Table of Contents

[Outlier Introduction 1](#_Toc52204180)

[Detecting Outliers Visually 1](#_Toc52204181)

[Treating Outliers 4](#_Toc52204182)

[Detecting Outliers with the Z-Score 5](#_Toc52204183)

[Detecting Outliers by Percentile 9](#_Toc52204184)

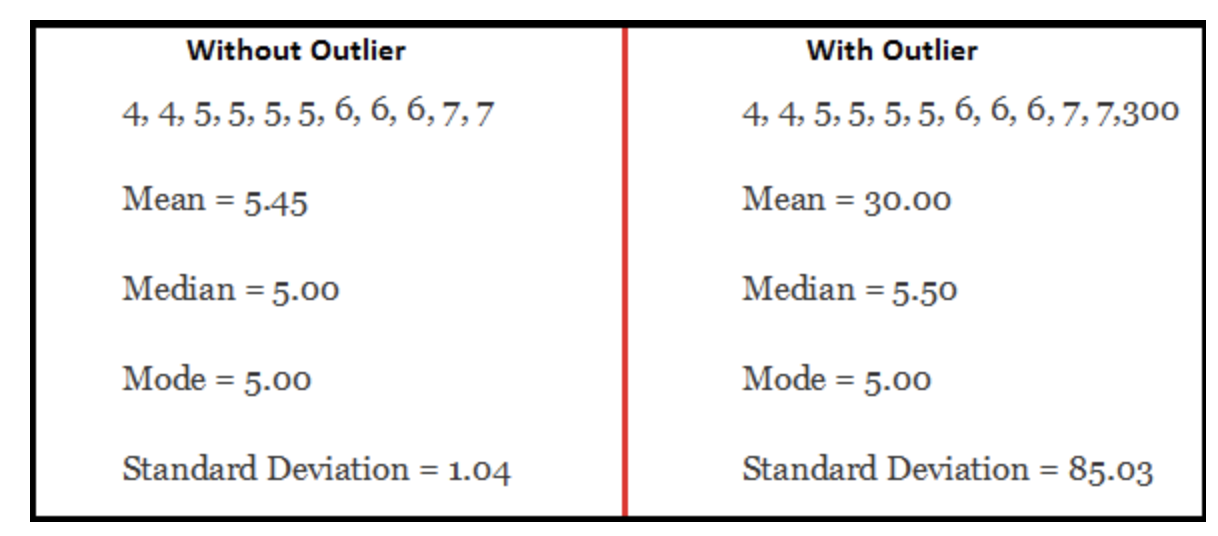
[Data Removal 11](#_Toc52204185)

[Data Clipping 12](#_Toc52204186)

## Outlier Introduction

An outlier is an extreme abnormal value. Outliers can unfairly influence the results of a prediction (see Table 1).

Table 1: The Impact of Outliers



Outliers may occur due to many reasons. Reasons could include data entry errors, measurement errors, experimental errors or intentional omissions. Respondents to a survey may even lie or hide the truth.

### Detecting Outliers Visually

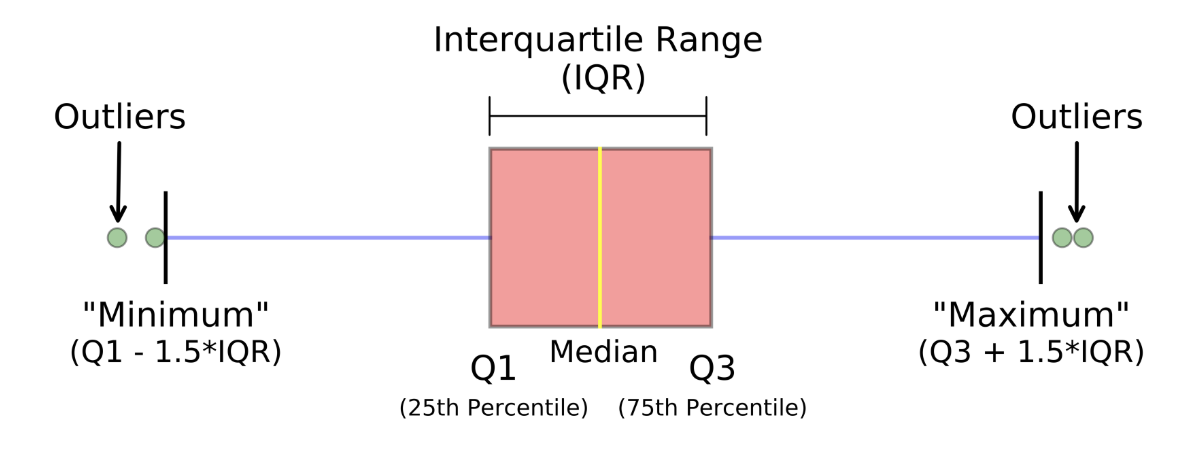
Two simple ways to detect outliers are visually are:

1. Box plots for a single variable.
2. A scatter plot of two variables.

#### Boxplots

Box plots are a common visualization which help explain the distribution of a specific variable. The box plot highlights the skew, median and outliers (extreme values). The box plot in Figure 1 shows where the median (middle) value is as well as the concentration of samples within the first to third quartiles. The range between the first quartile and the third quartile is called the interquartile range (IQR). Whiskers usually extend from both sides of the IQR. Each whisker is 1.5\*IQR in length. Anything outside the whisker boundaries is considered to be an outlier.

