CSE 572: Lab 18

In this lab, you will practice implementing anomaly detection (also known as outlier detection) techniques for a real-world dataset. Anomaly detection is the task of identifying instances whose characteristics differ significantly from the rest of the data. You should refer to the lecture slides and Chapter 9 of the "Introduction to Data Mining" book to understand some of the concepts in this tutorial.

Acknowledgment: This notebook was adapted from Introduction to Data Mining, 2nd Edition. Tan, Steinbach, Karpatne, Kumar.

To execute and make changes to this notebook, click File > Save a copy to save your own version in your Google Drive or Github. Read the step-by-step instructions below carefully. To execute the code, click on each cell below and press the SHIFT-ENTER keys simultaneously or by clicking the Play button.

When you finish executing all code/exercises, save your notebook then download a copy (.ipynb file). Submit the following **three** things:

1. a link to your Colab notebook,
2. the .ipynb file, and
3. a pdf of the executed notebook on Canvas.

To generate a pdf of the notebook, click File > Print > Save as PDF.

Model-based statistical approaches

This approach assumes that the majority of the data instances are governed by some well-known probability distribution, e.g., Binomial or Gaussian distribution. Anomalies can then be detected as observations that do not fit the overall distribution of the data.

In this example, our goal is to detect anomalous changes in the daily closing prices of various stocks. The input data *stocks.csv* contains the historical closing prices of stocks for 3 large corporations (Microsoft, Ford Motor Company, and Bank of America).

```
import pandas as pd

stocks = pd.read_csv('https://docs.google.com/uc?export=download&id=1UqHZmlfSoPDcZlTIr2TB6OadBhni9Kbv', header='infer')
stocks

stocks.index = stocks['Date']
stocks = stocks.drop(['Date'], axis=1)
stocks.head()
```

Date	MSFT	F	BAC
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

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, $t-1$.

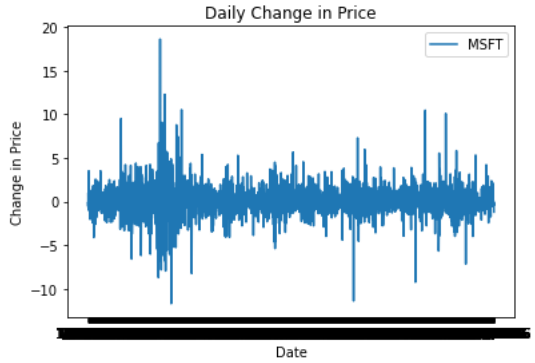
```
import numpy as np

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

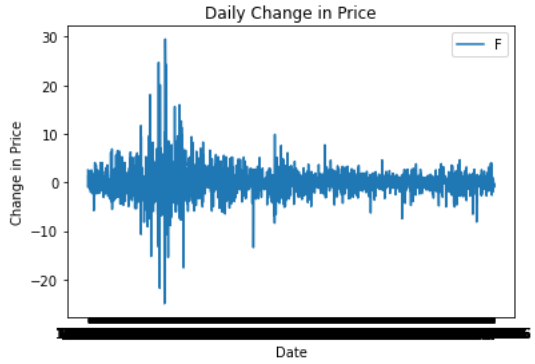
```
import pandas as pd
import matplotlib.pyplot as plt
```

Plot the daily change in price as a line for each of the companies.

```
# MSFT - YOUR CODE HERE

plt.plot(delta['MSFT'], label='MSFT')


# Add labels and legend to the plot
plt.title('Daily Change in Price')
plt.xlabel('Date')
plt.ylabel('Change in Price')
plt.legend()

