תרגיל בית 2

גם בתרגיל זה נעבוד עם סט הנתונים "london.csv". הנתונים מכילים כ17,400 רשומות, כאשר כל רשומה מתעדת את מספר האופניים החדשים שנשכרו בפרויקט בלונדון הדומה לתל אופן בתל אביב, פרמטרי מזג אוויר ותקופה בשבוע ובשנה.
כל רשומה חדשה מייצגת שעה עגולה, החל מה04/01/2015 בשעה 00:00 עד ל03/01/2017 בשעה 23:00.

הפעם נעבוד עם סט הנתונים המלא. לתיאור מלא של הנתונים אנא בקרו ב:

https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset

לתרגיל זה נגדיר סביבה חדשה במכונה, ע"י שימוש בקובץ env.yml המצורף לתרגיל.

כדי לייצר את הסביבה מתוך הקובץ יש להעבירו למכונה, ולאחר מכן (מאותה תיקייה) להריץ את הפקודה הבאה:

```
conda env create -f env.yml
```

לאחר מכן תיווצר לכם סביבה, ותוכלו להפעיל אותה באופן הבא:

```
conda activate hw2_ds_env
```

חלק א – **EDA and Pandas**

עליכם לממש את הפונקציות המתוארות לפי בmodule בשם data.py כאשר הmodule מכיל את הImports הבאים בלבד:

```
import pandas as pd
from datetime import datetime
```

```
def load_data(path):
"""" reads and returns the pandas DataFrame """"
```

1. קראו את קובץ הנתונים לתוך DataFrame ע"י הפקודה pd.read_csv(path) (הניחו שהקובץ נמצא באותה תיקייה עם הסקריפט). החזירו את הDataFrame הנוצר.

שימו לב: על 1 להיות ממומש בפונקציה load_data.

```
def add_new_columns(df):
"""adds columns to df and returns the new df"""
```

2. הוסיפו עמודה בשם season_name באמצעות פו' apply של pandas על העמודה season. ערכי העמודה הם שמות העונות לפי מספרי העונות הנתונים בעמודה season:

"season" - category field meteorological seasons: 0-spring ; 1-summer; 2-fall; 3-winter.

3. הוסיפו 4 עמודות בשם:

- Hour (an integer, 0-23)
- Day (an integer, 1-31)
- Month (an integer, 1-12)
- Year (an integer, 2015-2017)

באמצעות פו' apply של pandas על עמודת timestamp. ערכי העמודה הם השעה/יום/חודש/שנה שתועדה עבור כל רשומה. יש להיעזר בספריה datetime.

4. הוסיפו עמודה בשם is_weekend_holiday באמצעות פו' apply של pandas על העמודות is_holiday ו-is_weekend. ערכי העמודה יהיו:

is_holiday	is_weekend	is_weekend_holiday
0	0	0
0	1	1
1	0	2
1	1	3

5. הוסיפו עמודה בשם t_diff באמצעות פו' apply של pandas על העמודות t1 וt2. ערך העמודה יהיה ההפרש בין t2 לt1.

החזירו את הDataFrame הנוצר.

שימו לב: על 2-5 להיות ממומשים בפונקציה add_new_columns. (5 נק' לכל סעיף)

```
def data_analysis(df):
"""prints statistics on the transformed df"""
```

6. הוסיפו את הפקודות הבאות בתחילת הפונקציה data_analysis:

```
print("describe output:")
print(df.describe().to_string())
print()
print("corr output:")
corr = df.corr()
print(corr.to_string())
print()
```

7. בסעיף זה עליכם להציג פלט שעונה על השאלות:
   1. מהם חמשת זוגות הפיצ'רים (שונים זה מזה) בעלי הקורלציה הגבוהה ביותר בערך מוחלט?
   2. מהן חמשת זוגות הפיצ'רים בעלי הקורלציה הנמוכה ביותר בערך מוחלט?

המלצה: תחילה צרו מילון (dictionary) בו המפתחות הם tuples של שמות זוג פיצ'רים והערכים הם הקורלציה (בערך מוחלט) בין זוג הפיצ'רים (השתמשו בcorr לשם כך). לדוגמה, אחד המפתחות במילון יהיה (t1,t2) וערכו יהיה מספר בין 0 ל1.

הפלט צריך להיות מהצורה:

```
Highest correlated are:
1. ('x', 'y') with <val0>
2. ('x', 'y') with <val1>
3. ('x', 'y') with <val2>
4. ('x', 'y') with <val3>
5. ('x', 'y') with <val4>

Lowest correlated are:
1. ('x', 'y') with <val0>
2. ('x', 'y') with <val1>
3. ('x', 'y') with <val2>
4. ('x', 'y') with <val3>
5. ('x', 'y') with <val4>
```

שימו לב: החמישיה בHighest מסודרת מהגדול לקטן, החמישיה בLowest מסודרת מהקטן לגדול. יש לעגל ל6 ספרות אחרי הנקודה בסעיף זה.

8. הציגו את ערך העמודה t_diff הממוצע לכל season_name באמצעות פקודת groupby, ובנוסף את הממוצע לכל הרשומות.
הפלט צריך להיות מהצורה:

```
fall average t_diff is <fall_avg_t_diff>
spring average t_diff is <spring_avg_t_diff>
summer average t_diff is <summer_avg_t_diff>
winter average t_diff is <winter_avg_t_diff>
All average t_diff is <all_avg_t_diff>
```

ניתן להיעזר בפקודה הזאת. יש לעגל ל2 ספרות אחרי הנקודה. תיעוד על groupby מופיע כאן. לשימושכם בנוסף tutor.

שימו לב: על 8-7 להיות ממומשים בפונקציה data_analysis. (10 נק' לכל סעיף)

חלק ב – **Clustering**

חלק זה צריך להיות ממומש תחת הקובץ clustering.py. שימו לב, הקובץ clustering.py מצורף לקבצי התרגיל, וממומשות בו חלק מהפונקציות. יש להשתמש בקובץ זה. הmodule מכיל את הImports הבאים בלבד:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
np.random.seed(2)
```

בחלק זה נתעניין בתכונות cnt, hum בלבד, על-פיהן נרצה לבצע clustering על התצפיות.

לצורך המשימה נגדיר:

- מרחק אוקלידי:

$$dist(\vec{x},\vec{y}) = \left||\vec{x}-\vec{y}|\right| = \sqrt{\sum_{i=k}^{d}(\overrightarrow{x_k}-\overrightarrow{y_k})^2}$$

כאשר $\vec{x},\vec{y} \in R^d$ הם שני וקטורים.

1. עליכם לממש את הפונקציה

```
transform_data(df, features)
```

פרמטרים:

- df הינו pandas dataframe, המכיל את הדאטהסט המלא שנקרא מתוך ה-csv, (ע"י הפקודה pd.read_csv(path)). שימו לב שיש לקרוא את הcsv בשנית מהזיכרון, אין קשר בין שני החלקים בתרגיל. ניתן לצורך כך להפעיל שוב את הפונקציה load_data מחלק 1.
- features הוא רשימה של בדיוק 2 שמות מהעמודות מה-dataframe (תכונות).

פלט המתודה יהיה:

return transformed_data

transformed_data הינו numpy array בעל shape של (n, 2), כאשר n הוא כמות התצפיות בדאטה, ו-2 כמספר התכונות ברשימה features.

בפונקציה זו, על מנת להגיע מ-df ל-transformed_data, יש לבצע את הפעולות הבאות:

- שליפת העמודות הרלוונטיות מתוך הנתונים (על פי הסדר שהן מופיעות ברשימה features).
- Sum scaling על העמודות הללו (עבור כל וקטור עמודה x, נבצע $\frac{x-min(x)}{sum(x)}$).

- לבסוף, יש להשתמש בפונקציה

```
add_noise(data)
```

אשר ממומשת בקובץ clustering.py, על מנת להוסיף רעש קטן לנתונים (יש להפעיל את הפונקציה על הדאטא לאחר בחירת העמודות וsum normalization).

(10 נק')

2. עליכם לממש את הפונקציה:

```
kmeans(data, k)
```

פרמטרים:

- data הינו numpy array, בעל shape של (n, 2), כאשר n הוא כמות התצפיות בדאטה, וכל שורה במערך מייצגת תצפית של 2 התכונות cnt, hum (בסדר הזה).
- k הינו מספר שלם המייצג את מספר ה-clusters באלגוריתם kmeans.

פלט המתודה יהיה:

return labels, centroids

כאשר:

- labels הינו numpy array בגודל n, כאשר n הוא כמות התצפיות בדאטה. כל כניסה ב-labels היא ה-cluster שאליו שייכת התצפית ה-i בסוף ריצת האלגוריתם.
- centroids הינו numpy array בעל shape של (2,k), כאשר כל שורה i בו מייצגת את ה centroidשל ה-cluster ה-i.

מספר דגשים חשובים:

- ניתן לממש פונקציות עזר כרצונכם. יש לתעד כל פונקציה שאתם כותבים.
- בתחילת הקובץ clustering.py מוגדר np.random.seed(2) – אין לשנות או למחוק שורה זו.
- לצורך בחירת ה-centroids ההתחלתיים יש להשתמש בפונקציה:

```
choose_initial_centroids(data, k)
```

אשר ממומשת בקובץ clustering.py.

- תנאי העצירה של האלגוריתם הינו שה-centroids לא השתנו בין 2 איטרציות עוקבות.
לשם בדיקה זו, יש להשתמש בשורת הקוד:
np.array_equal(prev_centorids, current_centroids)
- יש להשתמש במרחק אוקלידי כפי שהוגדר בתחילת חלק זה.

(35 נק')

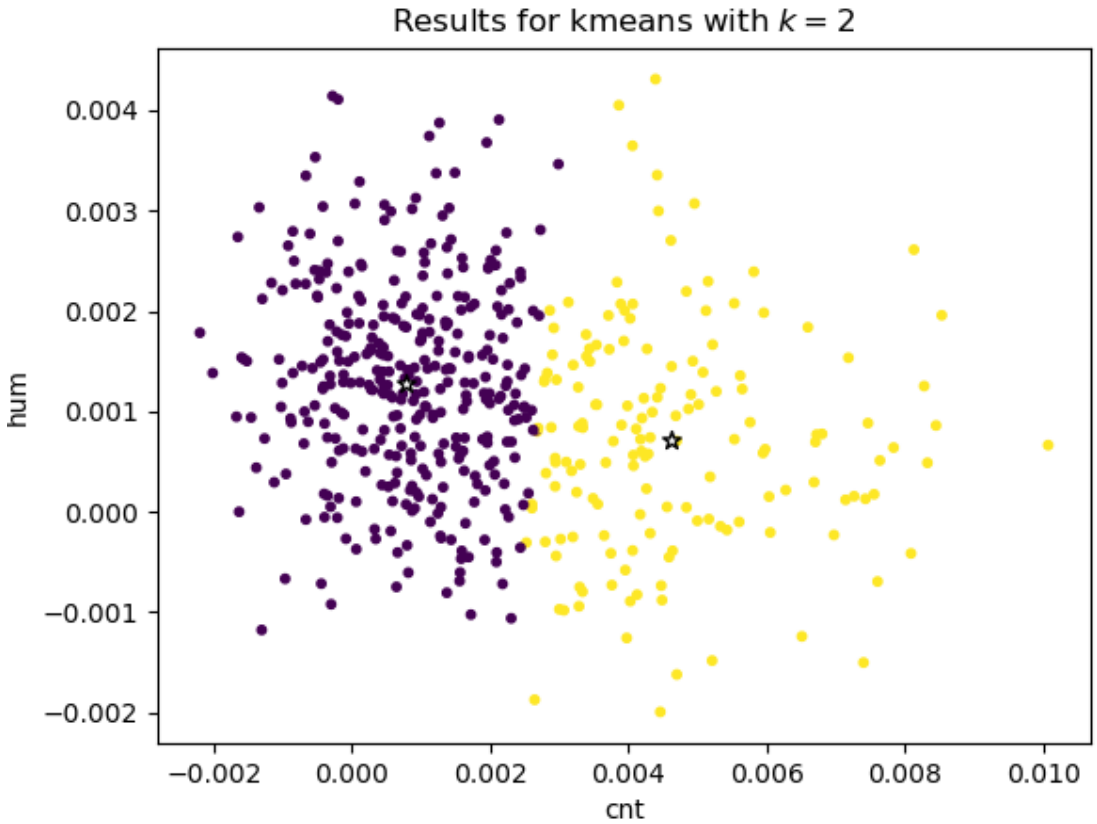
3. עליכם לממש את הפונקציה

```
visualize_results(data, labels, centroids, path)
```

פרמטרים:

- data הינו numpy array, בעל shape של (n, 2), כאשר n הוא כמות התצפיות בדאטה, וכל שורה במערך מייצגת תצפית של 2 התכונות cnt, hum (בסדר הזה) – פלט המתודה transform_data.
- labels ו-centroids הם פלט הפונקציה kmeans, (תיאור מלא מופיע למעלה).
- path - ניתוב שאליו ישמר הפלט של הפונקציה.

הפונקציה היא ללא ערך החזרה, אך במהלכה שומרת פלט של תרשים לקובץ (לפי הניתוב path). דוגמה לתרשים נראית כך:



על מנת לשמור את התרשים יש להשתמש בפקודה הבאה:

```
plt.savefig(path)
```

(15 נק')

פלט חלק זה מורכב כך:

- יש להריץ את kmeans עבור 3 ערכי k שונים ([2,3,5]). עבור כל הרצה כזו, יש:
  - להדפיס את ה-centroids ע"י הפקודה:

```
print(np.array_str(centroids, precision=3, suppress_small=True))
```

  - לייצר תרשים עם הפונקציה visualize_results .

דוגמה לפלט:

k = 2

centroids for k=2

k = 3

centroids for k=3

k = 5

centroids for k=5

את התרשימים ששמרתם יש לצרף בקובץ PDF אחד ששמו plots.pdf.

מבנה הפלט צריך להראות בדיוק על-פי הדוגמא המצורפת בקובץ הפלט לדוגמא output.txt, מאחר ומתבצעת השוואת קבצים אוטומטית. הקובץ הנ"ל מחזיק את הפלט עבור הקובץ london_sample_500.csv, שהינו דגימה קטנה מתוך כל הנתונים בקובץ london.csv.

הדגשה: בקובץ output.txt יש את הפלט של התוכנית כאשר מריצים את הקוד עם **london_sample_500.csv** ולא עם london.csv.

ניתן להניח שקבצי הcsv. נמצאים באותה תיקייה כמו main.py (אין חובה להשתמש בargv כמו בתרגיל 1).

דגשים נוספים:

1. עליכם לכתוב את הקוד בהתאם לדגשים והסטנדרטים לפי 8pep. לשימושכם המסמך " Code Quality Requirements " באתר ה-moodle של הקורס. קוד אשר לא יעמוד בסטנדרטים הנדרשים, יקבל ניקוד מופחת.

2. ניתן להוסיף מתודות/פונקציות נוספות, במידה ותמצאו לנכון. יש להימנע מכפילויות קוד.

3. ניתן להשתמש במתודות/פונקציות שהן in-built בשפה. קרי, מתודות אשר לא דורשות ייבוא של ספריות.

4. יש לתת שמות בעלי משמעות לכל משתנה.

5. חובה לתעד את הקוד באנגלית. בפרט עליכם לכתוב עבור כל מתודה docstring

6. יש להציג את כל הערכים המתקבלים עם עיגול של 2 ספרות לאחר הנקודה (לדוגמה: ⅓ יוצג כ-0.33, ו- 1 יוצג כ- 1.00), למעט במקומות בהם נאמר אחרת.

הוראות הגשה:

- התרגיל להגשה בזוגות בלבד.
- לפני ההגשה, חובה לוודא שהתוכנית עובדת במכונות הוירטואליות. בנוסף, על התוכנית לרוץ בפחות מ300 שניות **על המכונה ועם הסביבה שסופקה בenv.yml**.

- ההגשה חייבת להכיל קובץ אחד (קובץ zip) :
  - שם הקובץ חייב להיות hw2 _xxxxxxxx_yyyyyyyy.zip כאשר xxxxxxxx ו- yyyyyyyy הם מספרי תעודות הזהות של המגישים, כולל ספרת ביקורת.
  - הקובץ מכיל את כל קבצי הקוד וקובץ דו"ח שלכם עם תשובות לשאלות (בתרגיל זה – plots.pdf). אין להכיל תיקייה ובתוכה קבצי הקוד, אלא את קבצי הקוד עצמם.
  - הערה: עליכם לוודא שהתוכנית מתחילה לפעול מקובץ "main.py" בלבד.
  - תשובות לחלקים יבשים יש להקליד במעבד תמלילים. אין להגיש תשובות בכתב יד.
- ההגשה היא אלקטרונית בלבד, דרך אתר ה-moodle של הקורס. תרגילים שיוגשו בכל דרך אחרת לא ייבדקו.
- אין להגיש את אותו הקובץ פעמיים. התרגיל יוגש ע"י אחד מבני הזוג.
- שימו לב שההגשה תיחסם בדיוק בשעה 23:55 ביום ההגשה. מומלץ להגיש לפחות שעה לפני המועד האחרון.
- ניתן להגיש כמה פעמים. רק ההגשה האחרונה תישמר.
- תרגיל בית שלא יוגש לפי הוראות ההגשה – לא ייבדק (כלומר יקבל ציון 0).
- לצורך תרגיל הבית ייפתח פורום. ניהול שאלות ומתן תשובות בנושא התרגיל יתבצע דרך הפורום בלבד.

בהצלחה!