מעבדה 5 – למידה לא מונחית:

מגיש: איתי גיא, ת.ז. - 305104184

1. קריאת קובץ הנתונים $iris.data$:

```
    """
    path - iris data file path
    """
    def load_data(self, path):
        dataset = pd.DataFrame(columns=Kmeans.XHeaders)
        y = pd.DataFrame(columns=Kmeans.YHeader)
        with open(path.strip(), "r") as d:
            lines = d.readlines()
            for i, line in enumerate(lines):
                line = line.split(",")
                dataset.loc[i] = line[0:len(line) - 1]
                y.loc[i] = line[len(line) - 1]
        self.__trainset = dataset
        self.__ytrain = y
```

2. מימוש 3 אלגוריתמים אופציונאליים לאיתחול $KMEANS$:

א. רנדומלי לחלוטין – אנו מאתחלים את ה-$seeds$ להיות וקטורים 4 מימדיים רנדומליים

```
    """
    k - number of vector seeds we should create and fill-in with random numbers
    """
    def __random(self,k):
        kmeans = pd.DataFrame(columns=Kmeans.XHeaders)
        for i in range(0, k):
            kmeans.loc[i] = np.random.rand(len(Kmeans.XHeaders))
            for idx, item in enumerate(kmeans.loc[i]):
                kmeans.loc[i][idx] = random.uniform(1, Kmeans.RAND_INT)
        return kmeans
```

ב. אלגוריתם אתחול פורגי – נבחר k רשומות רנדומלית מתוך הקלט שיהווה $seeds$ התחלתיים

```python
    """
    n - number of seeds we have to sample randomly from the concrete data
    """
    def __forgy(self,k):
        kmeans = pd.DataFrame(columns=Kmeans.XHeaders)
        data_indecies = np.asarray(range(0,len(self.__trainset)))
        np.random.shuffle(data_indecies)
        for idx,r in enumerate(data_indecies):
            #break out after k steps:
            if idx >= k:
                break
            kmeans.loc[idx] = self.__trainset.iloc[r]
        return kmeans
```

ג. אלגוריתם אתחול ++ - ממקסם את ההסתברות לבחור את ה-$seed$ הבא רחוק כמה שיותר מה-$seed$ הנוכחי שנבחר:

```python
    """
    k - number of seeds we should pick in smart probability way
    """
    def __init_plusPlus(self,k):
        kmeans = pd.DataFrame(columns=Kmeans.XHeaders)
        samp_df = self.__trainset.sample(n=1)
        k_idx = [samp_df.index.values.astype(int)[0]]
        kmeans.loc[0] = np.asarray(samp_df)[0]
        #each iterate we pick the next vector:
        for i in range(1, k):
            last_mean = np.asarray(kmeans)
            last_mean = last_mean[len(last_mean) - 1]
            for idx, _ in enumerate(last_mean):
                last_mean[idx] = float(last_mean[idx])
            next_sample_vec = None
            max_prop = 0
            total_sum = 0.0
            #computing the sum:
            for control, row in self.__trainset.iterrows():
                if control not in k_idx:
                    row = np.asarray(row)
                    for idx, _ in enumerate(row):
                        row[idx] = float(row[idx])
                    local_sum = 0.0
                    for k1,k2 in zip(row,last_mean):
                        local_sum = (k1 - k2) ** 2
                    total_sum = abs(total_sum + local_sum)
            #computing the probability and pick the far away one:
            for control, row in self.__trainset.iterrows():
                if control not in k_idx:
                    row = np.asarray(row)
                    for idx, _ in enumerate(row):
                        row[idx] = float(row[idx])
                    local_sum = 0.0
                    for k1,k2 in zip(row,last_mean):
                        local_sum = (k1 - k2) ** 2
                    curr_prop = local_sum*(1/total_sum)
                    if curr_prop > max_prop:
                        max_prop = curr_prop
```

```python
                        next_sample_vec = row
            if next_sample_vec is not None:
                kmeans.loc[i - 1] = next_sample_vec
        return kmeans
```

