Assignment 4 [part 1 of 2]

Questions 1B and 2 are coding exercises. Questions 1A and 3 are pen-and-paper exercises.

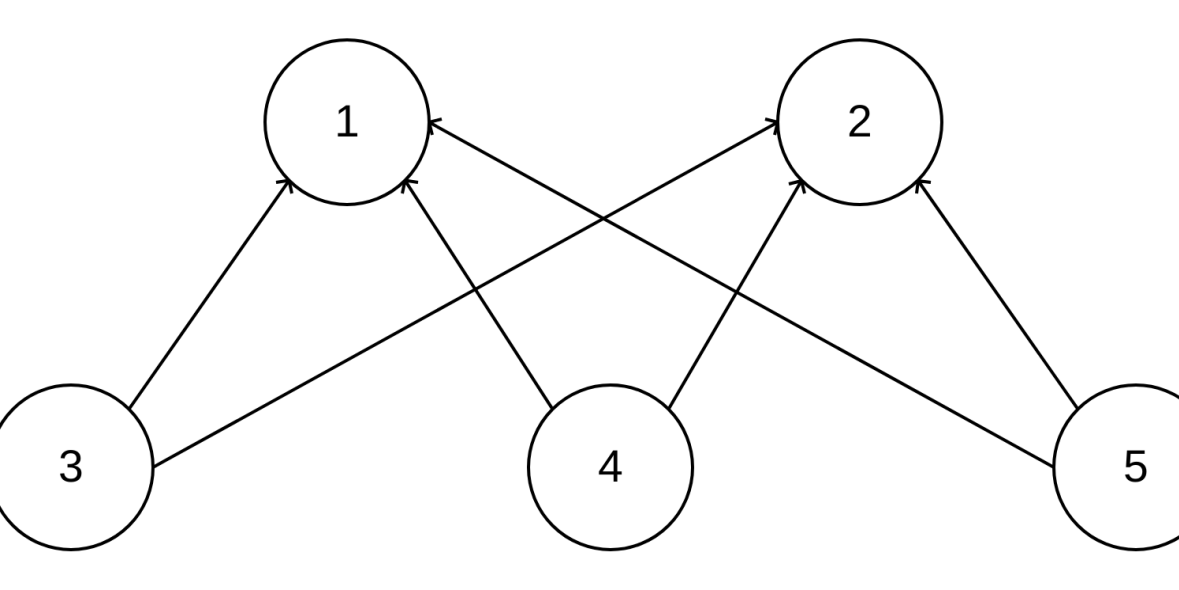
The supplementary material for this week contains offline copies of the assignment material in case there are internet connectivity issues with School websites. The supplementary material includes file "income_table.html" (for Question 1), file "programmes.html" (for Question 2), and file "Question-3-Graph.png" (for Question 3).

1. You are provided with the following URL: http://eecs.qmul.ac.uk/~emmanouilb/income_table.html. This webpage includes a table on individuals' income and shopping habits - the same that was used in the Week 3 lab.

   A. Inspect the HTML code of the above URL, and provide a short report on the various tags present in the code. What is the function of each unique tag present in the HTML code? [0.5 marks out of 5]
   B. Using Beautiful Soup, scrape the table and convert it into a pandas dataframe. Perform data cleaning when necessary to remove extra characters (no need to handle missing values). In the report include the code that was used to scrape and convert the table and provide evidence that the table has been successfully scraped and converted (e.g. by displaying the contents of the dataframe). [1 mark out of 5]
2. The list of the various MSc programmes offered by the School of EECS is provided at the following URL: http://eecs.qmul.ac.uk/postgraduate/programmes/. Perform web scraping on the table present in the above URL and convert it into a pandas dataframe that would include one row for each programme of study as shown in the webpage. The dataframe should include the following 5 columns: name of postgraduate degree programme (e.g. Advanced Electronic and Electrical Engineering), programme code for part-time study (e.g. H60C), programme code for full-time study (e.g. H60A), URL for part-time study programme details, and URL for full-time study programme details. Perform data cleaning to remove unecessary characters when needed. In the report include the code that was used to scrape, convert and clean the table and provide evidence that the table has been successfully scraped (e.g. by displaying the contents of the dataframe). [1 mark out of 5]

