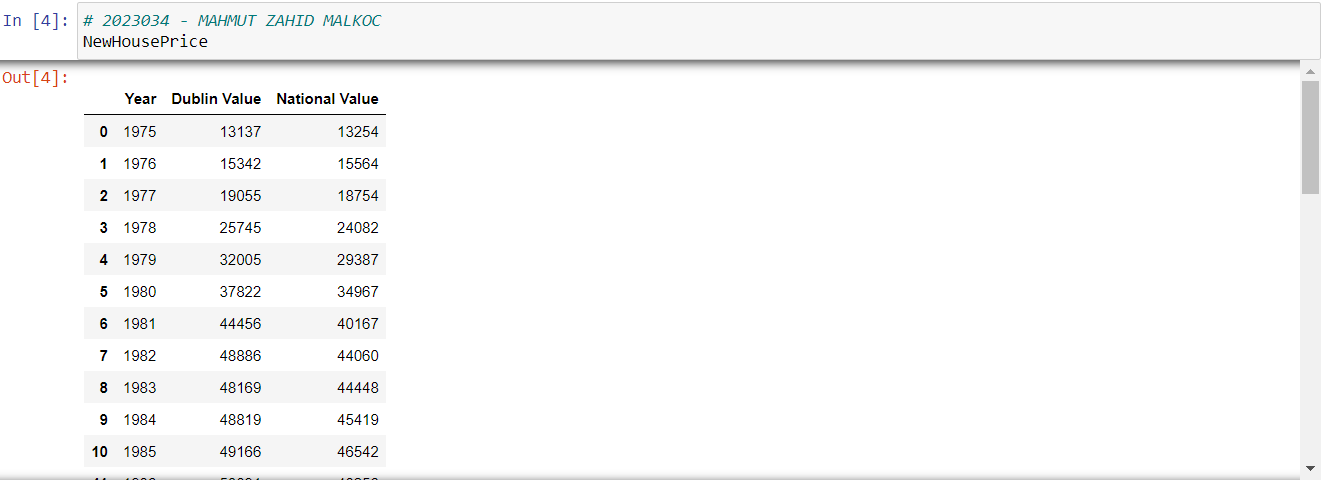
In this assessment, many datasets have been brought together using three different data sources. In order to make the assessment targeted here more efficient, some of the datasets were brought together in excel and called OtherDatasets.

Two other datasets from Ireland's open data source are named NewHousePrice and NationalHouseConstructionCostIndex. NewHousePrice, NationalHouseConstructionCostIndex datasets are imported in csv format, while Other Datasets are imported in excel that .xlsx format.



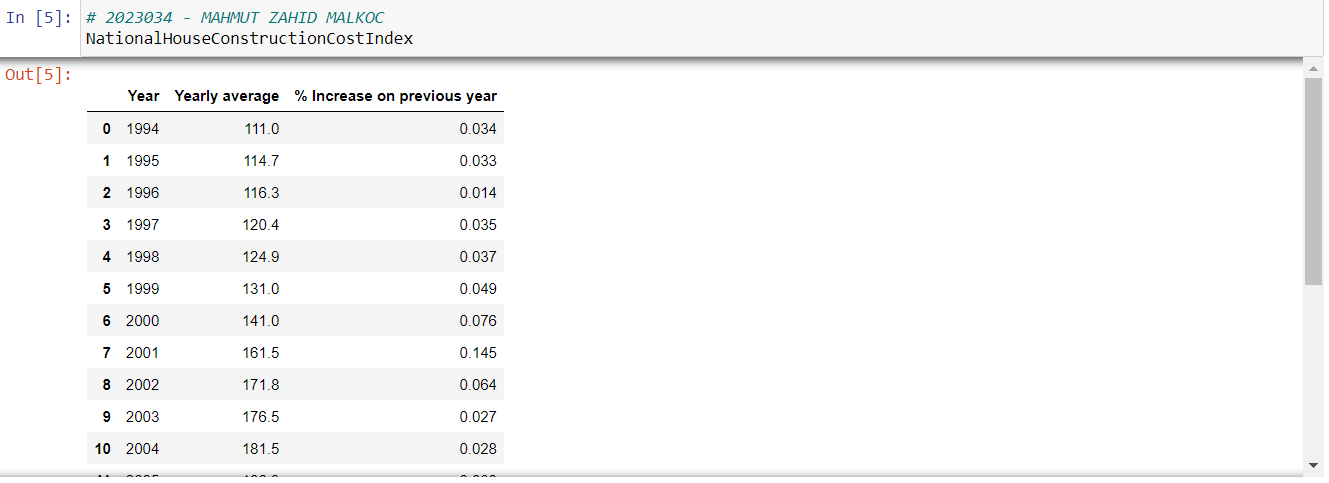
*Fig 3 - Code*

OtherDatasets is run and printed to the screen. As it seems, this dataset contains 14 different attributes from 1975 to 2016. These attributes were explained in detail above.



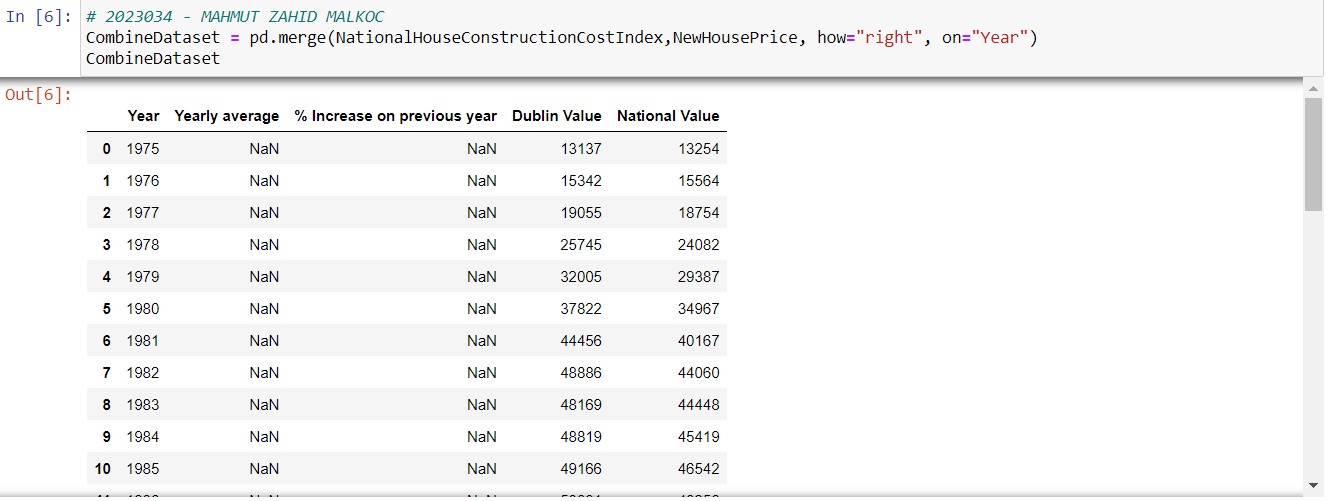
*Fig 4 - Code*

NewHousePrice is run and printed to the screen. As it seems, this dataset contains 3 different attributes from 1975 to 2016. The Dublin Value and National Value attributes are the new home prices of Dublin and Ireland in euros over the years.

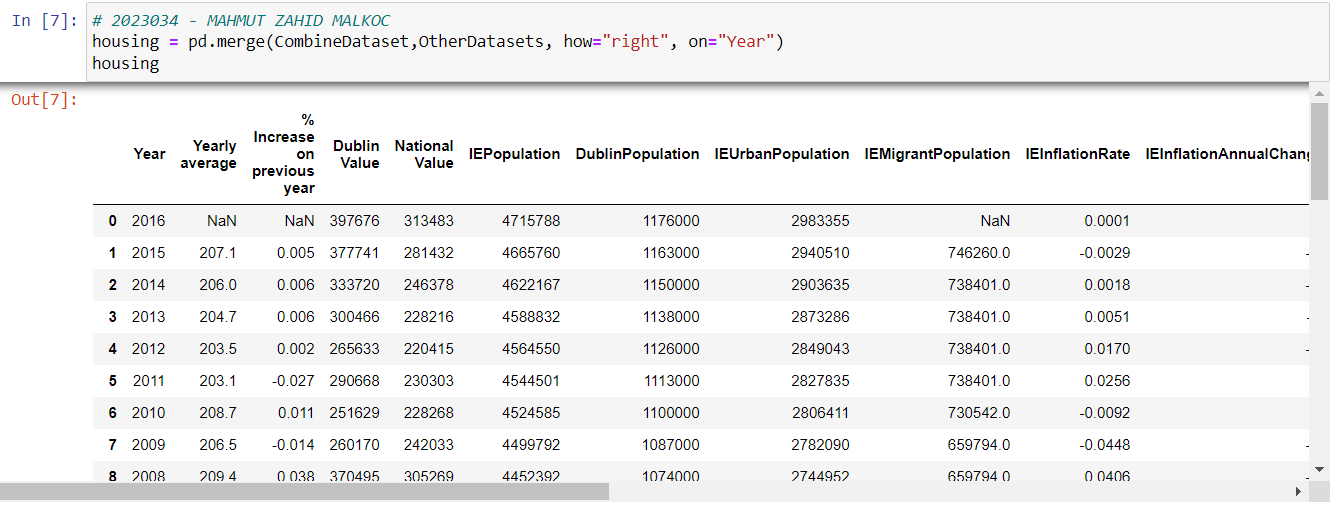


*Fig 5 - Code*

NationalHouseConstructionCostIndex is run and printed to the screen. As it seems, this dataset contains 3 different attributes from 1994 to 2015. The Yearly average and % Increase on previous year attributes represent Ireland's housing construction cost index. The Yearly average attribute is a data set based on the year 1991 as 100. The % Increase on previous year attribute is the percentage increase in the cost of housing construction compared to the previous year.

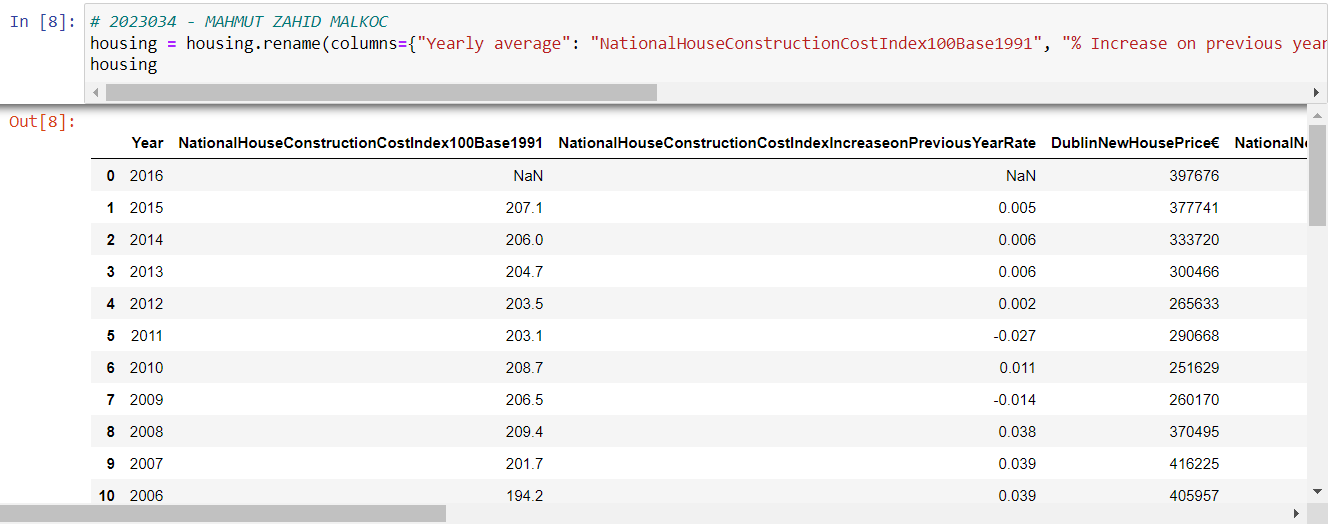


*Fig 6 - Code*

The NationalHouseConstructionCostIndex and NewHousePrice datasets have been merged with the year attribute fixed. 

*Fig 7 - Code*

CombineDataset and OtherDatasets datasets have been merged with the year attribute fixed.



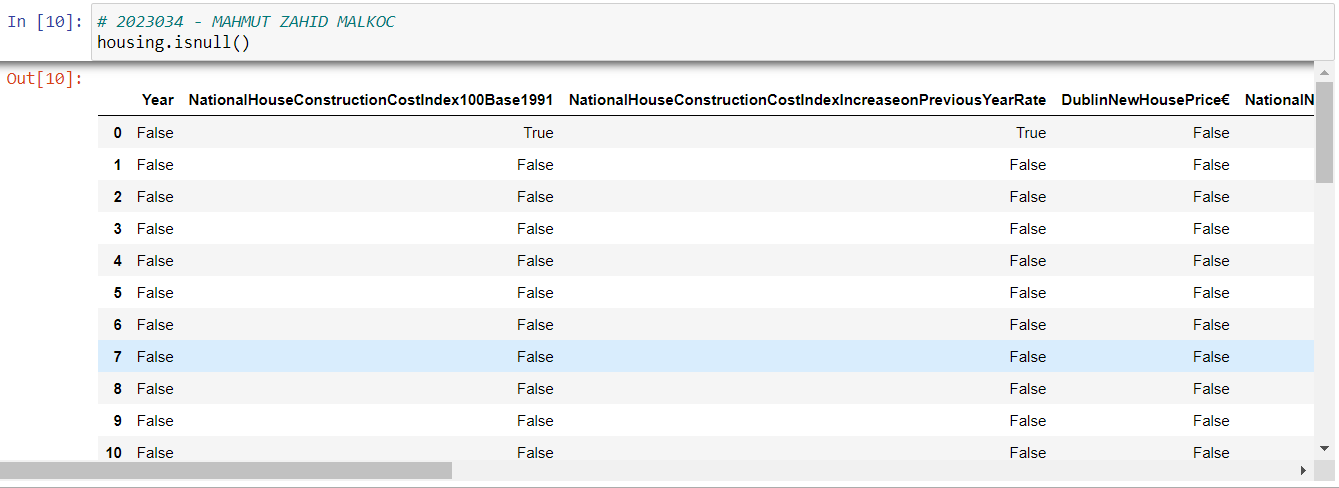
*Fig 8 - Code*

The attributes of these combined datasets have been given new names.



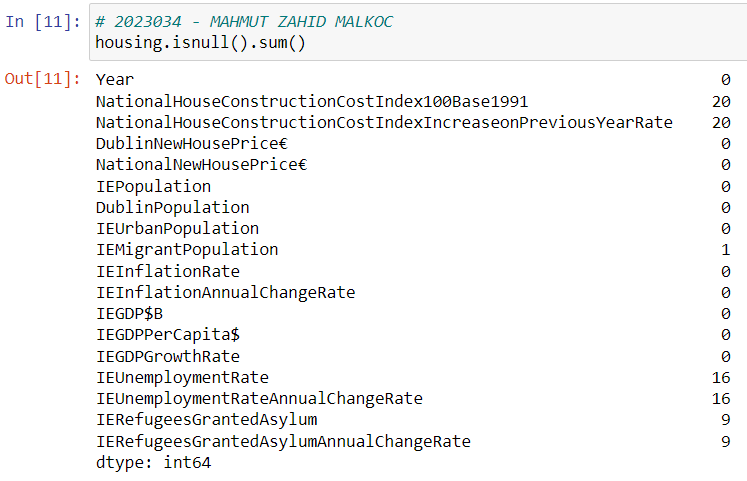
*Fig 9 - Code*

The number of data in all columns of the housing dataframe we created is learned. In this way, missing or empty values in the data set can be detected.



*Fig 10 - Code*

Here, it is checked whether each cell of the housing dataframe is missing or NaN (Not a Number) value.

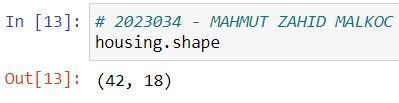
 *Fig 11 - Code*

Here, the total number of missing or NaN (Not a Number) values for each cell of the housing dataframe is displayed.



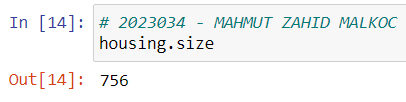
*Fig 12 - Code*

The size of the housing dataframe is queried.



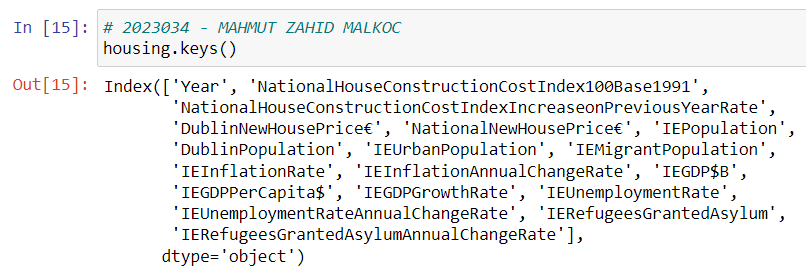
*Fig 13 - Code*

The size of the housing dataframe is queried in rows and columns.



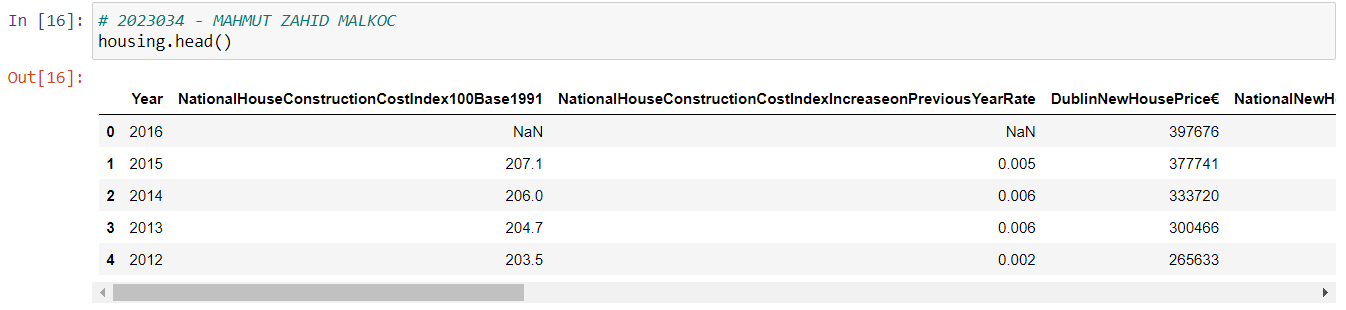
*Fig 14 - Code*

The total number of cells of the housing dataframe is queried.



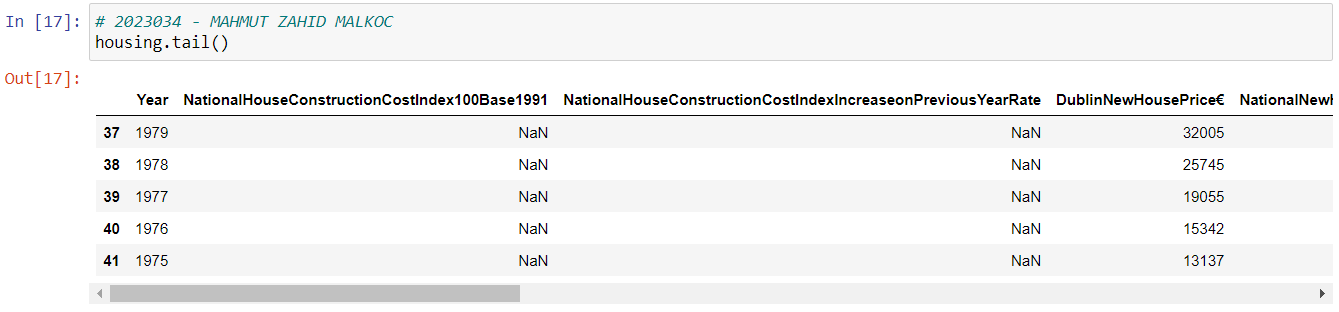
*Fig 15 - Code*

The column names of the housing dataframe are queried.



