Signitive TECHNOLOGIES

Signify the Learning

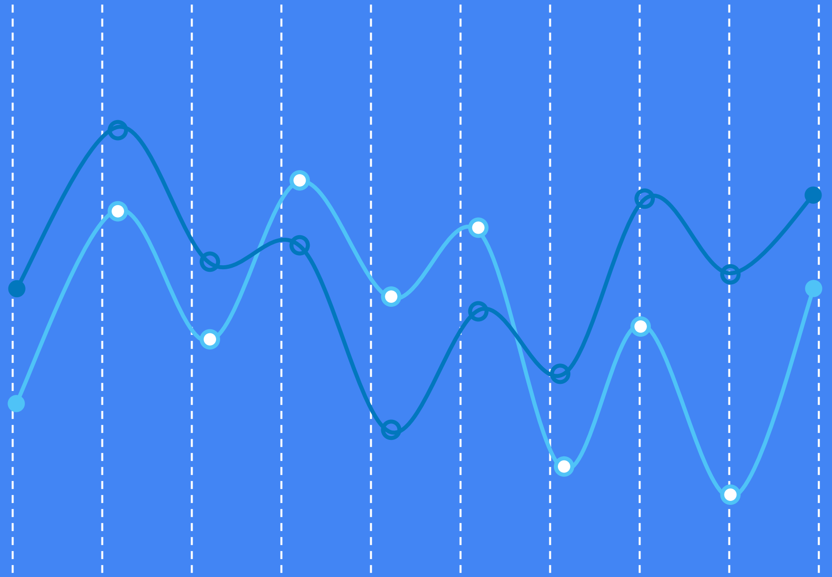
Certification Training Courses for Graduates and Professionals

www.signitivetech.com

Data Preprocessing

Agenda

- ❖ Convert Categorical Features to Numeric
- ❖ Missing Values Handling
- ❖ Rescaling the features
- ❖ Outliers Detection and Removal



Convert Categorical Features to Numeric

Convert Categorical Features to Numeric

- Categorical Variables: These can be Nominal or Ordinal
- Nominal Variables: Example
  - Gender can be Male or Female
  - Education can be Undergraduate, Graduate or Postgraduate

- Ordinal Variables: Example
  - Size can be Small, Medium or Large
  - Ratings can be Best, Better, Good, Satisfactory, Bad or Worst

- These Categorical variables need to be converted into Numeric Variables because most ML Algorithms don't support categorical variables while doing computations.

Convert Categorical Features to Numeric

- Techniques to convert categorical variables to Numeric

❖ Dummy Variables Creation:
❖ Dummy variables are those variables whose values are only 0 and 1

Consider this Data:

```
import pandas as pd
df = pd.read_csv('data.csv')
df
```

Age	Gender	City
22	M	Nagpur
25	M	Pune
26	F	Banglore
28	M	Pune
28	M	Pune
27	F	Nagpur

Convert Categorical Features to Numeric

To convert categorical features (Gender and City), we will create dummy variables using pandas libraries.

```
dummies = pd.get_dummies(data = df, columns = ['Gender', 'City'])
```

Age	Gender_M	Gender_F	City_Nagpur	City_Banglore	City_Pune
22	1	0	1	0	0
25	1	0	0	0	1
26	0	1	0	1	0
28	1	0	0	0	1
28	1	0	0	0	1
27	0	1	1	0	0

Convert Categorical Features to Numeric

Dummy Variable Trap:

- To avoid Dummy Variable trap we usually remove one column from these dummy variables. Because, when we create dummy variables, new entries will get added.
- Suppose there are 3 categories in one feature column, so 3 different entries of dummy variables will get created.
- Dummy variables trap is usually occurred while encountering Multilinear Regression Models which causes Multicollinearity. To avoid these issues we simply drop any column from these dummies.
- By dropping any one column the interpretation of categorical variables still remains the same.

Convert Categorical Features to Numeric

dummies = pd.get_dummies(data = df, columns = ['Gender', 'City'], drop_first = True)

Age	Gender_F	City_Banglore	City_Pune
22	0	0	0
25	0	0	1
26	1	1	0
28	0	0	1
28	0	0	1
27	1	0	0

Convert Categorical Features to Numeric

- Techniques to convert categorical variables to Numeric

❖ Label Encoding:
❖ Label encoding is simply converting each value in a column to a number.

