Outliers

February 15, 2024

1 Finding Outliers

Outliers are extreme values in a dataset. Are these true values or the result of an error?

Below is a sampling of methods to find outliers

I normally use boxplot.

[130]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

1.1 Load Diabetes dataset

[132]:
```
# Convert sklearn diabetes dataset to dataframe
def sklearn_to_df(sklearn_dataset):
    df = pd.DataFrame(sklearn_dataset.data, columns=sklearn_dataset.
 ↪feature_names)
    df['target'] = pd.Series(sklearn_dataset.target)
    return df

df_diabetes = sklearn_to_df(datasets.load_diabetes())
```

[133]:
```
# Diabetes dataset dimensions
# 442 rows and 11 columns

df_diabetes.shape
```

[133]:
```
(442, 11)
```

1.2 Method 1: Sorting using SQL

- Oracle LIVE SQL
- Sort column in ascending order.
- Look at first 10 rows.

There are no outliers.

STORE_NAME	TOTAL_SALES
S�o Paulo	3148.22
Tokyo	3263.82
Buenos Aires	3495.72
New York City	3582.33
Perth	3707.49
Chicago	3721.28
Berlin	3791.11
Bejing	3849.33
Johannesburg	3870.98
Utrecht	3934.56

- Oracle LIVE SQL
- Sort column in descending order.
- Look at first 10 rows.

The first two values are outliers.

STORE_NAME	TOTAL_SALES
-	299889.62
Online	211107.78
New Dehli	5291.76
Sydney	4605.82
Tel Aviv	4457.19
Mumbai	4443.81
London	4427.49
San Francisco	4380.91
Seattle	4294.2
Madrid	4187.88

1.3 Method 2: Boxplot using Python

Visualize outliers using a boxplot

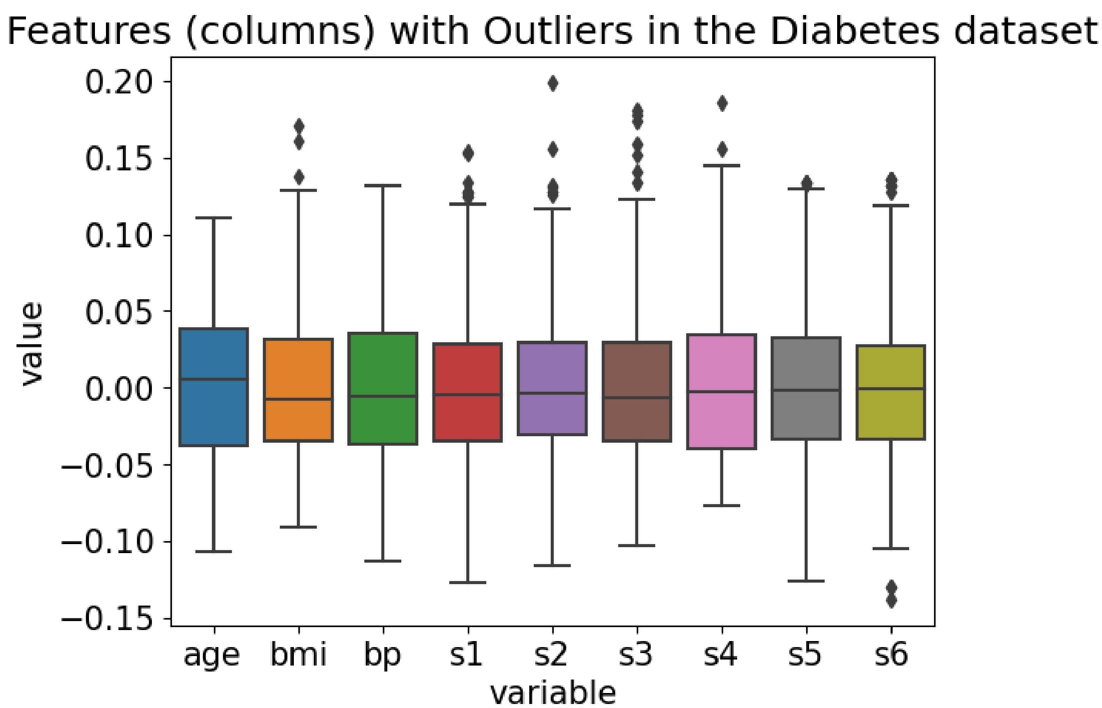
The markings above and below the box whiskers are outliers.

[140]:
```
#  Assign column names to a variable
cols = ['age', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

[141]:
```
# Generate a boxplot for each column name in the cols variable.

df_diabetes_g = df_diabetes[cols]
sns.boxplot(x="variable", y = "value", data=pd.melt(df_diabetes_g))
plt.title("Features (columns) with Outliers in the Diabetes dataset")
```

[141]:
```
Text(0.5, 1.0, 'Features (columns) with Outliers in the Diabetes dataset')
```



1.4 Method 3: Calculate Interquantile Range (IQR) using Python

A measure of the spread of the data

1.4.1 Use the describe() method to get the quantiles for each column

[144]:
```
# Descriptive statistics for the Diabetes dataset
# First quantile (25%)
# Second quantile (50%)
# Third quantile (75%)

q = df_diabetes.describe().transpose()
q[['25%','50%','75%']]
```

[144]:

	25%	50%	75%
age	-0.037299	0.005383	0.038076
sex	-0.044642	-0.044642	0.050680
bmi	-0.034229	-0.007284	0.031248
bp	-0.036656	-0.005670	0.035644
s1	-0.034248	-0.004321	0.028358
s2	-0.030358	-0.003819	0.029844
s3	-0.035117	-0.006584	0.029312
s4	-0.039493	-0.002592	0.034309
s5	-0.033246	-0.001947	0.032432
s6	-0.033179	-0.001078	0.027917
target	87.000000	140.500000	211.500000

1.4.2 Calculate the IQR for each column using the 25% and 75% quantiles

[146]:
```
q['IQR'] = q['75%'] - q['25%']
q[['25%', '75%', 'IQR']]
```

[146]:

	25%	75%	IQR
age	-0.037299	0.038076	0.075375
sex	-0.044642	0.050680	0.095322
bmi	-0.034229	0.031248	0.065477
bp	-0.036656	0.035644	0.072300
s1	-0.034248	0.028358	0.062606
s2	-0.030358	0.029844	0.060203
s3	-0.035117	0.029312	0.064429
s4	-0.039493	0.034309	0.073802
s5	-0.033246	0.032432	0.065678
s6	-0.033179	0.027917	0.061096
target	87.000000	211.500000	124.500000

1.4.3 Calculate the Upper and Lower limits for each column

[148]:
```
q['Upper'] = q['75%'] + (1.5 * q['IQR'])
q['Lower'] = q['25%'] - (1.5 * q['IQR'])

q[['25%', '75%', 'IQR','Upper', 'Lower']]
```

[148]:

	25%	75%	IQR	Upper	Lower
age	-0.037299	0.038076	0.075375	0.151139	-0.150362
sex	-0.044642	0.050680	0.095322	0.193663	-0.187624
bmi	-0.034229	0.031248	0.065477	0.129464	-0.132445
bp	-0.036656	0.035644	0.072300	0.144094	-0.145106
s1	-0.034248	0.028358	0.062606	0.122267	-0.128157
s2	-0.030358	0.029844	0.060203	0.120149	-0.120663
s3	-0.035117	0.029312	0.064429	0.125954	-0.131760

```
s4      -0.039493    0.034309    0.073802    0.145012  -0.150197
s5      -0.033246    0.032432    0.065678    0.130949  -0.131762
s6      -0.033179    0.027917    0.061096    0.119561  -0.124823
target  87.000000  211.500000  124.500000  398.250000 -99.750000
```

1.4.4 List the outliers for a given column

The outliers are identified as having a value greater than the upper or less than the lower

[150]:
```python
# Upper Outliers for bmi
df_diabetes.loc[(df_diabetes['bmi']>0.13), 'bmi']
```

[150]:
```
256    0.160855
366    0.137143
367    0.170555
Name: bmi, dtype: float64
```

[151]:
```python
# Upper Outliers for s1
df_diabetes.loc[(df_diabetes['s1']>0.12), 's1']
```

[151]:
```
123    0.152538
161    0.133274
202    0.126395
230    0.153914
248    0.127771
276    0.125019
287    0.125019
346    0.127771
Name: s1, dtype: float64
```

[152]:
```python
# Upper Outliers for s1
df_diabetes.loc[(df_diabetes['s1']>0.12), 's1']
```

[152]:
```
123    0.152538
161    0.133274
202    0.126395
230    0.153914
248    0.127771
276    0.125019
287    0.125019
346    0.127771
Name: s1, dtype: float64
```

[153]:
```python
# Upper Outliers for s3
df_diabetes.loc[(df_diabetes['s3'] > 0.12), 's3']
```

[153]:
```
35     0.133318
58     0.181179
260    0.151726
261    0.177497
266    0.122273
269    0.159089
286    0.140681
433    0.122273
441    0.173816
Name: s3, dtype: float64
```

[154]:
```python
# Upper Outliers for s6
df_diabetes.loc[(df_diabetes['s6']< -0.125),'s6']
```

[154]:
```
84    -0.129483
245   -0.129483
406   -0.137767
Name: s6, dtype: float64
```

[155]:
```python
# Lower Outliers for s6
df_diabetes.loc[(df_diabetes['s6'] < -0.12), 's6']
```

[155]:
```
84    -0.129483
245   -0.129483
406   -0.137767
Name: s6, dtype: float64
```

1.5 Method 4: Z-score using Python

z-score is measure of how many standard deviations a value is from the mean.

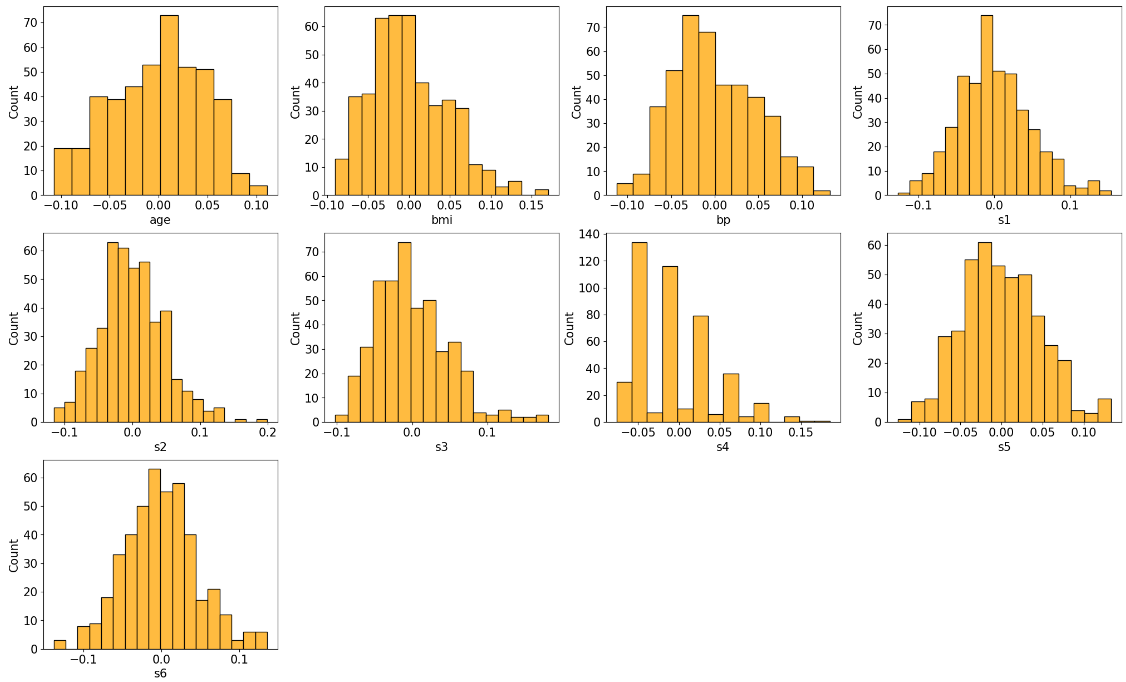
Per investofpedia.com, "a z-score can be used to determine how far a stock's return differs from it's average return. Z-scores are measures of an instrument's variability and can be used by traders to help determine volatiity."

[157]:
```python
import statsmodels.api as sm
```

Visual inspection shows most of the values for each column look normally distributed. A normal distribution is a requirement for z-scores. Histograms can also highlight outliers. Look at the chart for s4. The right-side shows outliers.

[190]:
```python
plt.figure(figsize=(25,15))
plt.rc('font', size=15)

for indx, colnm in enumerate(cols):
    plt.subplot(3,4, indx+1)
    sns.histplot(df_diabetes[colnm], color='orange')
```



1.5.1 Calculate z-scores for each column

[161]:
```
import scipy.stats as stats
df_zscore = df_diabetes.select_dtypes(include='number').apply(stats.zscore)
df_zscore = df_zscore.drop(['age','sex','target'], axis=1)
df_zscore
```

[161]:
```
          bmi        bp        s1        s2        s3        s4        s5  \
0    1.297088  0.459841 -0.929746 -0.732065 -0.912451 -0.054499  0.418531
1   -1.082180 -0.553505 -0.177624 -0.402886  1.564414 -0.830301 -1.436589
2    0.934533 -0.119214 -0.958674 -0.718897 -0.680245 -0.054499  0.060156
3   -0.243771 -0.770650  0.256292  0.525397 -0.757647  0.721302  0.476983
4   -0.764944  0.459841  0.082726  0.327890  0.171178 -0.054499 -0.672502
..        …         …         …         …         …         …         …
437  0.413360  1.256040 -0.119769 -0.053957 -0.602843 -0.054499  0.655787
438 -0.334410 -1.422086  1.037341  1.664355 -0.602843  0.721302 -0.380819
439 -0.334410  0.363573 -0.785107 -0.290965 -0.525441 -0.232934 -0.985649
440  0.821235  0.025550  0.343075  0.321306 -0.602843  0.558384  0.936163
441 -1.535374 -1.711613  1.760535  0.584649  3.654268 -0.830301 -0.088752

           s6
0   -0.370989
1   -1.938479
2   -0.545154
3   -0.196823
```

```
4   -0.980568
..        ...
437  0.151508
438  0.935254
439  0.325674
440 -0.545154
441  0.064426

[442 rows x 8 columns]
```

[162]:
```python
# Outliers based on z-score.  Three standard deviations from the mean.
df_zscore.loc[(df_zscore['bmi'] > 2.576) | (df_zscore['bmi'] < -2.576), 'bmi' ]
```

[162]:
```
32     2.634011
145    2.701990
256    3.381781
262    2.679330
366    2.883268
367    3.585718
405    2.588691
Name: bmi, dtype: float64
```

[163]:
```python
# Outliers based on z-score.  Three standard deviations from the mean.
df_zscore.loc[(df_zscore['bmi'] > 2.576) | (df_zscore['s1'] < -2.576), 's1' ]
```

[163]:
```
32    -1.132240
76    -2.665411
145   -0.698324
256   -0.611541
262    0.343075
366    0.863775
367    0.632353
405   -2.202567
Name: s1, dtype: float64
```

1.6 Method 4: z-score using SQL

1.6.1 Finding Outliers in an OracleLive SQL database using z-scores.

Assumption: Data has a normal distribution

CAL_MONTH_SALES_MV does not have any outliers

```
-- Oracle dummy tables.
-- Calculate z-score
-- Outliers: data that falls outside 3 standard deviations from the mean
-- CAL_MONTH_SALES_MV does not have any outliers

select *
from (
select calendar_month_desc, dollars
, round((dollars - (avg(dollars) over() ))/stddev(dollars) over(),2) as zscore
from SH.CAL_MONTH_SALES_MV
    ) score_table
where zscore > 2.576 or zscore < -2.576;
```

```
no data found
```

Assumption: Data has a normal distribution

CO.STORE_ORDERS has two outliers.

```
 -- Oracle dummy tables.
-- Calculate z-score.
-- Assumption:  data is normally distributed
-- Outliers: data that falls outside 3 standard deviations from the mean
-- CO.STORE_ORDERS has two outliers

select *
from (
select store_name, total_sales
, round((total_sales - (avg(total_sales) over() ))/stddev(total_sales) over(),2) as zscore
from co.store_orders
where order_status = 'COMPLETE'
    ) score_table
where zscore > 2.576 or zscore < -2.576;
```

STORE_NAME	TOTAL_SALES	ZSCORE
Online	211107.78	2.58
-	299889.62	3.81

```
[ ]:
```