pratical-2

April 17, 2024

---

1 Data Wrangling II

Create an "Academic performance" dataset of students and perform the following operations using Python. 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. 3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly.

[ ]:
```
# # 1) Scan all variables for missing values and inconsistencies. If there are␣
 ↪missing values and/or inconsistencies, use any of the suitable techniques to␣
 ↪deal with them
# import pandas as pd
# import numpy as np
# from numpy.random import seed
# from numpy.random import randn
# from numpy import mean
# from numpy import std
# import matplotlib.pyplot as plt
# from numpy.lib.function_base import percentile
# from sklearn.preprocessing import MinMaxScaler

# student = pd.read_csv("StudentsPerformance.csv")

# student.info()

# student.isnull().sum()

# # Filling Missing Value by mean
# student['math score'].fillna(int(student['math score'].mean()),inplace =True)

# student.isnull().sum()
```

```
# # Filling missing value with previous ones
# student['reading score'].fillna(method ='pad',inplace=True)

# student.isnull().sum()

# # Filling missing value by median
# student['writing score'].fillna(int(student['writing score'].
 ↪median()),inplace=True)

# student.isnull().sum()

# # 2) Scan all numeric variables for outliers. If there are outliers, use any␣
 ↪of the suitable techniques to deal with them.

# seed(1)
# # univaraite dataset: single variable/ attribute
# # multivariate dataset: multiple variable/ attribute
# data = 5*randn(10000)+ 50
# print('mean=%.3f stdv=%.3f' % (mean(data),std(data)) )

# # Standard Deviation Method
# data_mean = mean(data)
# data_std = std(data)
# cut_off = data_std * 3
# lower = data_mean - cut_off
# upper = data_mean + cut_off

# outliers = [x for x in data if x < lower or x > upper]
# outliers

# plt.plot(data)

# outliers_removed = [x for x in data if x >= lower and x <= upper]
# plt.plot(outliers_removed)

# # Interquartile Deviation Method

# q25 = percentile(data, 25)
# q75 = percentile(data, 75)
# IQR = q75 - q25
# cut_off_IQR = IQR * 2
# lower = q25 - cut_off_IQR
# upper = q75 + cut_off_IQR

# outliers_IQR = [x for x in data if x < lower or x > upper]
# outliers_IQR
```

```
# outliers_removed = [x for x in data if x >= lower and x <= upper]
# plt.plot(outliers_removed)

# # 3) Apply data transformations on at least one of the variables. The purpose␣
 ↪of this transformation should be one of the following reasons: to change the␣
 ↪scale for better understanding of the variable, to convert a non-linear␣
 ↪relation into a linear one, or to decrease the skewness and convert the␣
 ↪distribution into a normal distribution. Reason and document your approach␣
 ↪properly.
# mms = MinMaxScaler()

# student[['math score', 'reading score', 'writing score']] = mms.
 ↪fit_transform(student[['math score', 'reading score', 'writing score']])
# student.head()
```

1.0.1 1) Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them

[1]:
```
import pandas as pd
import numpy as np
from numpy.random import seed
from numpy.random import randn
from numpy import mean
from numpy import std
import matplotlib.pyplot as plt
from numpy.lib.function_base import percentile
from sklearn.preprocessing import MinMaxScaler
```

[2]:
```
student = pd.read_csv("StudentsPerformance.csv")
student.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

[3]:
```
student.isnull().sum()
```

[3]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

[4]:
```
# Filling Missing Value by mean
student['math score'].fillna(int(student['math score'].mean()),inplace =True)
```

[5]:
```
student.isnull().sum()
```

[5]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

[6]:
```
# Filling missing value with previous ones
student['reading score'].fillna(method ='pad',inplace=True)
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_7612\686343477.py:2: FutureWarning:
Series.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  student['reading score'].fillna(method ='pad',inplace=True)
```

[7]:
```
student.isnull().sum()
```

[7]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

[8]:
```
# Filling missing value by median
student['writing score'].fillna(int(student['writing score'].
 ↪median()),inplace=True)
```

[9]:
```
student.isnull().sum()
```

[9]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

1.0.2 2) Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

[10]:
```
from numpy.random import seed
from numpy.random import randn
from numpy import mean
from numpy import std
seed(1)
# univaraite dataset: single variable/ attribute
# multivariate dataset: multiple variable/ attribute
data = 5*randn(10000)+ 50

