תרגיל בית **3**

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 מדגם בגודל 2500 מתוך סט הנתונים המלא (מאילוצי זמן ריצה של אלגוריתם KNN).

לתיאור מלא של הנתונים אנא בקרו ב:

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

בתרגיל זה נשתמש באותה הסביבה בה השתמשנו בתרגיל בית 2 (hw2_ds_env). לקובץ ה-yml. של הסביבה ולהוראות על התקנתה (למי שלא התקין בתרגיל הקודם), ראו תרגיל בית 2.

בתרגיל זה נבנה מספר מודולים ומחלקות, שישמשו אותנו על מנת לייצר מודלי supervised, ולהעריך את ביצועיהם בצורה איכותית.

חלק א' - Classification:

המשתנים המסבירים (הפיצ'רים) יהיו t1, t2, wind_speed, hum

והמשתנה המוסבר (התיוג) יהיה בינארי: 0 אם העונה היא אביב או קיץ, ו-1 אם העונה היא סתיו או חורף.

כלומר, מטרתנו תהיה לחזות איזו תקופה בשנה מתוארת ע"י המשתנים המסבירים.

את משימה זו נבצע ע"י מודל KNN ל-classification, ונעריך ע"י f1_score.

חלק ב' – Regression:

המשתנים המסבירים (הפיצ'רים) יהיו t1, t2, wind_speed

והמשתנה המוסבר (התיוג) יהיה רציף: ערך המשתנה hum.

כלומר, מטרתנו תהיה לחזות מה יהיו אחוזי הלחות ע"י המשתנים המסבירים.

את משימה זו נבצע ע"י מודל KNN ל-regression, ונעריך ע"י RMSE.

את שתי המשימות נבצע בשיטת cross validation.

רוב חלקי העבודה משותפים לשני החלקים, ויתוארו בעמודים הבאים, בחלוקה לפי מודולים.

מכיוון שישנם קשרים בין החלקים השונים, מומלץ לקרוא את התרגיל כולו ולהבין כיצד החלקים מתחברים, ורק לאחר מכן להתחיל לממש.

data

עליכם לממש את הפונקציות והמחלקות המתוארות בmodule בשם **data.py**. שימו לב, הקובץ data.py מצורף לקבצי התרגיל, וממומשות בו חלק מהפונקציות. יש להשתמש בקובץ זה. הmodule מכיל את הImports הבאים בלבד:

```
import numpy as np
from sklearn.model_selection import KFold
import pandas as pd
np.random.seed(2)
```

```
def load_data(path):

""" reads and returns the pandas DataFrame """
```

1. קראו את קובץ הנתונים לתוך DataFrame ע"י הפקודה pd.read_csv(path) (נתיב הקובץ ניתן ע"י argv כמפורט בהמשך). החזירו את הDataFrame הנוצר.

```
def adjust_labels(y):

"""adjust labels of season from {0,1,2,3} to {0,1}"""
```

2. הפונקציה מקבלת מערך של ערכים מהעמודה season (ערכים מתוך {0,1,2,3}), כאשר:

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

ומחזירה מערך באותו גודל, כאשר כל הערכים אשר היו spring או summer יהיו כעת 0, וכל הערכים אשר היו fall או winter יהיו כעת 1.

3. עליכם לממש מחלקה בשם StandardScaler. על המחלקה לכלול את הפונקציות הבאות:

```
def __init__(self):
""" object instantiation """

def fit(self, X):
""" fit scaler by learning mean and standard deviation per feature """

def transform(self, X):
""" transform X by learned mean and standard deviation, and return it """

def fit_transform(self, X):
""" fit scaler by learning mean and standard deviation per feature, and then transform X """
```

כאשר X הוא מערך מסוג np.array, שהשורות שלו הן תצפיות, והעמודות שלו הן פיצ'רים. להזכירכם, הטרנספורמציה שמבצע standard scaler לכל פיצ'ר היא: $\frac{x_i-\bar{x}}{s(x)}$, כאשר $\bar{x}$ ו-$s(x)$ הם הממוצע וסטיית התקן המדגמית בהתאמה.

4. הפונקציות get_folds ו-add_noise ממומשות כבר בקובץ, ואין לגעת בהן או לשנות אותן.

evaluation

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

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f1_score(y_true, y_pred):

""" returns f1_score of binary classification task with true labels y_true and predicted labels y_pred
"""
```

1. הפונקציה מקבלת שני מערכים בגודל מספר התצפיות, כאשר y_true הם התיוגים האמיתיים, ו-y_pred הם התיוגים החזויים ע"י מודל ה-classification, ומחזירה את ה-f1_score שלהם. להזכירכם:
$f1-score = \frac{2*recall*precision}{recall+precision}$.

```
def rmse (y_true, y_pred):

"""returns RMSE of regression task with true labels y_true and predicted labels y_pred"""
```

2. הפונקציה מקבלת שני מערכים בגודל מספר התצפיות, כאשר y_true הם התיוגים האמיתיים (מסומנים בנוסחה ע"י y), ו-y_pred הם התיוגים החזויים ע"י מודל ה-regression (מסומנים בנוסחה ע"י $\hat{y}$), ומחזירה את ה-RMSE שלהם.
נוסחת הRMSE: $RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}$.

```
def visualize_results(k_list, scores, metric_name, title, path):
""" plot a results graph of cross validation scores """
```

3. הפונקציה מייצרת גרף לתוצאות של הרצת המודלים השונים עם cross validation. הפונקציה אינה מחזירה פלט, אלא שומרת את ה-figure לקובץ, באמצעות הפקודה plt.savefig(path)
פרמטרים:
- k_list – רשימת הערכים עבור k
- scores – רשימה באורך זהה ל-k_list, כאשר כל כניסה היא ממוצע התוצאות של תהליך ה-cross validation עבור המודל עם ה-k המתאים.
- metric_name – מחרוזת שמקבלת את אחד משני הערכים "f1_score" או "RMSE".
- title – מחרוזת שמקבלת את אחד משני הערכים "Classification" או "Regression".
- path – נתיב אליו ישמר הקובץ.

cross_validation

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

```
import numpy as np
```

```
def cross_validation_score(model, X, y, folds, metric):
"""run cross validation on X and y with specific model by given folds. Evaluate by given metric."""
```

1. הפונקציה מבצעת תהליך cross validation לפי folds נתונים. ערך ההחזרה של הפונקציה הוא רשימה בגודל מספר ה-folds (5), כאשר כל כניסה היא ערך המטריקה שהתקבל על סט ה-validation לכל fold.

פרמטרים:

- model –אובייקט של מודל (מסוג KNN שתגדירו) לאחר שלב ה-instantiation.
- X – מטריצה שהשורות שלה הן התצפיות והעמודות שלה הן הפיצ'רים.
- y – מערך בגודל מספר התצפיות, המכיל תיוג לכל תצפית.
- folds – אובייקט מסוג KFold של sklearn, פלט הפונקציה data.get_folds . אובייקט זה מכיל את החלוקה של הנתונים ל-folds שונים. על מנת להשתמש באובייקט זה כדי להשיג את החלוקה, יש להשתמש בפקודה:
  for train_indices, validation_indices in folds.split(X):
  שימוש בפקודה זו מחזיר רשימה של אינדקסים לחלוקה בהתאם (עבור ה-train ועבור ה-validation). לדוגמה, בהינתן אינדקסים אלו ניתן לגשת לסט ה-validation עבור חלוקה זו ע"י X[validation_indices] ו- y[validation_indices].
- metric – פונקציה שמקבלת 2 ארגומנטים y_true, y_pred, ומחזירה סקלר. (לדוגמה: f1_score).

```
def model_selection_cross_validation(model, k_list, X, y, folds, metric):
"""run cross validation on X and y for every model induced by values from k_list by given folds. Evaluate each model by given metric."""
```

2. הפונקציה מבצעת תהליך cross validation לפי folds נתונים עבור מספר מודלים. כל מודל מוגדר ע"י האובייקט model ועל ידי ערך הפרמטר k מתוך הרשימה k_list.

הפונקציה מחזירה 2 ערכים:

- רשימה בגודל מספר ה-folds (5), כאשר כל כניסה היא *ממוצע* ערכי המטריקה שהתקבלו על סט ה-validation לכל fold עבור אותו מודל.
- רשימה בגודל מספר ה-folds (5), כאשר כל כניסה היא *סטיית התקן המדגמית* ערכי המטריקה שהתקבלו על סט ה-validation לכל fold עבור אותו מודל.

פרמטרים:

- model - מחלקה של מודל (מסוג KNN שתגדירו).
- k_list – רשימה של ערכים אפשריים ל-k (מספר השכנים ב-KNN).
- X,y,folds,metric – זהים לפונקציה 1 ראו הסבר למעלה.

שימו לב להבדל בין הפונקציות בארגומנט model: בפונקציה הראשונה מדובר ב-instance של class מסוג KNN (של אחד הבנים שלו RegressionKNN או ClassificationKNN), בעוד שבפונקציה השנייה model הוא ה-class עצמו, ולא instance שלו.

knn

עליכם לממש את הפונקציות והמחלקות המתוארות בmodule בשם **knn.py** כאשר הmodule מכיל את הImports הבאים בלבד:

```
import numpy as np
from scipy import stats
from abc import abstractmethod
from data import StandardScaler
```

1. עליכם לממש מחלקה בשם KNN. על המחלקה לכלול את הפונקציות הבאות:

```
def __init__(self, k):
""" object instantiation, save k and define a scaler object """

def fit(self, X_train, y_train):
""" fit scaler and save X_train and y_train """

@abstractmethod
def predict(self, X_test):
""" predict labels for X_test and return predicted labels """

def neighbours_indices(self, x):
""" for a given point x, find indices of k closest points in the training set """

@staticmethod
def dist(x1, x2):
"""returns Euclidean distance between x1 and x2"""
```

הפונקציה predict אינה ממומשת בKNN, אלא זוהי רק הצהרה, שכן מחלקות הבן (בהמשך) יממשו אותה, כל אחת לפי הלוגיקה המתאימה לה.

2. עליכם לממש 2 מחלקות בשמות ClassificationKNN, RegressionKNN, כאשר שתיהן יורשות מKNN שהוגדר למעלה. על כל אחת מהמחלקות לכלול את הפונקציות הבאות:

```
def __init__(self, k):
""" object instantiation, parent class instantiation """

def predict(self, X_test):
""" predict labels for X_test and return predicted labels """
```

כאשר את הפונקציה predict מממשת כל מחלקת בן ע"י הכלל המתאים לה:

- עבור classification עליכם למצוא את התיוג השכיח מבין k השכנים הקרובים על מנת לחשב שכיח של מערך, יש להשתמש בפונקציה scipy.stats.mode .
- עבור regression עליכם למצוא את התיוג הממוצע מבין k השכנים הקרובים

main

חלק זה צריך להיות ממומש תחת הקובץ **main.py**. במודול זה נשתמש בפונקציות ובמחלקות שהגדרנו למעלה באופן הבא:

1. נקרא את קובץ ה-csv
2. נגדיר את ה-folds ע"י קריאה לפונקציה data.get_folds

פעולות אלו יקרו פעם אחת בלבד. לאחר מכן, עבור כל משימה:

1. נגדיר את X,y בהתאם למשימה (משתנים מסבירים ומשתנה מוסבר)
2. נוסיף רעש ע"י הפונקציה data.add_noise (ל-X בלבד)
3. עבור רשימת הערכים [3,5,11,25,51,75,101] עבור k, בצע cross validation, והדפס את *הממוצע* ואת *סטיית התקן המדגמית* של המטריקה המתאימה, שמתקבלים עבור סט ה-validation לכל מודל KNN עם k מהרשימה.
4. ייצר גרף שציר ה-x שלה הוא ערכי ה-k, וציר ה-y שלה הוא ממוצע המטריקה המתאימה עבור המודל המתאים עם אותו k.

ה-output צריך להיות מהפורמט הבא:

Part 1 – Classification

k=3, mean score: val, std of scores: val

...

k=101, mean score: val, std of scores: val

Part 2 – Regression

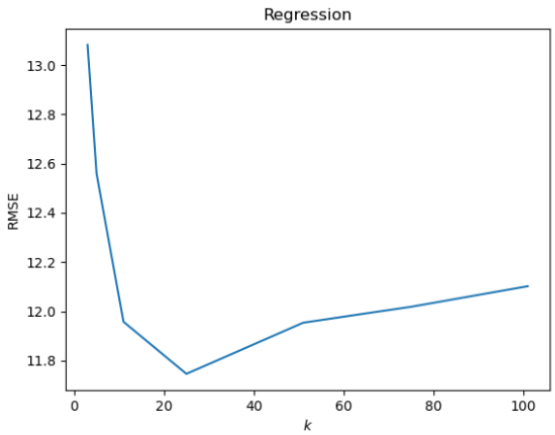
k=3, mean score: val, std of scores: val

...

k=101, mean score: val, std of scores: val

כאשר הערכים val צריכים להיות מעוגלים ל-4 ספרות אחרי הנקודה העשרונית.

דוגמה לגרף:



את 2 התרשימים ששמרתם יש לצרף בקובץ PDF אחד ששמו plots.pdf.

מספר דגשים חשובים:

- ניתן לממש פונקציות עזר כרצונכם. יש לתעד כל פונקציה שאתם כותבים.
- בתחילת הקובץ data.py מוגדר np.random.seed(2) – אין לשנות או למחוק שורה זו.

מבנה הפלט צריך להראות בדיוק על-פי הדוגמה המצורפת בקובץ הפלט לדוגמה 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.

בין היתר, בקובץ **main.py** יופיעו השורות:

```
def main(argv):
    pass


if __name__ == '__main__':
    main(sys.argv)
```

כאשר במקום "pass" יופיע קטע הקוד. עליכם לייבא את הספרייה sys בראש הקובץ, ע"י import sys. ע"י המבנה הנ"ל תוכלו להריץ את התוכנית שלכם באמצעות:

python **your_path**/main.py arguments

בתרגיל בית זה הארגומנט היחיד הוא הpath לקובץ ה-csv:

python /home/student/**your_path**/main.py /home/student/**your_path**/london.csv

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 4 ספרות לאחר הנקודה (לדוגמה: ⅓ יוצג כ- 0.3333, ו- 1 יוצג כ- 1.0000), למעט במקומות בהם נאמר אחרת.

הוראות הגשה:

- התרגיל להגשה בזוגות בלבד.
- לפני ההגשה, חובה לוודא שהתוכנית עובדת במכונות הוירטואליות.

- ההגשה חייבת להכיל קובץ אחד (קובץ zip) :
  - שם הקובץ חייב להיות hw3_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!