3. הדפסת הסיווגים של כל פרח לפי טבלה המבטאת כמה פרחים מכל סוג סווגו לכל אחד משלושת הצברים:

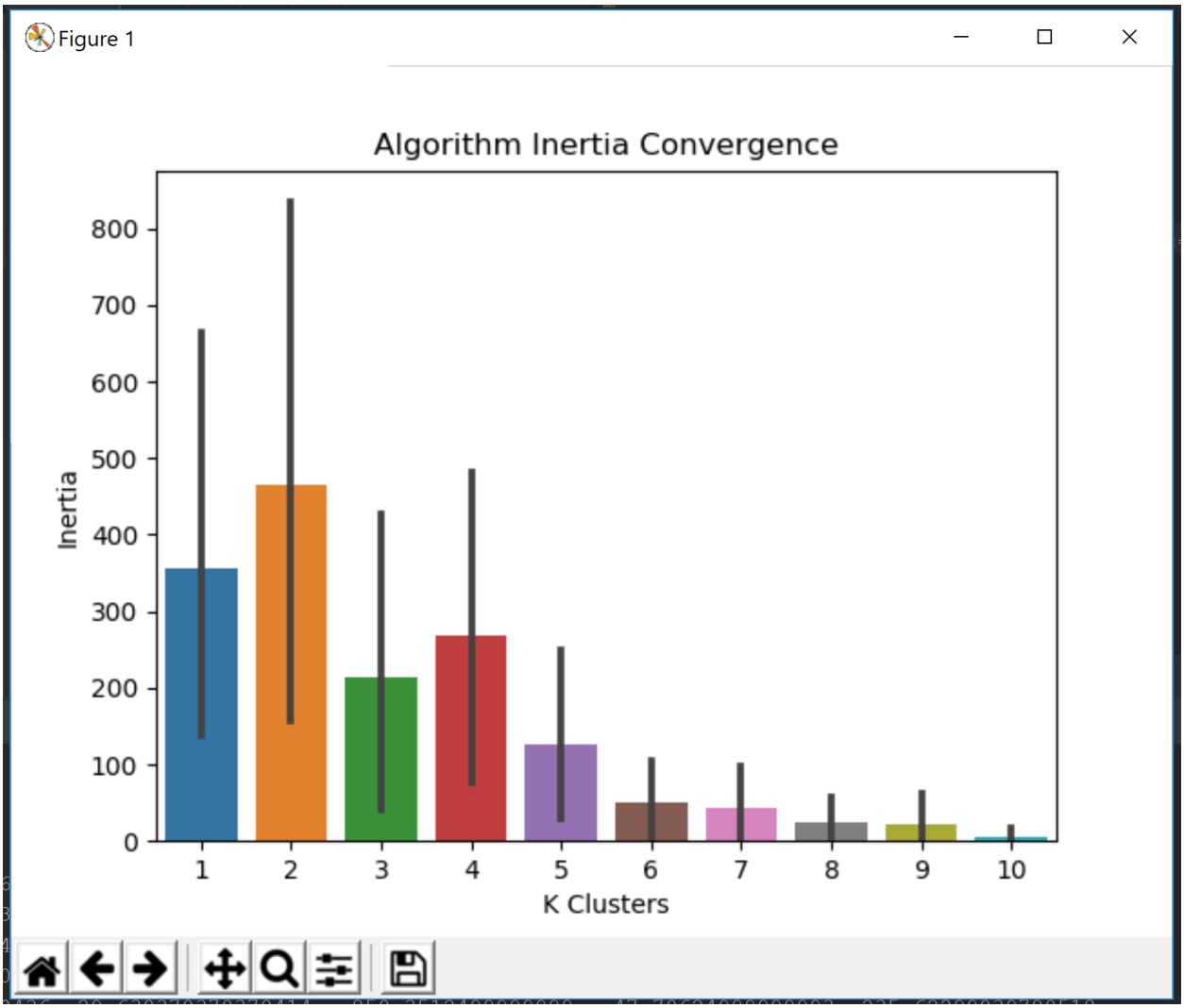
```
Using Selection method = random and k = 3 :
===================================
   setosa  versicolor  virginica
0       0          47         50
1      50           3          0
2       0           0          0
```

```
Using Selection method = forgy and k = 3 :
===================================
   setosa  versicolor  virginica
0       0          47         14
1       0           3         36
2      50           0          0
```

```
Using Selection method = init++ and k = 3 :
===================================
   setosa  versicolor  virginica
0      50           3          0
1       0          47         50
2       0           0          0
```