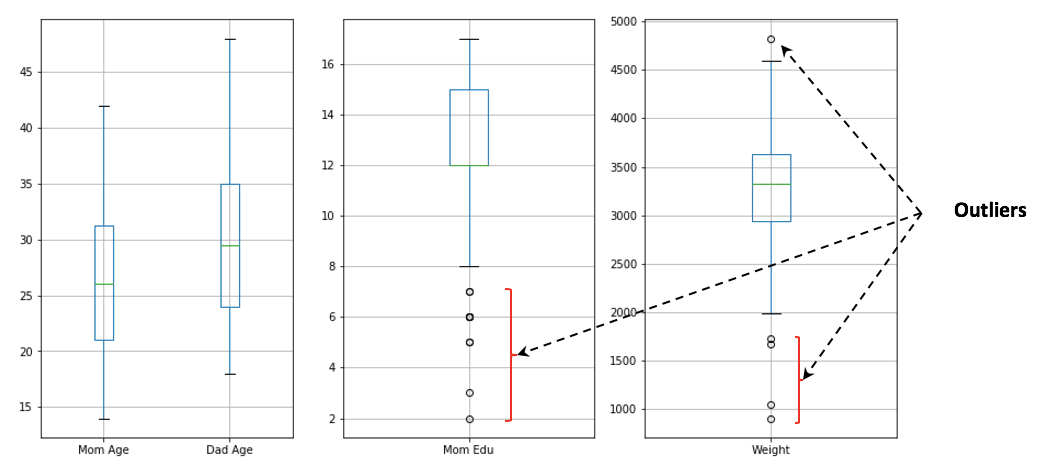
Figure 1: Box Plot Features



Example 1: Box Plots

This example shows how to draw box plots for the birth health statistics study (refer to Figure 2). A quick view of the age box plots tells us the age differences between Mothers and Fathers is around 3 to 4 years and are slightly older. The blue line at the center indicates their median ages. The box plot for education tells us that the median education level for Mother’s is grade 12 but 50% of Mother’s had education ranging between grade 12 to 3 years of post-secondary. A larger skew exists for lower education and several outliers exist for very little education. The birthweight median exists at around 3300 grams and it mostly ranges between 2900 grams and 3600 grams.

Figure 2: Box and Scatter Plots Showing Birth Health Outliers

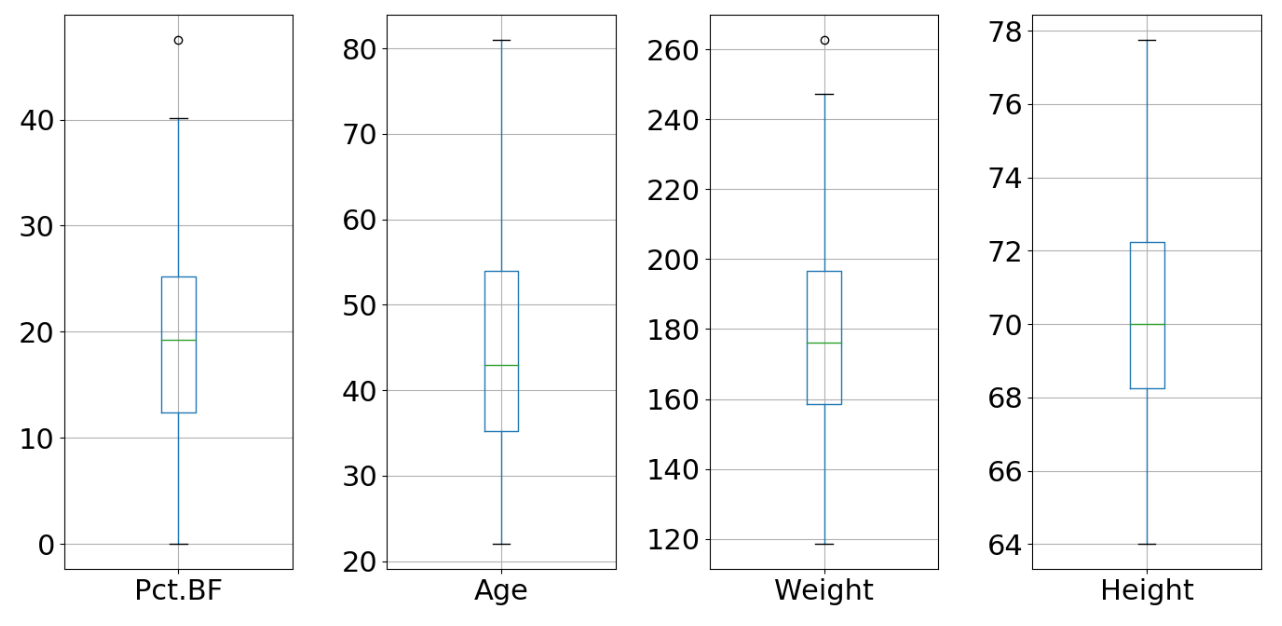


The code that is highlighted in green shows how to draw the Mother and Father’s age box plots together since they share the same unit scale. The other two lines of code which are highlighted in yellow show how to plot the Mother’s education level and the weight of the baby.

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  # Import data into a DataFrame.  path = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/babysamp-98.txt"  df = pd.read\_csv(path, skiprows=1,                     sep='\t',                     names=('MomAge', 'DadAge', 'MomEduc', 'MomMarital', 'numlive',                            "dobmm", 'gestation', 'sex', 'weight', 'prenatalstart',                            'orig.id', 'preemie'))  # Rename the columns so they are more reader-friendly.  df = df.rename({'MomAge': 'Mom Age', 'DadAge':'Dad Age',                  'MomEduc':'Mom Edu', 'weight':'Weight'}, axis=1)  # new method  # This line allows us to set the figure size supposedly in inches.  # When rendered in the IDE the output often does not translate to inches.  plt.subplots(nrows=1, ncols=3,  figsize=(14,7))  plt.subplot(1, 3, 1) # Specfies total rows, columns and image #                       # where images are drawn clockwise.  #plt.xlabel("Mom Age")  plt.xticks([], ())  boxplot = df.boxplot(column=['Mom Age', 'Dad Age'])  plt.subplot(1, 3, 2) # Specfies total rows, columns and image #                       # where images are drawn clockwise.  boxplot = df.boxplot(column=['Mom Edu'])  plt.subplot(1, 3, 3) # Specfies total rows, columns and image #                       # where images are drawn clockwise.  boxplot = df.boxplot(column=['Weight'])  plt.show() |

The next series of questions uses Figure 3.

Figure 3: Boxplots for Percent Body Fat, Age, Weight, and Height



Exercise 1 (4 marks0

1. Which diagrams show the outliers?

|  |
| --- |
| Bct.BF , Weight |

1. Exactly which percentage of the sample data is represented by the area in the rectangular section of the box plot?

|  |
| --- |
| 50% |

1. What point does the blue line in the box of the box plot represent?

|  |
| --- |
| Represented mean |

1. How much bigger is the length of the whisker than the box?

|  |
| --- |
| 1.5\*IQR |

Exercise 2 (2 marks)

Draw the plot that is shown in Figure 3. You can start with this code:

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  # Import data into a DataFrame.  path = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/bodyfat.txt"  df = pd.read\_csv(path, skiprows=1,  sep='\t',  names=('Density', 'Pct.BF', 'Age', 'Weight', 'Height',  'Neck', 'Chest', 'Abdomen', 'Waist', 'Hip', 'Thigh',  'Ankle', 'Knee', 'Bicep', 'Forearm', 'Wrist'))  plt.rcParams.update({'font.size': 22})  df.head()  plt.subplots(nrows=1, ncols=4, figsize=(14,7)) |

Show your code here:

|  |
| --- |
| import pandas as pd import matplotlib.pyplot as plt  # Import data into a DataFrame. path = "/Users/hyerimshin/PredictiveAnalytics/DataSet/bodyfat.txt" df = pd.read\_csv(path, skiprows=1,  sep='\t',  names=('Density', 'Pct.BF', 'Age', 'Weight', 'Height',  'Neck', 'Chest', 'Abdomen', 'Waist', 'Hip', 'Thigh',  'Ankle', 'Knee', 'Bicep', 'Forearm', 'Wrist'))   # Rename the columns so they are more reader-friendly. df = df.rename({'Pct.BF': 'Pct.BF', 'Age': 'Age',  'weight': 'Weight', 'Height': 'Height'}, axis=1) # new method  # This line allows us to set the figure size supposedly in inches. # When rendered in the IDE the output often does not translate to inches. plt.rcParams.update({'font.size': 22}) df.head() plt.subplots(nrows=1, ncols=4, figsize=(14, 7))   plt.subplot(1, 4, 1) # Specfies total rows, columns and image # # where images are drawn clockwise. # plt.xlabel("Mom Age") boxplot = df.boxplot(column=['Pct.BF'])   plt.subplot(1, 4, 2) # Specfies total rows, columns and image # # where images are drawn clockwise. # plt.xlabel("Mom Age") boxplot = df.boxplot(column=['Age'])  plt.subplot(1, 4, 3) # Specfies total rows, columns and image # # where images are drawn clockwise. boxplot = df.boxplot(column=['Weight'])  plt.subplot(1, 4, 4) # Specfies total rows, columns and image # # where images are drawn clockwise. boxplot = df.boxplot(column=['Height'])  plt.show() |

## Treating Outliers

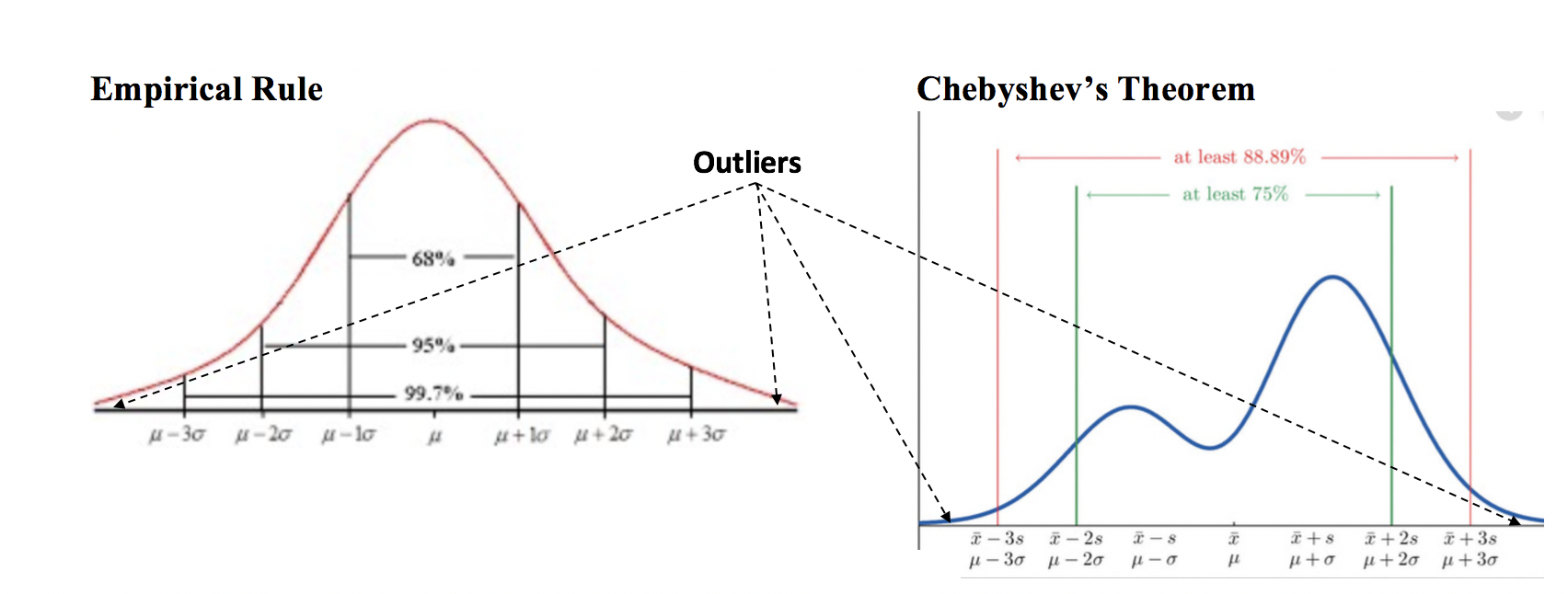
A common problem when preparing data for modelling is how to fix outliers without introducing bias. For example, most humans will almost never be 115 years of age or older. WNBA teams almost always play 36 games per season or less. A contradiction to these limits suggests either an error occurred or further investigation may be required if it is possible. Outliers may also skew our models so we may need to address them to ensure our models remain valid for regular cases.

