

## למידת מכונה עמוקה (חלק א) משימות ותרגילים

מחבר

ידידיה כהן

ייעוץ מדעי ופדגוגי:

גדי הרמן

אריאל בר יצחק

מפמ"ר מגמת הנדסת אלקטרוניקה ומחשבים:

שלומי אחנין

תשפ"ד

© כל הזכויות שמורות למשרד החינוך

מרכז המורים הארצי למקצועות הטכנולוגיים, מור-טק

הפרויקט מבוצע על ידי מוסד הטכניון עפ"י מכרז 22/11.2020

הפרויקט מבוצע עבור המזכירות הפדגוגית, משרד החינוך

האוגדן יצא לאור במימון האגף למדעים במזכירות הפדגוגית ומינהלת מל"מ המרכז הישראלי לחינוך מדעי טכנולוגי.

אין לשכפל, להעתיק, לצלם, להקליט, לתרגם, לאחסן במאגר מידע, לשדר או לקלוט בכל דרך או אמצעי אלקטרוני, אופטי או מכני או אחר כל חלק שהוא מהחומר שבחוברת זו. שימוש מסחרי מכל סוג שהוא בחומר הכלול בחוברת זו אסור בהחלט אלא ברשות מפורשת בכתב מהמו"ל.

## תוכן עניינים

1.....	מבוא וסביבת עבודה
1.....	מבוא ללמידת מכונה
2.....	טעינה ותצוגה של נתונים באתר Colab
10.....	רגרסיה ליניארית
10.....	מבוא ודוגמאות
14.....	חיזוי באמצעות רגרסיה ליניארית תוך שימוש ב- scikit-learn
18.....	חיזוי הכולל מאפיינים מרובים
19.....	תרגיל השלמת קוד Linear regression
20.....	פונקציית המחריר ו-gradient descent
23.....	רגרסיה ליניארית חד/רב ממדית
24.....	רגרסיה לוגיסטית
24.....	הערכת ביצועים של מסווג
26.....	מדידת ביצועים של מסווג – תרגיל
26.....	רגרסיה לוגיסטית – הגדרה
29.....	סיווג על ידי רגרסיה לוגיסטית – תרגיל
30.....	תהליך האימון ברגרסיה לוגיסטית
31.....	רגרסיה לוגיסטית ב-sklearn
33.....	רגרסיה לוגיסטית ב-sklearn – תרגיל
36.....	למידה עמוקה - רשת נוירונים
36.....	טעינת dataset לוגול Colab
37.....	למידה עמוקה
38.....	נוירון
42.....	Sklearn MLPClassifier
46.....	MLP Classifier – תרגיל
52.....	הפרדת ה-Dataset ל-Train ו-Test
54.....	חלוקה ל-Train, Test – תרגיל

## מבוא וסביבת עבודה

### מבוא ללמידת מכונה

מדעי הנתונים – מדע בין-תחומי עם שילוב של תחום מדעי המחשב, תחום המתמטיקה/סטטיסטיקה + התחום המדעי של התוכן של המידע. ולכן, פרויקטים מתבצעים כאן בדרך כלל מתבצעים ע"י שיתוף פעולה בין מומחים מהתחומים השונים.

הפרדיגמה החדשה של למידת מכונה – מזינים למכונה דוגמאות של נתוני כניסה ופתרונות והמכונה קובעת את חוקי הקשר (אלגוריתם) בין נתונים לפתרונות ולאחר מכן מפעילה את החוקים הללו על נתוני כניסה של בעיות "אמיתיות" ומחשבת את הפתרונות. וזאת, לעומת הגישה הקלאסית בתכנות – מזינים למכונה נתוני כניסה וחוקי הקשר (אלגוריתם) בין נתונים לפתרונות והמכונה מחשבת את הפתרונות.

Artificial Intelligence – (בינה מלאכותית) – ללמד מכונה לחשוב כמו בני אדם. בתוך התחום של הבינה המלאכותית יש את התחום של Machine Learning (לימוד מכונה) – המכונה פותרת בעיות באמצעות דוגמאות ללא חוקי קשר. בתוך התחום של לימוד מכונה יש את התחום של Deep Learning (למידה עמוקה) - למידה באמצעות מנגנון artificial neural networks (רשתות עצביות).

בתוך ה-Machine Learning קיימים 3 תחומים עיקריים:

- Supervised learning – דוגמאות עם תיגום כפי שדיברנו קודם.
- Un-Supervised learning – למשל, ניתן למחשב הרבה דוגמאות של אותיות האלף-בית (בלי לתייגן) והמחשב ידע לחלקן לקבוצות של האותיות השונות על בסיס הקירבה בין התמונות השונות של אותה אות.
- Reinforcement learning – למידה מתוך התנסות ללא השלב של האימון המוקדם שב-Supervised. רלוונטי, למשל, לתחום של משחקים בו המכונה לומדת מניצחונות/הפסדים ומשתפרת עם הניסיונות.

רוב השוק נמצא ב-Supervised ולכן אנו נתמקד בתחום הזה.

סוגי בעיות:

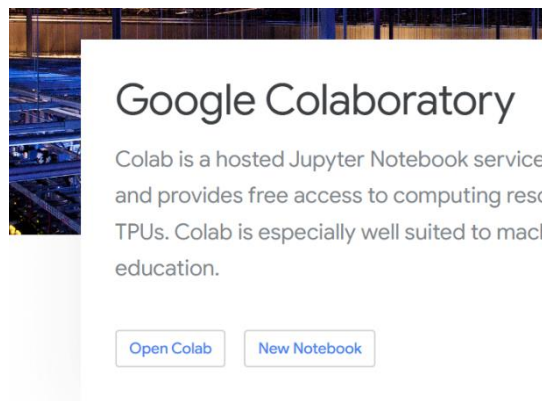
- זיהוי (Classification) (Classification)
  - Binary Classification – זיהוי מצב (מחלקה) מתוך 2 מצבים (מחלקות) אפשריים. למשל, קבלת mail והחלטה אם זה spam או לא. ניתן לפתור בעיות אלה בשיטות של Multiclass Classification אבל בגלל שלסוג הבעיות הללו יש פתרון יעיל יותר ואלו בעיות נפוצות, הגדירו לנושא תחום נפרד.

- Multiclass Classification – זיהוי מחלקה מתוך יותר מ-2 מחלקות אפשריות. למשל, זיהוי תמונת חתול מתוך אוסף האפשרויות של תמונת כלב, חתול, סוס, דג וציפור. הפתרון ייתן אחוזי הסתברות של כל אחת מהאפשרויות עם סכום הסתברות כוללת של 100% והאפשרות עם ההסתברות הגבוהה ביותר היא המומלצת.
- Multi-Label Classification – זיהוי יותר ממחלקה אחת מתוך יותר מ-2 מחלקות אפשריות. למשל, זיהוי תמונת חתול וציפור מתוך אוסף האפשרויות של תמונת כלב, חתול, סוס, דג וציפור. הפתרון ייתן אחוזי הסתברות של כל אחת מה- labels וסכום ההסתברויות יהיה גדול מ-100%.
- זיהוי ומיקום (Classification + Localization) - תיחום האובייקט המזוהה באמצעות מלבן.
- גילוי (Object Detection) זיהוי ותיחום מלבני של מספר אובייקטים (מאותו סוג ומסוגים שונים) באותה תמונה.
- Instance Segmentation - גילוי ברמה של פיקסלים. כמו ב-Object Detection אבל התחום הוא עם מעטפת של קו המקיף את גבולות התמונה של האובייקט.

