Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

Data Wrangling using Python

Agenda

1. Data Operations (selection , append , concat)
2. Univariate Analysis
3. Multivariate Analysis
4. Handling Missing Values
5. Handling Outliers

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is used:

1. To Solve simple business problems
2. To test business assumptions
3. To generate hypotheses for further analysis.
4. To prepare the data for modeling.

Data profiling: To summarize the dataset through descriptive statistics in order to better understand your dataset and to assess its quality.

Python Operators

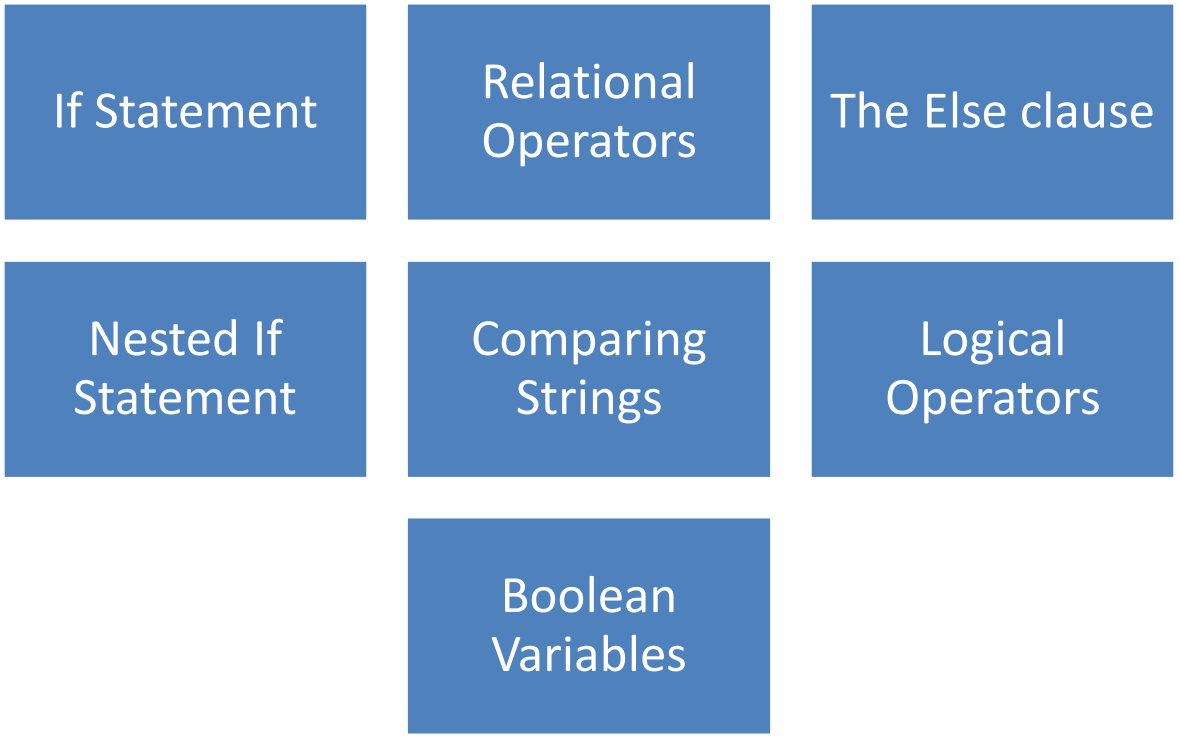
Operators are symbols with a specific action to be performed on the variables to make mathematical or logical manipulations. In python, we have the following data operators:

- Arithmetic Operators
- Comparison Operators
- Python Assignment Operators
- Logical Operators or Bitwise Operators
- Membership Operators
- Identity Operators
- Operator precedence

Operators	Meaning
**	Exponent
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<= < > >=	Comparison operators
= %= /= //= -= += *= **=	Assignment Operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Data Operation: Selection

Codes are generally executed on the order we write them or they appear. However, In python we have selection control statements to leverage to change the flow of control of the program. Below is the list of the selection statements:



Data Operation: Append

Append():

- This function **e**xpands an array by one element and adds data to the new position. The datatype of the element parameter must be the same as the datatype of the array.
- When using an array of objects, the data returned from the function must be cast to the object array's data type.