print('mean=%.3f stdv=%.3f' % (mean(data),std(data)) )
```

```
mean=50.049 stdv=4.994
```

1.0.3 Standard Deviation Method

[11]:
```
data_mean = mean(data)
data_std = std(data)
cut_off = data_std * 3
lower = data_mean - cut_off
upper = data_mean + cut_off
```

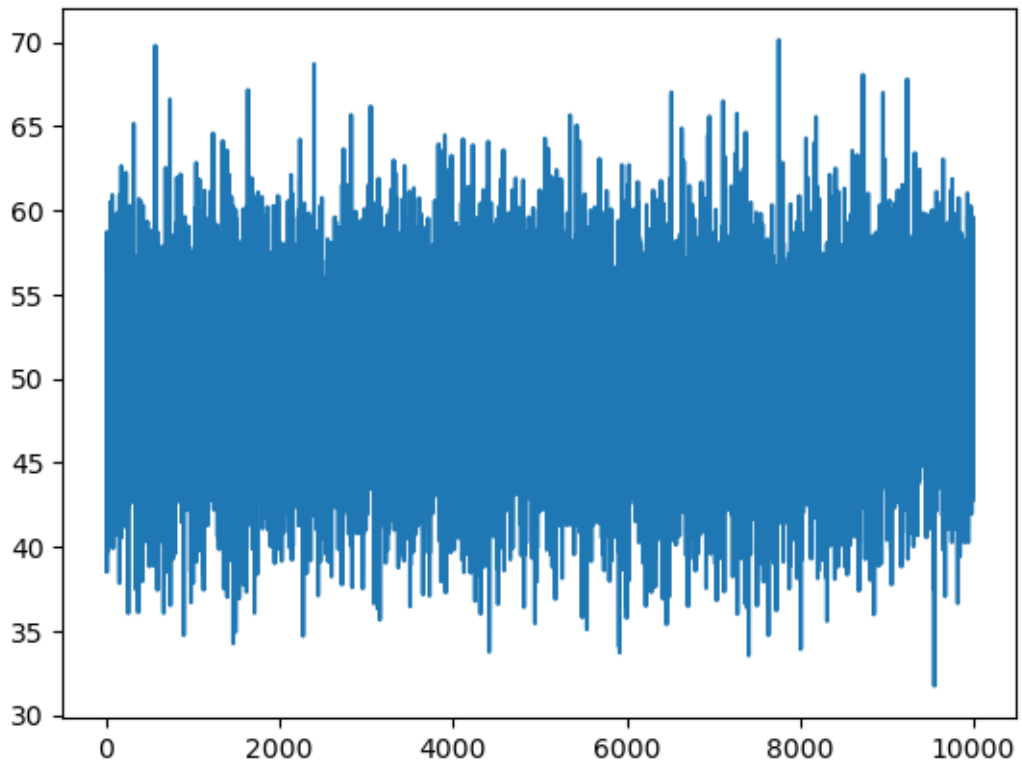
[12]:
```
outliers = [x for x in data if x < lower or x > upper]
outliers
```

[12]:
```
[65.15428556186015,
 69.79301352018982,
 66.60539378085183,
 34.73117809786848,
```

```
34.23321274904475,
34.91984007395351,
67.1633171589778,
34.679293219474495,
68.70124451852294,
65.67523670043954,
66.19171598376188,
33.73482882511691,
65.66014864070253,
65.06377284118616,
34.0469182658796,
33.6969245211173,
67.02151137874486,
65.59239795391275,
66.49270261640393,
65.74492012609815,
33.525707966507426,
34.72183379792847,
70.1342452227369,
33.90433947188079,
65.55945915508362,
68.06638503541573,
66.99057828251213,
67.80436660352774,
31.717799503726024]
```

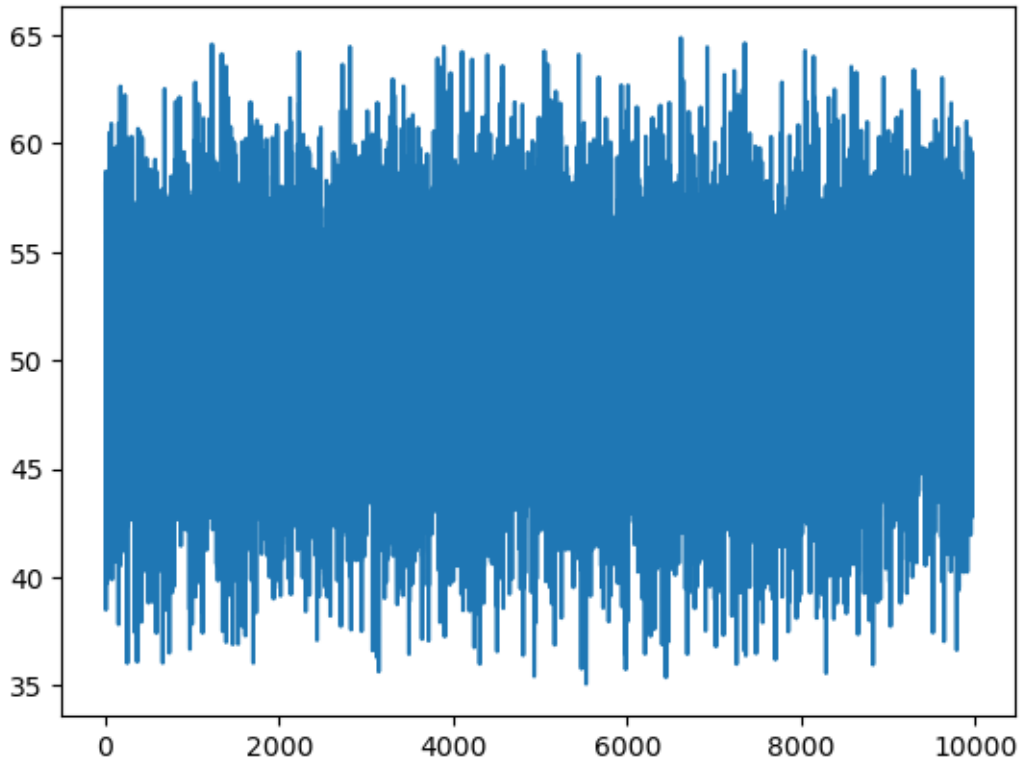
[13]:
```
import matplotlib.pyplot as plt
plt.plot(data)
```

[13]:
```
[<matplotlib.lines.Line2D at 0x24c7a7fc650>]
```



[14]:
```
outliers_removed = [x for x in data if x >= lower and x <= upper]
plt.plot(outliers_removed)
```

[14]: [<matplotlib.lines.Line2D at 0x24c7a859790>]



1.0.4 Interquartile Deviation Method

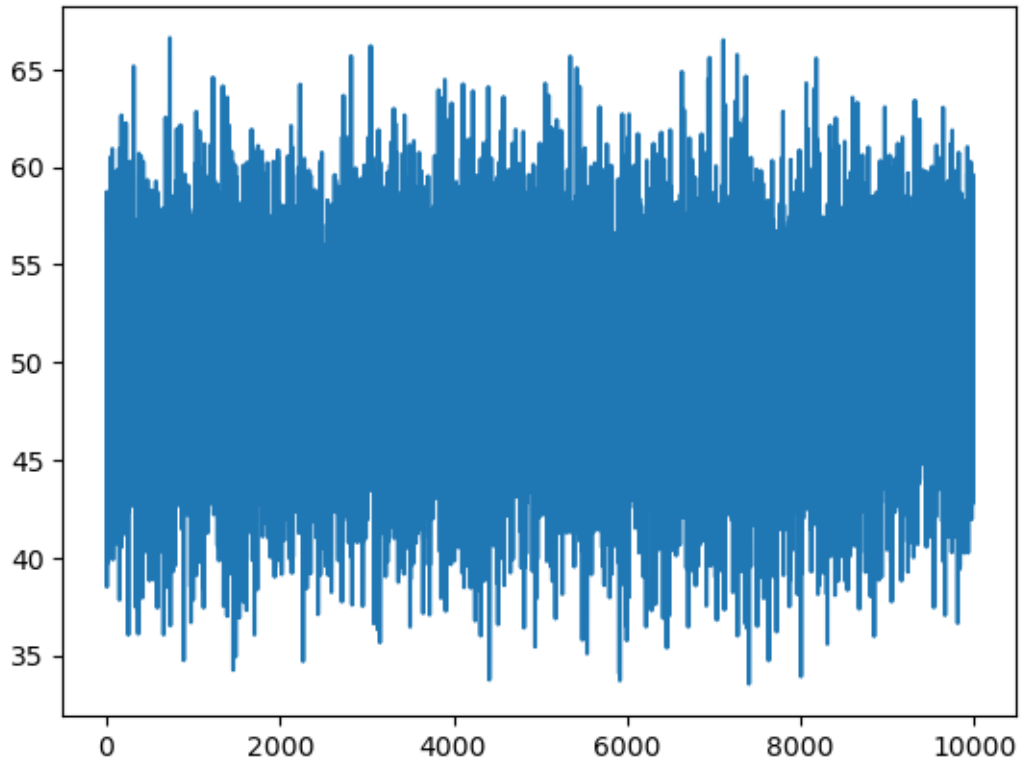
[15]:
```
from numpy.lib.function_base import percentile
q25 = percentile(data, 25)
q75 = percentile(data, 75)
IQR = q75 - q25
cut_off_IQR = IQR * 2
lower = q25 - cut_off_IQR
upper = q75 + cut_off_IQR
```

[16]:
```
outliers_IQR = [x for x in data if x < lower or x > upper]
outliers_IQR
```

[16]:
```
[69.79301352018982,
 67.1633171589778,
 68.70124451852294,
 67.02151137874486,
 70.1342452227369,
 68.06638503541573,
 66.99057828251213,
 67.80436660352774,
 31.717799503726024]
```

[17]:
```
outliers_removed = [x for x in data if x >= lower and x <= upper]
plt.plot(outliers_removed)
```

[17]:
```
[<matplotlib.lines.Line2D at 0x24c7de34650>]
```



1.0.5 3) Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly.

[18]:
```
from sklearn.preprocessing import MinMaxScaler
```

[19]:
```
mms = MinMaxScaler()
```

[20]:
```
student[['math score', 'reading score', 'writing score']] = mms.
↪fit_transform(student[['math score', 'reading score', 'writing score']])
```

[21]:
```
student.head()
```

[21]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	0.72	0.662651	0.711111
1	female	group C	some college	standard	completed	0.69	0.879518	0.866667
2	female	group B	master's degree	standard	none	0.90	0.939759	0.922222
3	male	group A	associate's degree	free/reduced	none	0.47	0.481928	0.377778
4	male	group C	some college	standard	none	0.76	0.734940	0.722222

[ ]: