מבוא למערכות לומדות – 67577 – תרגיל 2

אופק אבידן – 318879574

2. חלק תיאורטי:

2.1. Hard- & Soft-SVM

2.1.1

In class we saw the Hard-SVM classification model.

1. Prove that the following Hard-SVM optimization problem is a Quadratic Programming problem:

$$\underset{(\mathbf{w},b)}{\operatorname{argmin}} ||\mathbf{w}||^2 \quad \text{s.t.} \quad \forall i \; y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{1}$$

That is, find matrices Q and A and vectors $\mathbf{a}$ and $\mathbf{d}$ such that the above problem can be written in the following format

$$\underset{\mathbf{v}\in\mathbb{R}^n}{\operatorname{argmin}} \tfrac{1}{2}\mathbf{v}^\top Q\mathbf{v} + \mathbf{a}^\top \mathbf{v} \quad \text{s.t.} \quad A\mathbf{v} \leq \mathbf{d} \tag{2}$$

Hint: Observe that $||\mathbf{w}||^2 = \mathbf{w}^\top \mathbf{I}\mathbf{w}$

נתבונן בבעיה:

$$\arg_{w,b} \min ||w||^2 \; s.t. \forall i \;\; y_i(< w, x_i > +b) \geq 1$$

באופן שקול ניתן לכתוב:

$$\arg_{(w,b)} w^T I w$$

$$s.t. \forall i \; y_i(< w, x_i > +b) \geq \; 1 \rightarrow y_i x_i^T w + y_i b \geq 1 \rightarrow -y_i x_i^T w - y_i b \leq -1$$
$$\rightarrow -y_i x_i^T w \leq y_i b - 1$$

נגדיר:

$$Q = I, \;\; A_{i'th\,row} = \; -y_i x_i^T, \;\; a = 0^{\rightarrow} \;\; V = \sqrt{2}W$$

ונקבל שהבעיה שקולה ל-

$$\arg_{V\in\mathbb{R}^n} \min \frac{1}{2} V^T Q V + a^T V \; s.t. \; Av \leq d$$

2. The ***Gaussian*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Gaussian likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\pi) \\ x_j|y=k &\overset{ind.}{\sim} \mathcal{N}\left(\mu_{kj}, \sigma_{kj}^2\right) \end{aligned} \tag{4}$$

for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.

(a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i, y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (3) under assumptions (4). Fitting means finding the expressions for the maximum likelihood estimators.

(b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (3) under assumptions (4). You are encouraged to use the results from (3.a).

ראשית נמצא ביטוי ל-*likelihood*:

$$L(\Theta|X,y) = \prod_{i\in[m]} f_{x,y}(x_i, y_i|\Theta) =_{\text{מהנחה}} \prod_{i\in[m]} N\left(X_i \middle| M_{y_i}, \sigma_{y_i}^2\right) \cdot Mult(y_i|\pi)$$

פונקציית *log* הינה מונו' עולה, ולכן מקסום הביטוי ימקסם גם את ה-*log* של הביטוי, לכן:

$$\begin{aligned} \log\big(L(\Theta|X,y)\big) = l(\Theta|X,y) &= \Sigma_{i\in[m]}\left(\log N\left(X_i \middle| M_{y_i}, \sigma_{y_i}^2\right) + \log\big(Mult(y_i|\pi)\big)\right) \\ &= \Sigma_{i\in[m]}\left(\log\left(\frac{1}{\left(\sqrt{2\pi\sigma^2 y_i}\right)} \exp\left(-\frac{(x_i - \mu y_i)^2}{2\sigma^2 y_i}\right)\right) + \log(\pi y_i)\right) \\ &= -\frac{m}{2}\log(2\pi) + \Sigma_{k\in m} n_k \log(\pi k) - \frac{1}{2} n_k \log(\sigma_k^2) \\ &- \frac{1}{2\sigma_k^2}\Sigma_{i:y_i=k}(X_i - \mu_k)^2 \end{aligned}$$

כאשר $n_k = \Sigma_{i\in[m]} 1_{[y_i=k]}$.