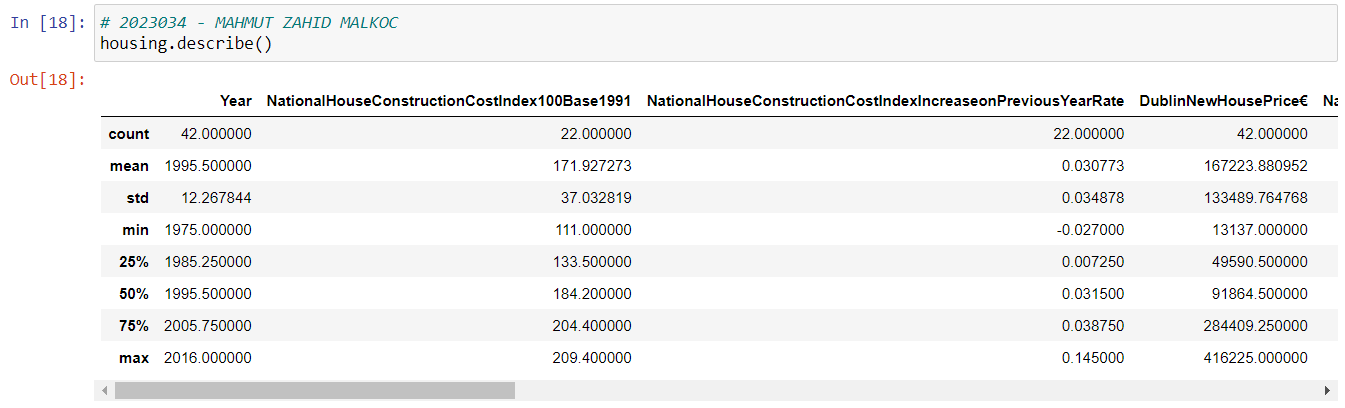
*Fig 16 - Code*

The first 5 rows of the housing dataframe are queried.



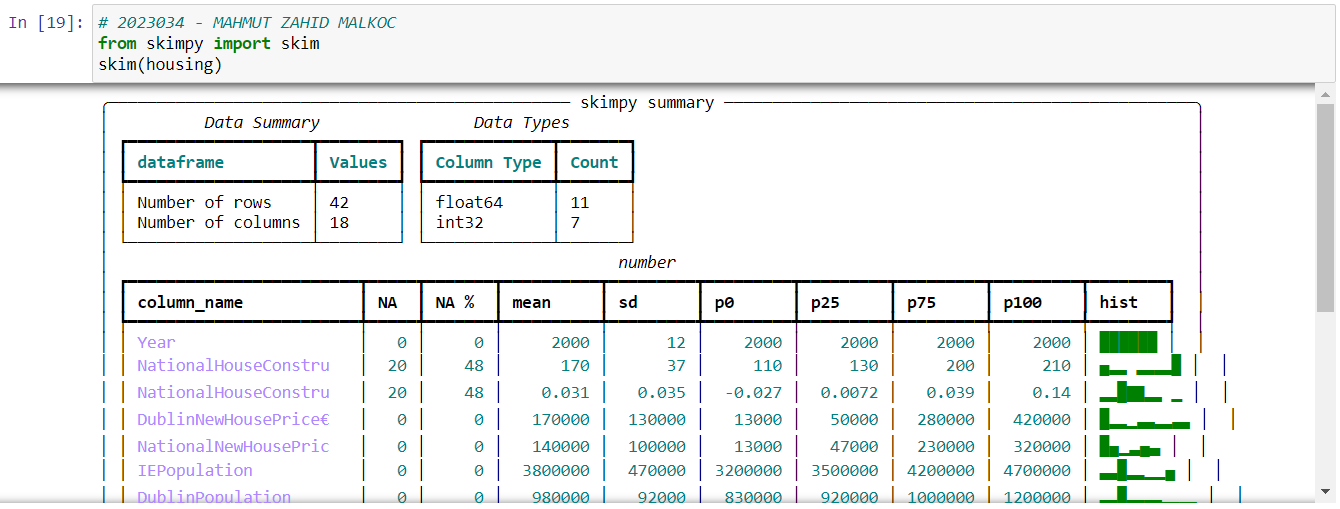
*Fig 17 - Code*

The last 5 rows of the housing dataframe are queried.



*Fig 18 - Code*

Fundamental statistics of numeric columns of housing dataframe are obtained. Basically, the values of count, mean, std, min, 25%, 50%, 75%, and max were obtained.



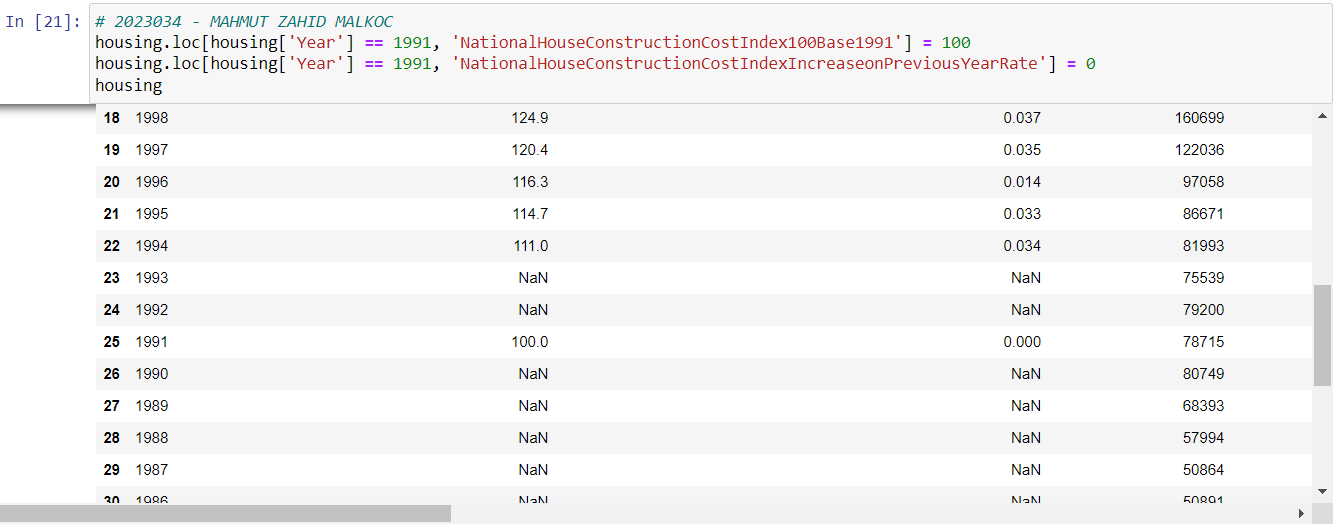
*Fig 19 - Code*

Used instead of .describe() method, skim() illustrates fundamental statistical values in a visualized way, as well as the number of columns and rows, the types and number of data, and the number and percentage of NaN values in the columns.



*Fig 20 - Code*

Each row of the housing dataframe is checked. This method checks if this value is in the housing dataframe. It renders True if present, False otherwise.



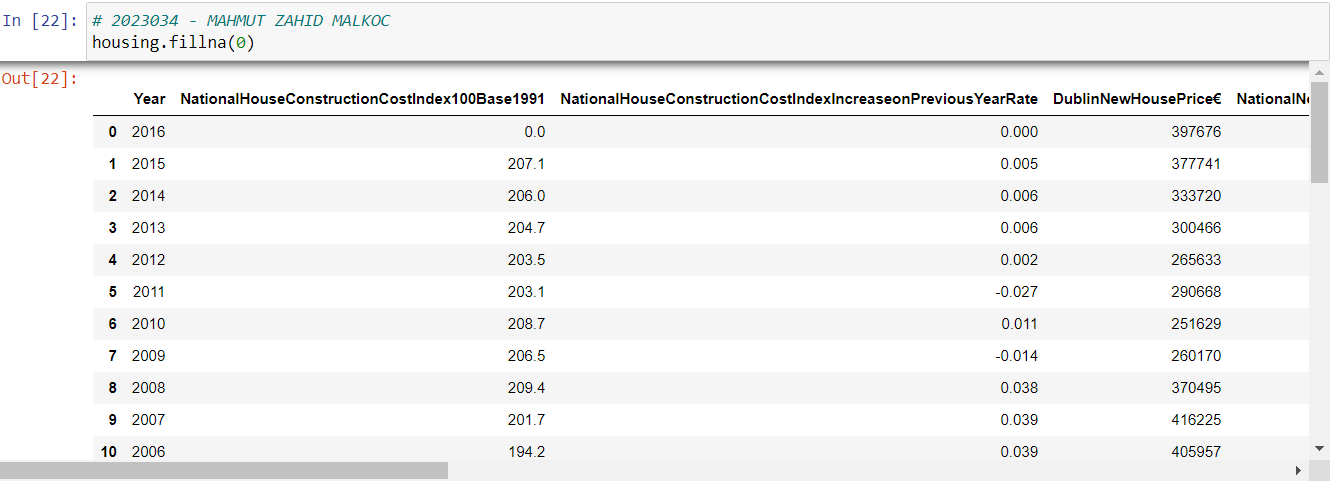
*Fig 21 - Code*

While selecting these datasets, according to the information in the abstract section, NationalConstructionCostIndex100Base1991 was defined based on 100 in 1991. In this case, NationalConstructionCostIndexIncreasePreviousYearRate is 0.

At this stage, it seems that the dataset has NaN values. Filling in these NaN values is important for the dataframe and future machine learning models. In this case, the methods to be used are as follows:

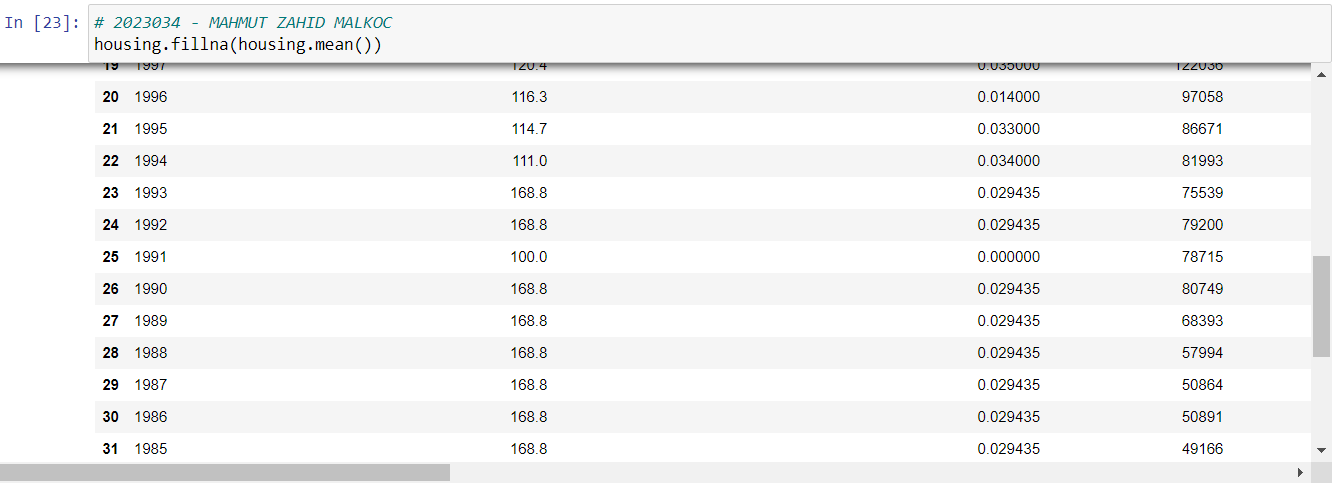
* Fill it with 0 Value
* Average Value
* Median Value
* Most Common Value
* Linear Interpolate Value
* Machine Learning Based Value

The properties of the dataset are taken into account to decide which method to use. It is necessary to determine which method is most suitable. In this case, methods that match the housing dataframe are tried.



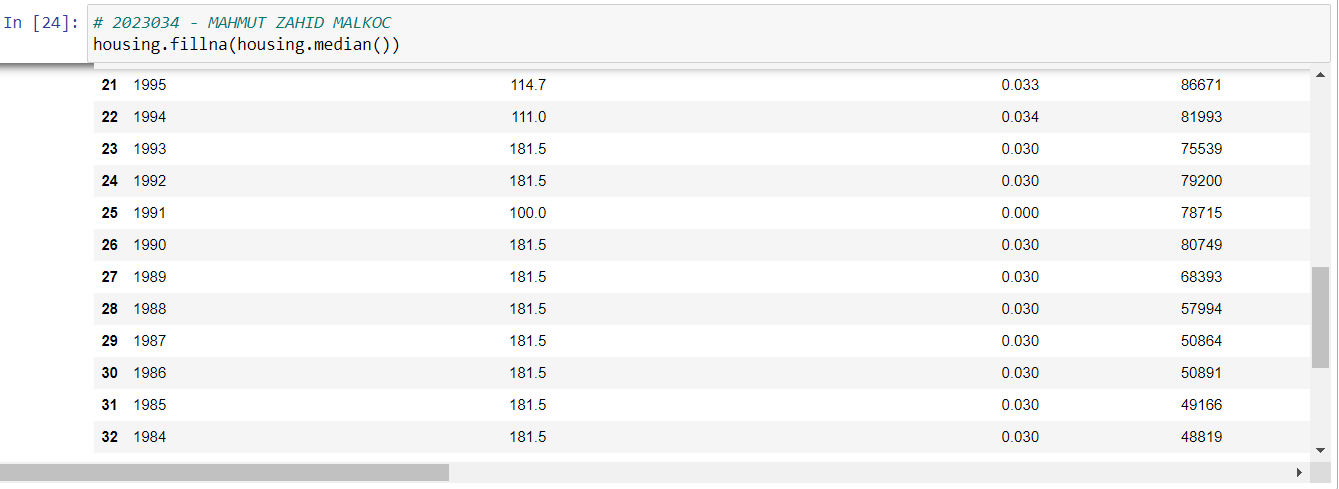
*Fig 22 - Code*

All values are filled with 0 using this method. However, All values filled with this method generally do not give such a good impression. Filling all values with 0 will reduce the accuracy of the machine learning model. Therefore, this method is not suitable in order to filling values.



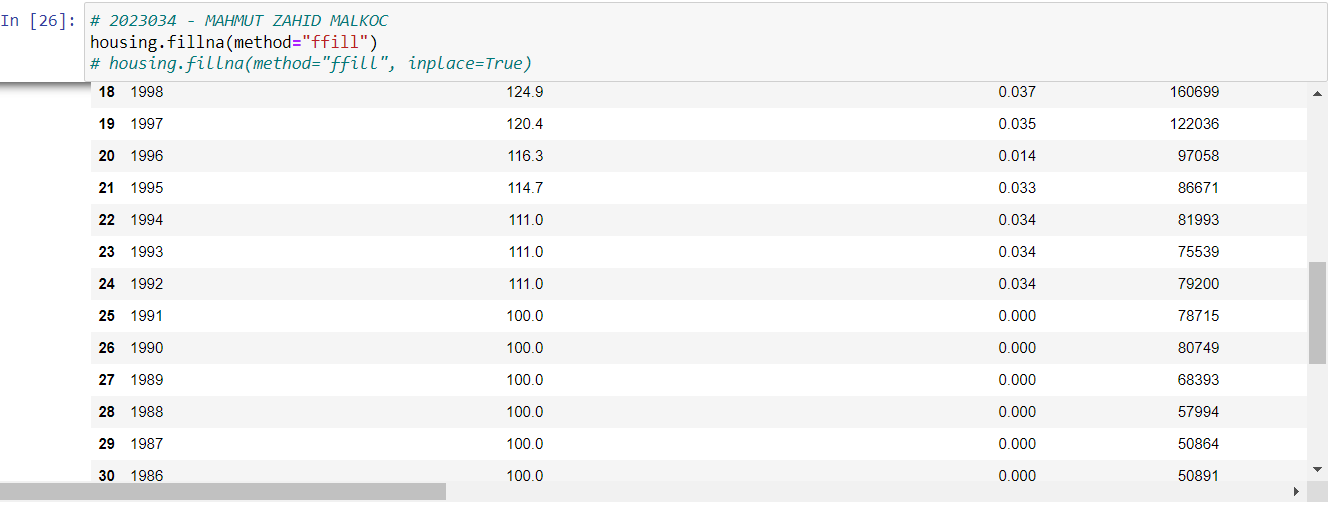
*Fig 23 - Code*

In the method of filling all values according to the mean of the column values, the value ranges based on 100 in 1991, that is, 1992 and 1993, were much higher than expected. As can be seen, 1992 and 1993 were filled 168.8. At the same time, it was filled same value from 1975 to 1990. Therefore, this method is not also suitable in order to filling values.

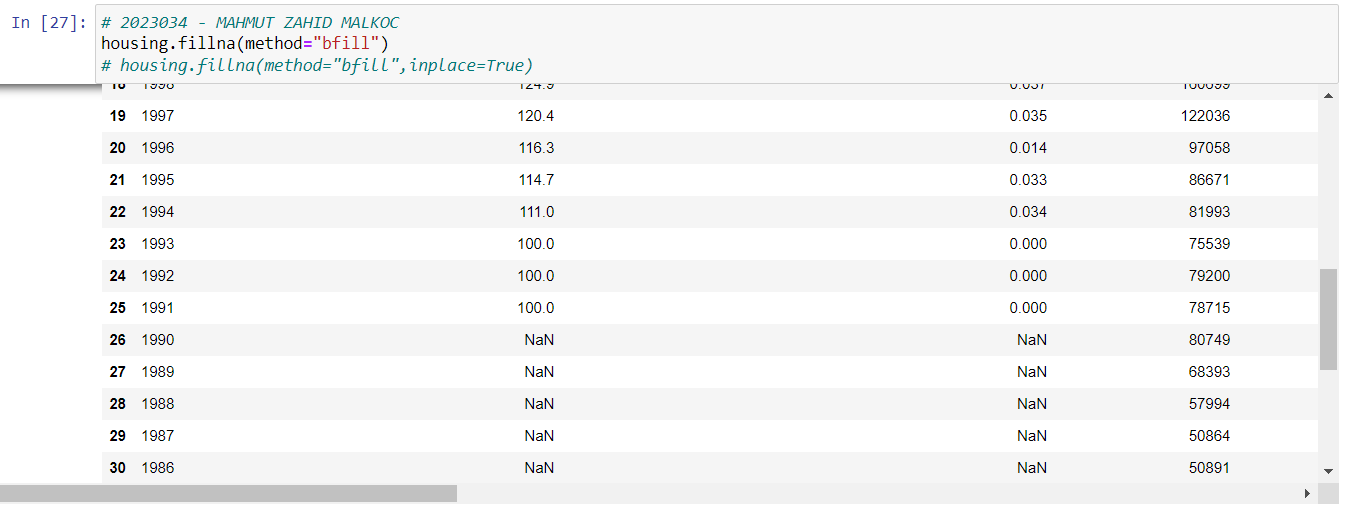


*Fig 24 - Code*

The filling method with the median value gave very similar results to the mean method. It even filled it with higher values than expected such as 181.5. This method is not also suitable.