For Example: append(array, element)

Parameters:

Array: boolean[], byte[], char[], int[], float[], or String[], or an array of objects

Element: New data for the array

Returns: Array (the same datatype as the input)

Data Operation: Concat

The concat() module lets us gather ***concat::fragment*** resources from our other modules and order them into a coherent file through a single concat resource.

It helps in maintaining a list of the major modules on a node

Types:

concat_file: Generates a file with content from fragments sharing a common unique tag.

concat_fragment: Manages the fragment.


```
class motd {
  $motd = '/etc/motd'

  concat { $motd:
    owner => 'root',
    group => 'root',
    mode  => '0644'
  }

  concat::fragment{ 'motd_header':
    target  => $motd,
    content => "\nPuppet modules on this server:\n\n",
    order   => '01'
  }
```

Univariate and multivariate Analysis

Predictor Variable
Target Variable

Character Variable
Numeric Variable

Categorical Variable
Continuous variable

Univariate Analysis

Univariate analysis refers to the analysis of one variable

Continuous Variables:- In case of continuous variables, we need to understand the central tendency and spread of the variable. These are measured using various statistical metrics visualization methods as shown below:

Pie chart
Bar Chart

Categorical

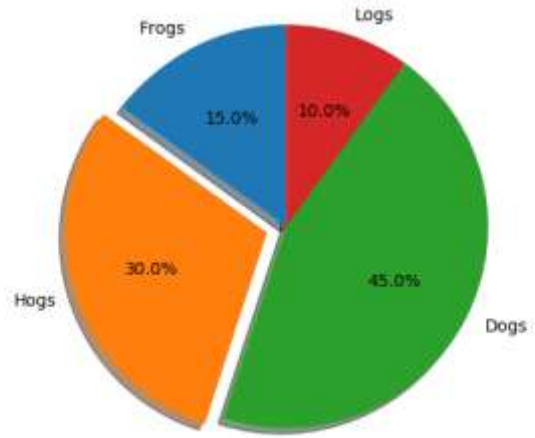
Histogram
Boxplot

Numerical

For categorical variables, we'll use frequency table to understand distribution of each category. We can also read as percentage of values under each category. We can also use charts like Pie and bar for analysis

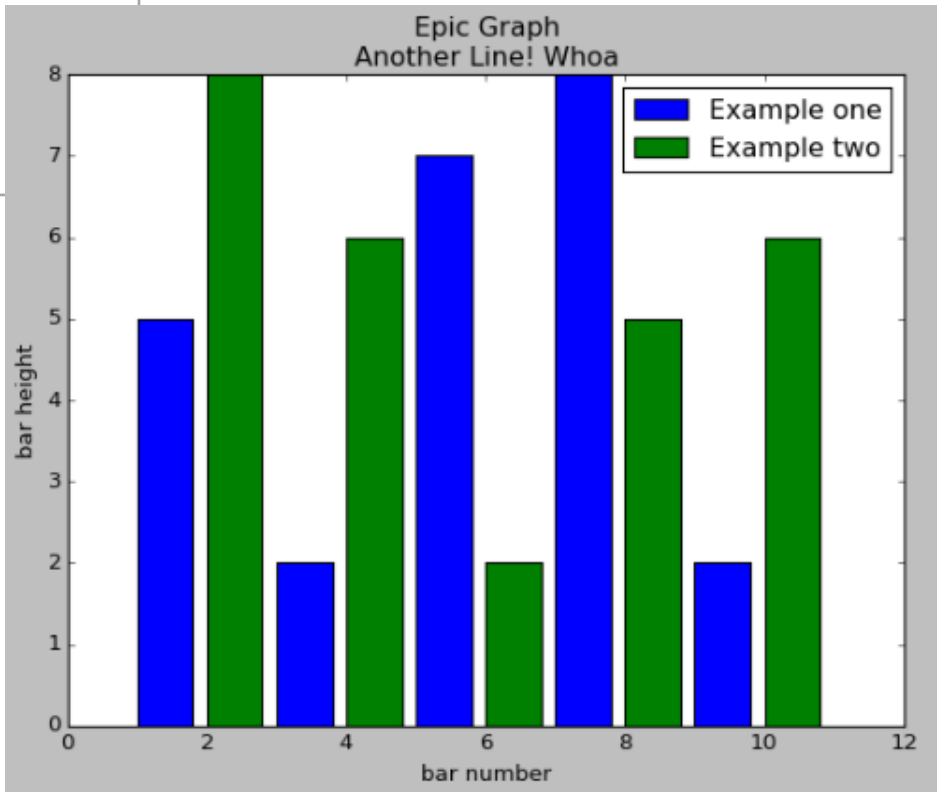
Matplotlib – Pie Chart

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



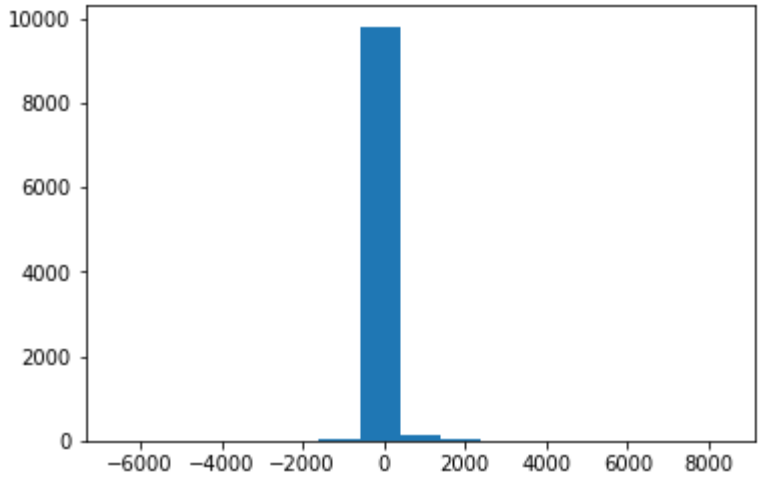
Matplotlib – Bar Charts

```
import matplotlib.pyplot as plt
      plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Example one")

plt.bar([2,4,6,8,10],[8,6,2,5,6], label="Example two", color='g')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')

plt.title('Epic Graph\nAnother Line! Whoa')

plt.show()
```



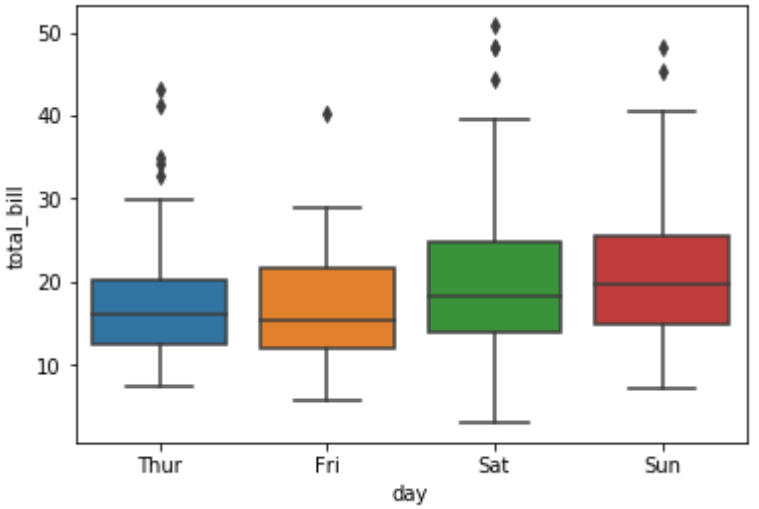
Matplotlib – Histogram

```
import pandas as pd
import os
os.chdir('D:\\Python A-Z\\Module 1 - Python Basics\\')

sample_super_store=pd.read_excel('Sample - Superstore.xls')

from matplotlib import pyplot as plt
import seaborn as sns

n1 = plt.hist(sample_super_store.Sales, bins=15)
```