3. Consider the graph in the figure below as displaying the links for a group of 5 webpages.

   A. Which of the 5 nodes would you consider hubs and which would you consider authorities? [0.5 marks out of 5]
   B. Assume that this graph is to be used as input to the PageRank algorithm. Add transitions with a uniform probability distribution in the case of dead-end nodes (do not consider cases of dead-end components). Then, calculate the transition probabilities p_{ji} for all 5 nodes in the graph (where i and j take values between 1 to 5). [1 mark out of 5].
   C. Using the modified graph of question 3B, derive the PageRank $\pi(i)$ for all nodes, where $i = \{1, \ldots, 5\}$ corresponds to the node index. Assume that the teleportation probability is set to α. [1 mark out of 5]



Solution for question 1A (0.5 marks)

The HTML code includes the following tags: html (marking the start/end of the html document itself), body (marking the visible part of the html document), h1 (indicating a heading), p (indicating a paragraph), table (indicating a table), thead (indicating a table header), tbody (indicating a table body), tr (indicating a table row), th (indicating a table header cell), and td (indicating a table cell in the main body of the table).

In [4]:

```
# Solution for question 1B

# marking scheme:
# 0.25 marks for scraping the table header and body
# 0.25 marks for converting the table into a dataframe
# 0.25 marks for performing data cleaning to remove extra characters
# 0.25 marks for displaying the scraped & cleaned table

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from urllib.request import urlopen
from bs4 import BeautifulSoup

url_question1 = "http://eecs.qmul.ac.uk/~emmanouilb/income_table.html"
html_question1 = urlopen(url_question1)
soup_question1 = BeautifulSoup(html_question1, 'lxml')

# Scrape table header
question1_header_list = []
col_labels = soup_question1.find_all('th')
col_str = str(col_labels)
cleantext_header = BeautifulSoup(col_str, "lxml").get_text()
question1_header_list.append(cleantext_header)

# Scrape table body
question1_table_list = []
question1_rows = soup_question1.find_all('tr')
for row in question1_rows:
    row_td = row.find_all('td')
    row_cells = str(row_td)
    row_cleantext = BeautifulSoup(row_cells, "lxml").get_text()
    question1_table_list.append(row_cleantext)

# Convert table header into a dataframe -
df_header_question1 = pd.DataFrame(question1_header_list)
df_header_question1 = df_header_question1[0].str.split(',', expand=True)

# Convert table body into a dataframe
df_table_question1 = pd.DataFrame(question1_table_list)
df_table_question1 = df_table_question1[0].str.split(',', expand=True)

# Concatenate the two dataframes
frames = [df_header_question1, df_table_question1]
df_question1 = pd.concat(frames)

# Perform data cleaning to remove extra characters
df_question1[0] = df_question1[0].str.strip('[')
df_question1[3] = df_question1[3].str.strip(']')
#  Assign the first row to be the dataframe header
df_question1 = df_question1.rename(columns=df_question1.iloc[0])
df_question1 = df_question1.drop(df_question1.index[0])

# Display evidence that the table has been scraped, converted, and cleaned successgfu
df_question1.head(10)
```

Out[4]:

	Region	Age	Income	Online Shopper
1	India	49	86400	No
2	Brazil	32	57600	Yes
3	USA	35	64800	No
4	Brazil	43	73200	No
5	USA	45		Yes
6	India	40	69600	Yes
7	Brazil		62400	No
8	India	53	94800	Yes
9	USA	55	99600	No
10	India	42	80400	Yes