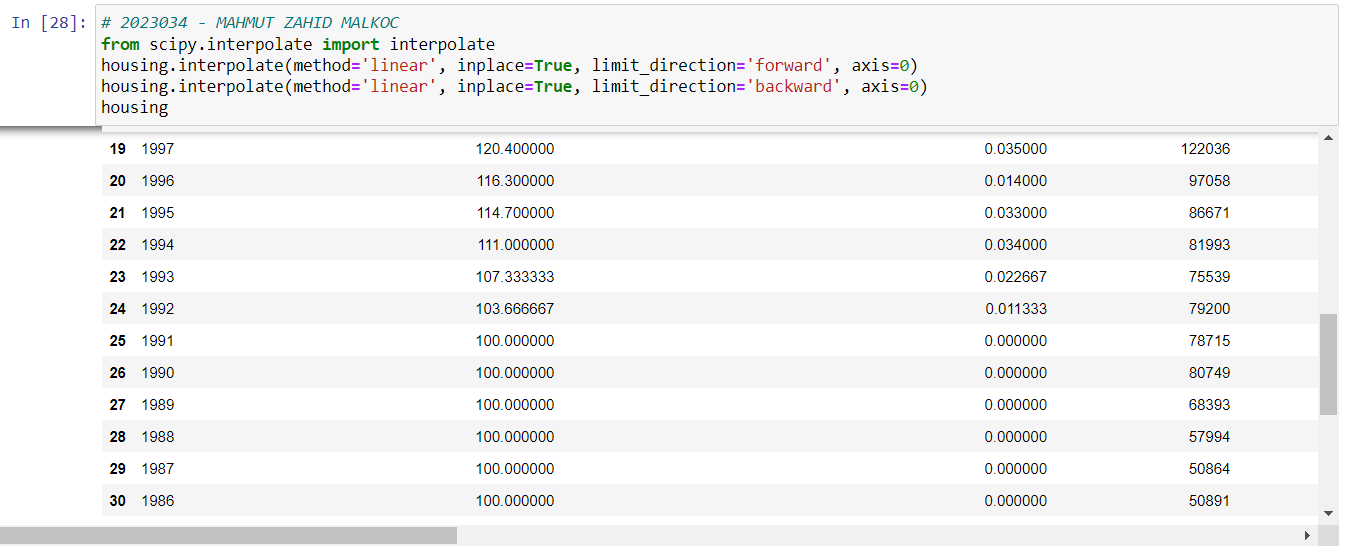


*Fig 25 - Code*



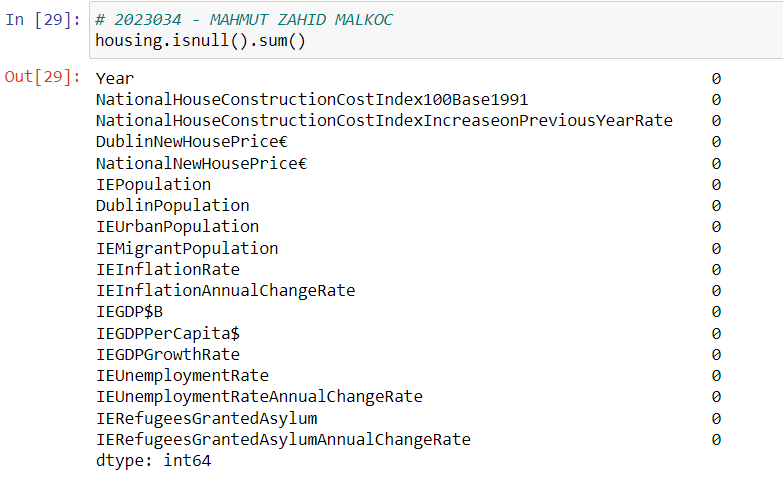
*Fig 26 - Code*

These two methods (ffill, bfill) are obtained better results than those applied so far. ffill completes forward values while bfill completes backward values based on values before NaN. Inplace=True must be added to the method for these two methods to complete the dataset permanently. However, since this value uses existing values according to forward and backward values, it will decrease the accuracy rate in the machine learning model. For instance, the values for 1992 and 1993 are completed to the previous or next value. A more logical method can be used for a dataset that shows the impression of a linear increase or decrease.



*Fig 27 - Code*

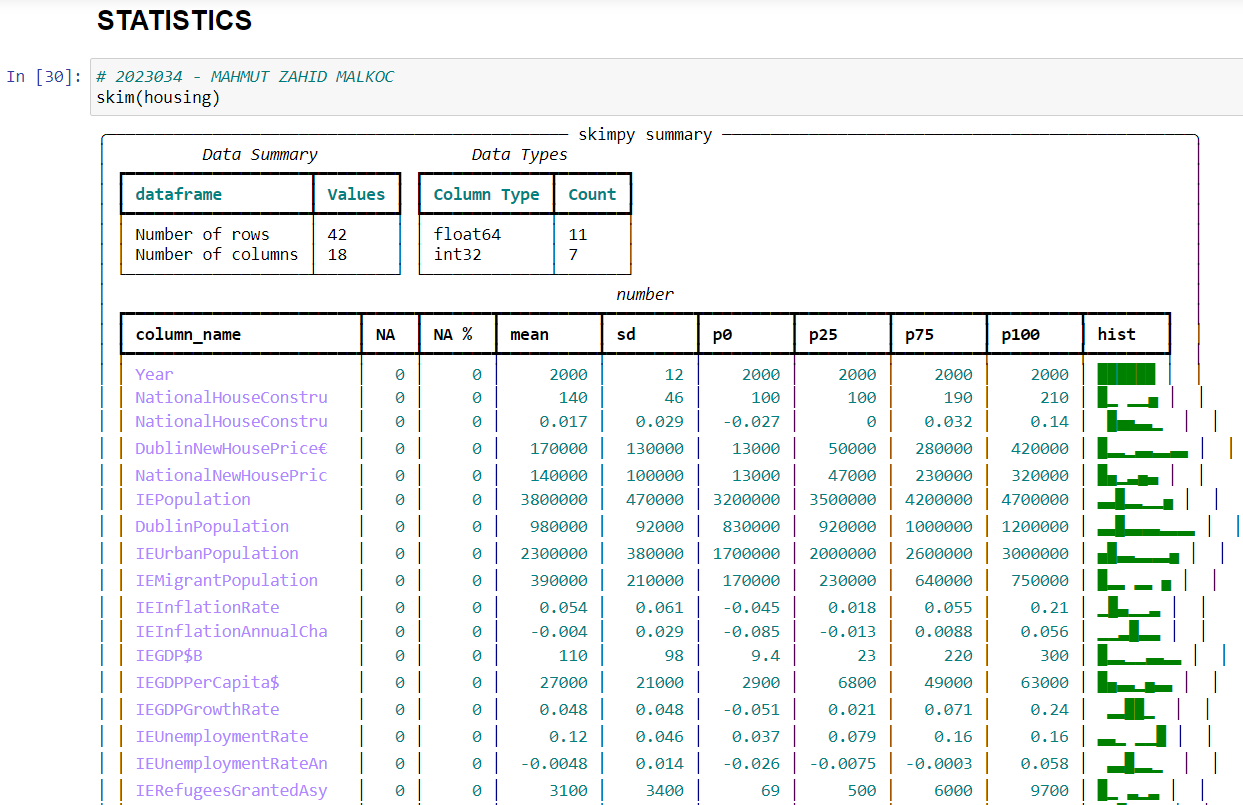
Linear interpolate, which completes with the most efficient and logical values among the methods, has been chosen as the most appropriate method for the housing dataset. The NationalConstructionCostIndex100Base1991 was completed with a linear increment of 103,666 in 1992 and 107,333 in 1993. At the same time, the NationalConstructionCostIndexIncreasePreviousYearRate value of 0.0113 in 1992 and 0.0226 in 1993 was completed according to the linear increment of this dataset. As a result of all the tried methods, the linear interpolation method was applied as the most logical result for the machine learning model.



*Fig 28 - Code*

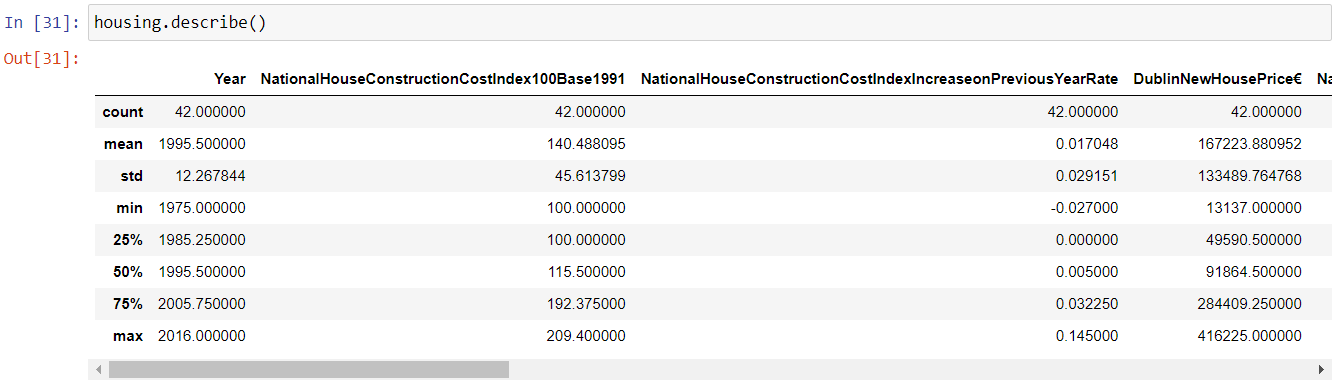
The above method is applied to see that there are no NaN values. As can be seen, there is no NaN value anymore. As a result, the Housing dataset is suitable to form for statistics, visualization and machine learning models when the entire dataset is checked for NaN value.

**Statistics and Programming:**



*Fig 29 - Code*

The above skim() method is applied to see the basic statistical results in the housing dataset. As it seems, this method gives by rounding to the nearest upper or lower value mean, standard deviation, percentages of 0, 25, 50, 75, 100, and percentage of NaN.



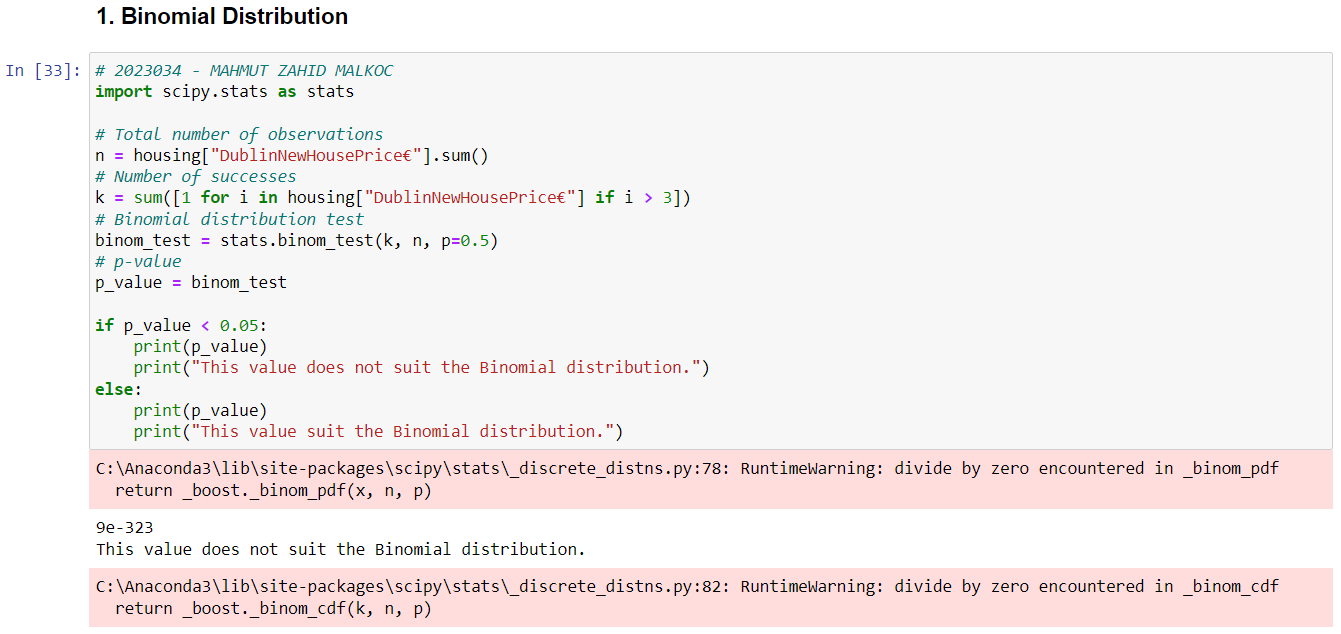
*Fig 30 - Code*

However, Calculations are carried out in considerable detail in statistical studies. Therefore, it is particularly desirable to see certain results by applying the describe() method here.



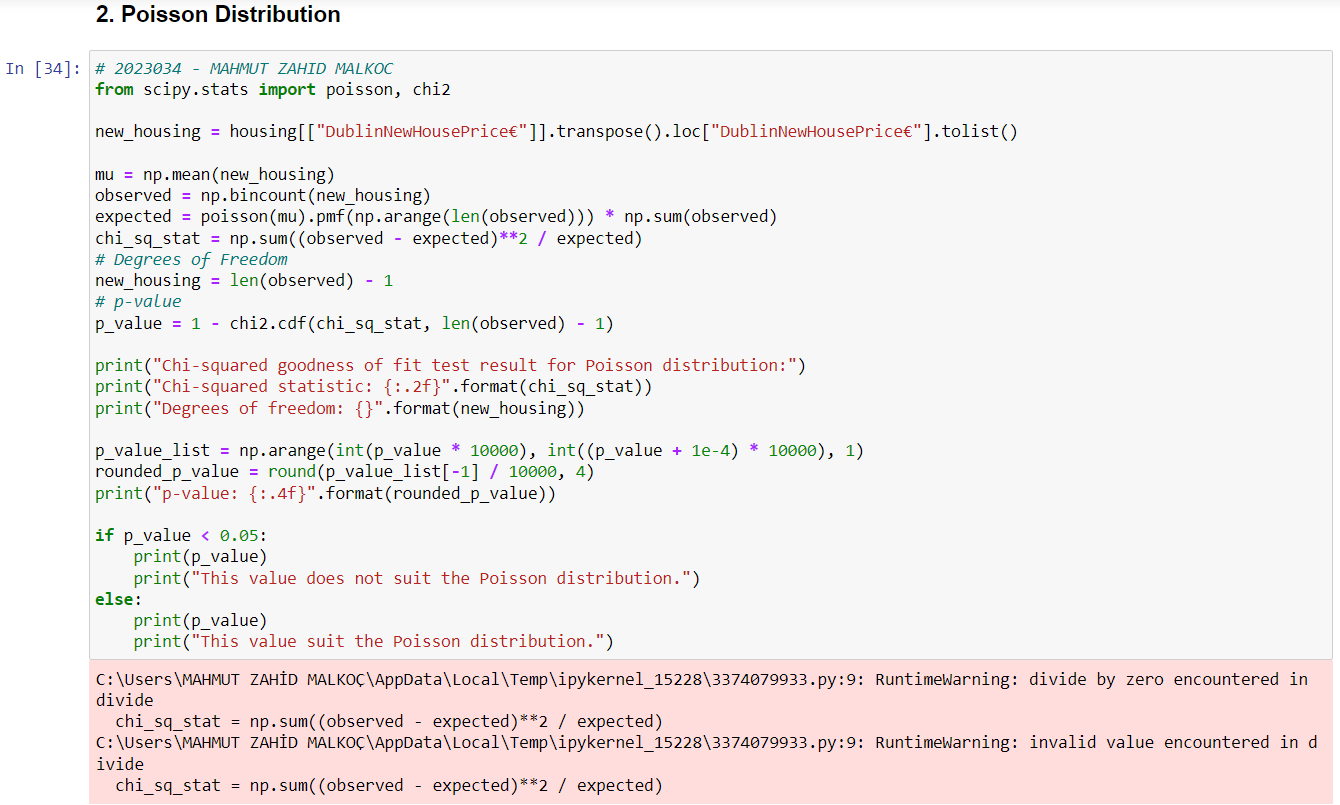
*Fig 31 - Code*

These results can be reached by hand coding to obtain results as above. For some of these, mean and standard deviation, have been calculated above. And also, as in the skim() method, rounding to the nearest up or down value was performed. The mean and standard deviations of the DublinNewHousePrice€ and DublinPopulation attributes were calculated and additionally rounded to the nearest values. When these outputs are compared with the outputs of the skim() and describe() methods, it will be seen that they are the same results.



*Fig 32 - Code*

Through the code seen above, it has been tested whether the binomial distribution is suitable for the housing dataset. It has been observed that the binomial distribution is not an appropriate distribution for this dataset, with the p-value of the binomial distribution quite below the expected result.



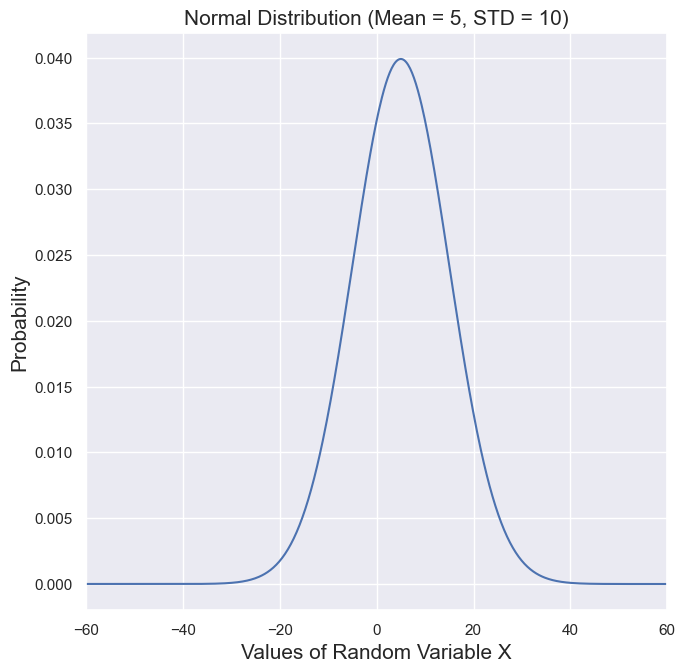
*Fig 33 - Code*

The Poisson distribution is a probability distribution used when a case is unusual. The distribution of a continuous variable such as house prices is not compatible with the Poisson distribution. As can be seen from the above Poisson distribution test code, this distribution type is also not suitable for the housing dataset.



*Fig 34 - Code*

Normal distribution is the default probability for real-world scenarios. It represents a symmetric distribution where most of the observations cluster around the central peak called as mean of the distribution. A normal distribution can be thought of as a bell curve or Gaussian distribution which typically has two parameters: mean and standard deviation (SD). The parameter used to measure the variability of observations around the mean is called standard deviation. The probabilities for values occurring near the mean are higher than the values far away from the mean. The parameters of the normal distribution plot defining the shape and the probabilities are mean and standard deviation. The area of the plot between two different points in the normal distribution plot represents the probability of the value occurring between those two points.