Consider this Data:

```
import pandas as pd
df = pd.read_csv('data.csv')
df
```

Age	Gender	City
22	M	Nagpur
25	M	Pune
26	F	Banglore
28	M	Pune
28	M	Pune
27	F	Nagpur

Convert Categorical Features to Numeric

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df['Gender'] = label.fit_transform(df['Gender'])
df['City'] = label.fit_transform(df['City'])
```

Age	Gender	City
22	0	0
25	0	1
26	1	2
28	0	1
28	0	1
27	1	0

Convert Categorical Features to Numeric

- Label encoding has the advantage that it is straightforward but it has the disadvantage that the numeric values can be “misinterpreted”.
- Here, city nagpur has 0, pune has 1 and bangalore has 2 values. So, does it mean, banglore is greater than pune or nagpur? Of course not.

Age	Gender	City
22	0	0
25	0	1
26	1	2
28	0	1
28	0	1
27	1	0

Convert Categorical Features to Numeric

- Techniques to convert categorical variables to Numeric

❖ One Hot Encoding:

❖ To avoid Label Encoding misinterpretation, one hot encoding is used. The basic strategy behind one hot encoding is to convert each category value into a new column and assigns a 1 or 0 (True/False) value to the column. This has the benefit of not weighting a value improperly but does have the downside of adding more columns to the data set.

Convert Categorical Features to Numeric

Consider this Data:

```
import pandas as pd
df = pd.read_csv('data.csv')

from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df['Gender'] = label.fit_transform(df['Gender'])
df['City'] = label.fit_transform(df['City'])

from sklearn.preprocessing import OneHotEncoder
ohc = OneHotEncoder(categorical_features=[2])

df = ohc.fit_transform(df).toarray()
```

Age	Gender	City
22	M	Nagpur
25	M	Pune
26	F	Banglore
28	M	Pune
28	M	Pune
27	F	Nagpur

Convert Categorical Features to Numeric

Here while using one hot encoding we will have to mention on which columns you want to do one hot encoding. We are giving here index positions for respective columns.

```
ohc = OneHotEncoder(categorical_features=[2])
```

Then we convert result into array

```
df = ohc.fit_transform(df).toarray()
```

```
[[1, 0,   0,   1,   22]
 [0,  1,   0,   1,   25]
 [0,  0,   1,   0,   26]
 [0,  1,   0,   1,   28]
 [0,  1,   0,   1,   28]
 [1,  0,   0,   0,   27]]
```

Missing Values Handling

Missing Values Handling

- Real-world data often has missing values.
- Data can have missing values for a number of reasons such as observations that were not recorded and data corruption.
- Handling missing data is important as many machine learning algorithms do not support data with missing values.

Missing Values Handling

❖ Mark or Find Missing Values:

❖ We can use summary statistics to help identify missing or corrupt data.

Consider the data

```
import pandas as pd
df = pd.read_csv('data.csv')
df.describe()
```

	bmi_index	weight	height
count	100	100	100
mean	31.25	46.25	33.65
std	7.88	8.25	2.25
min	0.00	0.00	15.25
25%	27.25	39.52	22.25
50%	32.25	41.25	30.25
75%	36.28	48.25	32.36
max	40.25	49.36	39.25

Missing Values Handling

- On finding these Missing Values Replace them with NaN values.

```
import numpy as np

df['bmi_index'] = df['bmi_index'].replace(0, np.NaN)
df['weight'] = df['weight'].replace(0, np.NaN)
```

Missing Values Handling

- ❖ Apply some criterias to handle those missing values.
- ❖ There are many options we could consider when replacing a missing value, for example:

- A constant value that has meaning within the domain, such as 99999, distinct from all other values.
- A value from another randomly selected record.
- A mean, median or mode value for the column.
- A value estimated by another predictive model.

Missing Values Handling

- Replace NaN values with either of the mentioned criterias

```
mean_bmi = df['bmi_index'].mean()
mean_weight = df['weight'].mean()
```

- Using pandas fillna() method, helps to replace NaN values.

```
df['bmi_index'].fillna(mean_bmi, inplace = True)

df['weight'].fillna(mean_weight, inplace = True)
```

- Note: inplace = True indicates the changes will be made to the original dataset

Missing Values Handling

- ❖ Another functionality to replace NaN values is using scikit learn imputer method.
- ❖ It is a flexible class that allows you to specify the value to replace (it can be something other than NaN) and the technique used to replace it (such as mean, median, or mode).