# Show the plot
plt.show()
```

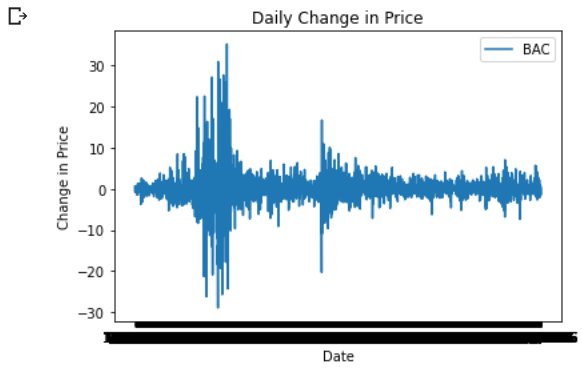



```
# F - YOUR CODE HERE
plt.plot(delta['F'], label='F')
plt.title('Daily Change in Price')
plt.xlabel('Date')
plt.ylabel('Change in Price')
plt.legend()

# Show the plot
plt.show()
```



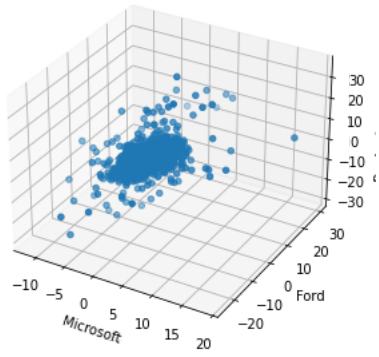

```
# BAC - YOUR CODE HERE
plt.plot(delta['BAC'], label='BAC')
```

```
plt.title('Daily Change in Price')
plt.xlabel('Date')
plt.ylabel('Change in Price')
plt.legend()

# Show the plot
plt.show()
```



We can also plot the *distribution* of the percentage daily changes in stock price for all three features (treated as attributes) as a 3D scatter plot.

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
%matplotlib inline

fig = plt.figure(figsize=(8,5))
ax = plt.axes(projection='3d')
ax.scatter(delta.MSFT, delta.F, delta.BAC)
ax.set_xlabel('Microsoft')
ax.set_ylabel('Ford')
ax.set_zlabel('Bank of America')
plt.show()
```



Assuming the data follow a multivariate Gaussian distribution, we can compute the mean and covariance matrix of the 3-dimensional data. Compute and print these in the cell below. Name your variables `mean` and `cov`.

```
# YOUR CODE HERE
mean = np.mean(delta.values, axis=0)
cov = np.cov(delta.values.T)
print("Mean vector:\n", mean)
print("\nCovariance matrix:\n", cov)
```

```
Mean vector:
 [0.04500277 0.06137418 0.03335074]

Covariance matrix:
 [[ 3.19167433  2.1363511   2.78886962]
 [ 2.1363511   8.52494366  4.99740498]
 [ 2.78886962  4.99740498 13.7707607 ]]
```

To determine the anomalous trading days, we can compute the Mahalanobis distance between the percentage of price change on each day against the mean percentage of price change.

$$\text{Mahalanobis}(x) = (x - \mu_x)\Sigma^{-1}(x - \mu_x)^T$$

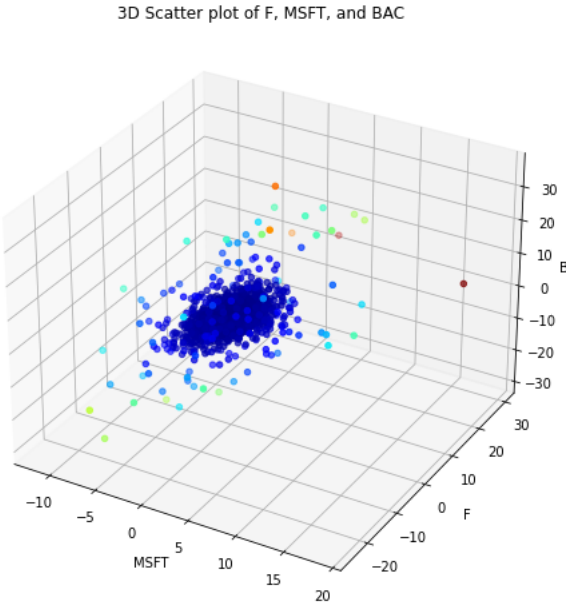
where x is assumed to be a row vector, μ_x is the feature-wise mean of x, and Σ^{-1} is the inverted covariance matrix of x.

```
from numpy.linalg import inv

# Invert the covariance matrix
cov_inv = np.linalg.inv(cov)

# Function to compute the Mahalanobis distance for one sample
def mahalanobis(row):
    sub = row - mean
    return np.matmul(sub, cov_inv).dot(sub)

# Compute the Mahalanobis distance for every sample
anomaly_scores = np.apply_along_axis(mahalanobis, axis=1, arr=delta)

# Result is one score for each sample
anomaly_scores.shape
```

```
(2517,)
```

