HUMBER INSTITUTE OF TECHNOLOGY
AND ADVANCED LEARNING
(HUMBER COLLEGE)

Group Assignment Report

Machine Learning 1 - BIA-5302-0GA

Group - 7

Data Set - **Boston Housing** (The Boston House-price Data, n.d.)

Submitted by:

Last Name	First Name	Student Number
Das	Subhanjan	N01431473
Patel	Hardi	N01480409
Shah	Divyansh	N01472284
Shand	Yuvraj	N01479401
Tyagi	Abhi	N01474042

Submitted to: Professor Salam Ismaeel

Submission Date: 2022-09-19

INDEX

PART A

Make a review of such techniques, data, and examples with references.

Outliers are the values that do not fit the data type or range of the data. Outliers can be of different data types and we can detect and treat outliers using the methods mentioned below. An everyday example of an outlier can be the salary disparity between an employee of an organisation and the CEO of the same organisation.

Outliers (Cause) -

- Data-entry errors - These errors include decimal misplacements, value misplacement, and errors that occur during data entry.
- Measurement errors - These errors occur when there is a discrepancy in the measurement process of the data. These values may not always be wrong but induce errors in models. (Shmueli et al., 2019)
- Missing Values - The errors may occur due to misinterpretation of data by humans, equipment error, losing of samples while recording values, and others.

Outliers (Review) -

- Box Plot - Outliers can easily be visualized with the help of box plots. A box plot divides and maps the data into 4 quartiles (0-25%, 25%-50%, 50%-75% and 75%-100%).
- Sorting Values -It is used to anticipate the locations of unsorted data components within a sorted sequence. We can sort a column in python with the help of the sort_values() function. (Shmueli et al., 2019)
- Statistical Methods - quantile(), z score
- isnull() - We can find null values with the isnull() command
- isnan() - We can find NaN values with the isnan() command
- Manually setting a range - Instead of using statistical methods to find the min-max limits, we can manually

Outliers (Treatment) -

- Outlier trimming: This is not the most effective method for treating outliers since we can lose a lot of data by using this method and can have an impact on the dataset.
- Mean, Median imputation: The process of replacing missing/NaN values.
- Outlier capping: It is used to set a limit above or below a particular value for the field. (Goyal, 2022)

PART B

Here, we will be using the Boston Housing Dataset. (The Boston House-price Data, n.d.)

Importing excel file to Pandas Data Frame

In [1]:
```
#Import necessary packages

import pandas as pd
import numpy as np
```

In [2]:
```
#Read Excel file into Pandas DataFrame

boston_housing = pd.read_excel('BostonHousing.xls',sheet_name='Data')
```

In [3]:
```
boston_housing.head()     #Top 5 rows
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	0.06905	0.0	7.07	0	0.458	7.147	54.2		3	222	18.7

In [4]:
```
boston_housing.shape   #Dimensions of the Dataframe
```

Out[4]:
```
(167, 11)
```

In [5]:
```
boston_housing.dtypes    #Datatypes of the dataframe
```

Out[5]:
```
CRIM       float64
ZN         float64
INDUS       object
CHAS         int64
NOX         object
RM         float64
AGE        float64
DIS         object
RAD          int64
TAX          int64
PTRATIO     object
dtype: object
```

Here we import the necessary packages and read the excel file using Pandas Data Frame.

B.1: Handling Missing Data

Creating Functions to highlight missing and wrong values with a yellow background

Highlighting Cell that do not have numbers in the cells (Except: PTRATIO)

In [6]:
```
#Function to highlight cells yellow with non-float or non-int values
def float_check_background(cell_value):

    highlight = 'background-color: yellow;'
    default = ''

    if type(cell_value) in [float,int]:
        return default
    else:
        return highlight


#Function to highlight cells yellow with null values
def check_nan_background(cell_value):

    highlight = 'background-color: yellow;'
    default = ''

    if pd.isnull(cell_value) is True:
        return highlight
    else:
        return default
```

The above function float_check_background returns yellow background to the cells that do not have any number in them along with that check_nan_background returns yellow background to the cells with null or NaN values with the code referenced from (Felipe, 2022):

In the image below we run the above two functions to apply that function to all the columns except 'PTRATIO'

In [7]:
```
#Applying the above functions to all columns except PTRATIO