Box plots and scatter plots provide a quick visual way to identify if outliers exist. However, visual plots do not precisely identify which samples contain the outliers. Several statistical methods can help to quickly find sample outliers.

## Detecting Outliers with the Z-Score

A large Z-score can be used to detect outliers (refer to Figure 4).

Figure 4: Normal Distributions Explained by the Empirical Rule and Any Distribution Explained by Chebyshev’s Theorem



#### Detecting Extreme Z-Scores for Multiple Predictor Variables

This code finds all indexes in a data frame that contains variables which have a Z score greater than 3 or less than -3. For a normal distribution, results obtained fall outside the standard 99.7% range.

# Get Z score for both tails based on subset of df that contains numeric columns.

z = np.abs(stats.zscore(df[['numericColumnA','numericColumnB']]))

print(z)

THRESHOLD = 3 # Determines if sample attribute value > 97.87% or < 0.13% range.

print(np.where(z > THRESHOLD))

Example 2: Detecting Outliers with the Z-Score

This code searches the USA Housing data frame to obtain indexes where sample attribute values exceed 3 standard deviations from the mean.

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  import statsmodels.api as sm  PATH = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/"  CSV\_DATA = "USA\_Housing.csv"  dataset = pd.read\_csv(PATH + CSV\_DATA,  skiprows=1, # Don't include header row as part of data.  encoding = "ISO-8859-1", sep=',',  names=('Avg. Area Income','Avg. Area House Age',  'Avg. Area Number of Rooms','Avg. Area Number of Bedrooms', "Area Population",  'Price', "Address"))  # Show all columns.  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  dfSub = dataset[['Avg. Area Income','Avg. Area House Age',  'Avg. Area Number of Rooms','Avg. Area Number of Bedrooms', "Area Population",  'Price']]  from scipy import stats  import numpy as np  z = np.abs(stats.zscore(dfSub))  print(z)  THRESHOLD = 3  print(np.where(z > THRESHOLD))  print(dfSub.loc[39][[0]]) |

Our output when printing *z* shows values for an array that contains row indexes and an array that contains column indexes:

|  |
| --- |
| (array([ 39, 228, 256, 263, 314, 353, 411, 465, 496, 693, 693,  924, 962, 1074, 1091, 1234, 1248, 1271, 1459, 1459, 1530, 1536,  1595, 1628, 1661, 1734, 1757, 1777, 1799, 1799, 1891, 2066, 2092,  2173, 2465, 2534, 2538, 2719, 2719, 2756, 2771, 2829, 2839, 2898,  3069, 3134, 3138, 3212, 3336, 3541, 3806, 3855, 3989, 3991, 4087,  4488, 4491, 4565, 4716, 4803, 4855]), array([0, 4, 5, 5, 4, 4, 0, 5, 2, 0, 5, 5, 0, 1, 1, 4, 5, 5, 0, 5, 4, 2,  4, 1, 5, 0, 2, 1, 2, 5, 0, 2, 0, 4, 1, 4, 5, 0, 5, 4, 2, 4, 4, 1,  0, 4, 1, 5, 2, 0, 2, 2, 1, 4, 0, 1, 4, 1, 4, 4, 0])) |

Later in our output the first outlier from the sample is printed using the first row and column from the outlier output with the instruction:

print(dataset.loc[39][[0]])

|  |
| --- |
| Avg. Area Income 17796.6  Name: 39, dtype: object |

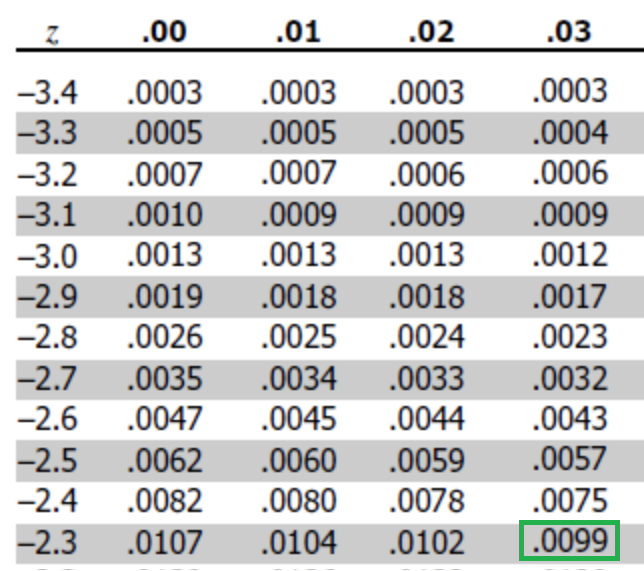
At this point, we may conduct an analysis to better understand how this extreme value appeared.

Exercise 3 (5 marks)

In this example, create a threshold for the Z-score of 2.33 which has an alpha region of 1% for each tail. According to the Z-score table, effectively you will be finding values in the within the 1st and 99th percentile. Run this test for all columns of the babysamp-98.txt file except do not include ﻿'sex', 'weight', 'prenatalstart', 'orig.id', 'preemie' columns. The code from Example 1 can help you get started. You can use the following subset of the original data frame:

﻿dfSub = df[['MomAge', 'MomMarital', 'numlive', "dobmm", 'gestation']]

Figure 5: Finding Cumulative Probability in the Z-score Table



Show the arrays of highlighted rows and columns here:

|  |
| --- |
|  |

Which column names are represented in the numeric output from the last step?

|  |
| --- |
| A picture containing text, device, meter  Description automatically generated |

Print out one value from each column. Show your print statements here:

|  |
| --- |
| print(df.loc[48][[2]]) |

Show your output from the print statements above here:

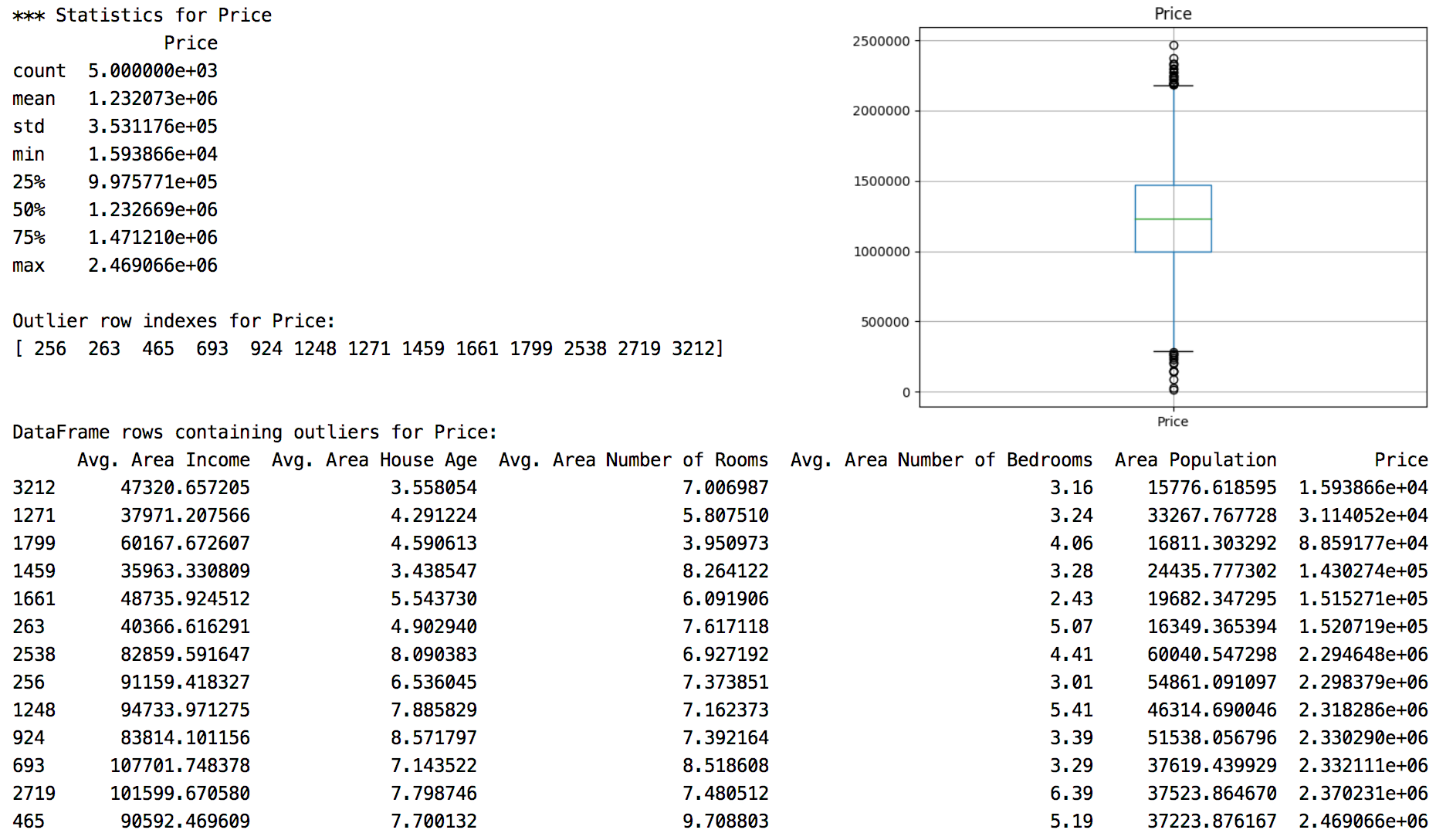
|  |
| --- |
| Graphical user interface, text  Description automatically generated |

Example 3: Identifying Outlier rows and Analysis for one Variable

This example shows how to conduct a univariate analysis to obtain all row indexes where outlier values of that variable exist. For this case, Price is analyzed and Price outlier values are identified.

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  import statsmodels.api as sm  import matplotlib.pyplot as plt  from scipy import stats  import numpy as np  PATH = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/"  CSV\_DATA = "USA\_Housing.csv"  dataset = pd.read\_csv(PATH + CSV\_DATA,  skiprows=1, # Don't include header row as part of data.  encoding="ISO-8859-1", sep=',',  names=('Avg. Area Income', 'Avg. Area House Age',  'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', "Area Population",  'Price', "Address"))  # Show all columns.  pd.set\_option('display.max\_columns', None)  pd.set\_option('display.width', 1000)  print(dataset.head())  # ------------------------------------------------------------------  # Show statistics, boxplot, extreme values and returns DataFrame  # row indexes where outliers exist.  # ------------------------------------------------------------------  def viewAndGetOutliers(df, colName, threshold, plt):  # Show basic statistics.  dfSub = df[[colName]]  print("\*\*\* Statistics for " + colName)  print(dfSub.describe())  # Show boxplot.  dfSub.boxplot(column=[colName])  plt.title(colName)  plt.show()  # Note this is absolute 'abs' so it gets both high and low values.  z = np.abs(stats.zscore(dfSub))  rowColumnArray = np.where(z > threshold)  rowIndices = rowColumnArray[0]  # Show outlier rows.  print("\nOutlier row indexes for " + colName + ":")  print(rowIndices)  print("")  # Show filtered and sorted DataFrame with outliers.  dfSub = df.iloc[rowIndices]  dfSorted = dfSub.sort\_values([colName], ascending=[True])  print("\nDataFrame rows containing outliers for " + colName + ":")  print(dfSorted)  return rowIndices  THRESHOLD\_Z = 3  priceOutlierRows = viewAndGetOutliers(dataset, 'Price', THRESHOLD\_Z, plt) |

The output shows basic statistics for our *Price* variable. The box plot quickly shows that most home prices are between $1,000,000 and $1,500,000. Several outliers exist outside the interquartile range. The outliers beyond the whiskers are identified by row and they are detailed in the data frame.



Exercise 4 (2 marks)

Run the code for Example 3 again but this time do it for 'Avg. Area Income'. Show a screen show of the output. Identify the row indexes that contain outlier values for ‘Avg. Area Income’ here:

|  |
| --- |
| Text  Description automatically generated |
| Chart, box and whisker chart  Description automatically generated |

Exercise 5 (5 marks)

Run the code for the viewAndGetOutliers() function again but for the ‘weight’ column in the ﻿babysamp-98.txt content. Use a threshold of 2.33 (<1% and >99%). Pass in the original dataframe so all columns are sent to the function. Code from Example 1 can help you start. Show your code here:

|  |
| --- |
| import pandas as pd from sklearn.model\_selection import train\_test\_split import statsmodels.api as sm import matplotlib.pyplot as plt from scipy import stats import numpy as np  path = "/Users/hyerimshin/PredictiveAnalytics/DataSet/bodyfat.txt" df = pd.read\_csv(path, skiprows=1,  sep='\t',  names=('MomAge', 'DadAge', 'MomEduc', 'MomMarital', 'numlive',  "dobmm", 'gestation', 'sex', 'weight', 'prenatalstart',  'orig.id', 'preemie'))  # Show all columns. pd.set\_option('display.max\_columns', None) pd.set\_option('display.width', 1000) print(df.head())   # ------------------------------------------------------------------ # Show statistics, boxplot, extreme values and returns DataFrame # row indexes where outliers exist. # ------------------------------------------------------------------ def viewAndGetOutliers(df, colName, threshold, plt):  # Show basic statistics.  dfSub = df[[colName]]  print("\*\*\* Statistics for " + colName)  print(dfSub.describe())   # Show boxplot.  dfSub.boxplot(column=[colName])  plt.title(colName)  plt.show()   # Note this is absolute 'abs' so it gets both high and low values.  z = np.abs(stats.zscore(dfSub))  rowColumnArray = np.where(z > threshold)  rowIndices = rowColumnArray[0]   # Show outlier rows.  print("\nOutlier row indexes for " + colName + ":")  print(rowIndices)  print("")   # Show filtered and sorted DataFrame with outliers.  dfSub = df.iloc[rowIndices]  dfSorted = dfSub.sort\_values([colName], ascending=[True])  print("\nDataFrame rows containing outliers for " + colName + ":")  print(dfSorted)  return rowIndices   THRESHOLD\_Z = 2.33 priceOutlierRows = viewAndGetOutliers(df, 'weight', THRESHOLD\_Z, plt)  print(priceOutlierRows) |

Show your output here:

|  |
| --- |
| **Graphical user interface  Description automatically generated with medium confidenceChart, box and whisker chart  Description automatically generated** |

Do you notice any pattern for other attributes with the low or high weights?

|  |
| --- |
| **Gestation mean is 38, std is 2.5 and highest gestation of the low weights(33) is 2 std away. That means low gestation have the low weight outliers too.** |

## Detecting Outliers by Percentile

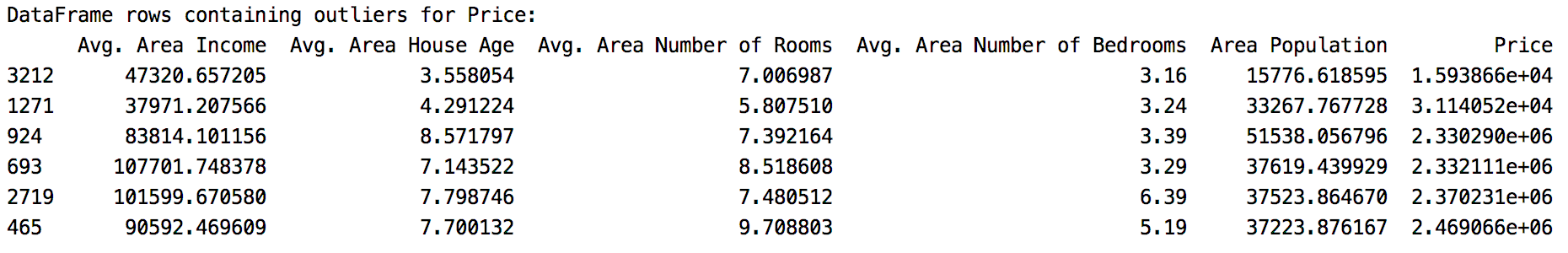
Another method for finding outliers involves filtering on the start and ending percentiles.

Example 4: Finding Outliers by Percentile

This example finds the largest and smallest values in the distribution. Since our sample is 5000 we need very small percentile markers.

|  |
| --- |
| # ------------------------------------------------------------------  # Show statistics, boxplot, extreme values and returns DataFrame  # row indexes where outliers exist outside an upper and lower percentile.  # ------------------------------------------------------------------  def viewAndGetOutliersByPercentile(df, colName, lowerP, upperP, plt):  # Show basic statistics.  dfSub = df[[colName]]  print("\*\*\* Statistics for " + colName)  print(dfSub.describe())  # Show boxplot.  dfSub.boxplot(column=[colName])  plt.title(colName)  plt.show()  # Get upper and lower perctiles and filter with them.  up = df[colName].quantile(upperP)  lp = df[colName].quantile(lowerP)  outlierDf = df[(df[colName] < lp) | (df[colName] > up)]  # Show filtered and sorted DataFrame with outliers.  dfSorted = outlierDf.sort\_values([colName], ascending=[True])  print("\nDataFrame rows containing outliers for " + colName + ":")  print(dfSorted)  return lp, up # return lower and upper percentiles  LOWER\_PERCENTILE = 0.00025  UPPER\_PERCENTILE = 0.99925  lp, up = viewAndGetOutliersByPercentile(dataset, 'Price',  LOWER\_PERCENTILE, UPPER\_PERCENTILE, plt) |

When running the example, the samples with the bottom two prices and top four prices are extracted.



Exercise 6 (3 marks)

Run the viewAndGetOutliersByPercentile() function again but for the ﻿babysamp-98.txt file. Use 2% and 98% as percentiles. Do this for the ‘gestation’ column. Show your code here:

Show your output here:

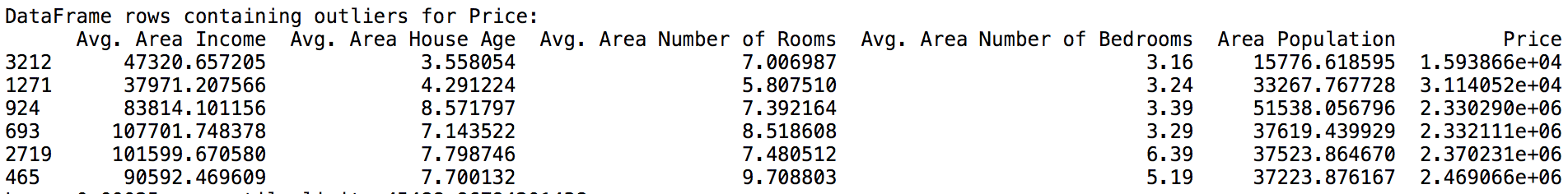
|  |
| --- |
| Text  Description automatically generated |

## Data Removal

Generally, deleting data is never a good idea. However, you can eliminate data from your current training set when samples are believed to not be representative of the population. A problem with removing data from the training sample though is the model will be unable to handle outlier data from new samples. In such cases, you may choose to **develop two models** where one model works with the usual sample inputs and the other model makes predictions with outlier inputs.

Example 5: Removing Outliers

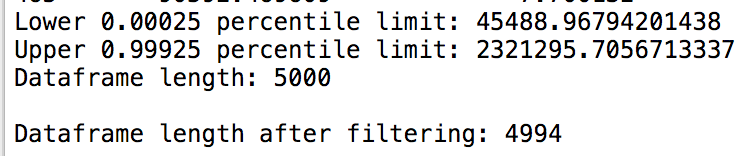
This example shows how to remove outlier data from a data frame. For this, example we will start with Example 4. The output initially displays data frame rows where *Price* exists in the ﻿ 0.00025 and 0.99925 percentiles.



We can create a new data frame which filters on our percentile limits. To do this, add this code to the end of Example 4:

|  |
| --- |
| LOWER\_PERCENTILE = 0.00025  UPPER\_PERCENTILE = 0.99925  lp, up = viewAndGetOutliersByPercentile(dataset  [['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',  'Avg. Area Number of Bedrooms', "Area Population", 'Price']], 'Price',  LOWER\_PERCENTILE, UPPER\_PERCENTILE, plt)  print("Lower " + str(LOWER\_PERCENTILE) + " percentile limit: " + str(lp))  print("Upper " + str(UPPER\_PERCENTILE) + " percentile limit: " + str(up))  print("Dataframe length: " + str(len(dataset)))  df\_filtered = dataset[(dataset["Price"] >lp) & (dataset["Price"] < up)]  print("\nDataframe length after filtering: " + str(len(df\_filtered))) |

When running the code after these changes, the new output contains all of the original data except the rows where the outlier *Prices* exist.



**Note:** If you remove data from your training set, you must ensure that your model can handle the test data or new data. You may choose to create a new model for handling the outlier data.

## Data Clipping

An alternative to removing data and an easy transformation involves clipping outliers to the desired boundary.

Example 6: Clipping

This example extends Example 3. Example 3 identified low and high price outliers in the USA Housing data set where -3 The outlier prices are shown in Table 2.

Table 2: Low and High Price Outliers for USA Housing where Abs(Z-Value)>3

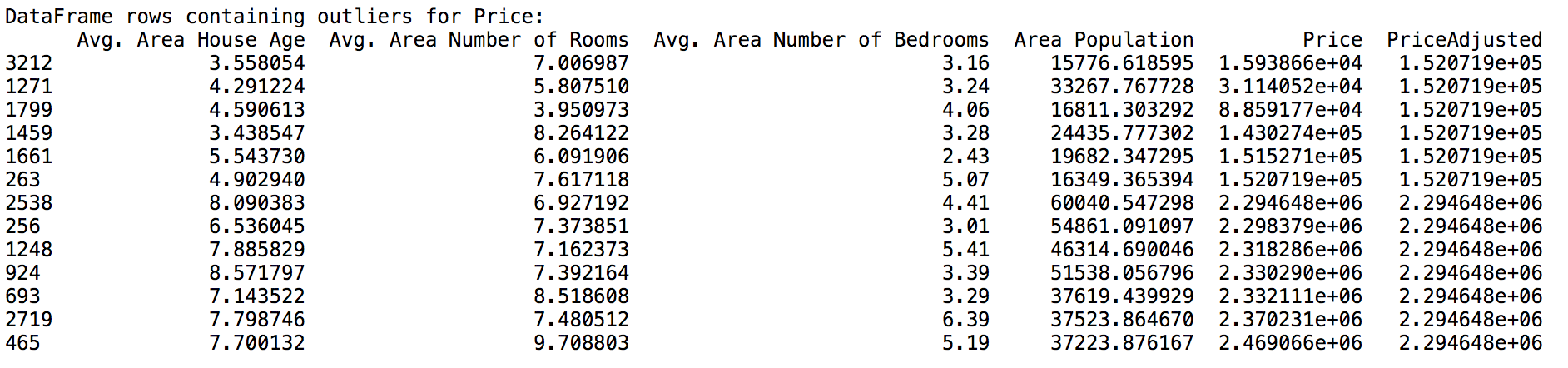
|  |  |
| --- | --- |
| **Low Price Outliers**  1.593866e+04  3.114052e+04  8.859177e+04  1.430274e+05  1.515271e+05  1.520719e+05 | **High Price Outliers**  2.294648e+06  2.298379e+06  2.318286e+06  2.330290e+06  2.332111e+06  2.370231e+06  2.469066e+06 |

If the outlier prices are negatively impacting our model we could clip the prices after choosing acceptable upper and lower bounds. After inspecting the outliers from Example 3 we may decide to restrict all prices to a range between 1.520719e+05 and 2.294648e+06.