כעת נגזור את הביטוי כדי למצוא מקסימום:

- נגזור לפי תוחלת:

$$\frac{1}{\sigma^2 k}\Sigma_{i:y_i=k}(X_i - \mu_k) = 0$$

$$* \mu_k = \frac{1}{n_k}\Sigma_{i:y_i=k} X_i$$

- נגזרת לפי שונות:

$$-\frac{n_k}{\sigma_k^2} - \frac{1}{2\sigma_k^4}\Sigma_{i:y_i=k}(X_i - \mu_k)^2 = 0 \rightarrow {\sigma^*}_k^2 = \frac{1}{n_k}\Sigma_{i:y_i=k}(X_i -* \mu_k)^2$$

כעת, נגדיר לגראנז'יאן:

$$G = l(\Theta|X,y) - 2\left(\Sigma_{k\in[m]}\pi_k - 1\right)$$

נגזור לפי π_k ונשווה ל-0:

$$0 = \left(l(\Theta|X,y) - 2\left(\Sigma_{k\in[m]}\pi_k - 1\right)\right)' = \frac{n_k}{\pi_k} - \lambda \to \pi_k = \frac{n_k}{\lambda}$$

נזכר כי $\Sigma_k \pi_k = 1$ ונמצא את λ:

$\Sigma_{k\in[m]}\pi_k = \Sigma_{k\in[m]}\frac{n_k}{\lambda} = 1 \to \lambda = m$.

לסיכום, המשערך של כל π_k הממקסם את ה-*likelihood* הינו:

$$\pi_k^{MLE} = \frac{n_k}{m}$$

כאשר $n_k = \Sigma_{i\in[m]} 1_{[y_i=k]}$.

כעת נחשב את הלייקליהוד:

$$L(\Theta|X,y) = \prod_{i\in[m]} f_{x,y}(x_i, y_i|\Theta) =_{\text{מהנחה}} \prod_{j\in[m]} \prod_{i\in[m]} N\left(X_i \middle| M_{y_i}, \sigma_{y_i}^2\right) \cdot Mult(y_i|\pi)$$

כעת, משיקולי מונוטוניות ומיקסום נפעיל לוג על כל הלייקליהוד:

$$\log\left(L(\Theta|X,y)\right) = \Sigma_{i\in[m]}((\log\left(mult(y_i,\pi_k)\right) + \Sigma_j \in [d] \log\left(N\left(x_{ij}\middle|\mu_{yij}, \sigma_{y_{ij}}^2\right)\right)$$
$$= \Sigma_k n_k \log \pi_k + \Sigma_k \Sigma_{j\in[d]} 1_{[y:=k]} \log\left((N\left(X_{ij}\middle|\mu_y := ij\ \sigma_y^2\right)\right)$$

נשים לב כי נוכל להסתכל על כל פיצ'ר j ולחשב את האומדים בעזרת החישוב שביצענו בסעיף הקודם ועבור כל k לקבל $\left(x_j \middle| y = k\right) \sim N(\mu_k, j\ , \sigma_k^2, j)$

עם אותם $\mu_{k,j}$, $\sigma_{k,j}^2$ שחושבו. בנוסף, מכך שהפיצרים בת"ל, לכל k אנו מקבלים מטריצה אלכסונית שונה ממחלקה למחלקה לכן נוכל לרשום את האומדים כך:

$$\pi_k^{MLE} = \frac{n_k}{m}, \quad M_{k,j}^{MLE} = \frac{1}{n_k}\Sigma_{i:y_i=k} K_{i,j}, {\sigma_{k,j}^2}^{MLE} = \frac{1}{n_k}\Sigma(X_{i,j} - M_{k,j}^{MLE})$$

3. The ***Poisson* Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Poisson likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\pi) \\ x_j | y = k &\overset{ind.}{\sim} \text{Poi}(\lambda_{kj}) \end{aligned} \tag{5}$$

for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.

(a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i, y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (3) under assumptions (5).

(b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (3) under assumptions (5). You are encouraged to use the results from (4.a).