*Fig 35 - Graph*

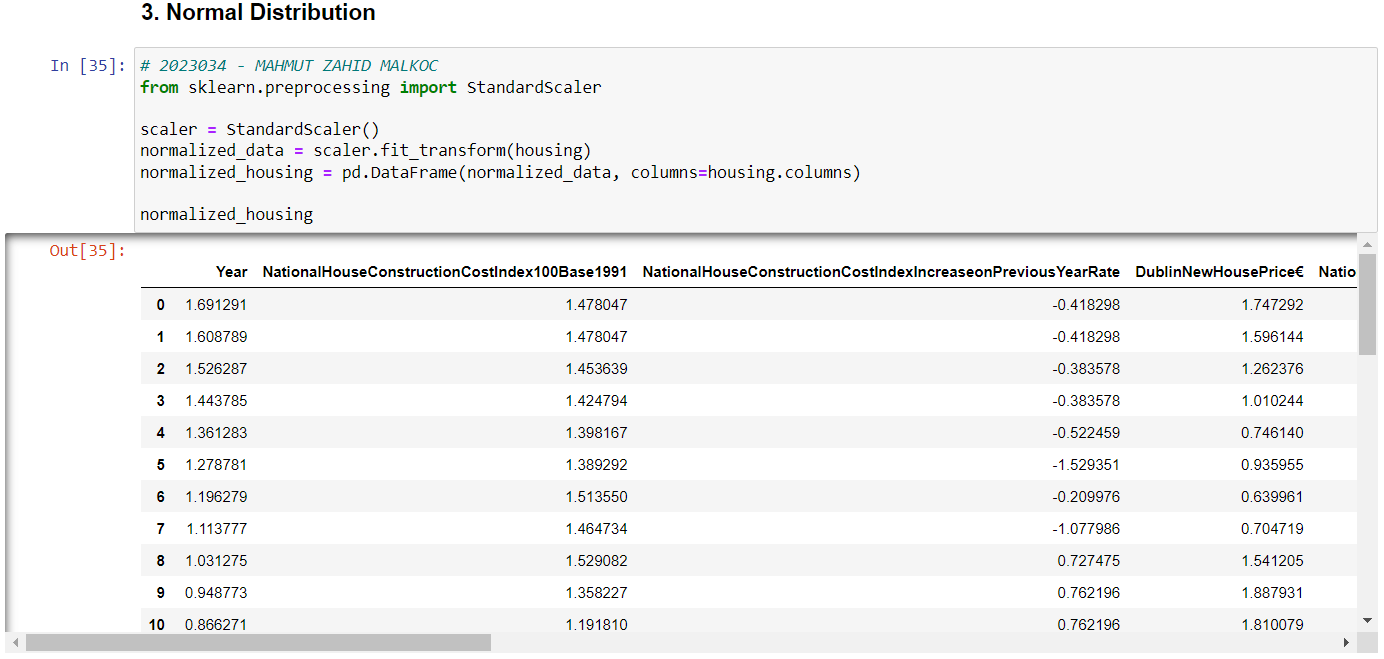
Many machine learning algorithms assume normal distribution in the data. If you have normally distributed inputs, use a normal probability function to calculate the probability of their occurrence. If your inputs are not normally distributed, transform them by applying log or square root transformations until they become normally distributed before feeding them into an algorithm that assumes normal distribution (such as linear regression).

The normal distribution of an obtained data set means that the data is distributed in a normal distribution. The normal distribution has an important place in probability theory and is frequently used in statistical analysis.

The main properties of the normal distribution are:

* Mean: The mean of the normal distribution is 0.
* Standard deviation: In a normal distribution, the standard deviation indicates how spread out the data points around the mean.
* A bell-shaped distribution: In a normal distribution, the majority of data points are concentrated in a region close to the mean, and the distribution appears to widen as the standard deviation increases. The graph of the normal distribution therefore resembles a bell shape.
* Symmetry: The normal distribution is a symmetrical distribution. That is, the left and right sides of the graph are the same around the mean.
* Probability theory: The normal distribution occupies an important place in probability theory. Many random events can be modeled according to a normal distribution.
* Central Limit Theorem: Normal distribution is also used in statistical theorems such as the central limit theorem. This theorem states that the sum or mean of many random variables converges to the normal distribution.

The normal distribution is an important form of distribution used in many statistical analysis and modeling operations. Data sets that do not fit the normal distribution can be subjected to some transformation to fit the normal distribution.

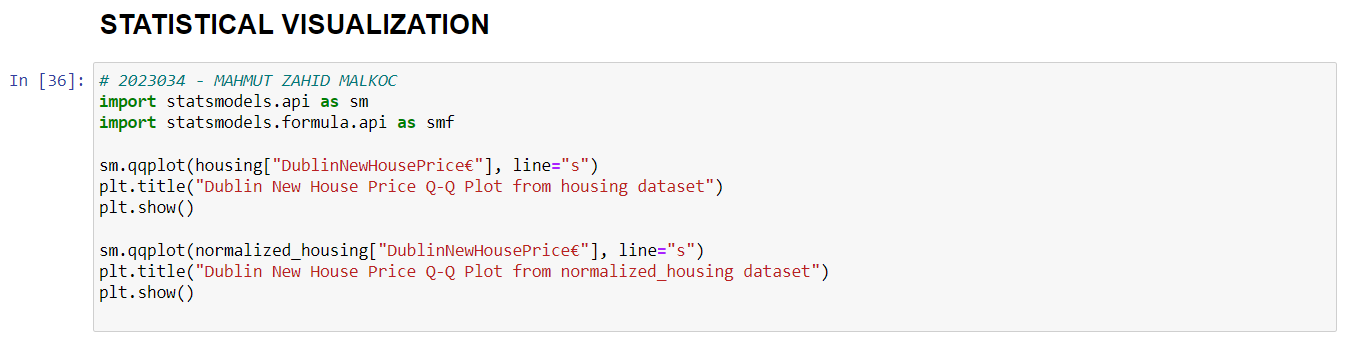


*Fig 36 - Code*

This code performs data pre-processing on a dataset, performing normalization and data visualization.

First, the dataset is loaded via the pandas library. Then, StandardScaler from Scikit-learn library is used to standardize on specific columns (DublinNewHousePrice€, NationalNewHousePrice€, IEPopulation) in the dataset. Normalization is done by setting the mean of each column to 0 and its standard deviation to 1. This ensures that each column is correctly compared to the other columns.

Next, histograms are plotted for the DublinNewHousePrice€, NationalNewHousePrice€ and IEPopulation columns of the dataset. Histograms visualize the distribution of columns. The bins parameter determines how many classes will be in the histograms. The xlabel and ylabel parameters and the x and y axes of the histograms are named. This dataset is called normalized\_housing because the housing dataset is normalized.



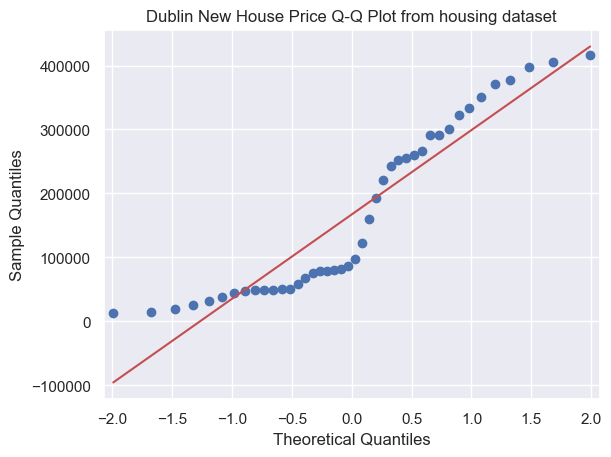
*Fig 37 - Code*

It uses the Q-Q (Quantile-Quantile) chart to check the normal distribution of the columns in a dataset.

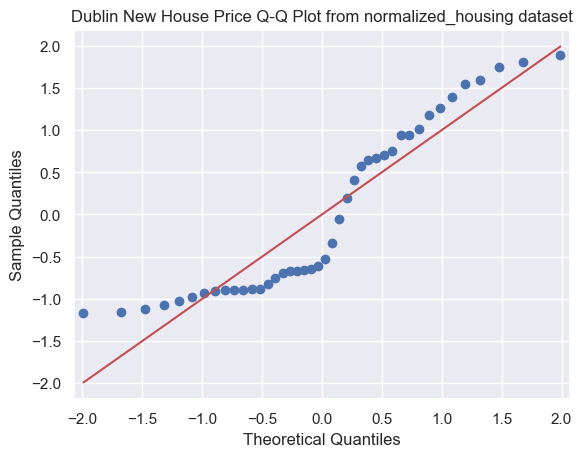
First, the Q-Q graph of the "DublinNewHousePrice€" column in the housing dataset is drawn using the qqplot function from the statsmodels library. With the line="s" parameter, the line formed by the normal distribution is drawn. The chart title is added with the plt.title function and the chart is displayed with the plt.show function.

Next, the same operation is applied on the normalized\_housing dataset. First, the Q-Q graph of the "DublinNewHousePrice€" column in the normalized dataset is plotted. With the line="s" parameter, the line formed by the normal distribution is drawn. The chart title is added with the plt.title function and the chart is displayed with the plt.show function.

This code is a common method used to check the normal distribution assumption of a dataset. If the data is normally distributed, the points on the Q-Q plot are evenly distributed along the line. If the data does not fit the normal distribution, the dots deviate from the line and an abnormal distribution can be seen on the graph. This method is a reliable and common tool in data analysis.



*Fig 38 - Graph*



*Fig 39 - Graph*

If the dataset is normally distributed, the points on the Q-Q plot are evenly distributed along the line. If the dataset does not fit the normal distribution, the points deviate from the line. As can be seen, both the housing dataset and the normalized\_housing dataset proceed close to the normal distribution.



*Fig 40 - Code*

A scatter plot is drawn to show the relationship between the "DublinNewHousePrice€" column and the "Year" column in a normalized housing dataset.

First, a scatter graph consisting of "DublinNewHousePrice€" and "Year" columns is drawn in the normalized dataset using the scatter function from the matplotlib library. The x and y axes are labeled with the plt.xlabel and plt.ylabel functions. Then the graph is drawn with the plt.show function.

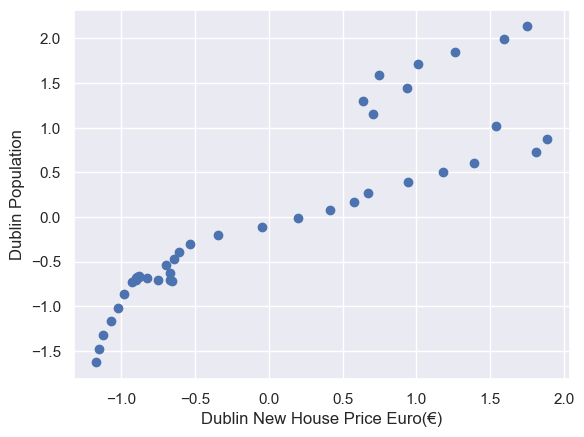
Scatter charts visualize the relationship between data. This chart examines the relationship between the "DublinNewHousePrice€" column on the x-axis and the "Year" column on the y-axis. The distribution of the points shows the relationship between these two columns. If the dots move towards the upper right, it indicates a positive relationship between the two columns. If the dots move towards the upper left, it indicates a negative relationship between the two columns. If the dots are unevenly distributed, there may be a weak correlation or no relationship between the two columns.

As can be seen here, all the attributes have the code associated with the DublinNewHousePrice€ attribute. Also, the codes of the relationships between all attributes and the DublinNewHousePrice€ attribute are commented out and hibernated. Since it will be difficult to plot each of them separately, in the future, relations with all attributes will be shown in a single plot with another method.



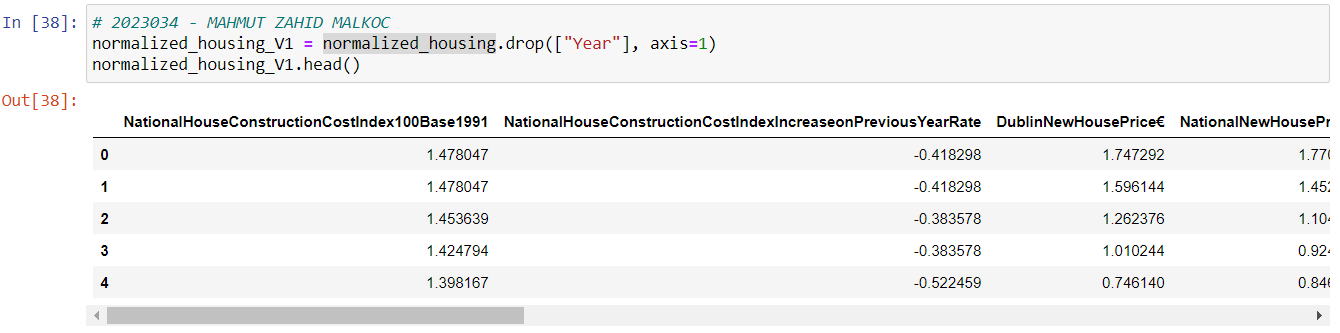
*Fig 41 - Graph*

The relationship between DublinNewHousePrice€ and Year attributes is shown with the scatter plot.



*Fig 42 - Graph*

The relationship between DublinNewHousePrice€ and DublinPopulation attributes is shown with the scatter plot.



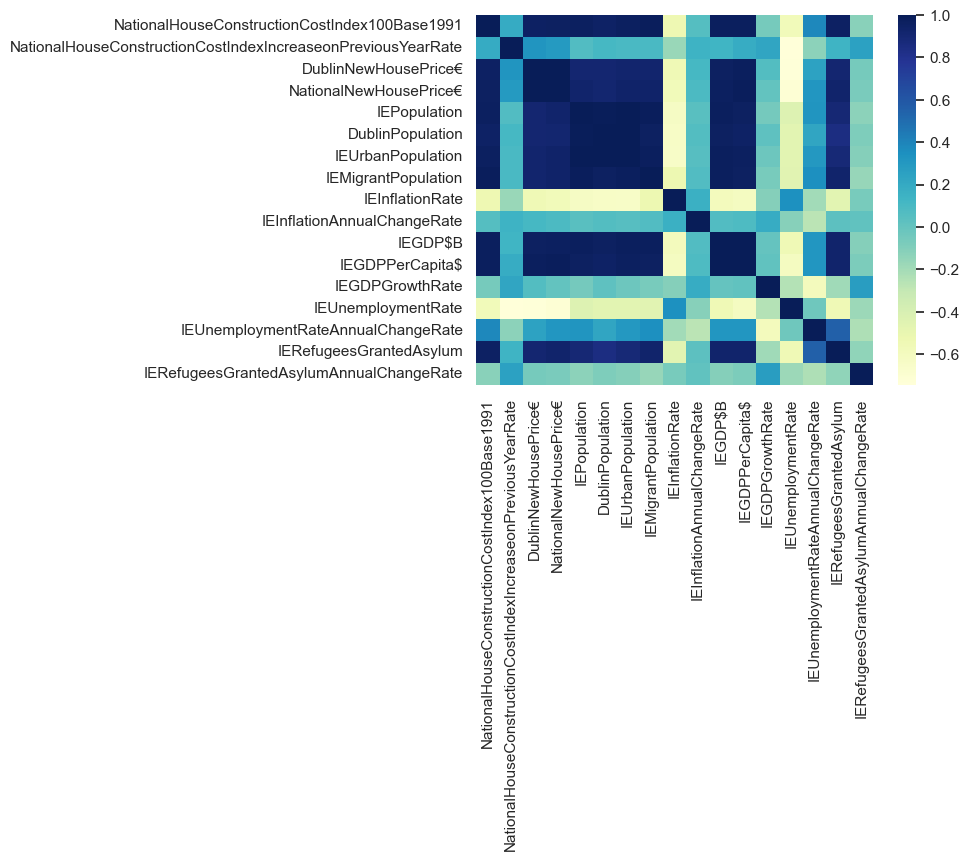
*Fig 43 - Code*

Looking at all the attributes, the Year attribute looks like a numerical value, but it is actually a categorical attribute. Therefore, the year attribute is dropped from the housing dataset. The new dataset is called normalized\_housing\_V1.

*Fig 44 - Code*

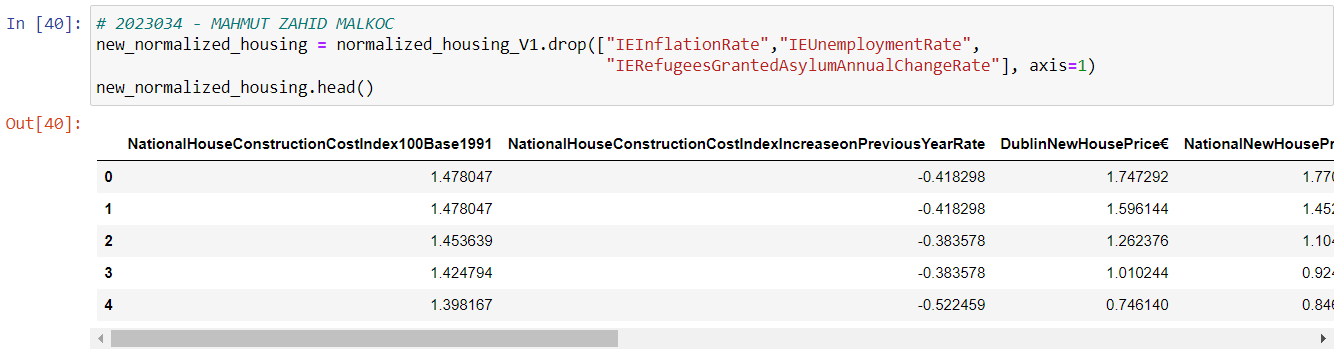
Shows the correlation matrix and heatmap between the normalized\_housing\_V1 columns normalized using the Pandas and Seaborn libraries.

First, a correlation matrix is created using the "corr" function to calculate the correlation between the normalized\_housing\_V1 columns. This matrix contains the Pearson correlation coefficients between the columns.



*Fig 45 - Graph*

As can be seen here, the heatmap was created using the seaborn library. Using the "heatmap" function, the correlation matrix is displayed as a colored heatmap. Colors correspond to temperature, and dark attributes (such as NationalHouseConstructionCostIndex100Base1991 and DublinNewHousePrice€) represent high correlation, and light attributes (such as IEInflationRate and IEUnemploymentRate) represent low correlation. As can be seen from this situation, the 3 lowest correlated attributes will be selected and dropped, and reviewed again.

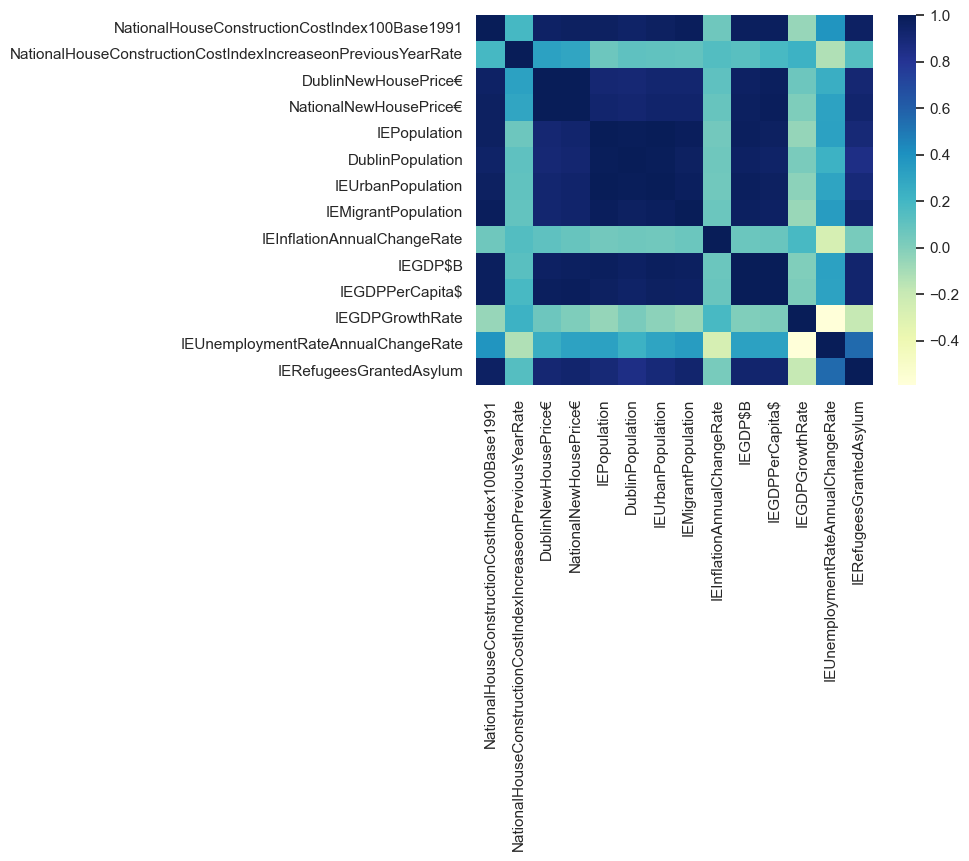


*Fig 46 - Code*

As can be understood from here, IEInflationRate, IEUnemploymentRate and IERefugeesGrantedAsylumAnnualChangeRate attributes have been dropped from the normalized\_housing\_V1 dataset. This new dataset is called new\_normalized\_housing.

