Practical #8: Anomaly Detection with Python

Objective:

To consolidate the understanding of anomaly/outlier scores and anomaly detection with Python.

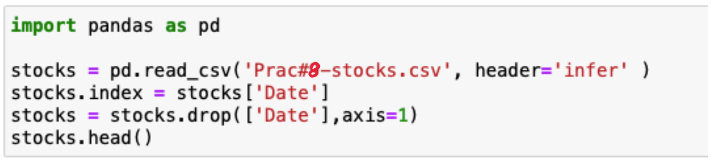
Submission:

- *What to submit*: The saved document (**.html**) containing the required steps done in this practical.
- *Deadline of the submission*: 11:59PM (Adelaide Time), Tuesday of **Week 11**.
- *Submission link*: "**Submission Link of Prac #8"** in **Week 10 section** on Learnonline course site.
- *Marks:* Prac#8 (part of the ongoing assessment of the course) is worth 2% of the total marks of the course.

Anomaly Detection with Python:

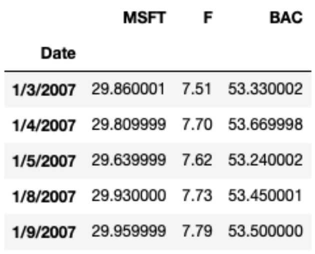
Anomaly detection can identify instances whose characteristics differ significantly from the rest of the data. Here, we will apply a distance-based anomaly detection technique using Python and its library packages.

1. In this practical, our goal is to detect anomalous changes in the daily closing prices of various stocks. The input data ("**Prac#8-stocks.csv**") contains the historical (represented as "date" information) closing prices of stocks for 3 large corporations (Microsoft (MSFT), Ford Motor Company (F), and Bank of America (BAC)). Note: you need to download the dataset ("**Prac#8-stocks.csv**") from Week 10 section on Learnonline course site and upload it to the Jupyter Notebook home folder.



```
import pandas as pd

stocks = pd.read_csv('Prac#8-stocks.csv', header='infer' )
stocks.index = stocks['Date']
stocks = stocks.drop(['Date'],axis=1)
stocks.head()
```



	MSFT	F	BAC
Date			
1/3/2007	29.860001	7.51	53.330002
1/4/2007	29.809999	7.70	53.669998
1/5/2007	29.639999	7.62	53.240002
1/8/2007	29.930000	7.73	53.450001
1/9/2007	29.959999	7.79	53.500000

2. We can compute the percentage of changes in the daily closing price of each stock as follows:

$$\Delta(t) = 100 \times \frac{x_t - x_{t-1}}{x_{t-1}}$$

where x_t denotes the price of a stock on day *t,* and x_{t-1} denotes the price on its previous day *t-1.*

The formula above can be converted into Python code as follows,

```
delta = pd.DataFrame(100*np.divide(stocks.iloc[1:,:].values-stocks.iloc[:N-1,:].values,
                         stocks.iloc[:N-1,:].values),
                         columns=stocks.columns,
                         index=stocks.iloc[1:].index)
```

The Python code for this step is shown as follows,

```
import numpy as np

N,d = stocks.shape
delta = pd.DataFrame(100*np.divide(stocks.iloc[1:,:].values-stocks.iloc[:N-1,:].values,
                                   stocks.iloc[:N-1,:].values),
                                   columns=stocks.columns,
                                   index=stocks.iloc[1:].index)
delta.head()
```

Date	MSFT	F	BAC
1/4/2007	-0.167455	2.529960	0.637532
1/5/2007	-0.570278	-1.038961	-0.801185
1/8/2007	0.978411	1.443570	0.394438
1/9/2007	0.100231	0.776197	0.093543
1/10/2007	-1.001332	-0.770218	0.149536

3. Now, we can deploy the *k*-nearest neighbour (such as *k*-means) approach to calculate the anomaly score based on the percentage of changes in the daily closing price of each stock (delta). Specifically, a normal instance is expected to have a small distance to its *k*-th nearest neighbour, whereas an anomaly is likely to have a large distance to its *k*-th nearest neighbour. For instance, we can apply the distance-based approach with ***k*=4** to identify the anomalous trading days from the stock market data.

   Please note the **NearestNeighbors** function and **kneighbors** comes from **sklearn.neighbors** library, which can be used for calculating distances and written in Python code as follow:

   nbrs = NearestNeighbors(n_neighbors=knn, metric=distance.euclidean).fit(delta.to_numpy())

   distances, indices = nbrs.kneighbors(delta.to_numpy())

   We must import the required library packages and define variables and formulas in Python as follows.

```
from sklearn.neighbors import NearestNeighbors
import numpy as np
from scipy.spatial import distance

knn = 4
nbrs = NearestNeighbors(n_neighbors=knn, metric=distance.euclidean).fit(delta.to_numpy())
distances, indices = nbrs.kneighbors(delta.to_numpy())

anomaly_score = distances[:,knn-1]
```

   Then, we can present the first 5 largest annoy scores as Table format in Python as follows.

```
anom = pd.DataFrame(anomaly_score, index=delta.index, columns=['Anomaly score'])
result = pd.concat((delta,anom), axis=1)
result.nlargest(5,'Anomaly score')
```

Date	MSFT	F	BAC	Anomaly score
10/13/2008	18.604651	20.100503	9.199808	15.642827
11/26/2008	2.501251	29.518072	4.256757	14.212749
10/7/2008	-6.744279	-20.867209	-26.225949	13.751302
11/28/2008	-1.317721	25.116279	5.314323	13.139586
9/30/2008	6.717317	24.700240	15.702479	12.599739

Save the .html document as your submission by following the instruction given on page 1 of the practical document.