In [5]:

```
# Solution for question 2

# marking scheme:
# 0.25 marks for scraping the table header and body
# 0.25 marks for converting the table into a dataframe
# 0.25 marks for performing data cleaning to remove extra characters and converting U
# 0.25 marks for displaying the scraped & cleaned table

url_question2 = "http://eecs.qmul.ac.uk/postgraduate/programmes/"
html_question2 = urlopen(url_question2)
soup_question2 = BeautifulSoup(html_question2, 'lxml')

# Scrape table header
question2_header_list = []
col_labels = soup_question2.find_all('th')
col_str = str(col_labels)
cleantext_header = BeautifulSoup(col_str, "lxml").get_text()
question2_header_list.append(cleantext_header)

# Scrape table body
question2_table_list = []
question2_url_list = []
question2_rows = soup_question2.find_all('tr')
for row in question2_rows:
    row_td = row.find_all('td')
    row_cells = str(row_td)
    row_cleantext = BeautifulSoup(row_cells, "lxml").get_text()
    programme_urls = [link.get('href') for link in row.findAll('a')]
    question2_table_list.append(row_cleantext)
    question2_url_list.append(programme_urls)

# Convert table header into a dataframe
df_header_question2 = pd.DataFrame(question2_header_list)
df_header_question2 = df_header_question2[0].str.split(',', expand=True)

# Convert table body into a dataframe
df_table_question2 = pd.DataFrame(question2_table_list)
df_table_question2 = df_table_question2[0].str.split(',', expand=True)

# Concatenate the two dataframes
frames = [df_header_question2, df_table_question2]
df_question2 = pd.concat(frames)

# Perform data cleaning to remove extra characters
df_question2[0] = df_question2[0].str.strip('[')
df_question2[0] = df_question2[0].str.strip(']')
df_question2[2] = df_question2[2].str.strip(']')

#  Assign the first row to be the dataframe header
df_question2 = df_question2.rename(columns=df_question2.iloc[0])
df_question2 = df_question2.drop(df_question2.index[0])

# Convert URLs into a dataframe
df_url_question2 = pd.DataFrame(question2_url_list)
df_url_question2 = df_url_question2.drop(df_url_question2.index[0])
df_url_question2.columns = ['URL (part-time)', 'URL (full-time)']

# Merge table dataframe with URL dataframe, and name URL columns
df_final = pd.concat([df_question2, df_url_question2], axis=1)
df_final.rename(columns={"0": "URL (part-time)", "1": "URL (full-time)"})
df_final.rename(columns={0: "URL (part-time)", 1: "URL (full-time)"})
#df_final.head(10)

# When there are missing values in the table, move URLs in correct column
df_final[" Part-time(2 year)"] = df_final[" Part-time(2 year)"].str.strip()
locs = df_final[df_final[" Part-time(2 year)"] == ''].index.values
for locations in locs:
    df_final.at[locations, "URL (full-time)"] = df_final.at[locations, "URL (part-time
    df_final.at[locations, "URL (part-time)"] = ''

# Display evidence that web scraping and porting into a dataframe has worked
df_final.head(10)
```

Out[5]:

	Postgraduate degree programmes	Part-time(2 year)	Full-time(1 year)	URL (part-time)	
1	Advanced Electronic and Electrical Engineering	H60C	H60A	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
2	Artificial Intelligence	I4U2	I4U1	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
3	Big Data Science	H6J6	H6J7	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
4	Computer Games		I4U4		https://www.qmul.ac.uk/postgradua
5	Computer Science	G4U2	G4U1	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
6	Computer Science by Research	G4Q2	G4Q1	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
7	Computing and Information Systems	G5U6	G5U5	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
8	Data Science and Artificial Intelligence by Co...		I4U5		https://www.qmul.ac.uk/postgradua
9	Electronic Engineering by Research	H6T6	H6T5	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua
10	Internet of Things (Data)	I1T2	I1T0	https://www.qmul.ac.uk/postgraduate/taught/cou...	https://www.qmul.ac.uk/postgradua

Solution for Question 3

A. Nodes 3,4,5 are considered hubs, since they have out-links to authorities *(0.25 marks)*. Nodes 1,2 are considered authorities, since they have in-links *(0.25 marks)*.

B. *0.2 marks for correctly stating the probabilities for each node*

Node 1 (dead end): $p_{11} = 0.2$, $p_{12} = 0.2$, $p_{13} = 0.2$, $p_{14} = 0.2$, $p_{15} = 0.2$.

Node 2 (dead end): $p_{21} = 0.2$, $p_{22} = 0.2$, $p_{23} = 0.2$, $p_{24} = 0.2$, $p_{25} = 0.2$.

Node 3: $p_{31} = 0.5$, $p_{32} = 0.5$, $p_{33} = 0$, $p_{34} = 0$, $p_{35} = 0$.

Node 4: $p_{41} = 0.5$, $p_{42} = 0.5$, $p_{43} = 0$, $p_{44} = 0$, $p_{45} = 0$.