(boston_housing.iloc[:,0:10].style
 .applymap(check_nan_background)
 .applymap(float_check_background))
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.006320	18.000000	2.310000	0	0.538000	6.575000	65.200000	4.090000	1	296
1	0.027310	0.000000	7.070000	0	0.469000	6.421000	78.900000	4.967100	2	242
2	0.027290	0.000000	7.070000	0	0.469000	7.185000	61.100000	4.967100	2	242
3	0.032370	0.000000	2.180000	0	0.458000	6.998000	45.800000	6.062200	3	222
4	0.069050	0.000000	7.070000	0	0.458000	7.147000	54.200000		3	222
5	0.029850	0.000000	****	0	0.458000	6.430000	58.700000	6.062200	3	222
6	0.088290	12.500000	7.070000	0	0.524000	6.012000	66.600000	5.560500	5	311
7	0.144550	12.500000	****	0	0.524000	6.172000	96.100000	5.950500	5	311
8	0.211240	12.500000	7.870000	0	0.524000	5.631000	100.000000	6.082100	5	311
9	0.170040	12.500000	****	0	0.524000	6.004000	85.900000	6.592100	5	311
10	0.224890	12.500000	7.870000	0	0.524000	6.377000	94.300000	6.346700	5	311
11	0.117470	12.500000	nan	0	0.524000	6.009000	82.900000	6.226700	5	311
12	0.093780	12.500000	7.870000	0	0.524000	5.889000	39.000000	5.450900	5	311
13	0.629760	0.000000	8.140000	0	nan	5.949000	61.800000	4.707500	4	307
14	0.637960	0.000000	8.140000	0	0.538000	6.096000	84.500000	4.461900	4	307

B.2: Find possible outliers

Creating a function to highlight outliers with a yellow background in the PTRATIO column

Highlighting outlier cells in PTRATIO column

In [8]:
```
#Function to highlight cells yellow with non-numeric and any outlier values

def outliers_background(cell_value):

    highlight = 'background-color: yellow;'
    default = ''

    if type(cell_value) in [float,int]:
        if cell_value >= 25 or cell_value <=10:
            return highlight
        else:
            return default
    else:
        return highlight
```

In the above function, we have set the limits for the PTRATIO column, where the values below 10 and above 25 will be highlighted in yellow. (Felipe, 2022)

In [9]:
```
#Applying the above functions to PTRATIO

(boston_housing.iloc[:,10:11].style
 .applymap(check_nan_background)
 .applymap(float_check_background)
 .applymap(outliers_background))
```

99	18
100	Alina
101	20.900000
102	20.900000
103	20.900000
104	20.900000
105	20.900000
106	20.900000
107	20.900000
108	20.900000
109	20.900000
110	##
111	17.800000

Here we were able to identify different outliers in the PTRATIO column

(a) **Typing non-numeric value:** Identified in index 100, 110, 117

(b) **Shift in decimal place while data entry error:** Identified in index 54

(c) **Genuine case of an outlier:** Identified in index 5, 13, 20, 31, 34, 74, 144, 145

PART - C

C.1: Omission and imputation on python

Omitting the unwanted cells from PTRatio & creating a new dataframe

```
In [10]: #Omitting the below index rows from the dataframe & creating a new dataframe

         df_boston = boston_housing.drop([5,13,20,31,34,74,144,145])
```

https://www.codegrepper.com/code-examples/python/show+all+rows+in+jupyter+notebook

```
In [11]: #Setting display to all rows
         pd.set_option('display.max_rows', None)
```

https://www.machinelearningplus.com/pandas/pandas-reset-index/

```
In [12]: #Resetting the index numbers after dropping the above rows
         df_boston.reset_index(drop=True,inplace=True)
         df_boston
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.006320	18.0	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3
1	0.027310	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	0.027290	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	0.032370	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	0.069050	0.0	7.07	0	0.458	7.147	54.2		3	222	18.7
5	0.088290	12.5	7.07	0	0.524	6.012	66.6	5.5605	5	311	15.2
6	0.144550	12.5	****	0	0.524	6.172	96.1	5.9505	5	311	15.2
7	0.211240	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2
8	0.170040	12.5	****	0	0.524	6.004	85.9	6.5921	5	311	15.2
9	0.224890	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2
10	0.117470	12.5	NaN	0	0.524	6.009	82.9	6.2267	5	311	15.2

In the above image, we drop the rows that we identified as ‘genuine outliers’ and would not have an impact on the data if removed. To display all the rows we used the pandas function set_index. (Codegrepper, 2020)

Once the rows are removed we lose the index of those columns as well so we reset the index of the whole data frame to make further analysis easier. (MachineLearningPlus, 2022)

In the below image we change the value of the cell we identified as a decimal typing error to 21.1 using iloc from pandas data frame

Correcting a decimal error in PTRATIO column