4. אלגוריתם האתחול $forgy$ **היה המוצלח ביותר** מבין השלושה (תיאר את הנתונים באופן יותר **נכון ועם חלוקה יותר עדינה** מיתר אלגוריתמי האתחול ובנוסף גם **תמיד** מוצא חלוקה ל-3 אשכולות).

5. שרטוט גרף שבציר ה-Y האנרציה שאליה התכנס האלגוריתם כאשר מריצים אותו עבור $K = 1,2,3,4,5,6,7,8,9,10$:



ניתן לראות שככל ש-k קטן גם האינרציה הולכת וקטנה עד שניהית זעירה ואף לא נראית בגרף. נתון זה לא מפתיע מהסיבה שככל ש-k גדל זה אומר שיש יותר $seeds$ במרחב שיכולים "לתפוס" יותר אירוסים קרובים ואז זה מגדיל את הסיכוי שעבור אירוס רחוק יהיה $seed$ קרוב שהוא יכול להשתייך אליו.

ה-k האופטימלי לפי הגרף הנ"ל הוא $k = 6$ מכיוון שעבורו סכום המרחקים הוא קטן מאוד ולכן מכיל **נקודות קרובות** בתוך האשכול ומייצר **מרחק גדול בין האשכולות** בנוסף החל מ-k זה יש התייצבות של ערכי האינרציה וזה אומר שדרושים **בסביבות** 6 אשכולות בכדי לקבל שגיאה קטנה שמחלקת את המרחב עם מינימום אשכולות ומקסימום נקודות בכל אשכול תחת התכונה שהוזכרה הנ"ל. לכן $k = 6$ בגרף **הנ"ל** הוא האופטימלי.

- <u>הערה –</u> מהבנת הנתונים ידוע ש-$setosa$ מופרד מהנתונים בצורה טובה ול-2 הסוגים דרושים עוד 2 אשכולות לפחות אבל מכיוון שהם "מעורבבים" קצת נצתרך יותר מ-2 אשכולות ולכן באופן כללי בין 3-5 אשכולות נוספים זה סביר בכדי לחלק את $versicolor$ ו-$virginica$.

<u>ניתן לראות את מטריצת הריצות הבאה שמכילה את המרחקים שיצרו את הגרף הנ"ל:</u>

	1	2	3	4	5	6	7	8	9	10
0	680.8244	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1619.4251975792097	845.7723562546502	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	372.12203125	1658.3829333333329	28.978645833333346	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	4.3073076923076944	1056.80984375	638.0414792899405	38.290819672131136	0.0	0.0	0.0	0.0	0.0	0.0
4	101.09760000000011	3.3362500000000006	234.6447258979206	642.3793047337278	410.3368175645087	0.0	0.0	0.0	0.0	0.0
5	126.58816326530618	3.176521739130436	29.630370370370414	850.3512499999999	47.70624999999993	235.62299039780518	0.0	0.0	0.0	0.0
6	58.816768	108.18065843621407	12.075013850415512	745.3250000000003	31.211198347107462	58.566796874999945	4.655000000000001	0.0	0.0	0.0
7	128.08419753086426	348.9508	937.57	2.7541071428571375	480.72512110726643	3.4816666666666666	214.77634920634918	136.80648760330587	0.0	0.0
8	97.47190399999997	575.5447500000001	0.95875	324.98571428571427	85.60333333333335	185.6438752362949	183.7625	23.63690000000002	20.5542148760330 58	0.0
9	367.0374999999999	51.83555555555564	257.0693750000001	77.84399999999997	204.6702	8.9546	28.572024793388437	86.30330812854442	203.47	54.4071597633136