```
from sklearn.preprocessing import Imputer
imput = Imputer(missing_values='NaN', strategy='mean')

df[['bmi_index', 'weight']] = imput.fit_transform(df[['bmi_index', 'weight']])
```

Note: strategy can be mean, median or most_frequent

Missing Values Handling

❖ In some cases, we can simply remove NaN values.

❖ Using pandas dropna() method, helps to remove NaN values.
❖ Note: It removes rows containing NaN values

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

❖ Note: inplace = True indicates the changes will be made to the original dataset

Missing Values Handling

- In some datasets missing values can be represented other symbols like NA, n/a, -, ?, and so on
- How to find these missing terms?
- Suppose you have a dataset df and you are looking missing values in Age column.
- Initially you look NaN values i.e. missing values represented by NaN

```
df['Age'].isnull().sum()
```

```
Output:
Age       0
```

- However, method shows zero NaN values

Missing Values Handling

❖ Using pandas unique() method we can determine unique values from Age column

```
df['Age'].unique()
```

Output:

```
array(['23', '24', '-', '27', '29', '28'])
```

❖ This unique() method shows unique values of age and clearly we can see an **“-”** sign is there which indicates missing term.

Missing Values Handling

❖ How we can check how many “-” missing terms are?

```
(df['Age'] == '-').sum()
```

Output:

```
20
```

❖ This sum() method sums all “-” terms and gives the total count of missing terms.

Rescaling the features

Rescaling the features

- ❖ Most of the time, as a data scientist, you will receive data in an inconvenient form. One of the first things that you will notice in those types of datasets is that the magnitude and range of different features varies a lot. One feature may be measured in hundreds, while the other in tens of thousands.

Why scale?

- ❖ Well, most machine learning algorithms take into account only the magnitude of the measurements, not the units of those measurements.
- ❖ That's why one feature, which is expressed in a very high magnitude (number), may affect the prediction a lot more than an equally important feature.

Rescaling the features

- Let’s say you have two lengths, *l1 = 250 cm* and *l2 = 2.5 m*. We humans see that these two are identical lengths (*l1 = l2),* but most ML algorithms interpret this quite differently.
- The algorithm is going to give a lot more weight to *l1*, just because it is expressed in a larger number, which, in turn is going to have a much larger impact on the prediction than *l2*.
- Note: Certain types of algorithms like Naive Bayes, Decision Trees, Random Forest and XGBoost do not require feature scaling, because they work in a different manner. Algorithms that exploit *distances* or *similarities* (e.g. in form of scalar product) between data samples, such as KNN and SVM, often require feature scaling.

Rescaling the features

The Standard Scalar:

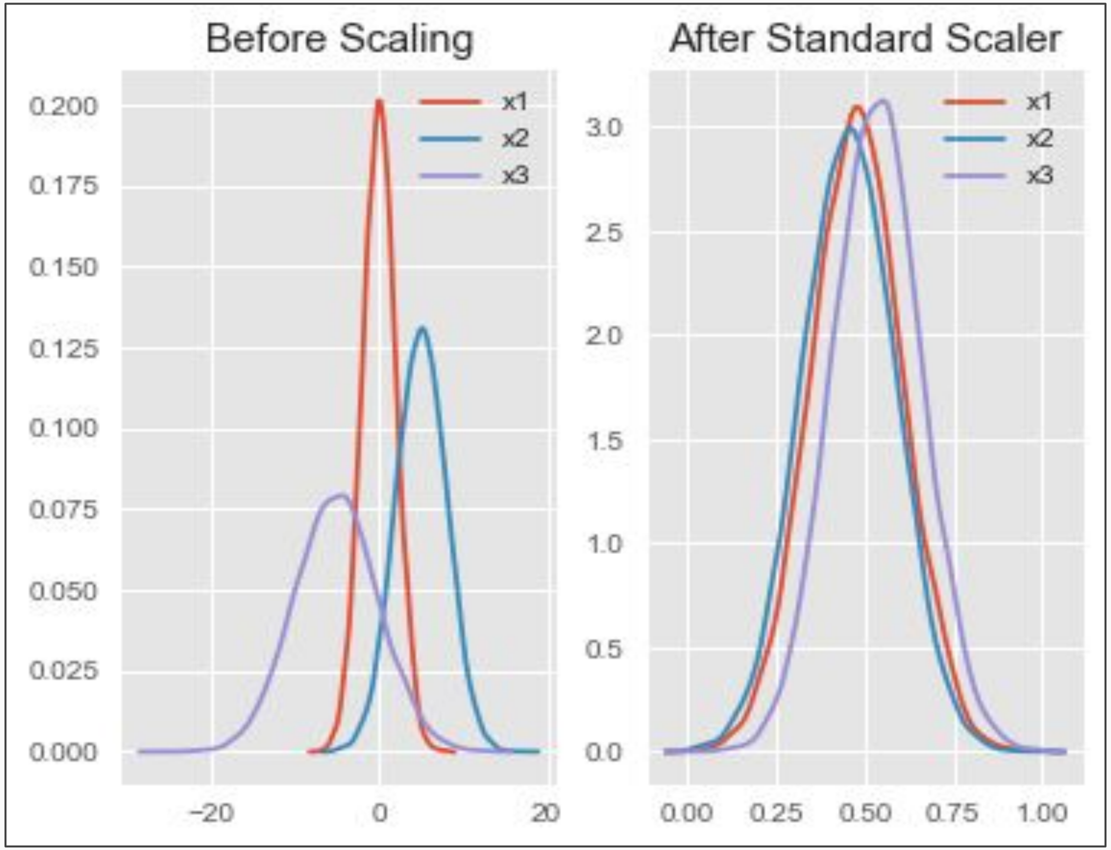
- ❖ The Standard Scaler is one of the most widely used scaling algorithms out there. It assumes that your data follows a Gaussian distribution (Gaussian distribution is the same thing as Normal distribution)
- ❖ The mean and the standard deviation are calculated for the feature and then the feature is scaled based on:

Standard_Scaled_Value = (xi–mean(x))/ stdev(x)

- ❖ The idea behind **Standard Scaler** is that it will transform your data, such that the distribution will have a mean of 0 and a standard deviation of 1.
- ❖ If the data is not normally distributed, it’s not recommended to use the **Standard Scaler**.

Rescaling the features