Node 5: $p_{51} = 0.5$, $p_{52} = 0.5$, $p_{53} = 0$, $p_{54} = 0$, $p_{55} = 0$.

C. *0.2 marks for correctly stating the PageRank equations of each node*

Using the equation from slide 46, using the above computed transition probabilities, and setting $n = 5$:

Node 1: $\pi(1) = \alpha/5 + (1-\alpha) \cdot (0.2\pi(1) + 0.2\pi(2) + 0.5\pi(3) + 0.5\pi(4) + 0.5\pi(5))$

Node 2: $\pi(2) = \alpha/5 + (1-\alpha) \cdot (0.2\pi(1) + 0.2\pi(2) + 0.5\pi(3) + 0.5\pi(4) + 0.5\pi(5))$

Node 3: $\pi(3) = \alpha/5 + (1-\alpha) \cdot (0.2\pi(1) + 0.2\pi(2) + 0\pi(3) + 0\pi(4) + 0\pi(5))$

Node 4: $\pi(4) = \alpha/5 + (1-\alpha) \cdot (0.2\pi(1) + 0.2\pi(2) + 0\pi(3) + 0\pi(4) + 0\pi(5))$

Node 5: $\pi(5) = \alpha/5 + (1-\alpha) \cdot (0.2\pi(1) + 0.2\pi(2) + 0\pi(3) + 0\pi(4) + 0\pi(5))$

Assignment 4 [part 2 of 2]

Questions 1-2 are pen-and-paper exercises; questions 3-5 are coding exercises. For all your answers please show your workings (equations or code when applicable).

1. Consider the following sentences related to data mining theory, and assume that each of the below sentences corresponds to a different document:

   - Data refers to characteristics that are collected through observation.
   - A dataset can be viewed as a collection of objects.
   - Data objects are described by a number of attributes.
   - An attribute is a characteristic or feature of an object.

   A. Construct and display the document-term matrix for the above documents. Remove all stop words (here consider as stop words: articles, prepositions, conjunctions, pronouns, and common verbs) and punctuation marks; convert any plural nouns/adjectives to their singular form; and convert verbs to the present tense and first person singular form, before you construct the matrix. [1 mark out of 5]
   B. Using the above constructed document-term matrix, calculate the inverse document frequency $idf(w)$ for all words w you have identified from question 1(a). [1 mark out of 5]
2. Consider the following timeseries $y = \{0.1, 0.15, 0.2, 0.2, 0.3, 0.4, 0.25, 0.6, 0.5\}$. Perform timeseries binning using $k = 3$ values per bin, and show the resulting timeseries after binning. [0.5 marks out of 5]

3. Load CSV file "timeseries.csv", which contains a univariate timeseries. Once loaded, convert the timeseries into a numpy array and use the numpy flatten() function to ensure that the loaded timeseries is one-dimensional. Compute the Discrete Fourier Transform (DFT) of the timeseries, and display plots for both the original timeseries and the magnitude of its DFT. How many predominant frequency components does the timeseries have? [1 mark out of 5]

4. Using the daily births dataset from this lab tutorial, smooth the timeseries using trailing moving average smoothing and a window size that corresponds to one week; then replace any NaN values with zeros. Perform timeseries forecasting using the smoothed dataset in order to predict daily births for the first 5 days of 1960, using the models below. Show your forecasting results. [0.75 marks out of 5]

   - AR model with $p = 2$
   - ARMA model with $p = 2$ and $q = 2$
5. Using a similar process used in section 1 of this lab notebook, perform document clustering using k-means on the following wikipedia articles: anomaly detection, cluster analysis, k-means clustering, data mining, data warehouse, association rule learning. As with section 1, use the elbow metric to find an appropriate number of clusters. Discuss and display the document clustering results. [0.75 marks out of 5]

Solution for Question 1

	data	refers	characteristics	collect	observation	dataset	view	collection	object	describe	num
Document 1	1	1	1	1	1	0	0	0	0	0	0
Document 2	0	0	0	0	0	1	1	1	1	0	0
Document 3	1	0	0	0	0	0	0	0	1	1	1
Document 4	0	0	1	0	0	0	0	0	1	0	0
IDF	0.3	0.6	0.3	0.6	0.6	0.6	0.6	0.6	0.12	0.6	0.6