6. לאחר שינויים "קלים" בנתונים הקלט ניתן לראות את האבחנות הבאות:

א. כאשר השינויים הם "קלים"-מספריים אז $random$ משנה את חלוקת הקבוצות לכן הוא אלגוריתם **שרגיש** לשינויים של נתוני הקלט (**ברוב** ההגרלות):

- *מצורף הקובץ עם השינויי* - $iris\_change.data$

```
Using Selection method = random and k = 3 :
=================================
   setosa  versicolor  virginica
0       0           1          1
1      50          49         49
2       0           0          0
```

ב. האלגוריתם $init++$ **בפעמים רבות** מבצע חלוקה יציבה יותר של הנתונים ובאופן עמיד לשינויים "קלים" בקלט לכן הוא פחות רגיש מ-$random$:

```
Using Selection method = init++ and k = 3 :
=================================
   setosa  versicolor  virginica
0       0          47         50
1      50           3          0
2       0           0          0
```

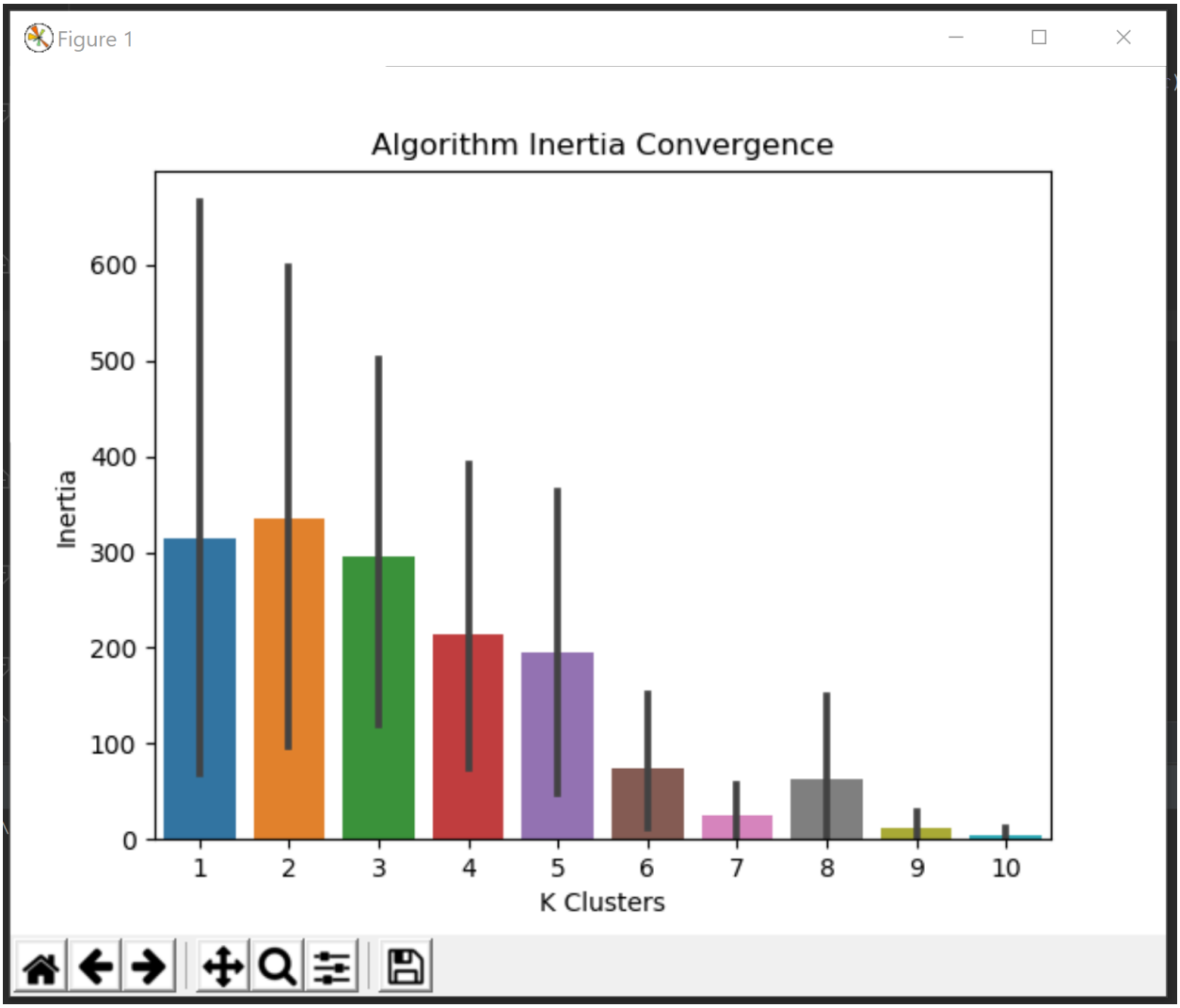
ג. כפי שניתן לראות פורגי מתאים את עצמו (באופן לא מפתיע) יותר לנתונים, יציב ומחלק את הקבוצות באופן יותר **נכון, עדין ועמיד** ביחס ל-$random$ ול-$init + +$ שמבצעים חלוקה יחסית **גסה ופחות נכונה**:

```
Using Selection method = forgy and k = 3 :
==================================
   setosa  versicolor  virginica
0       0          47         14
1       0           3         36
2      50           0          0
```

ניתן לראות זאת גם ב-k גבוה יותר בצורה ברורה יותר ש-$forgy$ מבצע אבחנה **עדינה** בין הקבוצות:

```
Using Selection method = forgy and k = 10 :
==================================
   setosa  versicolor  virginica
0       0           1          8
1      50           0          0
2       0          17          0
3       0          20          1
4       0           3          1
5       0           0         10
6       0           7          0
7       0           1         10
8       0           0         16
9       0           1          4
```

בנוסף ניתן לראות שגם גרף האינרציה של $forgy$ אכן שמר על צורתו:



7. לסיכום, $k-means$ זה אלגוריתם שמנסה לחלק את הנתונים לקבוצות באופן כזה שמנסה שהמרחקים בתוך כל קבוצה **קטנים** והמרחקים בין הקבוצות **השונות גדולים** ולכן במקרים בהם הקבוצות **לא מחולקות** באופן **ברור** במרחב למשל "מעורבבים" בצורה **צפופה** באופן **שלא ניתן** למצוא מישור/ישר מפריד כלשהו בניהם האלגוריתם לא ייצג טוב את חלוקת הקבוצות ויהיו בו המון שגיאות ולכן יגרור תוצאות לא טובות (למשל כמו סגנון החלוקה בין $versicolor$ ו-$virginica$ רק **באופן יותר קיצוני בהרבה**).

- למשל ניתן להבחין ב-$iris$ שקבוצת $setosa$ שמופרדת באופן ברור וכמעט מוחלט **מיתר** הקבוצות ולכן ברוב המוחלט של המקרים האלגוריתם כן הצליח לזהות את הקבוצה ולהפריד אותה מיתר הקבוצות במרחב **בצורה טובה**.

דוגמא להרצת התוכנית ב-*console*:

```
C:\Users\itgu1>C:\Users\itgu1\OneDrive\Desktop\Submission_5\Kmeans.py --file C:\Users\itgu1\OneDrive\Desktop\iris.data --selection_method all --plot_inertia_graph false

Using Selection method = forgy and k = 3 :
==================================
   setosa  versicolor  virginica
0       0           2         36
1      50           0          0
2       0          48         14
***
Using Selection method = random and k = 3 :
==================================
   setosa  versicolor  virginica
0      50           0          0
1       0           3         36
2       0          47         14
***
Using Selection method = init++ and k = 3 :
==================================
   setosa  versicolor  virginica
0       0          47         50
1      50           3          0
2       0           0          0
```