The Standard Scalar:



Rescaling the features

The Normalizer:

- The Normalizer normalises samples individually to unit norm.
- Each sample (i.e. each row of the data matrix) with at least one non zero component is rescaled independently of other samples so that its norm (*l1* or *l2*) equals one.
- Put more simply, the **normalizer** scales each value by dividing each value by its magnitude in *n*-dimensional space for *n* number of features.

Rescaling the features

The Min-Max Scalar:

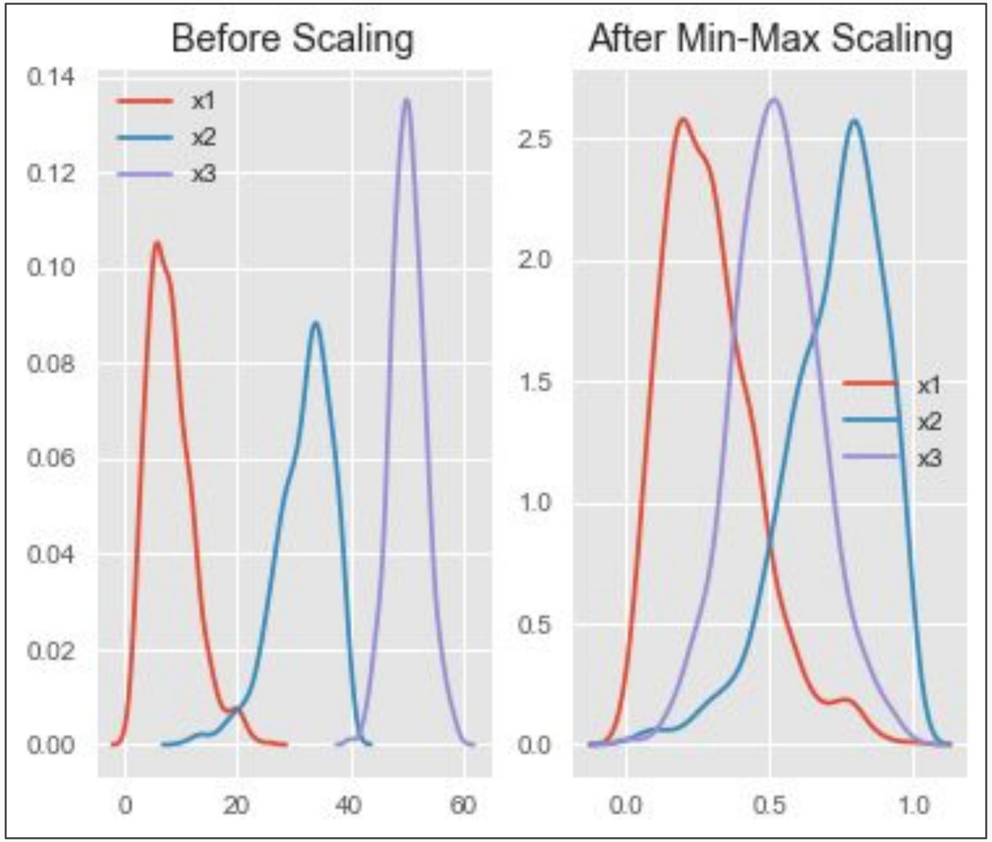
- The Min-Max Scaler uses the following formula for calculating each feature:

  Min_Max_Value = (xi–min(x))/ (max(x)–min(x))

- It transforms the data so that it’s now in the range you specified. You specify the range by passing in a tuple to the **feature_range** parameter.
- Note that, by default, it transforms the data into a range between 0 and 1 (-1 and 1, if there are negative values).
- It can be used as an alternative to The Standard Scaler or when the data is not normally distributed.
- It uses the *min* and *max* values, so it’s very sensitive to outliers. Be careful not to use it when your data has noticeable outliers.

Rescaling the features

The Min-Max Scalar:



Rescaling the features

The Robust Scalar:

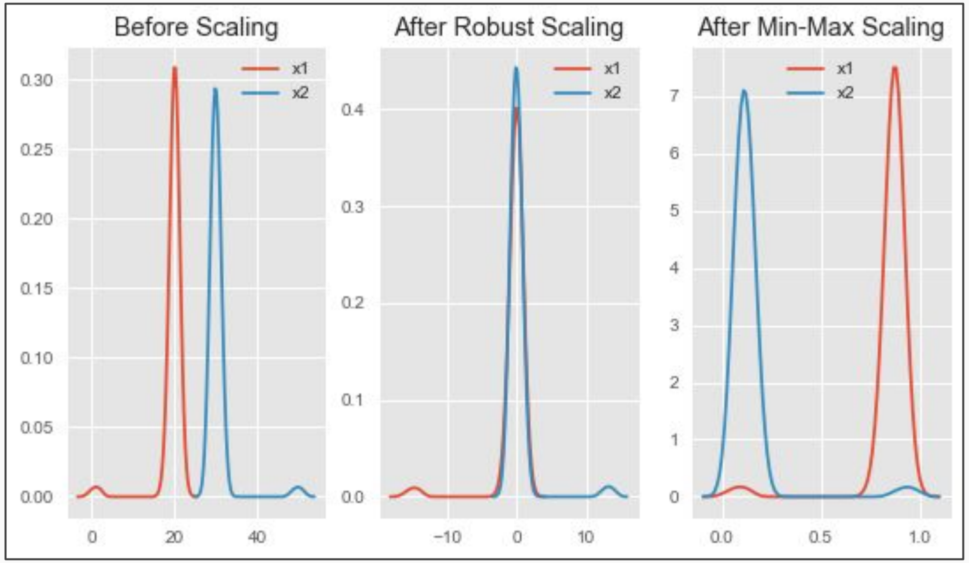
- ❖ The Robust Scaler uses statistics that are robust to outliers:

  Robust_Value = (xi–Q1(x))/(Q3(x)–Q1(x))

- ❖ It removes the median and uses the *interquartile range.* Q1 and Q3 represent quartiles.
- ❖ The **IQR** is the range between the 1st quartile and the 3rd quartile. This usage of interquartiles means that they focus on the parts where the bulk of the data is. This makes them very suitable for working with outliers.

Rescaling the features

The Robust Scalar:



❖ Notice that after Robust scaling, the distributions are brought into the same scale and overlap, but the outliers remain outside of bulk of the new distributions.However, in Min-Max scaling, the two normal distributions are kept separate by the outliers that are inside the 0 – 1 range.

Rescaling the features

Consider the dataset df and we have to rescale it. Scikit learn helps in rescaling the data:

1. Normalizer:

```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
result = norm.fit_transform(df)
```

fit_transform() method is used to rescale the data. It fits to the data and then transform it

Rescaling the features

Consider the dataset df and we have to rescale it. Scikit learn helps in rescaling the data:

2. Standard Scalar:

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

result = ss.fit_transform(df)
```

Rescaling the features

Consider the dataset df and we have to rescale it. Scikit learn helps in rescaling the data:

3. Min-Max Scalar:

```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
result = mms.fit_transform(df)
```

Rescaling the features

Consider the dataset df and we have to rescale it. Scikit learn helps in rescaling the data:

4. Robust Scalar:

```
from sklearn.preprocessing import RobustScaler
rs = RobustScaler()
result = rs.fit_transform(df)
```

Rescaling the features: When to Scale?

- **k-nearest neighbors** with an Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
- Scaling is critical, while performing **Principal Component Analysis(PCA)**. PCA tries to get the features with maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features.
- We can speed up **gradient descent** by scaling. This is because θ will descend quickly on small ranges and slowly on large ranges.
- **Tree based models** are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modelling trees.
- Algorithms like **Naive Bayes** are by design equipped to handle this and gives weights to the features accordingly. Performing a features scaling in these algorithms may not have much effect.

Rescaling the features: When to Scale?

- K-Means where clusters are made up depends on similarities calculated with an Euclidean distance measure
- Logistic regression, SVMs, perceptrons, neural networks, if we are using gradient descent optimization, otherwise some weights will update much faster than others
- Min-Max Scaling can be for KNN, Neural Networks and Gradient Descent if the data does not follows Normal Distributions.
- Standard Scaling can be useful in Linear and Logistic Regressions.
- Standard Scaling is most popular scaling method and can be used with many ML algorithms

Outliers Detection

Outliers Detection

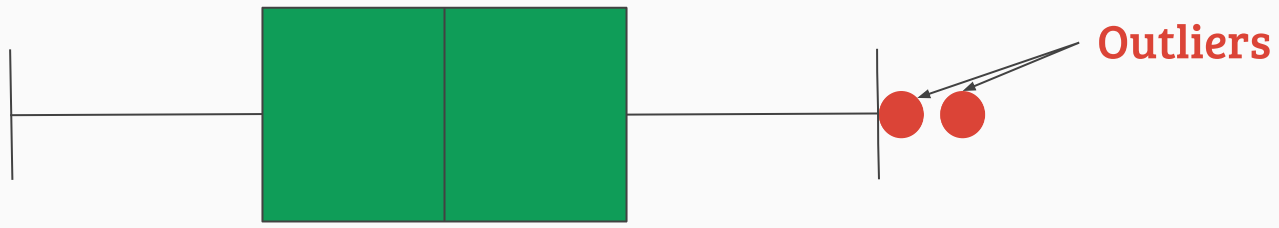
- ❖ What is Outlier?
- ❖ In statistics, an **outlier** is an observation point that is distant from other observations.
- ❖ The outliers can be a result of a mistake during data collection or it can be just an indication of variance in your data.

- ❖ Finding Outliers:
- ❖ There are two types of analysis we will follow to find the outliers- Univariate(one variable outlier analysis) and Multivariate(two or more variable outlier analysis).

Outliers Detection

Discover outliers with visualization tools:

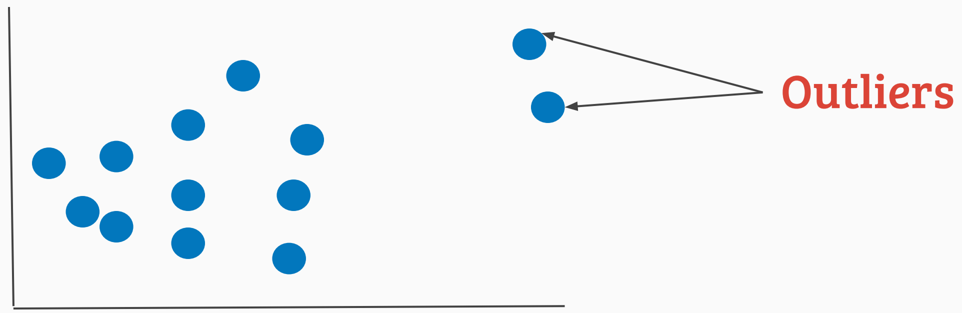
- ❖ **Box plot:**
- ❖ In descriptive statistics, a **box plot** is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have **lines extending vertically** from the boxes (whiskers) **indicating variability** outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. **Outliers** may be **plotted** as **individual** points.



Outliers Detection

Discover outliers with visualization tools:

- ❖ **Scatter plot:**
- ❖ A **scatter plot**, is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. The data are displayed as a **collection of points**, each having the value of **one variable** determining the position on the **horizontal** axis and the value of the **other variable** determining the position on the **vertical** axis.