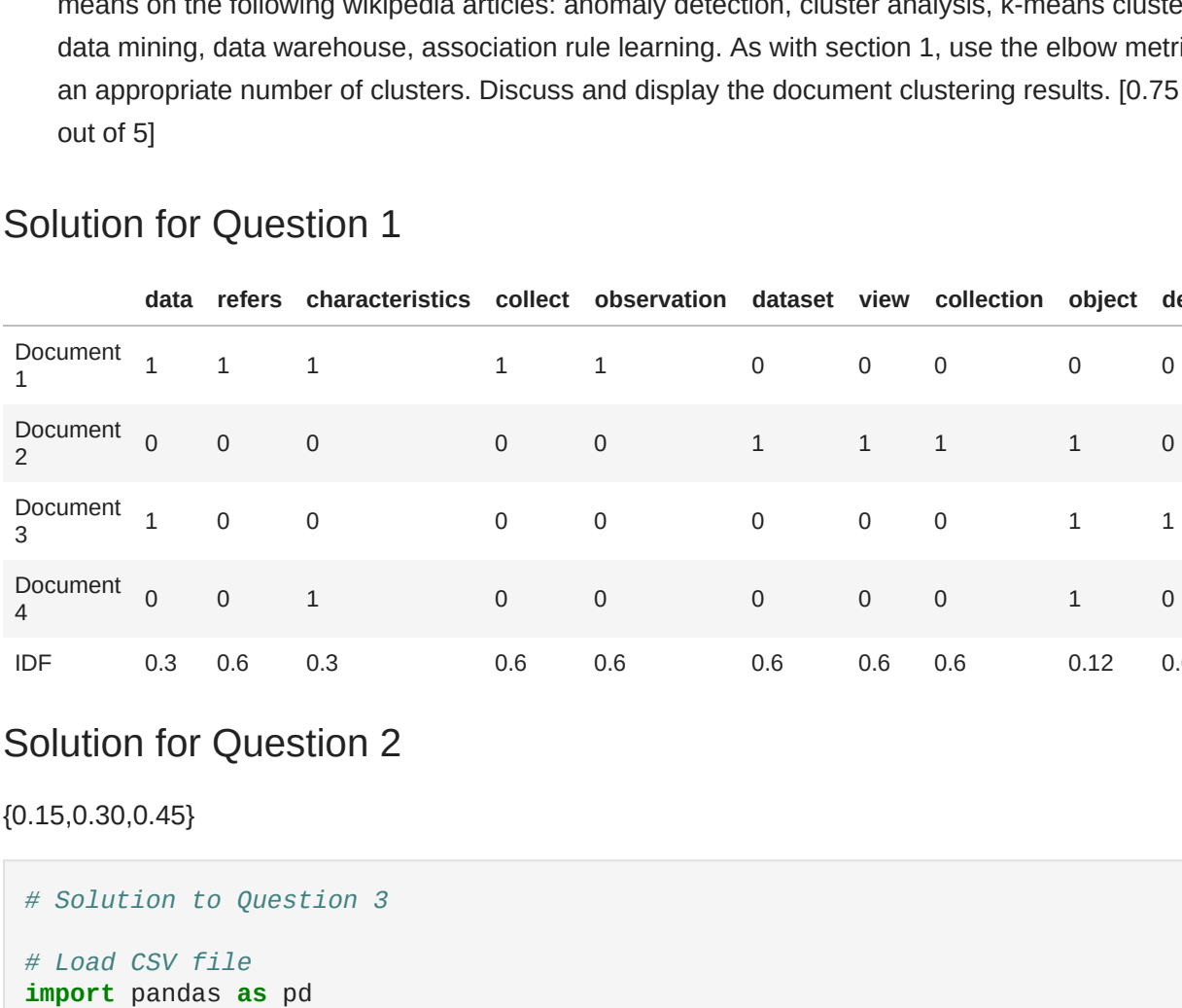
Solution for Question 2

{0.15,0.30,0.45}

In [6]:

```
# Solution to Question 3

# Load CSV file
import pandas as pd
df=pd.read_csv('timeseries.csv', sep=',',header=None)
timeseries = df.values
timeseries = timeseries.flatten() # Flatten to 1 dimension, otherwise DFT does not wo

# Plot timeseries [0.25 marks out of 1]
plt.figure(figsize=(10, 4))
plt.plot(timeseries)
plt.title('timeseries')

# Compute and plot the magnitude DFT of timeseries [0.25 marks out of 1]
timeseriesDFT = np.fft.fft(timeseries)
plt.figure(figsize=(10, 4))
plt.title('magnitude of DFT')
plt.plot(np.abs(timeseriesDFT))
plt.xlabel('Frequency (index $k$)')
```