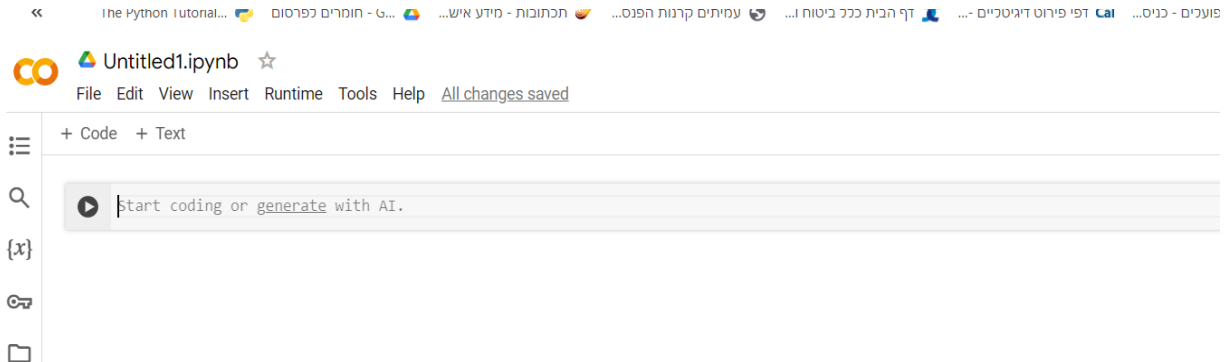
## טעינה ותצוגה של נתונים באתר Colab

אתר Google Colab Notebooks הוא סביבת פיתוח בפייתון בענן של Google. האתר משתמש במשאבים (כוח חישוב, זיכרון, ספריות, ...) של google וביצוע restart מתבצע למעשה על ה-session שלנו באתר ולא על המחשב שלנו. כל העבודות שנבצע באתר הזה מתבצעות כ"מחברת אינטראקטיבית" נשמרות באתר וזמניות להמשך פיתוח בכניסות הבאות.

[קישור](#) לאתר Google Colab. ניתן להגיע לאתר גם על ידי חיפוש בגוגל: "גוגל קולאב" וכניסה ללינק הראשון.

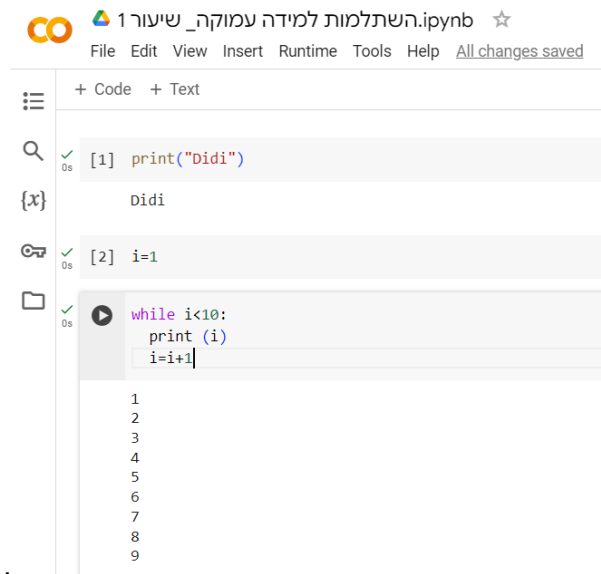


היות וזו המחברת הראשונה שלכם, יש לבחור ב-New Notebook ומקבלים את המסך:



יתרונות של פיתוח פייתון ב-Colab (לעומת Visual Studio, למשל):


- אין צורך לטעון ספריות.
- הכל נשמר באופן מסודר בענן.
- ניתן להריץ קטעים של התוכנה.
- ניתן לשלב במחברת קטעי טקסט, תמונות, טבלאות.
- שימוש חינמי במשאבים (כולל GPU) של גוגל.
- כל מה שנדרש מהמחשב עליו עובדים הוא דפדפן.




את המשך התיעוד המפורט של פרק זה רושמים


במחברת הראשונה שנתחה ושומרים אותה בשם/קישור "[למידה עמוקה שיעור 1](#)". במחברת מתועדים הנושאים הבאים:

## כתיבת קוד (פייתון) והרצתו

כתיבת הקוד מתבצעת כתא (cell) המורץ בנפרד באמצעות  ומקבל מספר שוטף של ההרצה (המספר שבסוגריים []). המחשב זוכר את ערכי המשתנים ולכן הרצה נוספת של הבלוק הארון לא תדפיס כלום כי הערך הנוכחי של i הוא 10.