Outliers Detection

Discover outliers with Z-Score:

- ❖ The **Z-score** is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.
- ❖ The intuition behind Z-score is to describe any data point by finding their relationship with the Standard Deviation and Mean of the group of data points. Z-score is finding the distribution of data where mean is 0 and standard deviation is 1 i.e. normal distribution.
- ❖ While calculating the Z-score we re-scale and center the data and look for data points which are too far from zero. These data points which are way too far from zero will be treated as the outliers. In most of the cases a threshold of 3 or -3 is used i.e if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

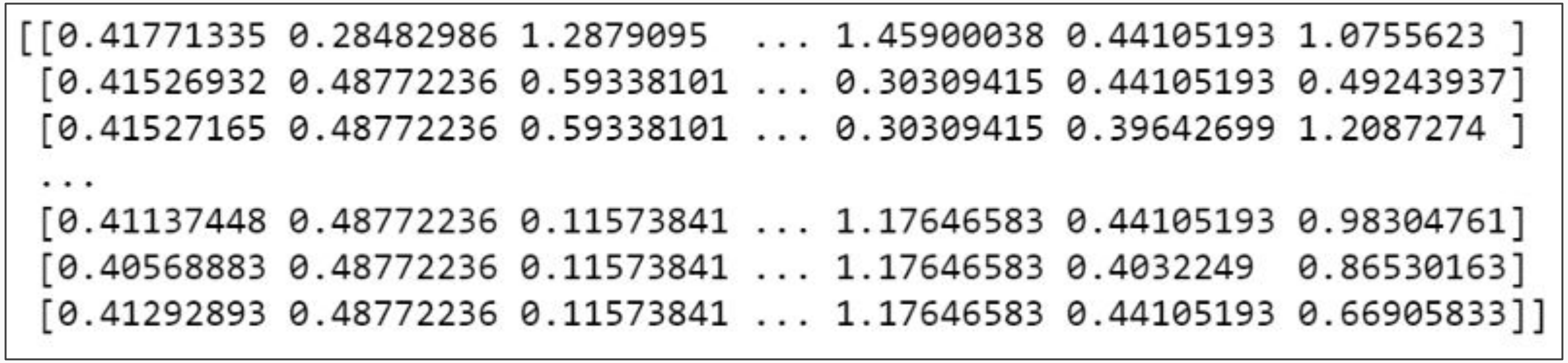
Outliers Detection

from scipy import stats

import numpy as np

z = np.abs(stats.zscore(df))

print(z)

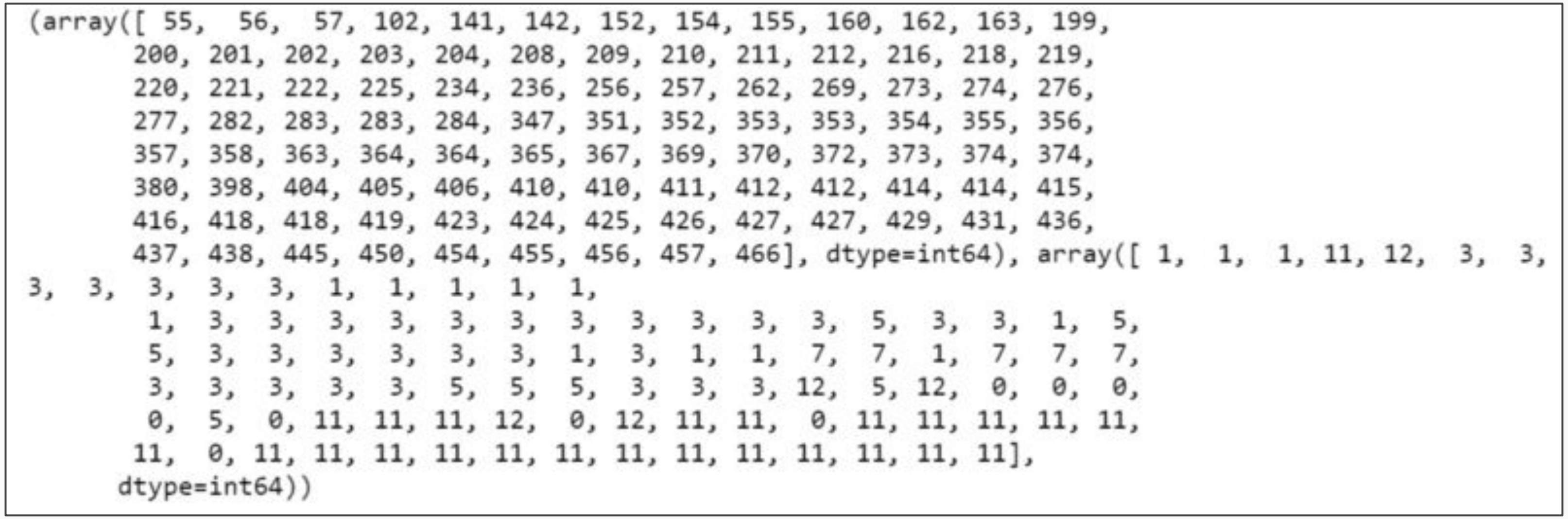


```
[[0.41771335 0.28482986 1.2879095  ... 1.45900038 0.44105193 1.0755623 ]
 [0.41526932 0.48772236 0.59338101 ... 0.30309415 0.44105193 0.49243937]
 [0.41527165 0.48772236 0.59338101 ... 0.30309415 0.39642699 1.2087274 ]
 ...
 [0.41137448 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.98304761]
 [0.40568883 0.48772236 0.11573841 ... 1.17646583 0.4032249  0.86530163]
 [0.41292893 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.66905833]]
```