Out[6]: `Text(0.5, 0, 'Frequency (index $k$)')`



[0.5 marks out of 1]: The DFT magnitude of the timeseries has two peaks, therefore has 2 predominant frequency components (remember that the DFT is symmetric, and we discard the upper half of the DFT)

In [9]:

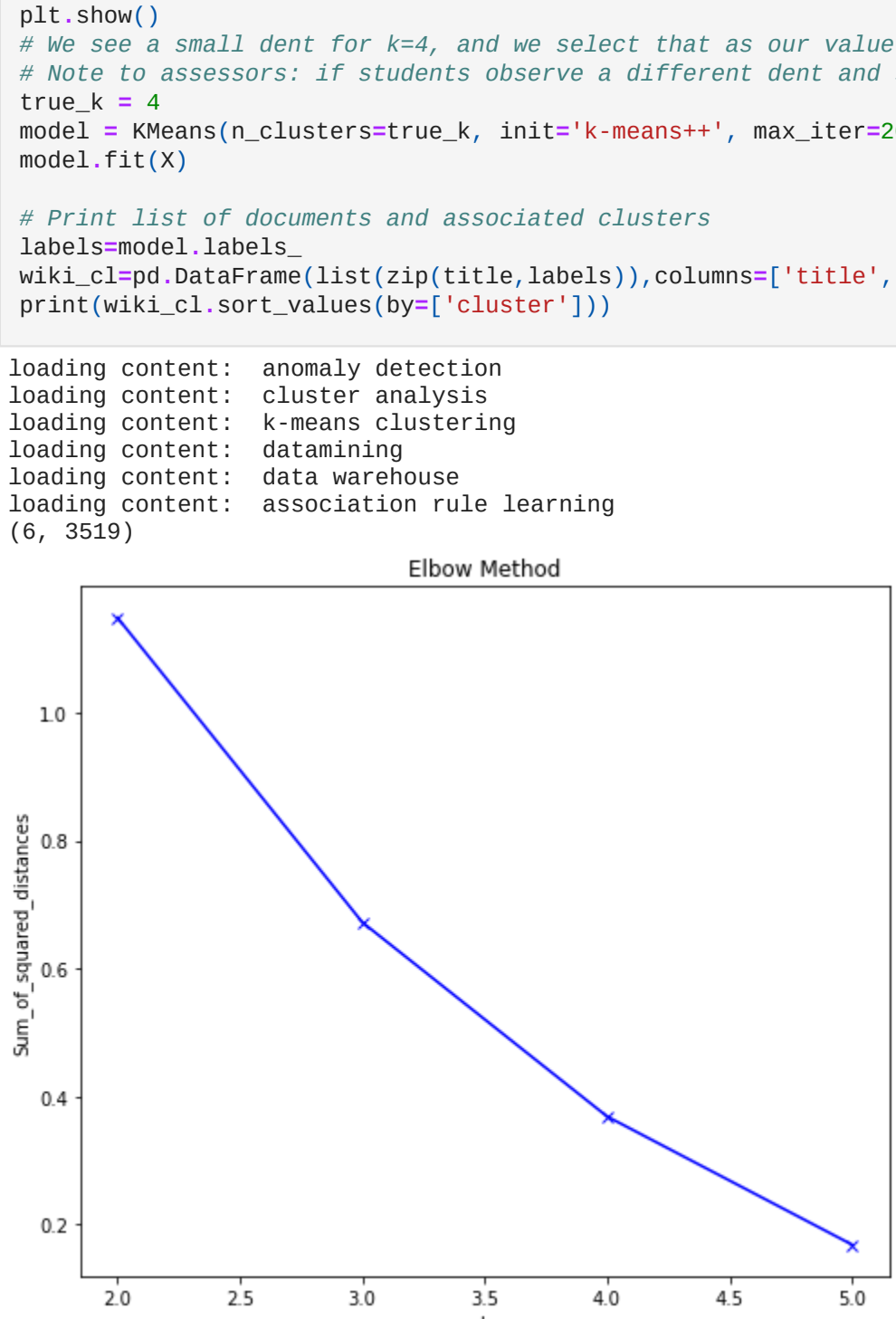
```
# Solution to question 4

from pandas import read_csv
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.arima.model import ARIMA


# Load CSV file
series = read_csv('births.csv', header=0, index_col=0)

# Perform trailing moving average smoothing with a 7-sample window [0.25 marks out of
rolling = series.rolling(window=7)
rolling_mean = rolling.mean()
rolling_mean = np.nan_to_num(rolling_mean, nan=0) # Replace NaN values with 0

# AR model forecasting [0.25 marks out of 0.75]
AR_model = AutoReg(rolling_mean, lags=2,old_names=False) # "lags" indicates the model
AR_model_fit = AR_model.fit()
AR_yhat = AR_model_fit.predict(len(rolling_mean), len(rolling_mean)+4) # Predict next
print("AR predictions: ",AR_yhat)

# ARMA model forecasting [0.25 marks out of 0.75]
ARMA_model = ARIMA(rolling_mean, order=(2, 0, 2)) # p=2, q=2
ARMA_model_fit = ARMA_model.fit()
ARMA_yhat = ARMA_model_fit.predict(len(rolling_mean), len(rolling_mean)+4) # Predict 
print("ARMA predictions: ",ARMA_yhat)
```

```
AR predictions:  [45.38017715 44.96085234 44.59067614 44.27169885 43.99739484]
ARMA predictions:  [45.81024785 45.81876676 45.72809324 45.56401772 45.34730671]
```