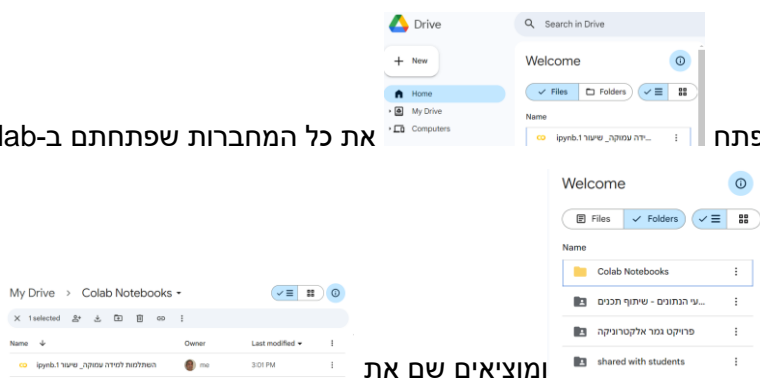
אפשר להריץ:

- תא בודד - באמצעות  או Runtime/Run the focused cell
- את כל התאים – באמצעות Runtime/Run All
- את כל התאים שלפני התא הנבחר – באמצעות Runtime/Run before
- את כל התאים החל מהתא הנבחר – באמצעות Runtime/Run after
- את כל התאים הנבחרים (באמצעות Cntl ולחצן שמאל של העכבר) - באמצעות Runtime/Run selection

המחברות נשמרות ב-google drive האישי. ניתן להגיע ל-drive האישי באמצעות  בסרגל הכלים ובחירת




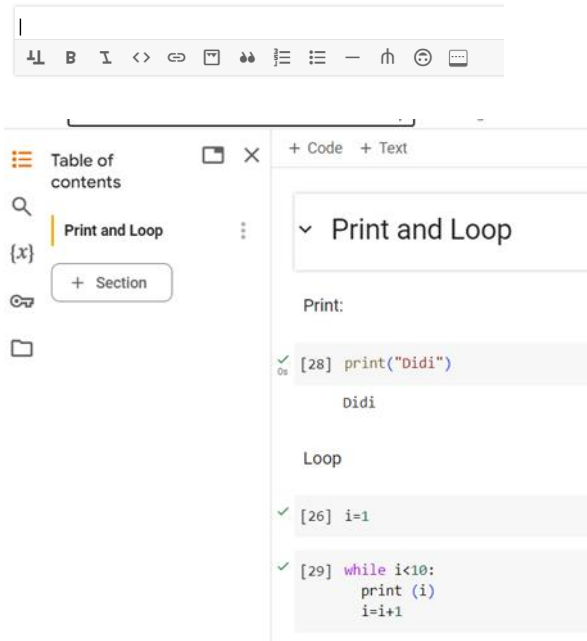
על מנת לאתר בקלות במסך שיפתח את כל המחברות שפתחתם ב-Colab, יש




## שילוב טקסט ותמונות

שילוב שורות טקסט ותמונות בין תאי הקוד.

שילוב שורת טקסט ככותרת למספר תאים ושורות טקסט באמצעות  בסרגל שורות הכותרת נכנסות לתוכן העניינים המוצג בחלון משמאל ומאפשרות הסתרה וגילוי של קטעים במחברת לנוחות ויעילות העבודה.



שילוב תמונות באמצעות . אם מקבלים כתובת קישור במקום התמונה, יש ללחוץ על Shift Enter או ללחוץ בכל מקום מחוץ לתא.

### טיפול בטבלאות והצגת גרפים

על מנת לעבוד בפייתון יש לבצע יבוא של ספריות (כל הספריות הנדרשות מותקנות ב-Colab). מתחילים עם הספריות הנדרשות על מנת לטפל בטבלאות ונציג גרפים:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- ספריית numpy פועלת על מערכים
- ספריית pandas פועלת על טבלאות נתונים
- ספריית matplotlib היא ספריית גרפית (בסיס)
- ספריית seaborn היא ספרייה גרפית (הרחבה לסטטיסטיקה)

כל הקיצורים כאן הם השמות המקובלים בכל העולם ויש להשתמש בהם ולא בשמות אחרים.