Matplotlib – Box and Whisker Plot

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

tips=sns.load_dataset("tips")
ax=sns.boxplot(x="day", y=tips["total_bill"], data=tips)
```



Multivariate Analysis

Multivariate analysis is a graphical in its approach. This allows users to examine the inner or hidden structure of large data sets, and to visually identify the factors which influence the results.

For Categorical and Continuous comparison:
Time series or a line chart is one of the multivariate chart where we are comparing two variables such as time and sales

For Continuous and Continuous comparison:
Scatter plot is other way to compare two measures i.e. Sales and Profit

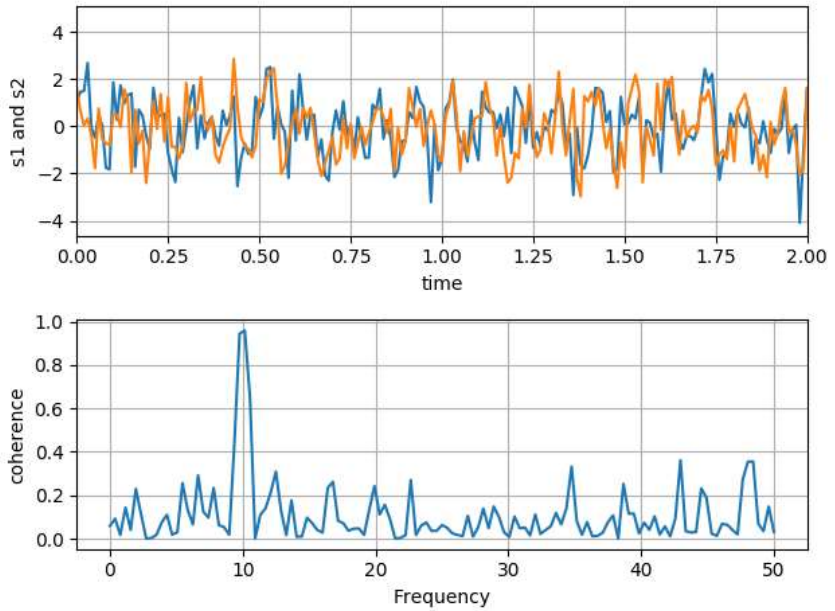
For Categorical and Categorical comparison:
To find the relationship between two categorical variables, we can use following methods:

Two-way table: We can start analyzing the relationship by creating a two-way table of count and count%. The rows represents the category of one variable and the columns represent the categories of the other variable. We show count or count% of observations available in each combination of row and column categories.

Stacked Column Chart: This method is more of a visual form of Two-way table.

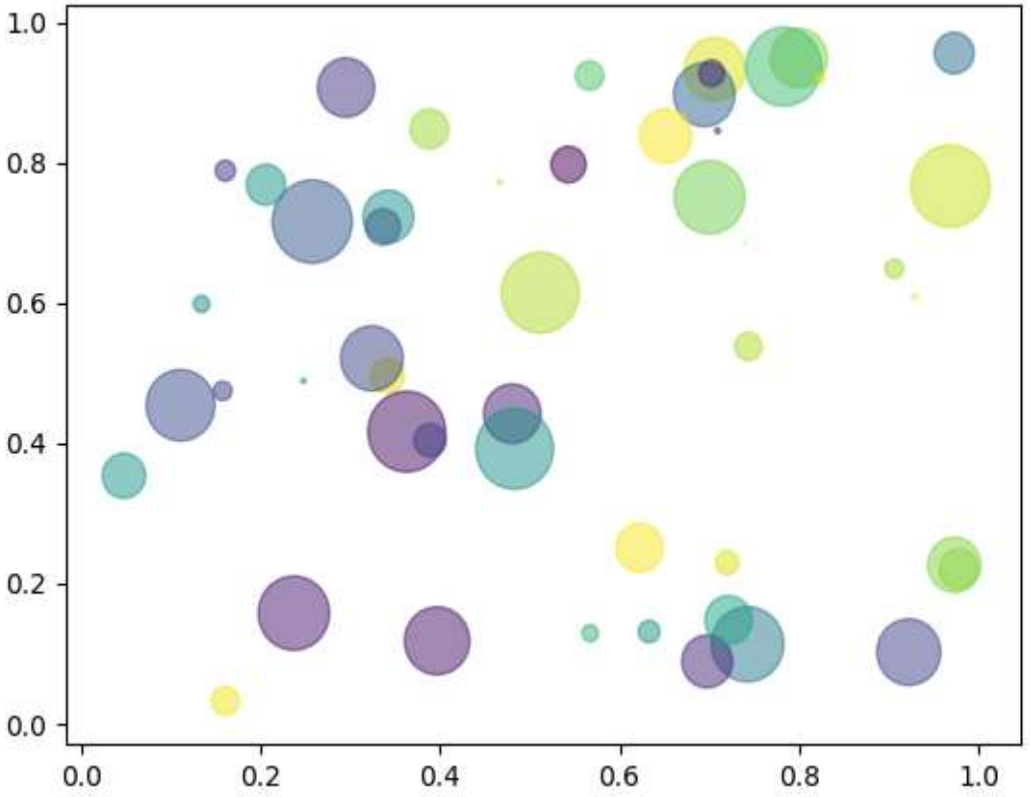
Matplotlib – Line chart

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

dt = 0.01
t = np.arange(0, 30, dt)
nse1 = np.random.randn(len(t))                 # white noise 1
nse2 = np.random.randn(len(t))                 # white noise 2

# Two signals with a coherent part at 10Hz and a random part
s1 = np.sin(2 * np.pi * 10 * t) + nse1
s2 = np.sin(2 * np.pi * 10 * t) + nse2

fig, axs = plt.subplots(2, 1)
axs[0].plot(t, s1, t, s2)
axs[0].set_xlim(0, 2)
axs[0].set_xlabel('time')
axs[0].set_ylabel('s1 and s2')
axs[0].grid(True)

cxy, f = axs[1].cohere(s1, s2, 256, 1. / dt)
axs[1].set_ylabel('coherence')

fig.tight_layout()
plt.show()
```