א. ראשית נחשב את הלייקליהוד:

$$L(\Theta|X,y) = \prod_{i\in[m]} f_{x,y}(x_i, y_i|\Theta) = \prod_{i\in[m]} Pois(Xi|\lambda_i y_i) \cdot Mult(y_i|\pi)$$

כעת, משיקולי מונוטוניות ומיקסום נפעיל לוג על כל הלייקליהוד:

$$\log\left(L(\Theta|X,y)\right) = \Sigma_{i\in[m]}((\log\big(Pois(Xi|\lambda_i y_i)\big) + \Sigma_j \in [d] \log\big(Mult(y_i|\pi)\big)$$
$$= -m\lambda + \log(\lambda)\,\Sigma_i X_i - \Sigma\log\left(Xid\right))$$

$$\Theta^{MLE} = \lambda$$

נגזור, נשווה ל-0 ונקבל:

$$0 = (-m\lambda + \log(\lambda)\,\Sigma_i X_i - \Sigma\log(Xid)))' = -m + 1/\lambda \ * \Sigma\_i \ \ X\_i$$

לכן:

$$\pi_k^{MLE} = \frac{nk}{m}, \qquad \lambda_k^{MLE} = \frac{1}{n_k}\Sigma_i 1_{[y_i=k]} X_i$$

כאשר $n_k = \Sigma_i 1_{[y_i=k]}$

ב. נניח כי $X \in \mathbb{R}^d$, כלומר לכל דגימה יש d פיצ'רים.

נוכל להסתכל על כל פיצ'ר j ולחשב את האומדים בעזרת החישוב שביצענו בסעיף הקודם ועבור כל מחלקה k לקבל $\left(X_j, y = k\right) \sim Pois(\lambda_k, j)$ עד אותו λ_k, j שחישבנו.

לכן נוכל לרשום את האמדים כך:

$$\lambda_{k,j}^{MLE} = \frac{1}{n_k}\Sigma_i 1_{[y_i=k]} X_{i,j}$$

$$\pi_k^{MLE} = \frac{n_k}{m}$$

כאשר $n_k = \Sigma_i 1_{[y_i=k]}$

חלק פרקטי:

3.1

3 Practical Part

In the following part, you will implement some classifiers you have seen in class and the recitation. You will compare the behavior of the different classifiers over different data scenarios.

3.1 Perceptron Classifier

Based on Lecture 2 and Recitation 5. Complete the following implementations

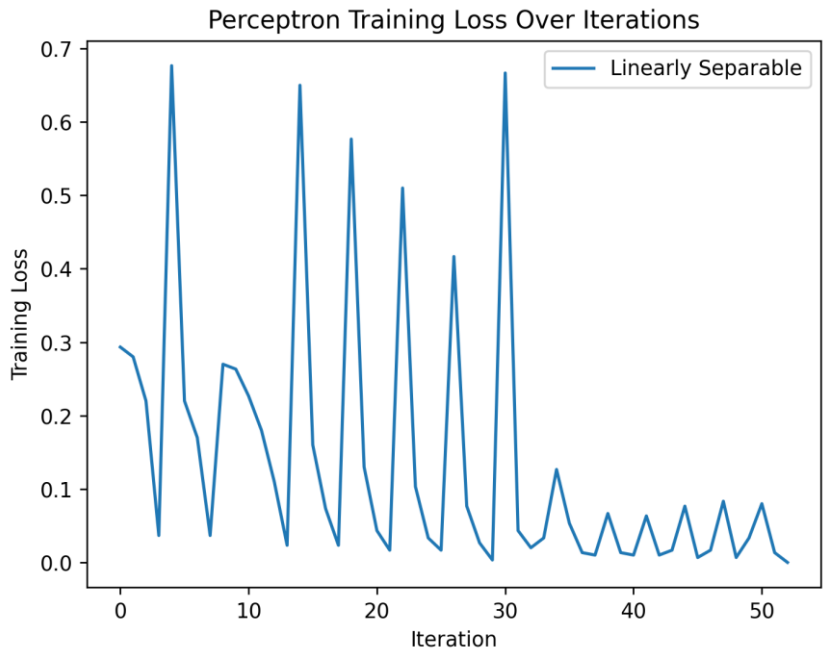
- Implement the `misclassification_error` function in the `loss_functions.py` file as described in the function documentation.
- Implement the Perceptron algorithm in the `classifiers.py` file as described in the class documentation. In the implementation use the misclassification error implemented above.

In the `classifiers_evaluation.py` file, implement the `run_perceptron` function as described in documentation.

- As good practice, to retrieve the loss at each iteration, we recommend not changing the previously implemented Perceptron class. Instead **specify a callback function** which receives the object and uses its `loss` function to calculate the loss over the training set. Store these values in an array to be used for plotting.

1. Fitting and plotting over the `linearly_seprable.npy` dataset, what can we learn from the plot?

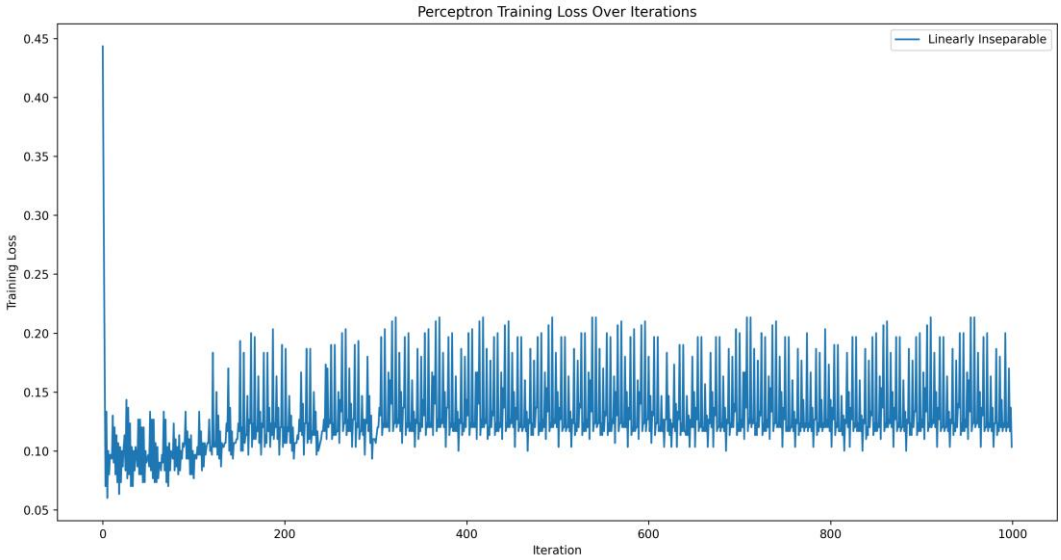
1. ראשית, מאחר והדאטה אכן פריד לינארית (כלומר, קו ישר (בדו מימד), מישור (בתלת מימד), או היפר מישור (במימדים גבוהים יותר) יכולים לחלק בצורה מושלמת את נקודות הנתונים של מחלקות שונות), ניתן לראות שהשגיאה האמפירית הסופית (לאחר כ-50 איטרציות) היא 0 כצפוי (אם לא היה פריד לינארית, לא היינו מגיעים ל-0). בנוסף, עד לאיטרציה 30 אפשר להבחין בקפיצות גדולות בשגיאות, זאת בשל שינוי הערכים שחוזרים מה-*misclassification error*. באופן כללי, אם הדאטה פריד לינארית, אז האלגוריתם פרספטרון אמור להתכנס. בכל איטרציה הוא "משפר" את הוקטורים w ו-b, בכיוון השיפור לפי הגודל של ה-*learning rate*. ייתכן שה-*learning rate* גדול מדי במקרים מסויימים, כלומר התזוזה תהיה גדולה מדי ומכאן הקלסיפיקציה תפגע והלוס גדל משמעותית.



.2

2. Next run the Perceptron algorithm over the `linearly_inseprable.npy` dataset and plot its loss as a function of the iterations. What is the difference between this plot and to the one in the previous question? How can we explain the difference in terms of the objective and parameter space?

כפי שהשם מרמז, הדאטה לא פריד לינארית. אנחנו רואים שהאלגוריתם פרספטרון לא מצליח להתמודד עם מידע כזה בדיוק כפי שנלמד בכיתה, והוא לא מצליח להתכנס גם אחרי 1000 איטרציות (לשם השוואה, בסעיף 1 ביצענו פחות מ-60). כלומר, לא קיים קו (או כאמור בהכללה, היפר מישור), שמחלק את הדאטה כך שהשגיאה האמפירית (לפי *misclassification error*) היא 0. מכאן שהאלגוריתם ימשיך בחיפוש אחר הפתרון עד שהוא מגיע למספר מקסימלי של איטרציות שהוגדר מראש (כאמור, במקרה זה, 1000).



3.2 Bayes Classifiers

Based on Lecture 2 and Recitation 5.

Complete the following implementations

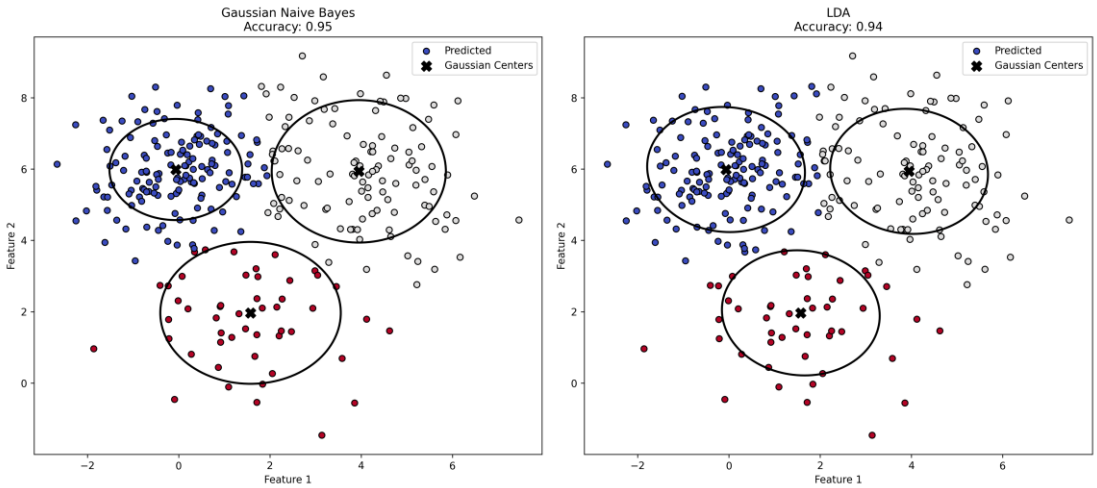
- Implement the `accuracy` function in the `loss_functions.py` file as described in the function documentation.
- Implement the `LDA` classifier in the `classifiers.py` file as described in the class documentation. Use expressions derived in class.
- Implement the `GaussianNaiveBayes` classifier in the `classifiers.py` file as described in the class documentation. Use expressions derived in question 3b of the theoretical part.

Then, implement and answer the following questions:

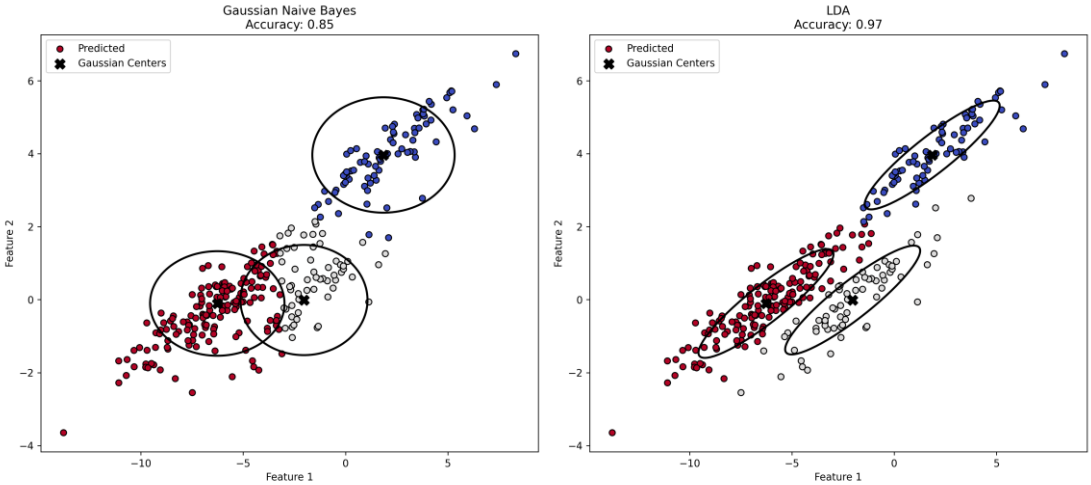
1. In the `compare_gaussian_classifiers` function, `classifiers_evaluation.py` file, load the `gaussians1.npy` dataset. Fit both the Gaussian Naive Bayes and LDA algorithms previously implemented. Plot the following:
   - A single figure with two subplots:
     (a) 2D scatter-plot of samples, with marker color indicating Gaussian Naive Bayes *predicted* class and marker shape indicating *true* class.
     (b) 2D scatter-plot of samples, with marker color indicating LDA *predicted* class and marker shape indicating *true* class.
     (c) Provide classifier name and accuracy (over train) in sub-plot title
   - For both subplots add:
     (a) Markers (colored black and shaped as 'X') indicating the center of fitted Gaussians.
     (b) An ellipsis (colored black) centered in Gaussian centers and shape dictated by fitted covariance matrix.
   - Specify dataset name in figure title.

   Explain what can be learned from the plots above regarding the distribution used to sample the data?

1. ניתן לראות שהגרפים שהתקבלו משני המודלים דומים מאוד, עם יתרון קטן לטובת ה-GNB. זאת מאחר וההתפלגות שממנה דגמו את אוסף הנתונים (הדאטאסט שקיבלנו) היא בעלת שונות משותפת אפס (כלומר, הדגימות בלתי תלויות). הדבר נובע מההנחה של GNB שהנתונים בת"ל ולכן התוצאה של שני המודלים תהיה זהה. כמו כן, אליפסה עגולה (שבה הצירים דומים באורכם) מעידה על כך שהשונות במשתנים השונים דומה. אם האליפסה מאורכת בכיוון מסוים, זה מצביע על כך שהשונות במשתנה אחד גבוהה יותר מאשר במשתנה אחר.



2. Repeat the procedure above (while avoiding code repetition) for `gaussians2.npy`. What is the difference between the two scenarios? What can be learned regarding the distribution used to sample the data? Which of the two classifiers better matches this dataset and why?



ניתן לראות שהדיוק ממודל *LDA* גבוה מהדיוק במודל *GNB* בניגוד לגרף הקודם. התופעה נובעת ככל הנראה מהיותם של הפיצ׳רים תלויים זה בזה בניגוד להנחה של *GNB* שהם בת״ל. בנוסף, ניתן להבחין שהאליפסה בגרף ה *LDA* מתוחה בכיוון הדגימות מה שמסביר את הדיוק הגבוה יותר (כאמור, אם האליפסה מאורכת בכיוון מסוים, זה מצביע על כך שהשונות במשתנה אחד גבוהה יותר מאשר במשתנה אחר). גם כיוון האליפסה מראה את השונות בין המשתנים. אליפסה מוטה מעידה על כך שיש קשר (קורלציה) בין המשתנים. כיוון האליפסה יצביע על הכיוון שבו המשתנים מתפזרים יחד. אם האליפסה מוטה בכיוון מסוים, זה מעיד על כך ששני המשתנים משתנים באותו כיוון. אם האליפסה ישרה, זה מעיד על כך שאין קורלציה בין המשתנים. נשים לב שהאליפסות אינן מקבילות לצירים ב-*LDA*, כלומר האליפסות מוטות, בניגוד לסעיף הקודם, וזו עשויה להיות עדות נוספת לכך שהפיצ׳רים תלויים.

שימוש ב-Chat GPT

השתמשתי בצ׳אט ג׳יפיטי ברוב החלק התכנותי. לצערי אין לי ניסיון ב-*numpy* ולכן נעזרתי בו לא מעט, בעיקר בהדפסת הגרפים. אני כן מבין לאחר מכן את מה שהוא ביצע, ואני משתדל לפרט המון בדו״ח.