לתרגול פתחו את הקישור הבא:

[https://colab.research.google.com/drive/1\\_s1uH0BseMDTbBGvJ6lgkJk1aMt5dok?usp=share\\_link](https://colab.research.google.com/drive/1_s1uH0BseMDTbBGvJ6lgkJk1aMt5dok?usp=share_link)

לאחר פתיחת המחברת מקבלים קטעי קוד המדגימים יבוא נתונים המייצגים מידע על עלי כותרת בפרחי IRIS:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import ConfusionMatrixDisplay, r2_score
from sklearn.model_selection import train_test_split
```

Iris dataset

```
[ ] df = pd.read_csv('https://raw.githubusercontent.com/arielb30/datasets/main/Iris.csv')
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...

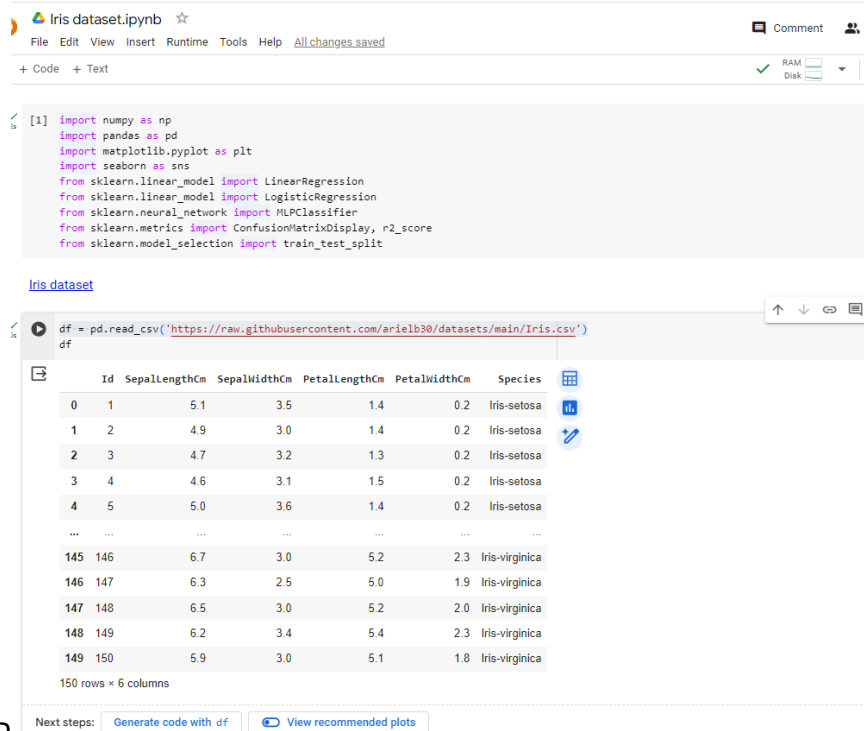
בקצה הימני של שורת התפריט רשום Changes will not be saved וזאת משום שזו המחברת של משתמש אחר. ולכן יש לשמור אותה באמצעות Copy To drive. השמירה יוצרת מחברת חדשה בשם Copy of Iris dataset עם הקישור הזה. ניתן לשנות את השם (אני שיניתי מ-Copy of Iris dataset ל-Iris dataset) ואז מקבלים:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import ConfusionMatrixDisplay, r2_score
from sklearn.model_selection import train_test_split
```

Iris dataset

```
[ ] df = pd.read_csv('https://raw.githubusercontent.com/arielb30/datasets/main/Iris.csv')
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa



```

[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import ConfusionMatrixDisplay, r2_score
from sklearn.model_selection import train_test_split

iris_dataset

df = pd.read_csv("https://raw.githubusercontent.com/arielb30/datasets/main/Iris.csv")
df
    
```