Outliers Detection

Looking the code and the output above, it is difficult to say which data point is an outlier. Let’s try and define a threshold to identify an outlier.

```
threshold = 3
print(np.where(z > 3))
```



```
(array([ 55,  56,  57, 102, 141, 142, 152, 154, 155, 160, 162, 163, 199,
       200, 201, 202, 203, 204, 208, 209, 210, 211, 212, 216, 218, 219,
       220, 221, 222, 225, 234, 236, 256, 257, 262, 269, 273, 274, 276,
       277, 282, 283, 283, 284, 347, 351, 352, 353, 353, 354, 355, 356,
       357, 358, 363, 364, 364, 365, 367, 369, 370, 372, 373, 374, 374,
       380, 398, 404, 405, 406, 410, 410, 411, 412, 412, 414, 414, 415,
       416, 418, 418, 419, 423, 424, 425, 426, 427, 427, 429, 431, 436,
       437, 438, 445, 450, 454, 455, 456, 457, 466], dtype=int64), array([ 1,  1,  1, 11, 12,  3,  3,
3,  3,  3,  3,  3,  1,  1,  1,  1,  1,
        1,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  5,  3,  3,  1,  5,
        5,  3,  3,  3,  3,  3,  3,  1,  3,  1,  1,  7,  7,  1,  7,  7,  7,
        3,  3,  3,  3,  3,  5,  5,  5,  3,  3,  3, 12,  5, 12,  0,  0,  0,
        0,  5,  0, 11, 11, 11, 12,  0, 12, 11, 11,  0, 11, 11, 11, 11, 11,
       11,  0, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11],
      dtype=int64))
```

Outliers Detection

The first array contains the list of row numbers and second array respective column numbers, which mean z[55][1] have a Z-score higher than 3.

```
print(z[55][1])
```

```
Output: 3.375038763517309
```

This will be our outlier

Outliers Detection

Discover outliers IQR:

❖ Box plot use the IQR method to display data and outliers(shape of the data) but in order to be get a list of identified outlier, we will need to use the mathematical formula and retrieve the outlier data.

❖ The **interquartile range** (**IQR**), also called the **midspread** or **middle 50%**, or technically **H-spread**, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, IQR = Q3 − Q1.

❖ In other words, the IQR is the first quartile subtracted from the third quartile; these quartiles can be clearly seen on a box plot on the data.

❖ It is a measure of the dispersion similar to standard deviation or variance, but is much more robust against outliers.

❖ IQR is somewhat similar to Z-score in terms of finding the distribution of data and then keeping some threshold to identify the outlier.

Outliers Detection

- **Discover outliers IQR:**

Using formula we can detect outliers:

$C1 = Q3 + 1.5*IQR$

$C2 = Q1 - 1.5*IQR$

Any value below C2 or above C1 will treat as an outlier

Outliers Detection

Suppose we have age column in dataset df and we have to find outliers:

```
import numpy as np

Q1 = df['age'].quantile(0.25)
Q3 = df['age'].quantile(0.75)

# alternative
# Q1 = np.percentile(df['age'], 25)
# Q3 = np.percentile(df['age'], 75)

IQR = Q3 - Q1
print(df['age'] < (Q1 - 1.5 * IQR)) | (df['age'] > (Q3 + 1.5 * IQR))
```

Result will be contains values which can be treated as an outliers

Thanks!

Signitive Technologies

Sneh Nagar, Behind ICICI Bank,
Chhatrapati Square, Nagpur 15

Landmark:
Bharat Petrol Pump
Chhatrapati Square

Contact: 9011033776
www.signitivetech.com

“Keep Learning, Happy Learning”

Best Luck!

Have a Happy Future