In [14]:

```
# Solution to question 5

import pandas as pd
import wikipedia
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load Wikipedia articles and create vector representation [0.25 marks out of 0.75]
articles=['anomaly detection', 'cluster analysis', 'k-means clustering', 'datamining',
wiki_lst=[]
title=[]

for article in articles:
    print("loading content: ",article)
    wiki_lst.append(wikipedia.page(article).content)
    title.append(article)

vectorizer = TfidfVectorizer(stop_words={'english'})
X = vectorizer.fit_transform(wiki_lst) # Create tf-idf feature of the wikipedia datase
print(X.shape) # Print dimensions of tf-idf feature

# Select best number of clusters using elbow metric and fit k-means model [0.25 marks
Sum_of_squared_distances = []
K = range(2,6)

for k in K:
    km = KMeans(n_clusters=k, max_iter=200, n_init=10)
    km = km.fit(X)
    Sum_of_squared_distances.append(km.inertia_)

plt.figure(figsize=(8, 7))
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method')
plt.show()
# We see a small dent for k=4, and we select that as our value
# Note to assessors: if students observe a different dent and select a different valu
true_k = 4
model = KMeans(n_clusters=true_k, init='k-means++', max_iter=200, n_init=10)
model.fit(X)

# Print list of documents and associated clusters
labels=model.labels_
wiki_cl=pd.DataFrame(list(zip(title,labels)),columns=['title','cluster'])
print(wiki_cl.sort_values(by=['cluster']))
```

```
loading content:  anomaly detection
loading content:  cluster analysis
loading content:  k-means clustering
loading content:  datamining
loading content:  data warehouse
loading content:  association rule learning
(6, 3519)
```




```
                       title  cluster
3                 datamining        0
4             data warehouse        0
1           cluster analysis        1
2         k-means clustering        1
0          anomaly detection        2
5  association rule learning        3
```

Discussion of clustering results: [0.25 marks out of 0.75]

- We see that k-means clustering and cluster analysis are grouped together, which makes sense since they belong to the same topic. There are separate clusters for anomaly detection and associatin rule learning, which is appropriate since these are separate topics from the rest.We also see that data mining and data warehouse form a cluster, which can make sense since data mining is often viewed from the perspective of infrastructure and storage.

In [ ]: