Getting Started

Resources

- Slides posted in google classroom.
- Python tutorial book
- Python for data analysis

Offline Compilers

- Python shell
- Anaconda Distribution (recommended)

Online Compilers

- Google Colaboratory (recommended)
- Programiz
- Tutorialspoint
- w3schools

TA Meetings

- TBD

Assignment 1 - (20 Marks, Due on 26/06/2022)

Instructions

- The assignment has to be done individually. It has two exercises.
- You can download the data files here.
- You need to submit two zip files, one for each of the exercises. They should be named as rollnumber_exercisenumber.zip. Create a folder rollnumber_exercisenumber containing one source code file and a report. The code file should be named as rollnumber_exercisenumber.py and should be commented/documented properly by mentioning the code for each sub-problems of the exercise. The report should have the results of the print statements mentioned in the exercises and should be named as rollnumber_exercisenumber_report.pdf. The report template can be found here.

  For example: Create a zip file named ai21btech11021_exercise1.zip. This file contains a folder named ai21btech11021_exercise1 that contains the two files ai21btech11021_exercise1.py and ai21btech11021_exercise1_report.pdf. In a similar manner create ai21btech11021_exercise2.zip.

  PS: You can also submit .ipynb files instead of .py files

Problems

[10 Points] Exercise 1: The shopping.csv file contains details of the items purchased in different transactions at a store. Note that each line in the CSV file corresponds to a different transaction. The first field in each of the transactions corresponds to the number of items purchased and the remaining fields correspond to the name of the respective items purchased. Write python code to perform the following on the shopping data:

1. Load the shopping.csv file to a dataframe.

```
import pandas as pd
df = pd.read_csv('C:/Users/abc/Desktop/file_name.csv')
print(df)
```

2. Create Dictionary: Create a python dictionary to store all the transaction details. Associate a transaction ID starting from T1, then T2, T3, and so on, to each of the transactions in the dataset. Note that, the key of the dictionary must be the transaction ID and the value should be the list of items purchased in that transaction. Print the first five key-value pairs.

3. Use the '**re'** module to convert the name of all the items purchased to a single format. The different words associated with an item should be connected by '_'. For example, items like 'juice', 'brown bread', 'milk/fruit juice', 'Samsung J 12-S' gets updated to 'juice' (no change here), 'brown_bread', 'milk/fruit_juice', 'Samsung_J_12-S', etc. Finally, arrange them in lexical order. Use this dataframe for the rest of the problems below.

4. Write a function that takes a transaction ID as an input argument and prints all the items purchased in that transaction. In the report, list the items for the transaction IDs: 32, 68, 78.

5. Generate a list of frequent one-itemsets purchased - one-itemset is the itemset that contains only 1 item. A one-itemset is frequent if its support is greater than the given support threshold. Support of one-itemset is defined as follows:
   $support = \frac{|T_{i_m}|}{N_{total}}$, where $|T_{i_m}|$ is the number of transactions in which the item i_m appears in the dataset and N_{total} is the total number of transactions.
   Generate the following table for different support thresholds. Note: support@10 means that all the items that appeared at least 10% of the time in the dataset.

Support	support@0.5	support@1	support@2	support@3	support@5	support@10
No. of items						

   Also, list all the frequent one-itemsets having support@3, as given in the below table.

Support@3	Items …..

6. Generate the most frequent two-itemsets from the dataset. A two-itemset is the itemset that contains 2 items. A two-itemset is frequent if its support is greater than the minimum support threshold. Support of two-itemset is defined as follows:
   , $support = \frac{|T_{i_m,i_n}|}{N_{total}}$ where $|T_{i_m,i_n}|$ is the number of transactions in which both the items i_m and i_n appear together in the dataset and N_{total} is the total number of transactions.

Generate the following table having varied support thresholds. Note: support@10 means that all the two-itemsets that appeared at least 10% of the time in the dataset.

Support	support@0.5	support@1	support@2	support@3	support@5	support@10
No. of items						

Also, list all the frequent one-itemsets having support@3, as given in the below table.

Support@3	Items …..

[10 Points] Exercise 2: For this exercise, consider the four text documents, namely Doc1, Doc2, Doc3, and Doc4, that contain film reviews. The objective of this exercise is to process text files and find the similarity between them using cosine similarity.

1. Read the reviews given in the four text files and store them in four different string variables. Perform the following operations on each of these strings.
    a. Tokenization: From the strings, remove all punctuation (i.e., only keep the alphabets, numbers, and whitespaces) and convert all alphabets to lowercase as part of text pre-processing. Transform the strings into lists composed of words (words are split from the string using blank space(s)).

    b. Vocabulary building: Combine all tokens (words) from all the four lists and store them into a single vocabulary set, where each token is stored only once. Print the total number of words in the vocabulary.

2. Create a frequency table
    a. Create a dictionary corresponding to each of the four lists each of which is associated with a particular review. In the dictionary, the keys are the tokens in the vocabulary and the value is the frequency of that token in the corresponding review.

    b. Combine all the dictionary information into a single dataframe of the size 4 X M, where M is the vocabulary size. The columns of the dataframe are the tokens in the vocabulary and a particular row corresponds to the number of times these tokens are present in a particular review.