Plot the dataset as a 3d scatter plot again, but this time color the points by their anomaly score. Use `cmap=jet` for the colormap.

```
# YOUR CODE HERE
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8,8))
ax = plt.axes(projection='3d')
ax.scatter(delta['MSFT'],delta['F'], delta['BAC'], cmap='jet', c=anomaly_scores)

# Add labels and title
ax.set_xlabel('MSFT')
ax.set_ylabel('F')
ax.set_zlabel('BAC')
ax.set_title('3D Scatter plot of F, MSFT, and BAC')

plt.show()
```



The top anomalies are shown as brownish-orange points in the figure above. To see which samples the top anomalies correspond to, create a new dataframe named `result` that appends the anomaly scores as a new column to `delta`.

```
# YOUR CODE HERE
```

```
result = delta.copy()
result['Anomaly score'] = anomaly_scores
```

Now we can print the N samples (here, 10 samples) with the largest anomaly scores using the `nlargest()` function.

```
result.nlargest(10, 'Anomaly score')
```

Date	MSFT	F	BAC	Anomaly score
10/13/2008	18.604651	20.100503	9.199808	126.992257
11/26/2008	2.501251	29.518072	4.256757	121.360808
4/9/2009	2.501297	7.341772	35.269122	100.466343
11/28/2008	-1.317721	25.116279	5.314323	98.117362
1/21/2009	4.870124	-1.877934	30.980392	95.630501
11/19/2008	-6.778797	-25.000000	-14.022383	73.876496
9/30/2008	6.717317	24.700240	15.702479	71.393091
1/20/2009	-6.240482	-2.739726	-28.969359	71.165307
12/8/2008	5.737287	24.264706	17.060367	69.706051
10/7/2008	-6.744279	-20.867209	-26.225949	69.513349

Double-click (or enter) to edit

See the Wikipedia article for the global financial crisis for context on what happened during these dates:
https://en.wikipedia.org/wiki/Global_financial_crisis_in_October_2008

Question 1: We can see that the top anomaly corresponds to the sample from October 13, 2008, which represents the change between October 12-13, 2008. According to the Wikipedia article, what happened on this date?

Answer:

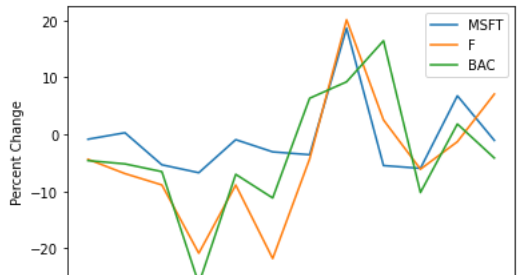
YOUR ANSWER HERE There were several events happened:

1. Governments and central banks from all over the world took a number of synchronized moves on October 12–13, 2008, to confront the growing global financial crisis.
2. Plans for bank recapitalization, deposit guarantees, interest rate reductions, and the addition of liquidity to the financial system were a few of these measures. These steps were taken in an effort to stabilize the banking industry and avert the financial system's collapse, which had been put at risk by Lehman Brothers' demise in September 2008.
3. Investors' positive response to these concerted activities and the expectation that they would help lessen the consequences of the financial crisis certainly led to the market anomalies, such as the record increase in the Dow Jones Industrial Average and the rise of European stock markets.
4. Several European nations announced plans to recapitalize their banks and guarantee bank deposits for five years on October 13, 2008.
5. The G7 nations committed to helping financially institutions that are crucial to the system and preventing their failure. Global stock markets increased, with the Dow Jones industrial average posting a record increase of 936.42 points, or 11.08%, to set a new high.
6. The process of nationalization was launched when the UK government invested £37 billion in the three biggest banks in the country. Italy and the Netherlands agreed to step in as needed to stop bank collapses in their nations in exchange for the European Central Bank's weekly infusions of unrestricted euro cash into the credit market.