To enforce this range, add this code to the code in Example 3. Here, the clip() function trims the data so outliers conform to the specified lower and upper bounds.

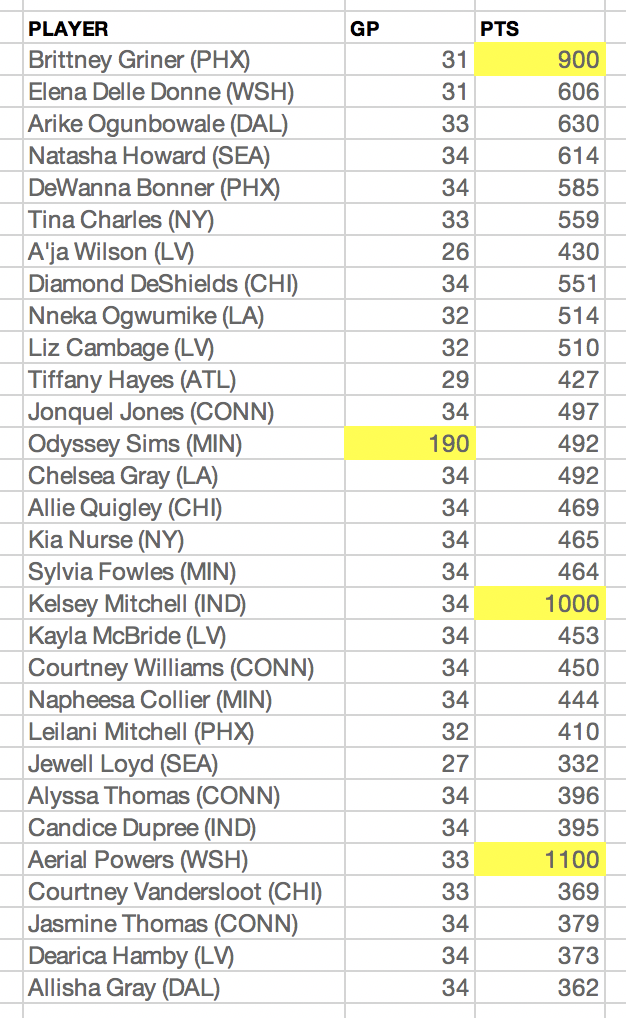
|  |
| --- |
| # Trim prices to fixed boundararies.  dfAdjusted = dataset['Price'].clip(1.520719e+05, 2.294648e+06)  # Add adjusted price to the current data frame. Avoid deleting original data.  dataset['PriceAdjusted'] = dfAdjusted  # Show reviesed results.  priceOutlierRows = viewAndGetOutliers(dataset[[  'Avg. Area House Age','Avg. Area Number of Rooms',  'Avg. Area Number of Bedrooms', "Area Population",'Price', 'PriceAdjusted']],  'Price', THRESHOLD\_Z, plt) |

After running the revised code, the new output confirms that the outliers were trimmed:



Exercise 7 (4 marks)

You are preparing some data models about basketball player performance. You have a small amount of data from last year from your project sponsor but you expect to receive more data from previous years. The data set is expected to grow by tens of thousands of rows. You will not have time to individually check each record and the model will need to make predictions for thousands of players in the future. A manager from the league speaks with you and notices that the statistics in the current dataset are inaccurate. The manager points out that there are only 36 games per season and the highest point count that anyone has ever earned is 860 in one season. You decide the best approach to handle the errors with the time that you have is to clip the games to 36 and limit the number of points to 860. These are the statistics that you currently have and the errors are highlighted in yellow (This sample, **wnba.csv**, is located in the datasets folder)



Here is some starter code:

|  |
| --- |
| import pandas as pd  PATH = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/"  CSV\_DATA = "wnba.csv"  df = pd.read\_csv(PATH + CSV\_DATA,  skiprows=1, # Don't include header row as part of data.  encoding="ISO-8859-1", sep=',', names=('PLAYER', 'GP', 'PTS'))  print(df.head(30)) |

Show your code after enabling clipping here:

|  |
| --- |
| df = pd.read\_csv(PATH + CSV\_DATA,  skiprows=1, # Don't include header row as part of data.  encoding="ISO-8859-1", sep=',', names=('PLAYER', 'GP', 'PTS')) dfAdjusted = df['GP'].clip(0, 36) df['GPAdjusted'] = dfAdjusted dfAdjusted = df['PTS'].clip(0, 860) df['PTSAdjusted'] = dfAdjusted print(df.head(30)) |

Show a screenshot of your output after the changes here:

|  |
| --- |
| Calendar  Description automatically generated |

Exercise 8 (3 marks)

Later, instead of clipping, you decide you want to eliminate the outliers from your data set where 36 games are exceeded and when 860 points or more reported. Starting with the initial code from Exercise 7 create a new data frame which only contains the non-outlier data. Show your code here:

|  |
| --- |
| df = pd.read\_csv(PATH + CSV\_DATA,  skiprows=1, # Don't include header row as part of data.  encoding="ISO-8859-1", sep=',', names=('PLAYER', 'GP', 'PTS'))  df\_filtered = df[(df["GP"] <= 36) & (df["PTS"] <= 860)] print("\nDataframe length after filtering: " + str(len(df\_filtered))) print(df\_filtered.head(30)) |