Matplotlib – Scatter Plot

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



Advantages of univariate and multivariate analysis

Univariate and Multivariate techniques allow researchers to look at

- Relationships between variables in an overarching way and to quantify the relationship between variables.
- Introduce other variables to determine the links between the independent and dependent variables or to specify the conditions under which the association takes place.
- Advantages of these analysis include an ability to glean a more realistic picture than looking at a single variable.

Why Missing Values treatment is required?

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behaviour and relationship with other variables correctly. It can lead to wrong prediction or classification.

Player Id	Weight	Gender	Play Cricket/Not
1	56	M	Y
2	67	M	Y
3	65		N
4	67	F	N
5	78		N
6	87		N
7	65	F	Y
8	45	F	Y
9	65	F	N
10	76	M	Y

Gender	#Players	#Play Cricket	%Play Cricket
F	4	2	50%
M	3	3	100%
Missing	3	0	0%

Player Id	Weight	Gender	Play Cricket/Not
1	56	M	Y
2	67	M	Y
3	65	M	N
4	67	F	N
5	78	M	N
6	87	F	N
7	65	F	Y
8	45	F	Y
9	65	F	N
10	76	M	Y

Gender	#Players	#Play Cricket	%Play Cricket
F	5	2	40%
M	5	3	60%

No shown above: In the left scenario, we have not treated missing values. The inference from this data set is that the chances of playing cricket by males which is 100% and is higher than females. On the other hand, if you look at the second table, which shows data after treatment of missing values (based on gender), we can see that males chances of playing cricket drastically reduced.

Missing Values

Step1: Identifying Missing value

- Use describe() to see unusual zero's in the statistical description of the data.
- For example:

	1	2	3	4
Count	571	455	352	512
Mean	350	52	7892	12
Std dev.	1	2	1	0
Min	200	20	2512	8
25%	499	59	0	15
50%	999	118	0	30
75%	1498	177	0	46
Max	1800	190	58215	22

We can observe that column 3 and 4 may have more number of missing values since there are unusual zeros in the summary

Missing Values

Step2: Call Columns with unusual zeros found in statistical description

```
Print((dataset[[3,4]] = =0), sum()
```

1	2	3	4
0	52	112	120

3	4
3587	12
2685	10
5001	0
3621	11
0	0
0	0
0	8
0	10
.	.
.	.
.	.
2828	0

We can clearly see, columns 3 and 4 have large number of zeros and it is affecting the statistical calculations of the data.