The plot below visualizes what happened to the stock prices during the first couple weeks of October 2008.

```
fig, ax = plt.subplots(1)

delta[440:452].plot(ax=ax)
ax.set_ylabel('Percent Change')
```

```
Text(0, 0.5, 'Percent Change')
```



The second largest anomaly corresponds to the sample from November 26, 2008. We plot the time series around that date below.

```
fig, ax = plt.subplots(1)

delta[475:488].plot(ax=ax)
ax.set_ylabel('Percent Change')
```

```
Text(0, 0.5, 'Percent Change')
```

Distance-based (model free) approaches

This is a model-free anomaly detection approach as it does not require constructing an explicit model of the normal class to determine the anomaly score of data instances. The example code shown below employs the k-nearest neighbor approach to calculate anomaly score.

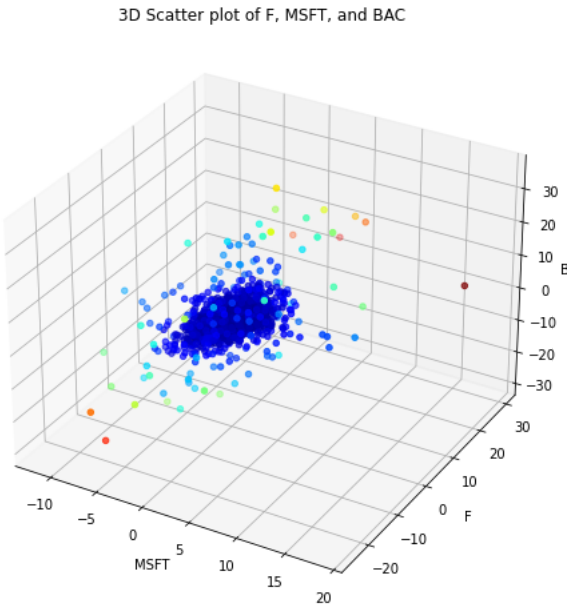
Specifically, a normal instance is expected to have a small distance to its k-th nearest neighbor whereas an anomaly is likely to have a large distance to its k-th nearest neighbor.

In the example below, we apply the distance-based approach with k=4 to identify the anomalous trading days from the stock market data described in the previous section.

```
from sklearn.neighbors import NearestNeighbors
from scipy.spatial import distance

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

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

Plot the dataset as a 3d scatter plot again with points colored by their anomaly score computed using the distance to k-nearest neighbors approach. Use `cmap=jet` for the colormap.

```
# YOUR CODE HERE
fig = plt.figure(figsize=(8,8))
ax = plt.axes(projection='3d')
ax.scatter(delta['MSFT'], delta['F'], delta['BAC'], cmap='jet', c=anomaly_score)

# Add labels and title
ax.set_xlabel('MSFT')
ax.set_ylabel('F')
ax.set_zlabel('BAC')
ax.set_title('3D Scatter plot of F, MSFT, and BAC')

plt.show()
```



The results are slightly different than the results from the previous section because we have used Euclidean distance (instead of Mahalanobis distance) to detect the anomalies.

We can examine the dates associated with the top-10 highest anomaly scores as follows.

```
anom = pd.DataFrame(anomaly_score, index=delta.index, columns=['Anomaly score'])
result = pd.concat((delta, anom), axis=1)
result.nlargest(10, '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
11/19/2008	-6.778797	-25.000000	-14.022383	12.375693
12/8/2008	5.737287	24.264706	17.060367	11.220257
4/9/2009	2.501297	7.341772	35.269122	10.440667
7/16/2008	4.244742	18.064516	22.408207	9.752602
1/21/2009	4.870124	-1.877934	30.980392	9.646659

✓ 0s completed at 6:06 AM