```
In [13]: #The below index location cell has a typing error
         df_boston.iloc[49,10]
```

```
Out[13]: 2.11
```

```
In [14]: #Replacing the cell value to 21.1

         df_boston.iloc[49,10] = 21.1
         df_boston.iloc[49,10]
```

```
Out[14]: 21.1
```

Next, we identified all the other outliers and replaced them with NaN for them to be imputed by the median of the columns.

C.2: Substituting the missing Data by NaN

Replacing outliers with NaN

```
In [15]: #Replacing wrong values with NaN in the dataframe

df_boston.replace(['****','*****','Sara',' ','Alina','##','Adam','&&&'],np.nan,inplace=True)
df_boston
```

139	0.091780	0.0	NaN	0	0.5100	6.416	84.1	2.6463	5	296	16.6
140	0.084470	0.0	4.05	0	0.5100	5.859	68.7	2.7019	5	296	16.6
141	0.066640	0.0	4.05	0	0.5100	6.546	33.1	3.1323	5	296	16.6
142	0.070220	0.0	4.05	0	0.5100	6.020	47.2	3.5549	5	296	16.6
143	0.054250	0.0	4.05	0	0.5100	6.315	73.4	3.3175	5	296	16.6
144	0.066420	0.0	4.05	0	0.5100	6.860	74.4	2.9153	5	296	16.6
145	0.057800	0.0	2.46	0	0.4880	6.980	58.4	2.8290	3	193	17.8
146	0.065880	0.0	2.46	0	0.4880	7.765	83.3	2.7410	3	193	17.8
147	0.068880	0.0	NaN	0	0.4880	6.144	62.2	2.5979	3	193	17.8
148	0.091030	0.0	2.46	0	0.4880	7.155	92.2	2.7006	3	193	17.8
149	0.100080	0.0	2.46	0	0.4880	6.563	95.6	2.8470	3	193	17.8
150	0.083080	0.0	2.46	0	0.4880	5.604	89.8	2.9879	3	193	17.8
151	0.060470	0.0	2.46	0	0.4880	6.153	68.8	3.2797	3	193	17.8

Filing NaN values with Median

```
In [16]: #Checking a random median value
median_ptratio = df_boston['PTRATIO'].median()
median_ptratio
```

```
Out[16]: 18.55
```

```
In [17]: #Replacing all the NaN values with median
df_boston.fillna(df_boston.median(),inplace= True)
```

FINAL RESULT - CLEAN DATASET

Final DataFrame after basic cleaning

In [18]:
```
#Final DataFrame after cleaning
df_boston
```

Out[18]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.006320	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296	15.30
1	0.027310	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242	17.80
2	0.027290	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.80
3	0.032370	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.70
4	0.069050	0.0	7.07	0	0.4580	7.147	54.2	3.9769	3	222	18.70
5	0.088290	12.5	7.07	0	0.5240	6.012	66.6	5.5605	5	311	15.20
6	0.144550	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.20
7	0.211240	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.20
8	0.170040	12.5	7.87	0	0.5240	6.004	85.9	6.5921	5	311	15.20
9	0.224890	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.20
10	0.117470	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.20

REFERENCES

Felipe. (2022, May 7). *Pandas dataframe examples: Styling cells and conditional formatting*. queirozf.com. Retrieved September 19, 2022, from https://queirozf.com/entries/pandas-dataframe-examples-styling-cells-and-conditional-formatting

Frantic Fowl. (2020, February 25). *Show all rows in Jupyter notebook code example*. Show all rows in jupyter notebook Code Example. Retrieved September 19, 2022, from https://www.codegrepper.com/code-examples/python/show+all+rows+in+jupyter+notebook

Goyal, C. (2022, August 25). *How to detect and remove outliers: Outlier Detection and Removal*. Analytics Vidhya. Retrieved September 19, 2022, from https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/

MachineLearningPlus. (2022, March 8). *Pandas reset index - how to reset the index and convert the index to a column?* Machine Learning Plus. Retrieved September 19, 2022, from https://www.machinelearningplus.com/pandas/pandas-reset-index/

Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2020). *Data mining for Business Analytics: Concepts, techniques and applications in Python*. John Wiley & Sons, Inc.

The Boston house-price data. (n.d.). http://lib.stat.cmu.edu/