Treating Missing Values

Once, missing values are identified. Next step is to decide weather to treat these missing values by filling plausible values with respect to each column or to drop a particular column completely.

Method 1:
Replace all zero values with *NAN* value using Replace() function.

```
Dataset[[1,2,3,4]] = dataset[[1,2,3,4]].replace(0,numoy.nan
Print(dataset.isnull().sum())
```

By replacing zero values with *NaN,* python will not consider these values while doing any mathematical calculation, such as finding mean, variance etc.

It would simply omit them and hence give a better picture.

Treating Missing Values

Method 2:

Drop the column completely using drop() function.

```
Dataset.dropna(inplace = True)
```

If a particular column has large number of missing values like more than 25% (or depends upon the business scenario), then sometimes, treating them with nan function is not a good option because the efficiency of overall model would be weak. In that case, we sometimes have to drop the complete column itself. So that overall reliability of model doesn't get affected due to one weak column

Treating Missing Values

Method 3:

Replace the missing values with mean of the column.
Note: Outliers will impact the missing value as the mean will impact because of the outliers.

```
Dataset.fillna(dataset.mean(), inplace = True)
```

If a particular column has moderate number of missing values, less than 25% (or depends upon the business scenario) and also, If data is normally distributed., then, we can opt for filling those missing values by mean of the column. To make the algorithm work properly

Treating Missing Values

Method 4:

Replace the missing values with median of the column.
Note: Outliers will not impact the missing value as the median will not impact because of the outliers.

```
Dataset.fillna(dataset.median(), inplace = True)
```

Treating Missing Values

Method 5:

Replace the missing values with mode of the column.
Note: This will be the best option of imputation for categorical data

```
Dataset.fillna(dataset.mean(), inplace = True)
```

Treating Missing Values

Method 6:

KNN Imputation: In this method of imputation, the missing values of an attribute are imputed using the given number of attributes that are most similar to the attribute whose values are missing. The similarity of two attributes is determined using a distance function. It is also known to have certain advantage & disadvantages.

Advantages:

- k-nearest neighbour can predict both qualitative & quantitative attributes
- Creation of predictive model for each attribute with missing data is not required
- Attributes with multiple missing values can be easily treated
- Correlation structure of the data is taken into consideration

Disadvantage:

- KNN algorithm is very time-consuming in analyzing large database. It searches through all the dataset looking for the most similar instances.
- Choice of k-value is very critical. Higher value of k would include attributes which are significantly different from what we need whereas lower value of k implies missing out of significant attributes.

Treating Missing Values

Method 7:

Prediction Model: Prediction model is one of the sophisticated method for handling missing data. Here, we create a predictive model to estimate values that will substitute the missing data.

- In this case, we divide our data set into two sets: One set with no missing values for the variable and another one with missing values.
- First data set become training data set of the model while second data set with missing values is test data set and variable with missing values is treated as target variable.
- Next, we create a model to predict target variable based on other attributes of the training data set and populate missing values of test data set.
- We can use regression, ANOVA, Logistic regression and various modeling technique to perform this.

There are 2 drawbacks for this approach:

- The model estimated values are usually more well-behaved than the true values
- If there are no relationships with attributes in the data set and the attribute with missing values, then the model will not be precise for estimating missing values.

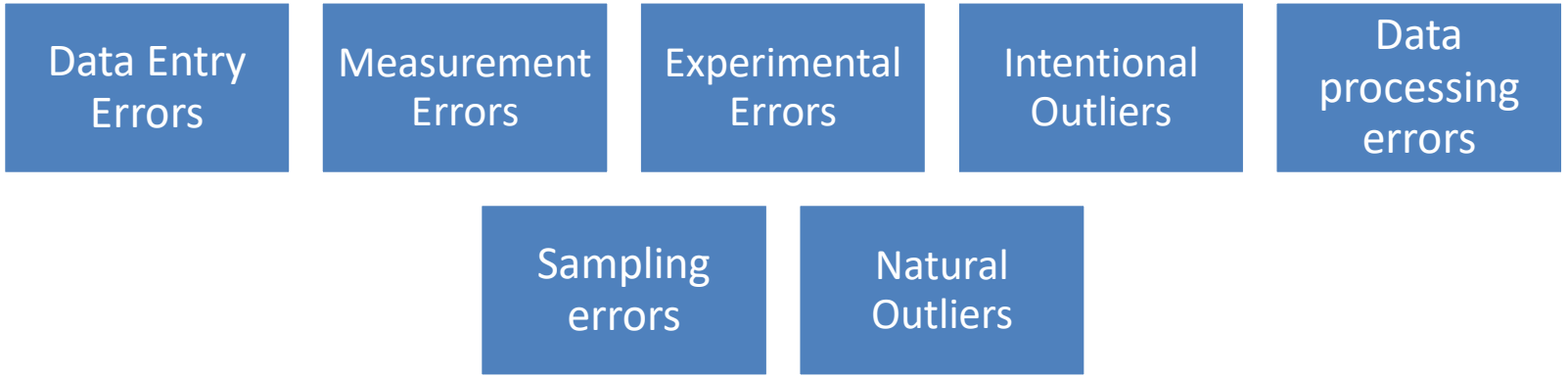
Libraries in Python

Scikit-learn is the library very much in use to do all the data pre processing and cleaning work.

```
>>> import numpy as np
>>> from sklearn.impute import SimpleImputer
>>> imp = SimpleImputer(missing_values=np.nan, strategy='mean')
>>> imp.fit([[1, 2], [np.nan, 3], [7, 6]])
SimpleImputer(copy=True, fill_value=None, missing_values=nan, strategy='mean', verbose=0)
>>> X = [[np.nan, 2], [6, np.nan], [7, 6]]
>>> print(imp.transform(X))
[[4.          2.        ]
 [6.          3.666...]
 [7.          6.        ]]
```

Types of Outliers

In any given dataset, there is always a possibility of :



Data Entry Errors:- Human errors such as errors caused during data collection, recording, or entry can cause outliers in data. For example: Monthly income of a customer is $15,000. Accidentally, the data entry operator puts an additional zero in the figure. Now the income becomes $150,000 which is 10 times higher. Evidently, this will be the outlier value when compared with rest of the population.

Measurement Error: It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty. For example: There are 10 weighing machines. 8 of them are correct, 2 is faulty. Weight measured by people on the faulty machine will be higher / lower than the rest of people in the group. The weights measured on faulty machine can lead to outliers.

Types of outliers

Experimental Error: Another cause of outliers is experimental error. For example: In a 100m sprint of 5 runners, one runner missed out on concentrating on the 'Go' call which caused him to start late. Hence, this caused the runner's run time to be more than other runners. His total run time can be an outlier.

Intentional Outlier: This is commonly found in self-reported measures that involves sensitive data. For example: Teens would typically under report the amount of TV they watch. Only a fraction of them would report actual value. Here actual values might look like outliers because rest of the teens are under reporting.

Data Processing Error: Whenever we perform data mining, we extract data from multiple sources. It is possible that some manipulation or extraction errors may lead to outliers in the dataset.

Sampling error: We have to measure the height of athletes. By mistake, we include a few basketball players in the sample. This inclusion is likely to cause outliers in the dataset.

Natural Outlier: When an outlier is not artificial (due to error), it is a natural outlier.

What are the impact of outliers in the dataset?

Outliers can drastically change the results of the data analysis and statistical modeling. There are numerous unfavourable impacts of outliers in the data set:

- It increases the error variance and reduces the power of statistical tests
- If the outliers are non-randomly distributed, they can decrease normality
- They can bias or influence estimates that may be of substantive interest
- They can also impact the basic assumption of Regression, ANOVA and other statistical model assumptions.

To understand the impact deeply, let's take an example to check what happens to a data set with and without outliers in the data set.