3. Similarity Calculation: Consider each row of the above-mentioned dataframe as a vector. In order to find the similarity between any two vectors (i.e., reviews) r_i and r_j, you can compute the cosine similarity between them as $CosSim(r_i, r_j)$, which is defined as follows:

$$CosSim(r_i, r_j) = \frac{r_i.r_j}{\|r_i\| \times \|r_j\|} = \frac{\sum_{k=1}^{n} r_{ik} \times r_{jk}}{\sqrt{\sum_{k=1}^{n} {r_{ik}}^2}\sqrt{\sum_{k=1}^{n} {r_{jk}}^2}}$$

where,

$r_i.r_j$ = dot product of the 2 vectors

$||r_i||$ = L2 norm of the vector r_i

$\|r_i\| \times \|r_j\|$ is the product of the length of the 2 vectors.

NOTE: L2 norm of a vector r_i can be calculated using numpy.linalg.norm(r_i)

Using the above formula, calculate the similarity between all pairs of reviews (vectors) and store it in a square matrix. Print the top 3 review pairs with the highest similarity in decreasing order along with their cosine similarity score.

Assignment 2 - (30 Marks, Due on 08/07/2022)

Instructions

- The assignment has to be done individually. It has two exercises.
- You can download the data files here.
- You need to submit two zip files, one for each of the exercises. They should be named as rollnumber_exercisenumber.zip. Create a folder rollnumber_exercisenumber containing one source code file and a report. The code file should be named as rollnumber_exercisenumber.py and should be commented/documented properly by mentioning the code for each sub-problems of the exercise. The report should have the results of the print statements mentioned in the exercises and should be named as rollnumber_exercisenumber_report.pdf. The report template can be found here.

  For example: Create a zip file named ai21btech11021_exercise1.zip. This file contains a folder named ai21btech11021_exercise1 that contains the two files ai21btech11021_exercise1.py and ai21btech11021_exercise1_report.pdf. In a similar manner create ai21btech11021_exercise2.zip.
  PS: You can also submit .ipynb files instead of .py files

Problems

[20 Points] Exercise 1: The 'USUnemployment.csv' file has details of the unemployment rate recorded per month in the USA from 1948 to 2019. Use the dataset for the following:

1. Add a new column named "Avg_Unemployment_Rate" that contains the average unemployment for each year across all the months. Print in decreasing order the 5 rows of the dataframe with maximum unemployment rate.

2. Unemployment rate is an indicator of a country's unemployment status. Categorize each year into three different employment classes, namely High, Medium and Low. See Table 1 for this categorization. Write a python function that takes in the argument Average unemployment rate as a value from the Avg_Unemployment_Rate column and returns the unemployment category. Also, add a new column to the dataframe created in the first question and name it as "Unemployment_Status". Plot a pie-chart utilizing the created dataframe that displays the percentage of the three classes of Unemployment_Status.

Unemployment Status	Average Unemployment Rate
Low	<4
Medium	<6 and >=4
High	>=6

Table 1: Unemployment Status

3. Add an additional column to each row to specify the decade. For example, all rows belonging to the years 1950-1959 would have the “Decade” label as 1950s, all rows belonging to the years 1960-1969 would have the “Decade” label as 1960s and so on. Print the first row of each decade from the dataframe.

4. Perform season-wise average on the original dataframe such that the 12 months are grouped into 4 seasons, namely Winter, Spring, Summer and Autumn. The 4 seasons in terms of months are Winter: December, January, February; Spring: March, April, and May; Summer: June, July, August; Autumn: September, October, November. Hence, the new dataframe created would contain 4 columns (season names) for each row.

   Create 4 line plots, one for every season in the same graph. Add legend to your plot that mentions the season. In the x-axis plot the years between 1990 to 2005, and in the y-axis the unemployment rate for that season.

5. For this exercise, use the dataframe from the 4th question. Compute horizontal sliding window averages for each season based on the current and the previous 4 years. For example, if the row is for 2019, then take the average of 2015, 2016, 2017, 2018 and 2019. Create 4 new columns for storing these season-wise sliding window average unemployment rate values. Thus, your new dataframe should contain 8 columns (4 season-wise averages from question 4 and 4 sliding window season-wise averages for the last 5 years).

   For the years 1948, 1949, 1950, and 1951, the 4 new columns created should have the same values as that of season-wise averages. Print the row data for the years 1950, 1960, 1970, 1980, 1990, and 2000 in your report.

[10 Points] Exercise 2: Use the image.png file provided and write codes to perform the following operations on the image: You need to use the skimage package to perform the following image transformations in Python.

1. Load the image information into a numpy nd-array.
   Hint: From skimage.io import imread to read the image

2. Print the number of pixels in the given image.

3. The primary colors/channels of an image are red, blue and green. These colors are added together in various ways to reproduce a broad array of colors. Convert the images to their individual respective channels. That is, represent the given image as red, blue, and green separately. Your report should contain all the three coloured image plots. Also, print the index and value of the first occurrence of maximum intensity value in all the three cases in your report. Thus, you should print the following:

   Red: Maximum intensity value and its index
   Green: Maximum intensity value and its index
   Blue: Maximum intensity value and its index

   Hint: The last dimension of the numpy array represents the colour channels. Channel 0 is for red, channel 1 is for green and channel 2 is for blue.

4. Apply the following transformations and plot the original image and the transformed image as subplots side by side (Please use the subplot function from matplotlib package). You should use the modules and functions from skimage package to perform the following image transformations.
   a. Convert the image into grayscale. Print the means of the original image array as well as the transformed image.
   b. Flip the image horizontally
   c. Flip the image vertically