 *Fig 47 - Code*

The correlation heatmap of the new\_normalized\_housing dataset is plotted.



*Fig 48 - Graph*

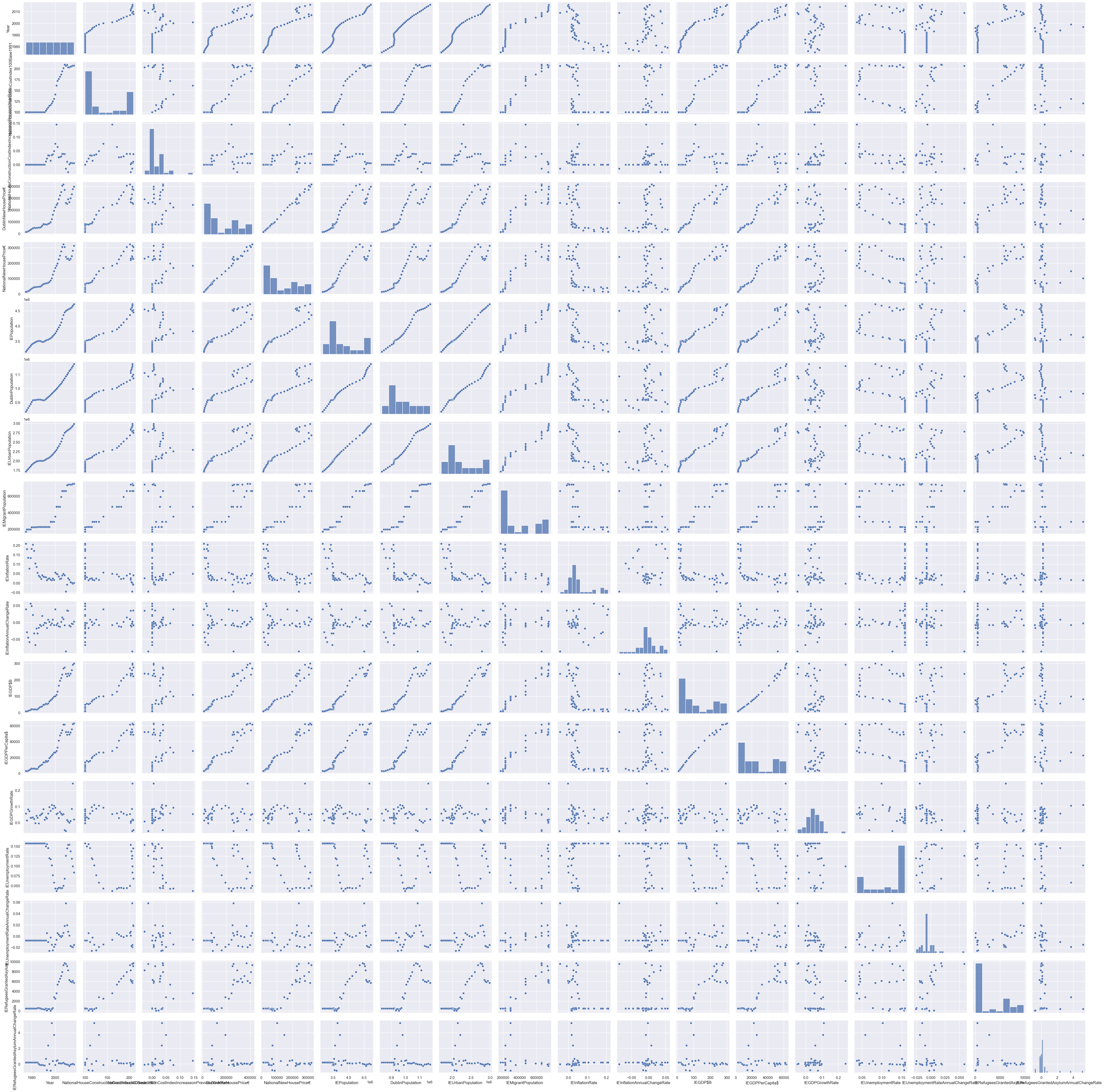
As can be seen, the new\_normalized\_housing heatmap shows a higher correlation density than the previous heatmap.

**Data Visualization and Programming:**



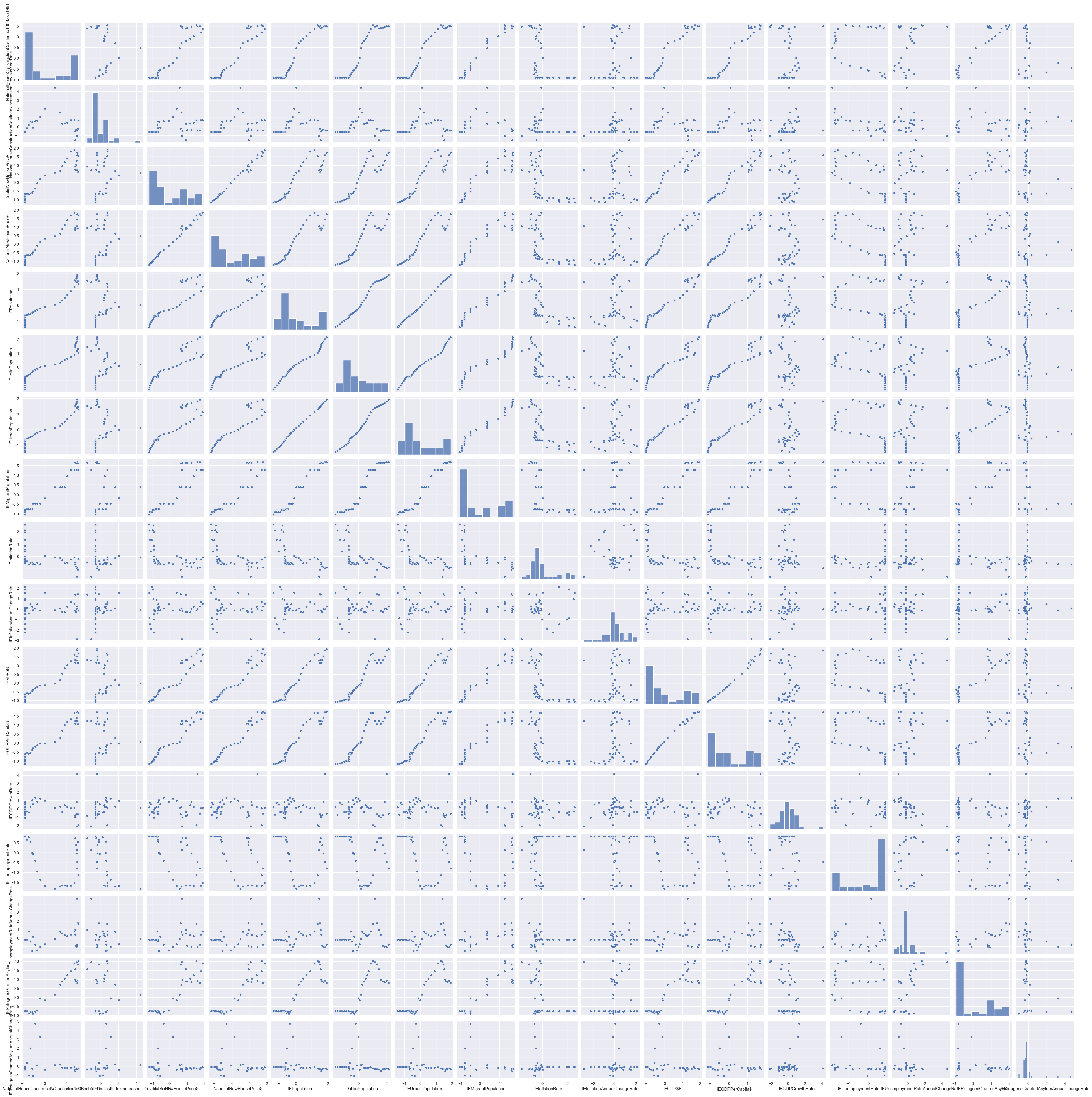
*Fig 49 - Code*

A pairplot is created using the seaborn library to show the relationship of the housing dataset attributes to each other. By comparing the attribute pairs of housing, normalized\_housing\_V1 and new\_normalized\_housing datasets, a pair of charts is created using the scatterplot graph. This method helps to visualize any relationship or patterns in the housing dataset and to see the relationship between all attributes in a single plot.



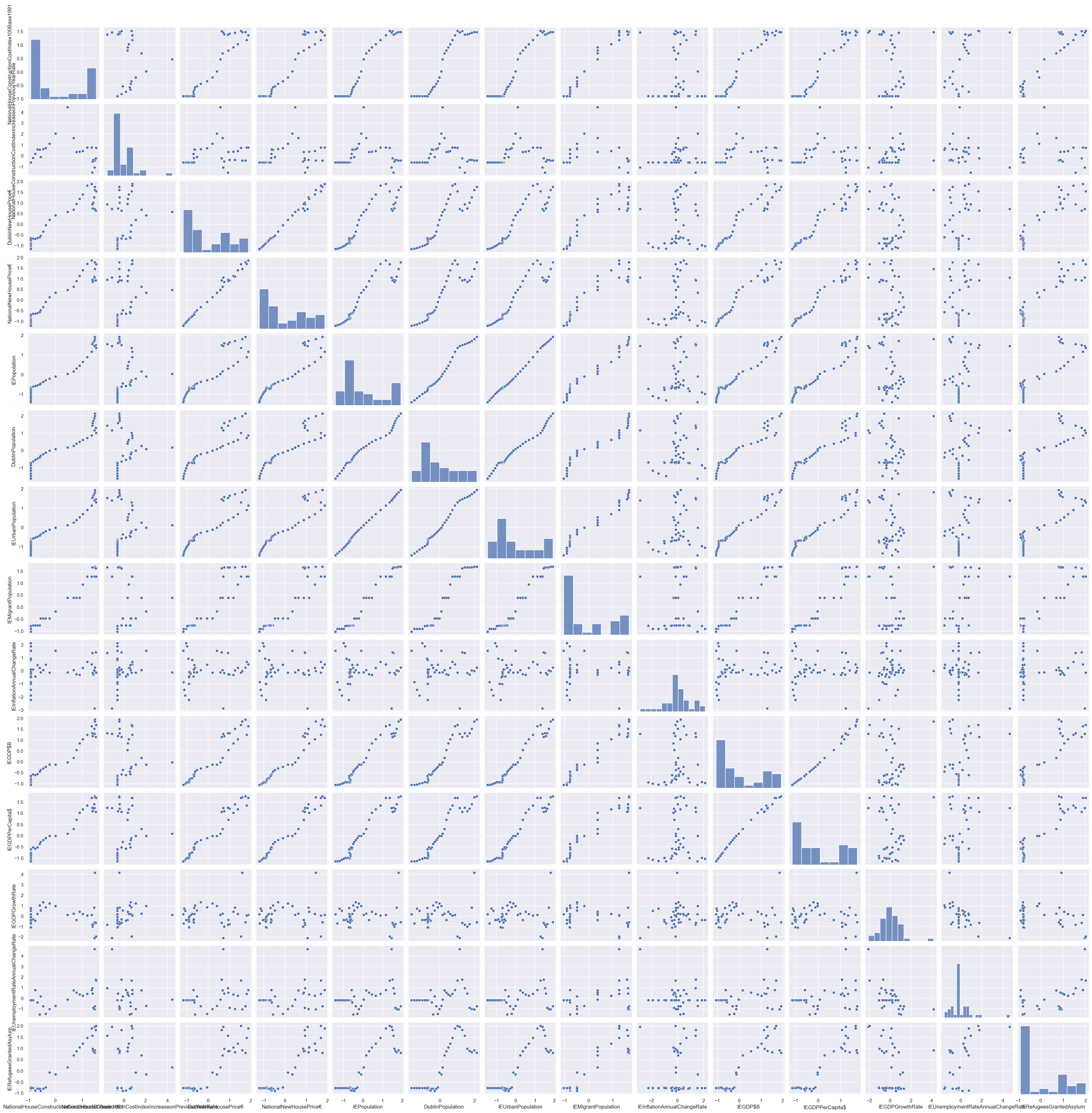
*Fig 50 - Graph*

As can be seen, the housing dataset is drawn in this plot.



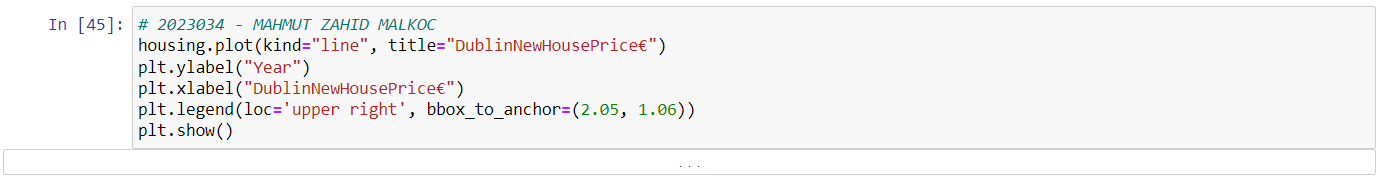
*Fig 51 - Graph*

As can be seen, normalized\_housing\_V1 which is the dropped version of the year attribute from the housing dataset is drawn in this plot.



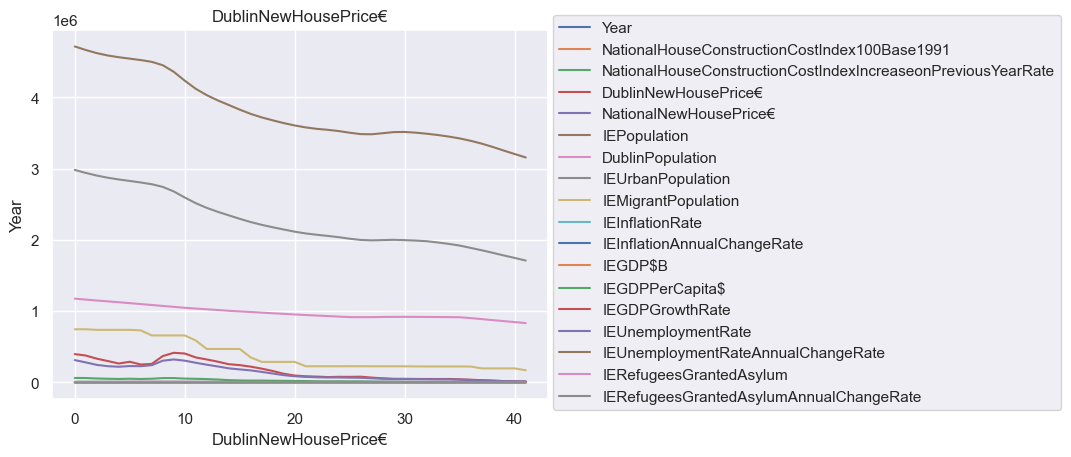
*Fig 52 - Graph*

As can be seen, in this plot, new\_normalized\_housing, which is the drop of 3 different attributes from the normalized\_housing\_V1 dataset, is drawn. It is explained why attributes are dropped from this dataset when normalized is applied. As a result, it is seen that the relationship between the new\_normalized\_housing dataset attributes is more positive.



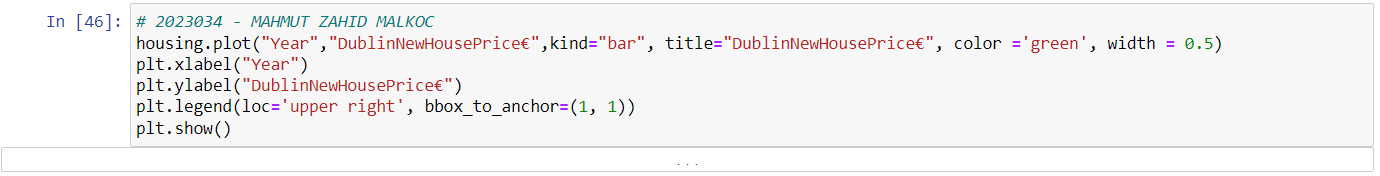
*Fig 53 - Code*

A line chart is created to visualize how the data in the DublinNewHousePrice€ column in the housing dataset changes over time. This type of visualization is used to show how a variable in the housing dataset changes over time. This works especially well for visualizing time-varying data, such as time series data. The chart has years on the y-axis and DublinNewHousePrice€ on the x-axis. The DublinNewHousePrice€ values for each year are shown on the line chart. Also, a legend is placed anywhere in the graphic.



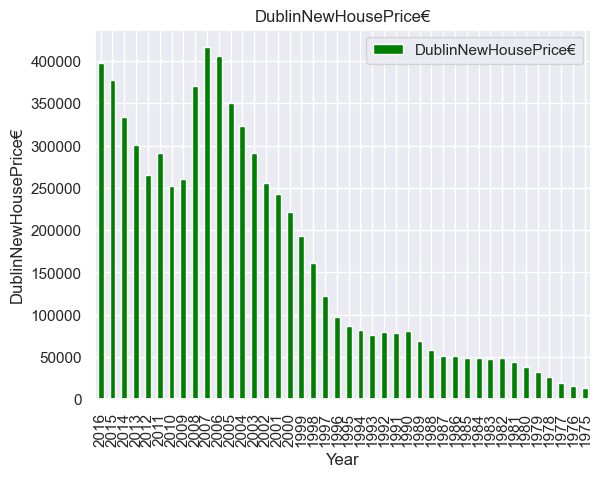
*Fig 54 - Graph*

This line graph is given the relation of all attributes according to DublinNewHousePrice€ and year columns.



*Fig 55 - Code*

It is used to visualize how the data in the DublinNewHousePrice€ column in the housing dataset changes in the bar graph for each year. The chart has year on the x-axis and DublinNewHousePrice€ on the y-axis. The DublinNewHousePrice€ values for each year are shown in the bar graph. There is also a legend at the top of the chart.



*Fig 56 - Graph*

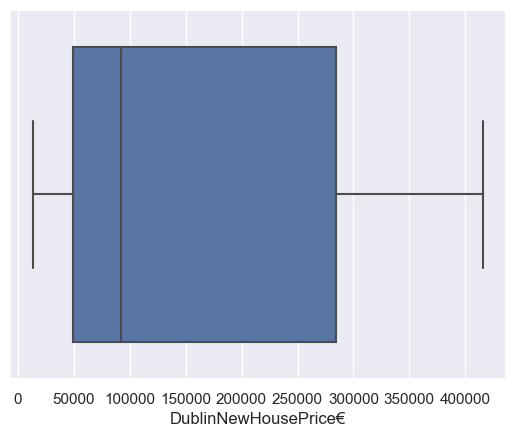
The change of Dublin New House Price€ values over the years is shown with a bar graph.



*Fig 57 - Code*

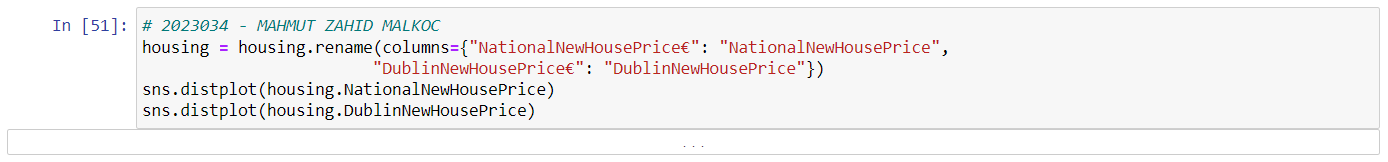
It is used to visualize the boxplot of the data in the DublinNewHousePrice€ column in the housing dataset. The boxplot is plotted based on a five-number summary of the data (minimum, first quartile, median, third quartile, and maximum). Lines (whiskers) usually show minimum and maximum values, but they can be of different lengths depending on the distribution of the data. The horizontal line inside the box represents 50% (median) of the data. Also, the circles shown at the top and bottom of the box indicate data that is called "outlier" and that differs significantly from other data.

In this dataset, the boxplot of the column DublinNewHousePrice€ is drawn using the sns.boxplot function. Because there is only one column on the x-axis, only a boxplot results. The visualization can be used to analyze the central trends, distributions and outliers of the data in the DublinNewHousePrice€ column.



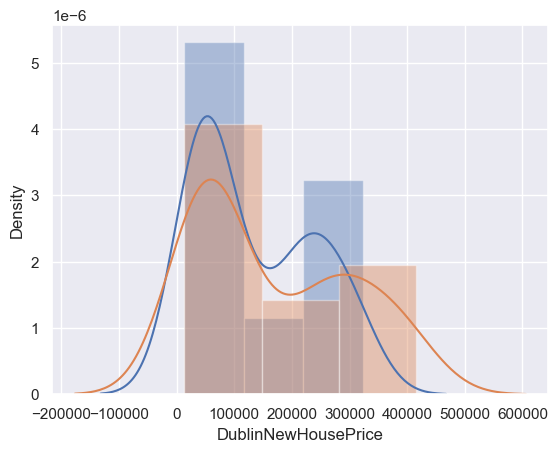
*Fig 58 - Graph*

According to DublinNewHousePrice€, the outlier min value is 20 thousand, while the outlier max value is 450 thousand. Moreover, the first quartile value is 50 thousand while the third quartile value is at 270 thousand. The median value is approximately 90 thousand.

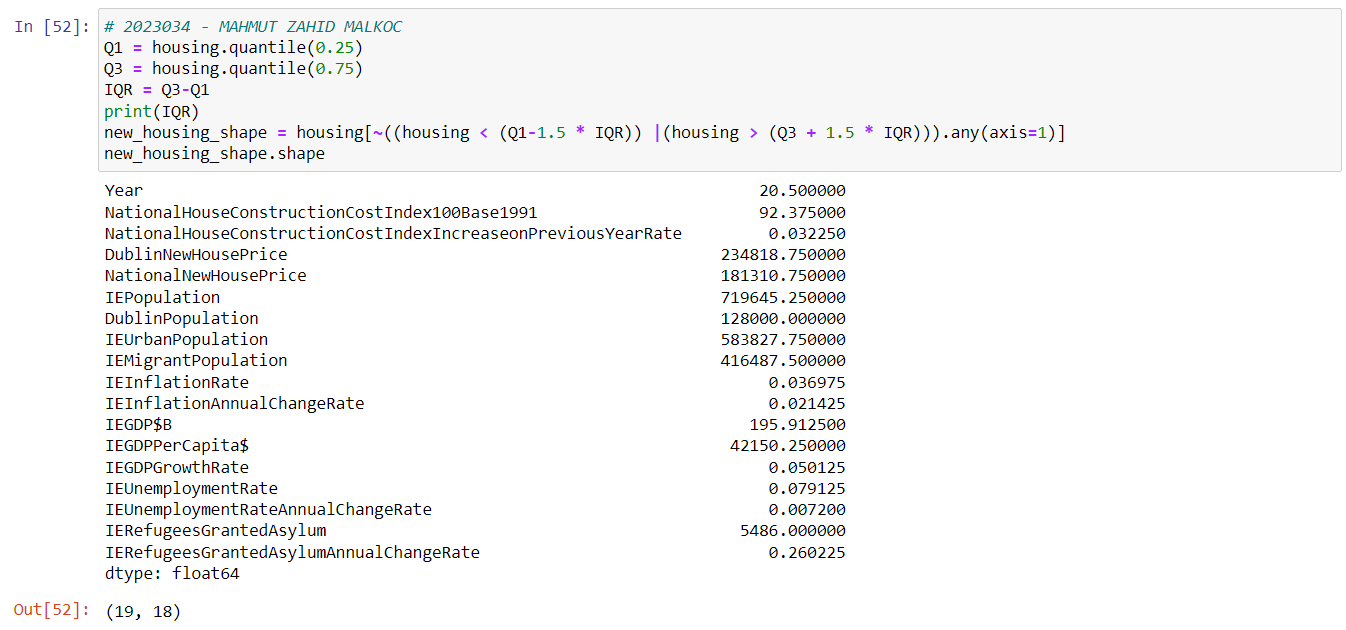


*Fig 59 - Code*

It is used to visually compare the distribution of the NationalNewHousePrice and DublinNewHousePrice variables. The density plot of the DublinNewHousePrice variable is plotted. This density graph is implemented on top of the NationalNewHousePrice density graph. If the distribution of two variables is different from each other, using this chart can see these differences more easily and can be used to understand the relationship between the variables.



*Fig 60 - Graph*

The density of the graph is shown on the y-axis, as it appears in this graph. The spread of the graph is drawn between -200k and 600k. However, this spread range varies between 20 thousand and 420 thousand. The graph is not skewed. DublinNewHousePrice€ and NationalNewHousePrice€ are seen symmetrically. The median value of the graph is 200 thousand, which is the point that separates the half of the graph.

*Fig 61 - Code*

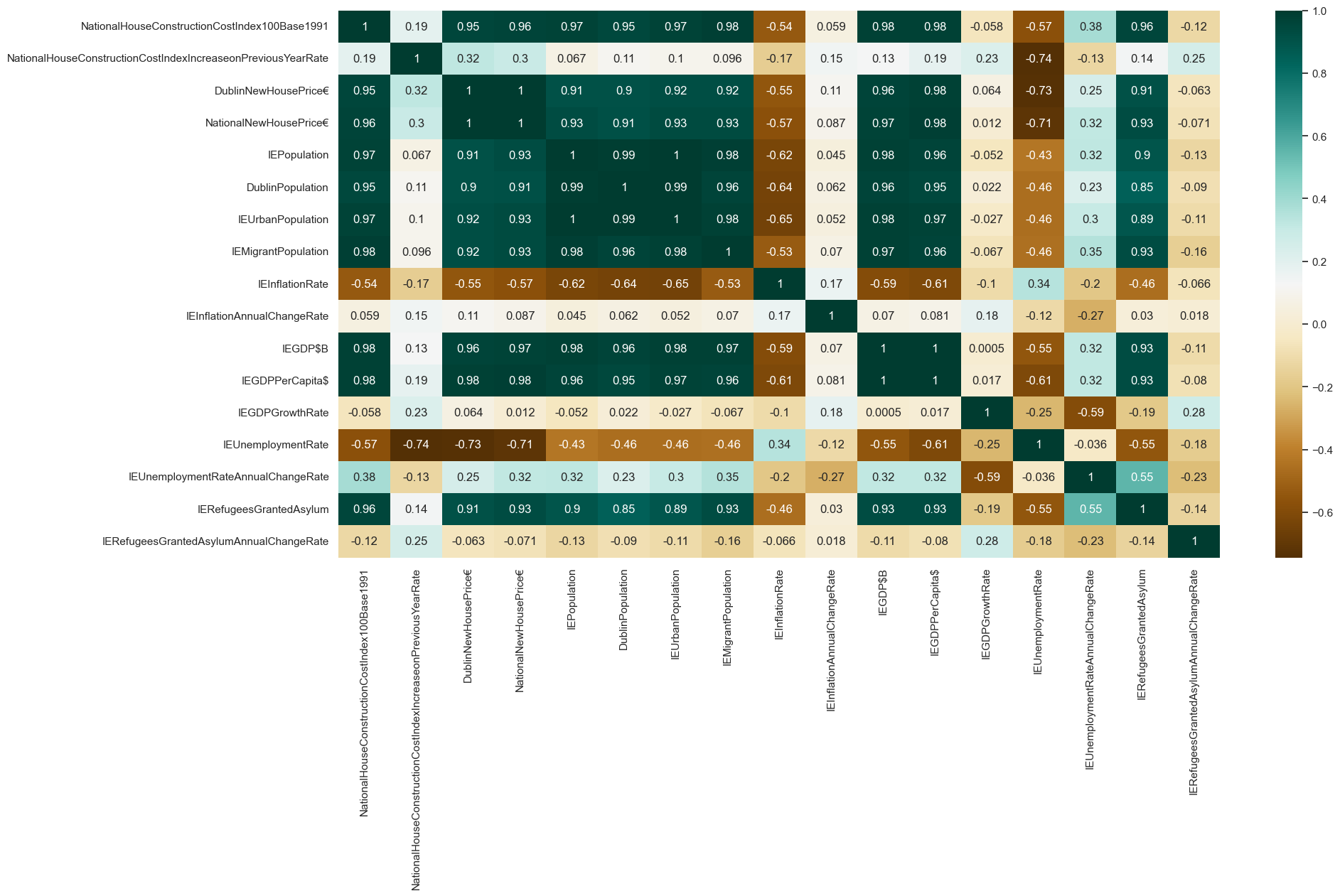
This method is used to detect and remove outliers from the "housing" dataset. The operations are as follows:

* Q1 = housing.quantile(0.25) command calculates the first quartile (Q1) value corresponding to 25 percent of the housing data frame and assigns it to variable Q1.
* Q3 = housing.quantile(0.75) command calculates the third quartile (Q3) value corresponding to 75 percent of the housing data frame and assigns it to variable Q3.
* The IQR = Q3-Q1 command calculates the IQR (Interquartile Range) value representing the interval between Q3 and Q1 and assigns it to the IQR variable.
* The print(IQR) command prints the IQR value to the screen.
* The expression (housing < (Q11.5 \* IQR)) |(housing > (Q3 + 1.5 \* IQR)) creates a mask where all values in the housing dataset correspond to those outside the range of Q1 and Q3. Here, a threshold value for detecting outliers is determined using an IQR of 1.5.
* The expression ~((housing < (Q11.5 \* IQR)) |(housing > (Q3 + 1.5 \* IQR))).any(axis=1) determines whether any rows in the housing dataset have an outlier. The "~" sign, the tilde character, selects all lines outside the mask by inverting the mask. The expression "any(axis=1)" selects rows with any columns with outliers.
* The command new\_housing\_shape = housing[~((housing < (Q11.5 \* IQR)) |(housing > (Q3 + 1.5 \* IQR))).any(axis=1)] creates a new copy of the housing dataset without outliers and new\_housing\_shape assigns it to the variable.
* The new\_housing\_shape.shape command displays the dimensions of the new housing dataset and returns the number of rows and columns.



*Fig 62 - Code*

As explained previous, the Year attribute of the normalized\_housing dataset was dropped from the normalized\_housing\_V1 dataset. It is used to visualize the correlation between the columns of the normalized\_housing\_V1 dataset. The plt.figure(figsize=(20,10)) command determines the size of the heatmap graph to be created. In this case, it is specified that a 20x10 graphic will be created. The c= normalized\_housing\_V1.corr() command calculates the Pearson correlation coefficients between the columns in the dataframe "normalized\_housing\_V1" and assigns them to variable "c". The sns.heatmap(c,cmap="BrBG",annot=True) command creates a heatmap of the correlation matrix. The "c" variable provides the data in the heatmap to be created. The "cmap" parameter determines the colormap to be used. In this case, a colormap named "BrBG" is used. The "annot" parameter is used to insert numbers representing the value of each cell. The c command prints the Pearson correlation matrix.



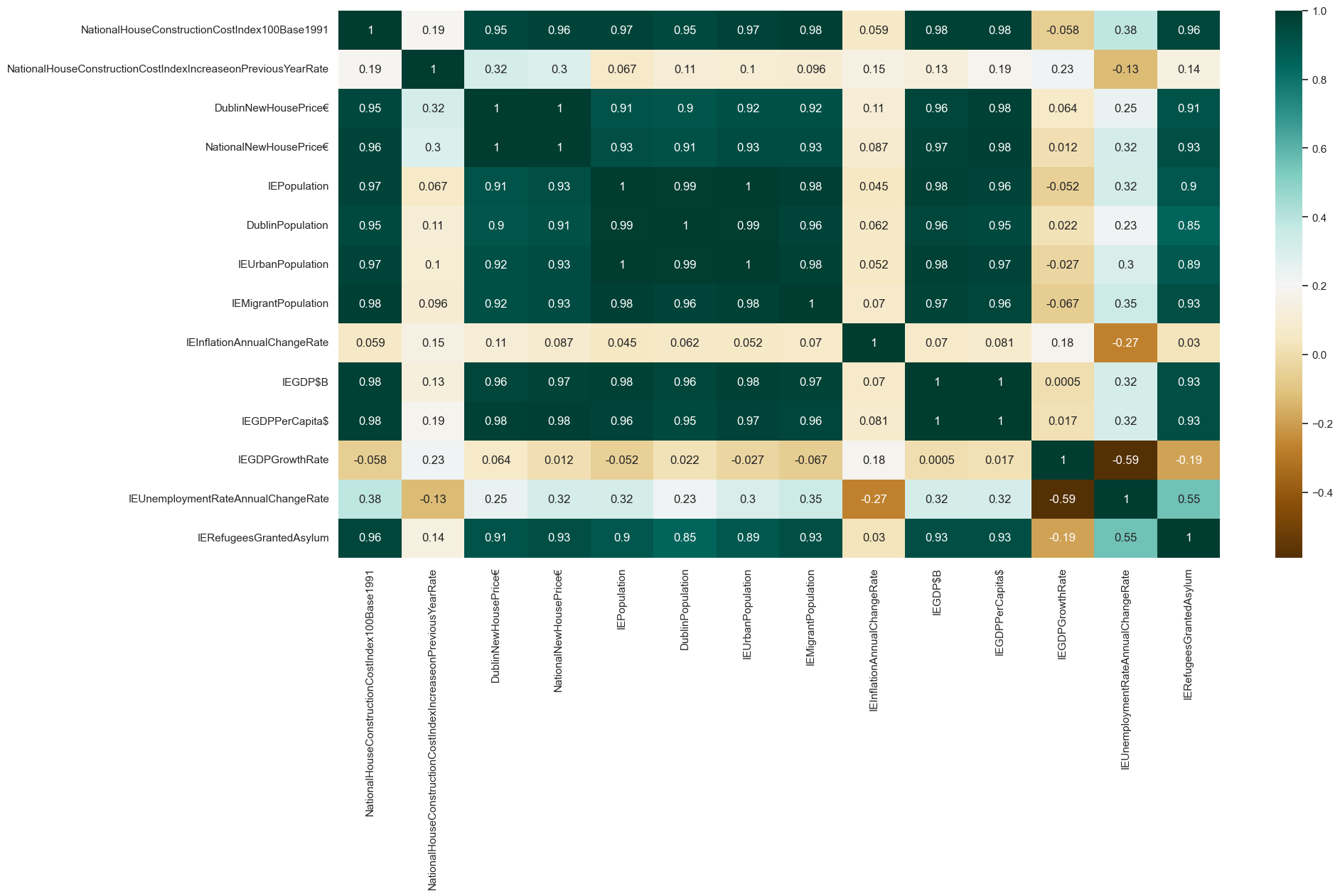
*Fig 63 - Graph*

As seen in the heatmap, the relationships between the attributes are explained with numerical values and colors. As seen in the heatmap, the relationships between the attributes are explained with numerical values and colors. As it will appear from this heatmap, like the previous heatmap, dropping the same 3 attributes will be better for attribute relationships.



*Fig 64 - Code*

The new\_normalized\_housing dataset, which is the recently reduced version of the attributes, is executed.



*Fig 65 - Graph*

The heatmap visualization of the new\_normalized\_housing dataset gave a more relationally logical result than the previous heatmap. The new\_normalized\_housing heatmap also has better relationships and higher density values.

**Machine Learning and Programming:**



*Fig 66 - Code*

The housing dataset is reviewed again.



*Fig 67 - Code*

The necessary libraries and methods for machine learning models are imported.



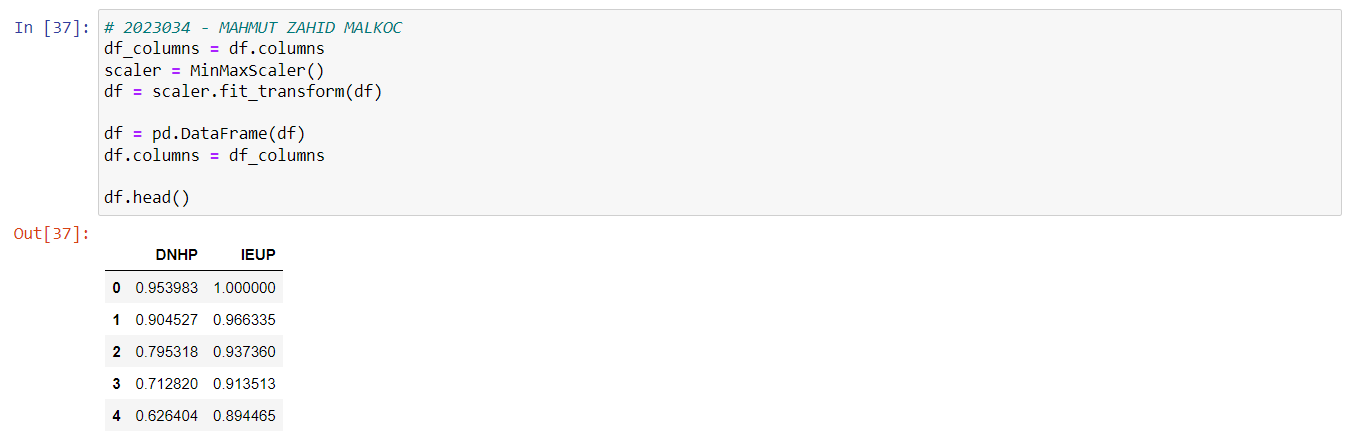
*Fig 68 - Code*

The index length of the housing dataset is checked.



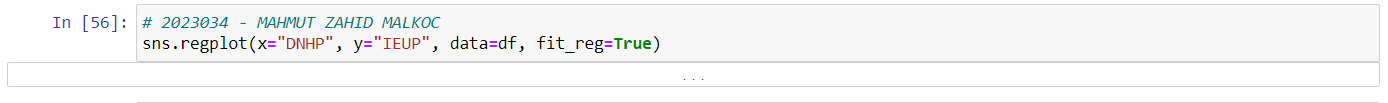
*Fig 69 - Code*

A new dataframe is created by selecting the DublinNewHousePrice and IEUrbanPopulation attributes from a housing dataset. Next, the DublinNewHousePrice column is renamed DNHP and the IEUrbanPopulation column is renamed IEUP. Finally, the first five rows of the new dataframe are printed using the .head() method. It was made to provide more convenient access to Attributes.



*Fig 70 - Code*

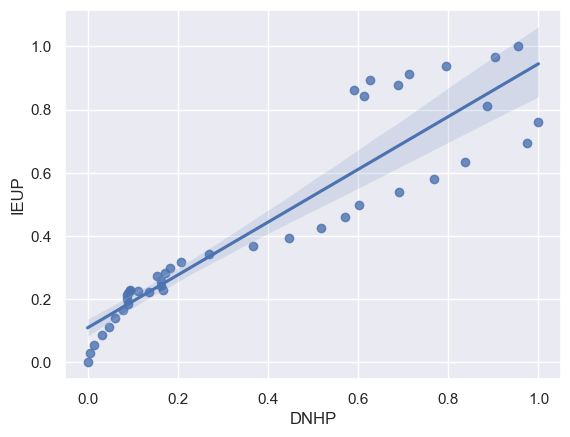
The MinMaxScaler() class is used to normalize the values of df 's properties. First, df holds the dataframe column names in a variable named df\_columns. It then assigns the MinMaxScaler() class as an example to the scaler variable.Then, using the fit\_transform() method, the columns of the dataset are normalized using the scaler object. Normalized data is returned as a Numpy array and then converted to a new Pandas dataframe. This is done using the pd.DataFrame() function, and the columns of normalized data are assigned df\_columns names. This allows the column values in the dataset to be normalized using the minimum-maximum scaling method. The normalization process ensures that the property values have the same scale and a more consistent comparison.



*Fig 71 - Code*

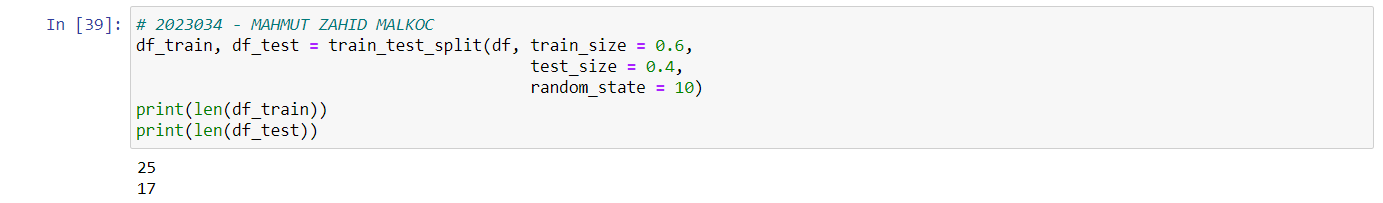
Generates a regression graph using the Seaborn library. The x parameter specifies DNHP the column of the data set to be displayed on the x-axis of the graph, while the y parameter specifies IEUP the column of the data set to be displayed on the y-axis of the graph. The data parameter specifies the data frame df from which the chart will be created.

The fit\_reg parameter determines whether to show the regression line. When set to true, the regression line is drawn. In this case, assuming the relationship between the x and y variables is linear, a regression line is drawn that predicts the best fit.



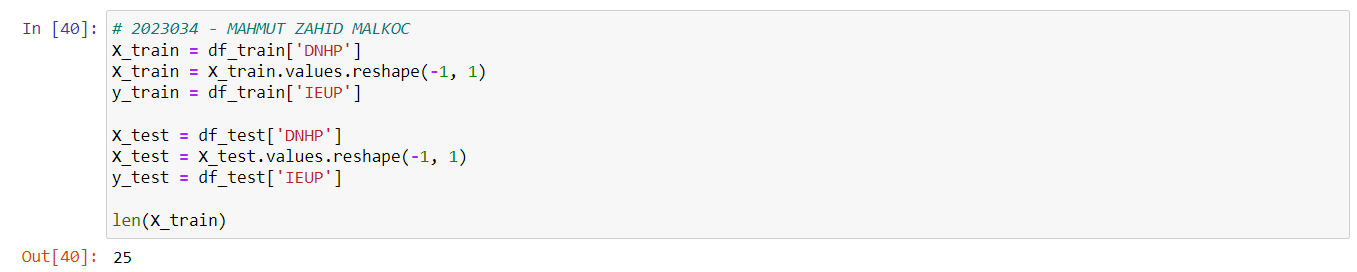
*Fig 72 - Graph*

As seen in this graph, some values may remain above this line, while others may remain in the upper right and upper left. Here, most of the dots stay on the right side.



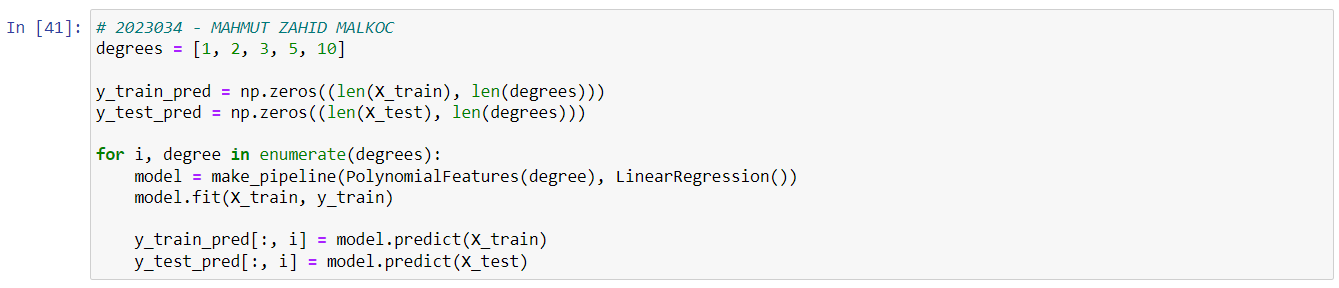
*Fig 73 - Code*

The train\_test\_split function splits the df dataframe into training and test sets and stores them in the df\_train and df\_test dataframes, respectively. The train\_size and test\_size parameter values are 60% and 40% respectively.



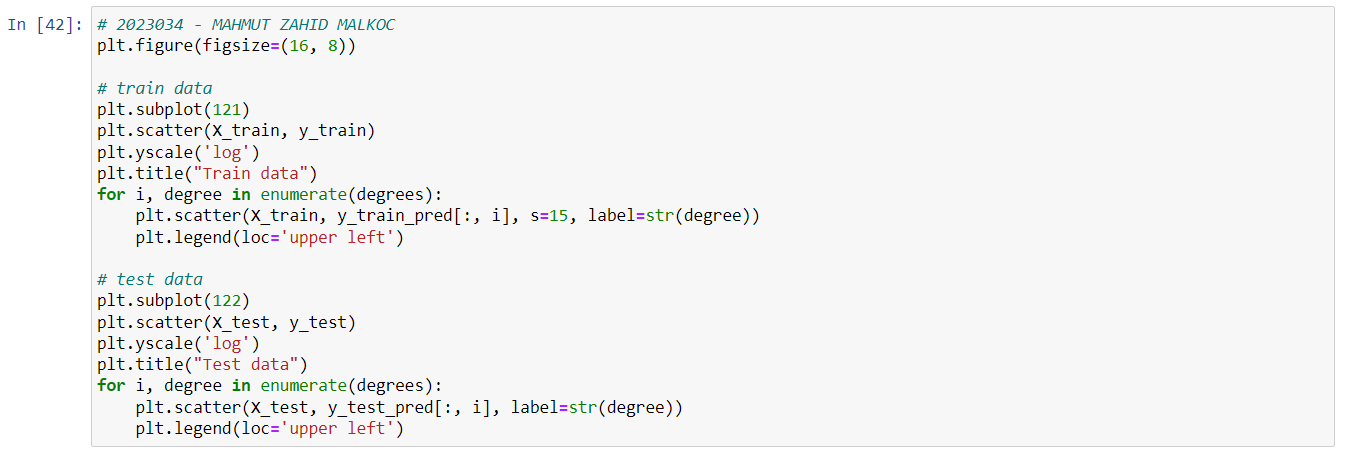
*Fig 74 - Code*

The independent variable (X) and dependent variable (y) sets are separated from the training and test data frames. The x\_train and y\_train variables are set from the df\_train dataframe as data in column DNHP and data in column IEUP. The data in the x\_train variable is reshaped as a one-dimensional array using the values.reshape() method. The x\_test and y\_test variables are set from the df\_test dataframe as data in column DNHP and data in column IEUP. The data in the x\_test variable is reshaped as a one-dimensional array using the values.reshape() method. Finally, the len(X\_train) function returns the number of rows in the variable X\_train. In this case, the number of rows in the X\_train dataset specifies the size of the dataset.



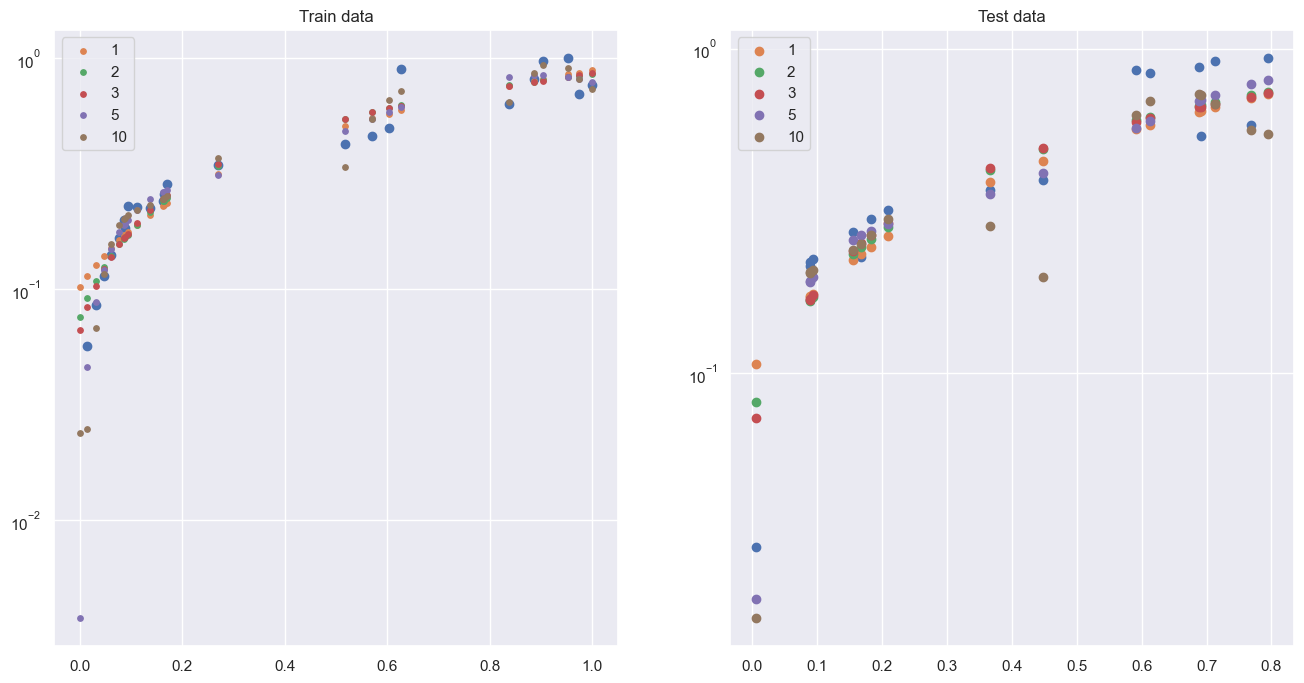
*Fig 75 - Code*

Regression models with different polynomial degrees (1, 2, 3, 5, 10) are created and these models are made suitable in the training dataset. The np.zeros() function creates an array of zeros. The variables y\_train\_pred and y\_test\_pred are created as matrices of zeros, each with a column for each degree in the degrees list. The loop creates a regression model by applying a polynomial feature extraction for each degree. The make\_pipeline function chains together the polynomial feature extraction followed by the linear regression model. The model is trained using the fit() method on the training dataset. For each degree, predictions are made on the X\_train data in the training dataset and assigned to the corresponding column in the y\_train\_pred matrix. Similarly, predictions are made on the X\_test data in the test dataset and assigned to the corresponding column in the y\_test\_pred matrix.



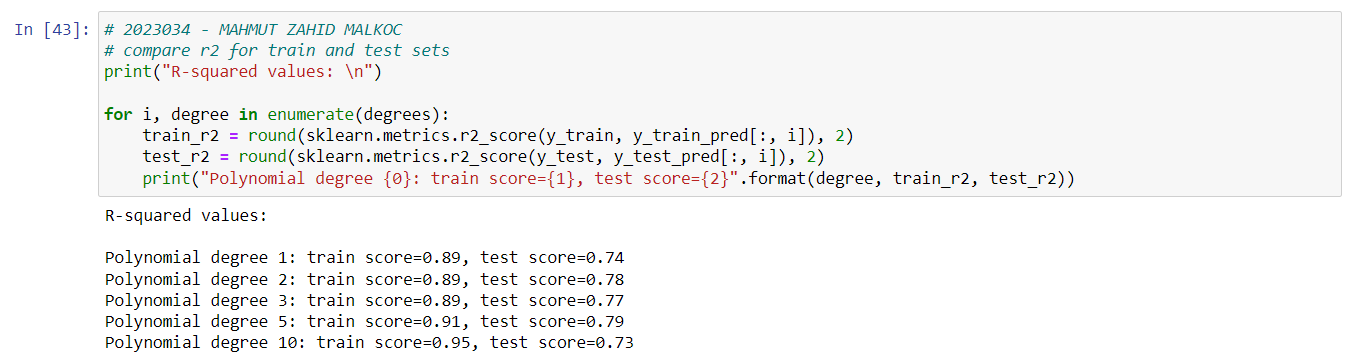
*Fig 76 - Code*

A visualization of the actual values for the training and test dataset and the predicted values for each grade is made. The plt.subplot() function is used to create multiple subplots. The plt.scatter() function is used to visualize point data. Using the pairs x\_train and y\_train or X\_test and y\_test, the actual data is plotted. The for loop is used to plot the predicted values on the training and test data for each degree. Relevant columns are selected from the y\_train\_pred and y\_test\_pred matrices and drawn as points using the plt.scatter() function.



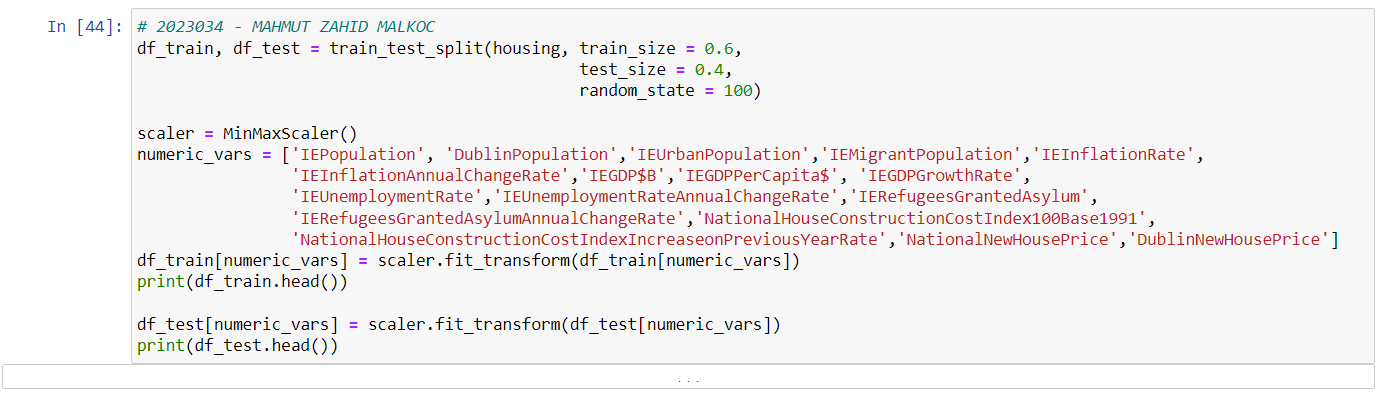
*Fig 77 - Graph*

The scatter graph of Train and Test data was obtained as above.



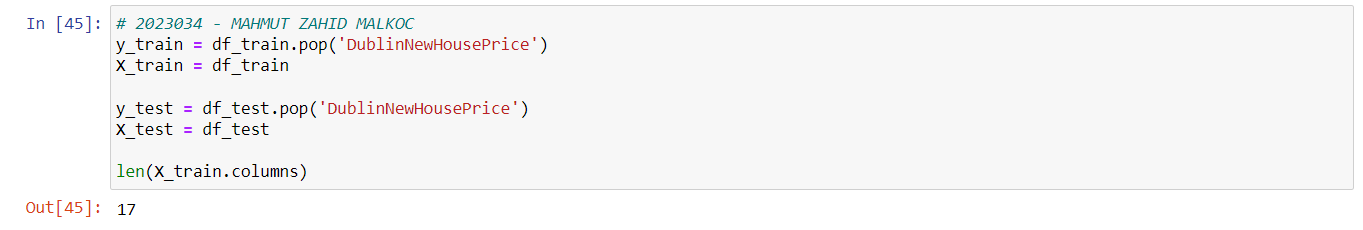
*Fig 78 - Code*

Evaluates the performance of a regression model created using different degrees of polynomial on training and test data with R-square score. The R-squared score is a metric used to measure how well a regression model fits. The range of values ranges from 0 to 1, and the closer to 1, the better the model fits the data. As can be seen, the best result is the 4th degree polynomial.



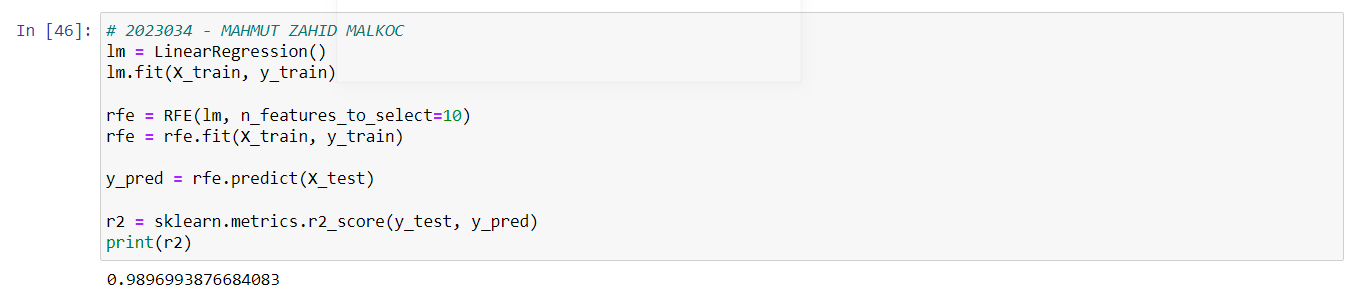
*Fig 79 - Code*

An instance of class "MinMaxScaler" is created and the values of the numeric columns in the "numeric\_vars" list are scaled. The "fit\_transform" method performs the scaling and assigns the transformed data to the variable.



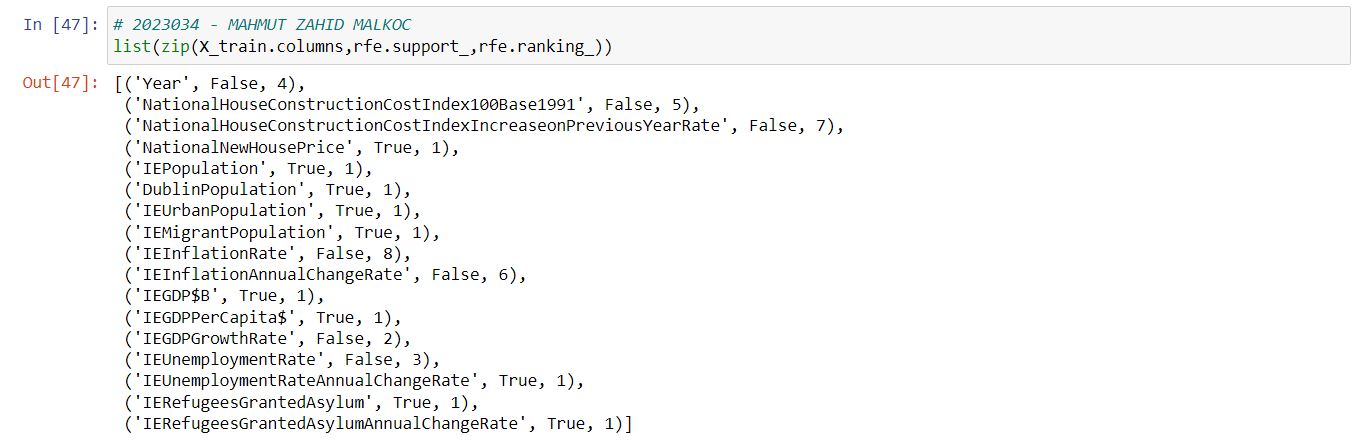
*Fig 80 - Code*

By specifying the DublinNewHousePrice column as the target variable, training and test datasets are created. It is used to assign target variables to the y\_train and y\_test variables. The pop method subtracts the selected column and assigns all remaining columns to the variables X\_train and X\_test.



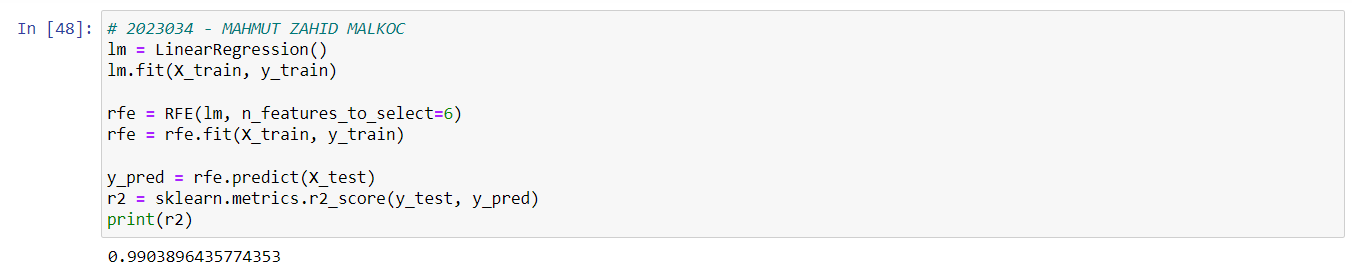
*Fig 81 - Code*

It is used to construct a linear regression model and then select the most important variables using the RFE (Recursive Feature Elimination) method. The most important variables are selected using the RFE class. This operation specifies how many variables will be selected using the n\_features\_to\_select parameter. R-squared is a measure of the explainability of a regression model and takes a value between 0 and 1. As can be seen, R-square score close to 1 indicates that the model explains the data well. The r-square result of the model is quite close to 1. This indicates that the data of the model is highly explicable.



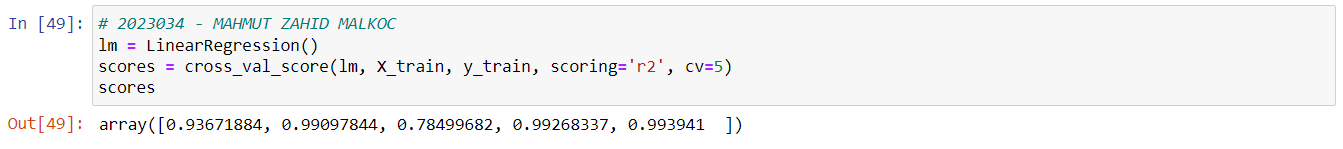
*Fig 82 - Code*

The zip function combines different lists such as column names, status of selected properties, and sorting information to form a group. The first column contains the column names, the second column contains the status of the selected features (True or False), and the third column contains the sorting information. The "rfe.ranking\_" property returns a list with ranking information for each column. This property is a value used in sorting in the RFE method and determines the importance of the columns. Low numbers indicate columns that are more important, while high numbers indicate less important columns.



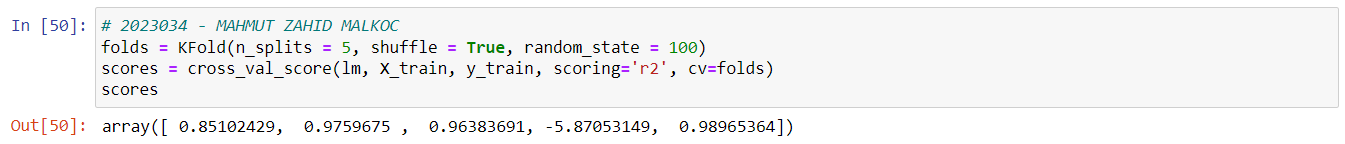
*Fig 83 - Code*

When the n\_features\_to\_select value is set to 6, a value closer to 1 is obtained than the previous r-square result. This means that the explainability of the data of this model has increased.



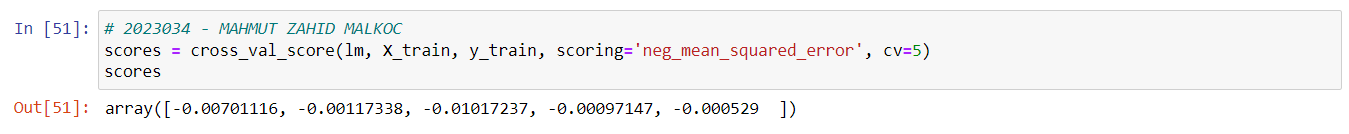
*Fig 84 - Code*

Performs 5-fold cross validation to determine the performance of a linear regression model on the X\_train dataset. The linear regression model models a relationship and learns using the least squares method. The R square score is used to measure the performance of the model, and the cross\_val\_score method provides more reliable results by performing multiple cross validations. This method is used to more reliably measure the performance of the model and helps reduce problems such as overfitting.



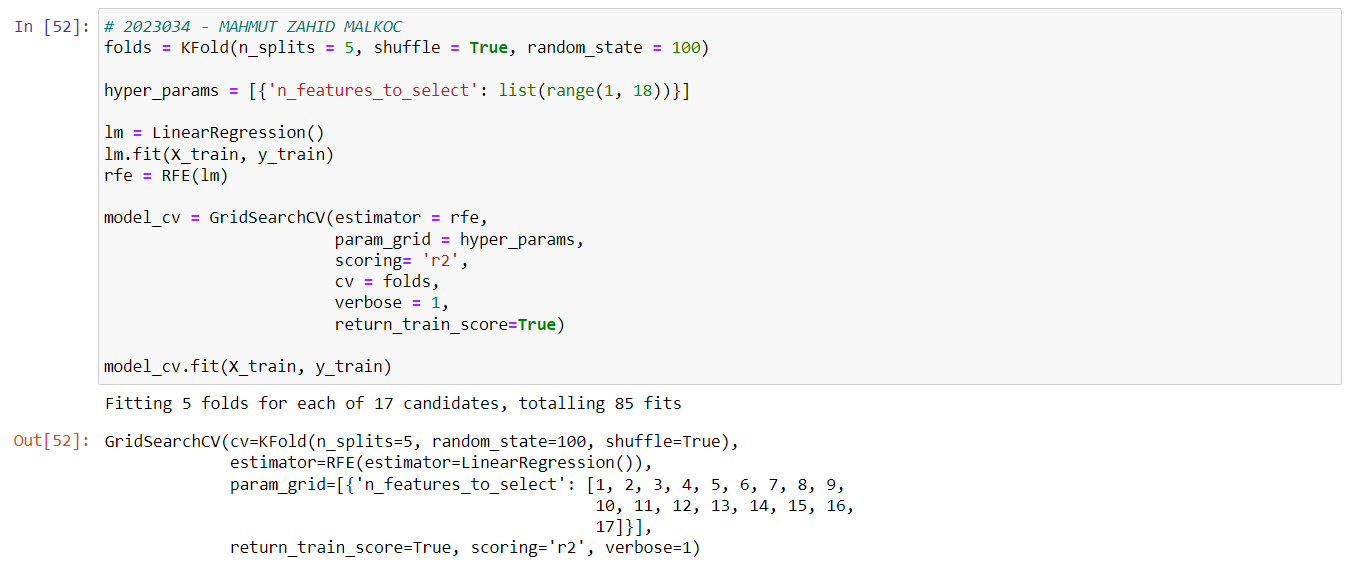
*Fig 85- Code*

Demonstrates the use of the KFold and cross\_val\_score methods. The KFold method divides the dataset into a specified number of pieces, and each piece is used as a test set, while the other pieces are used as a training set. In this way, it is aimed to measure the performance of the model more reliably.



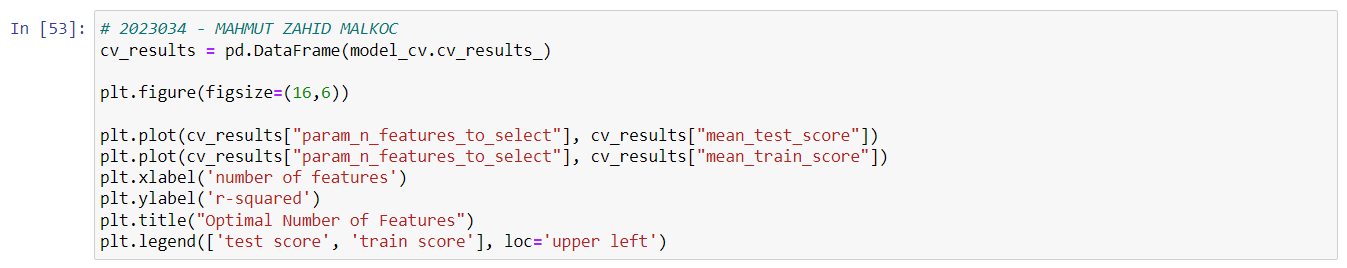
*Fig 86 - Code*

The scoring parameter is set to neg\_mean\_squared\_error which means negative values will be calculated for the mean squared error (MSE). The results are stored in the "scores" variable and are calculated as negative values of the MSE. This method is a technique used to more reliably measure the performance of the model.

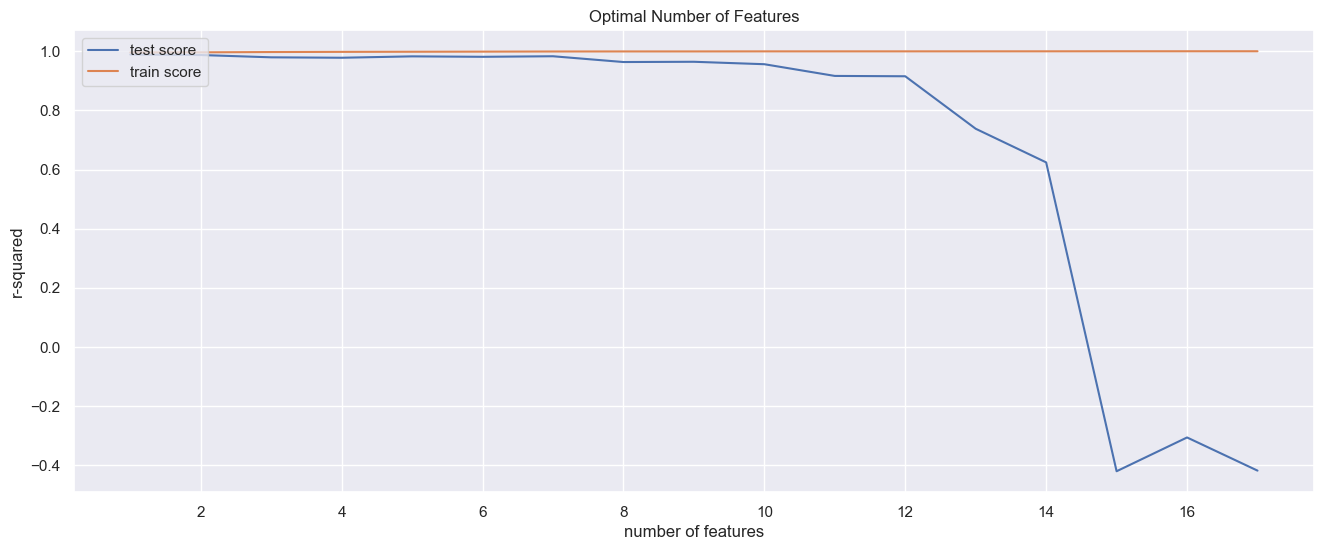


*Fig 87 - Code*

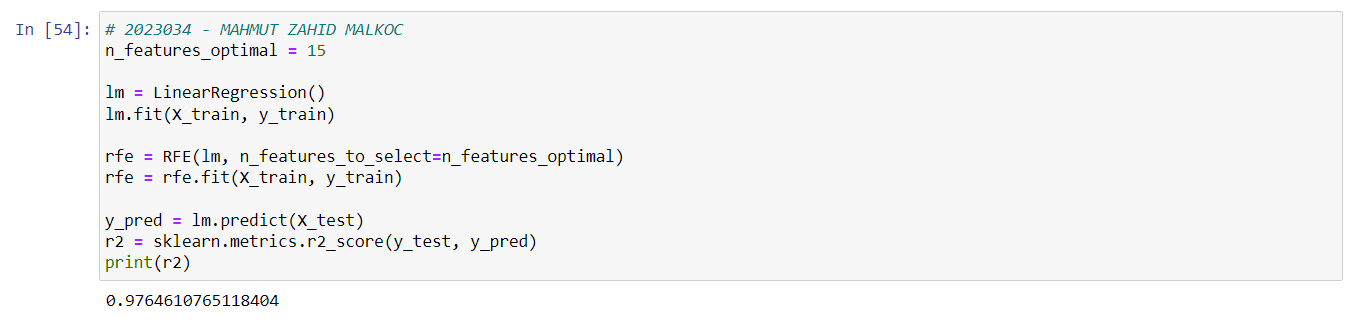
It uses the GridSearchCV method to determine the hyperparameters of the RFE method for feature selection. First, a linear regression model is created and the RFE method is adjusted to this model. Then, the hyperparameters of the RFE method are determined using the GridSearchCV method. This method helps to choose the best model with cross validation.



*Fig 88 - Code*

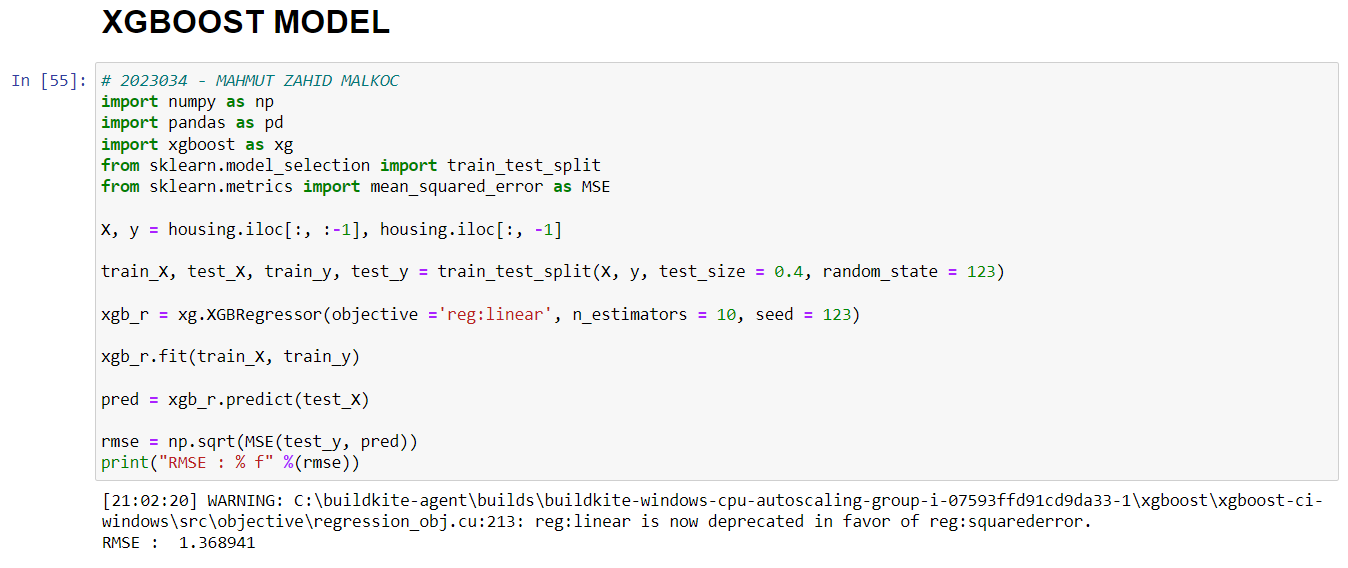


*Fig 89 - Graph*

This graph visualizes the relationship between test and training R-square scores when a given number of features is used. The highest point of the graph indicates the optimal number of features that give the best R squared score.

*Fig 90 - Code*

RFE method is applied using the optimal number of features determined for feature selection and the results obtained are evaluated with the measured R square score. The final r-square value of the model appears.



*Fig 91 - Code*

It creates a regression model using the XGBoost (eXtreme Gradient Boosting) method and measures the performance of this model. First, the independent variables "X" and the dependent variable "y" are assigned from the housing dataset. Using the train\_test\_split method, the dataset is split into 60% training and 40% test data. The "random\_state" parameter is used to achieve the same randomness every time.

Next, an XGBoost regression model is created using the XGBRegressor class. Since the objective parameter is set to reg:linear, the regression problem is solved. Since the n\_estimators parameter is set to 10, 10 trees are used. The seed parameter is used to get the same randomness every time.

The model is trained using the fit method and predictions are made on the test data with the predict method. Using MSE (Mean Squared Error) and np.sqrt (square root) functions, RMSE (Root Mean Squared Error) is calculated and printed on the screen.

The XGBoost method creates tree-based learning models using the gradient boosting technique. In this way, high performance and fast working models can be obtained.

The RMSE (Root Square Mean Error Squared) value is the root mean square error of the differences between the predicted values and the actual values. Therefore, the lower the RMSE, the better the model's performance.

**Programming:**

The advantages of the Python programming language when implementing machine learning models are:

* Ease of use: Python is a language that is easy to read and understand, so it is simply familiar. Also, machine learning algorithms can be easily implemented, as many libraries and tools provide support for the Python language.
* Widespread use: Because Python is an open source programming language, users can find resources and community support for libraries and tools that are updated frequently.
* Libraries: Python has many popular machine learning libraries (eg NumPy, Pandas, Scikit-Learn, TensorFlow, PyTorch). These libraries are more convenient than other languages to perform the operations required for machine learning models.
* Performance: Although the Python language is slower compared to languages such as C, C++ and Java, operations can be accelerated using many libraries such as Numpy, Scipy, Pandas.
* Multi-platform support: Python supports many platforms that can run on Windows, Mac OS, Linux and many other operating systems, allowing models to be implemented in different environments.

**References:**

The World Bank, 2023. *Macrotrends.* [Online]   
Available at: https://www.macrotrends.net/countries/IRL/ireland/urban-population  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/SM.POP.NETM?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/NY.GDP.MKTP.CD?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/NY.GDP.PCAP.CD?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/SM.POP.NETM?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/SL.UEM.TOTL.ZS?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/SM.POP.REFG?locations=ie  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG?locations=IE  
[Accessed 7 April 2023].

The World Bank, 2023. *The World Bank.* [Online]   
Available at: https://data.worldbank.org/indicator/SL.UEM.TOTL.ZS?locations=IE  
[Accessed 7 April 2023].

United Nations - World Population Prospects, 2023. *Macrotrends.* [Online]   
Available at: https://www.macrotrends.net/cities/21542/dublin/population  
[Accessed 7 April 2023].

United Nations - World Population Prospects, 2023. *Macrotrends.* [Online]   
Available at: https://www.macrotrends.net/countries/IRL/ireland/population#:~:text=The%20population%20of%20Ireland%20in,a%201.02%25%20increase%20from%202019.  
[Accessed 7 April 2023].

United Nations - World Population Prospects, 2023. *Macrotrends.* [Online]   
Available at: https://www.macrotrends.net/countries/IRL/ireland/net-migration  
[Accessed 7 April 2023].