Dataset 1	Dataset 2
4	4
4	4
5	5
5	5
5	5
5	5
6	6
6	7
7	7
7	300

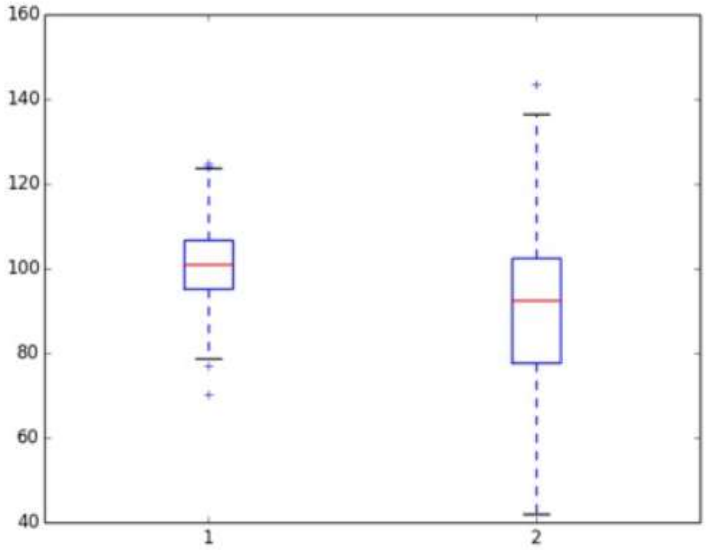
Mean	5.4	34.8
Median	5	5
Mode	5	5
Standard Deviation	1.019804	88.40566

Identification of Outlier

We can identify the outlier by graphically viewing the data used to plot box whisker plot

Code in python :

```
import numpy as np
>>> import matplotlib.pyplot as plt
>>> np.random.seed(10)
>>> array1 = np.random.normal(100, 10, 200)
>>> array2 = np.random.normal(90, 20, 200)
>>> data = [array1, array2]
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> res = ax.boxplot(data)
>>> plt.show()
```



Handling Outliers

To handle outliers we have four options

1) Transform the values
2) Imputing
3) Treat separately
4) Remove the outliers using Inter quartile range formula
    a) Upper hinge – Q3+ 1.5IQR
    b) Lower hinge – Q1 – 1.5IQR

Handling Outliers :Transformation

In any given scenario, if we can't just omit the outlier, we can fix the outlier as well. One such option is "Transformation"

Square root and log transformations both pull in high numbers. This can make assumptions work better if the outlier is a dependent variable and can reduce the impact of a single point if the outlier is an independent variable.

Cap the Data

- Put the value at a level that keeps the data intact

Assign New Value

- Put the value by taking mean of a variable, or utilizing a regression model to predict the missing value.

If the outlier can't be transformed, we can probably try the different model which can fit the outlier in the data set.

Handling Outliers - Contd

2) Imputing

Like imputation of missing values, we can also impute outliers. We can use mean, median, mode imputation methods. Before imputing values, we should analyse if it is natural outlier or artificial. If it is artificial, we can go with imputing values. We can also use statistical model to predict values of outlier observation and after that we can impute it with predicted values.

3) Treat Separately

If there are significant number of outliers, we should treat them separately in the statistical model. One of the approach is to treat both groups as two different groups and build individual model for both groups and then combine the output.

Handling Outliers – Inter Quartile Range (Box and Whisker)

To handle outliers we have two options

1) Remove the outliers using Inter quartile range formula
   a) Upper hinge – Q3+ 1.5IQR
   b) Lower hinge – Q1 – 1.5IQR

```
def remove_outlier(df_in, col_name):
    q1 = df_in[col_name].quantile(0.25)
    q3 = df_in[col_name].quantile(0.75)
    iqr = q3-q1
    fence_low = q1 - 1.5*iqr
    fence_high = q3 + 1.5*iqr
    df_out = df_in.loc[(df_in[col_name]> fence_low) & (df_in[col_name] < fence_high)]
    return df_out


df_in=FileNameDesc[(FileNameDesc.column_type=='int64') | (FileNameDesc.column_type =='float64')]


for col_na in df_in['column_name']:
    if col_na != IV:
        remove_outlier(dataset,col_na)
```


Cognixia
A Collabera LEARNING SOLUTIONS COMPANY
THANK YOU
45%