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	Iris-setosa
1	2	4.9	3.0	1.4	Iris-setosa
2	3	4.7	3.2	1.3	Iris-setosa
3	4	4.6	3.1	1.5	Iris-setosa
4	5	5.0	3.6	1.4	Iris-setosa
...	...	...	...	...	...
145	146	6.7	3.0	5.2	Iris-virginica
146	147	6.3	2.5	5.0	Iris-virginica
147	148	6.5	3.0	5.2	Iris-virginica
148	149	6.2	3.4	5.4	Iris-virginica
149	150	5.9	3.0	5.1	Iris-virginica

הפלטים המוצגים במחברת הם

תוצאת הריצה האחרונה אצל המשתמש האחר. על מנת להריץ את המחברת שלי, יש לבצע Runtime/Run all ואז מקבלים:

השינוי באזור הימני של סרגל הכלים מ- Connect ל- RAM Disk אומר שהדפדפן התחבר לשרת ה-Colab והשתמש במשאביו על מנת לבצע את הריצה.

השורה [iris dataset](#) היא שורת טקסט של קישור לאתר caggle, הידוע לכל קהילת ה-Data Science, בו נמצא בסיס הנתונים המקורי של פרח ה-IRIS (ועוד הרבה בסיסי נתונים אחרים). ההוראה:

```
df = pd.read_csv('https://raw.githubusercontent.com/arielb30/datasets/main/Iris.csv')
```

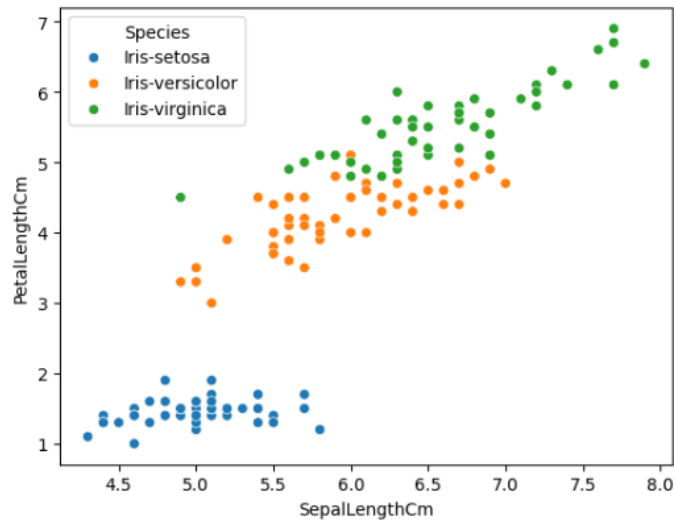
עושה השמה לעצם בשם df (קיצור של Data Frame) שהוא למעשה טבלה (=Frame).

pd.read\_csv – מספריית pandas (pd) היא פעולת קריאה מקובץ CVS (קובץ בו כל שורה היא רשומה עם ערכים מופרדים בפסיקים). בתוך הסוגריים נמצא הקישור לקובץ ב-github של בעלי הקובץ.

ההוראה df בשורה שלאחר מכן מציגה את הטבלה.

פקודה נוספת של הצגת נתוני הטבלה כגרף פיזור של שני משתנים :

```
[4] sns.scatterplot(data=df, x='SepalLengthCm', y='PetalLengthCm', hue='Species');
```

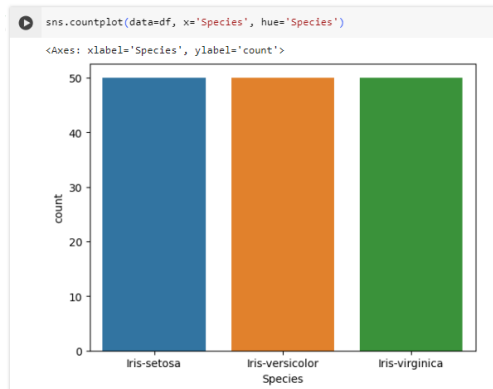


המקרא והצבעים נוצרים אוטומטית לפי 'hue='Species'.

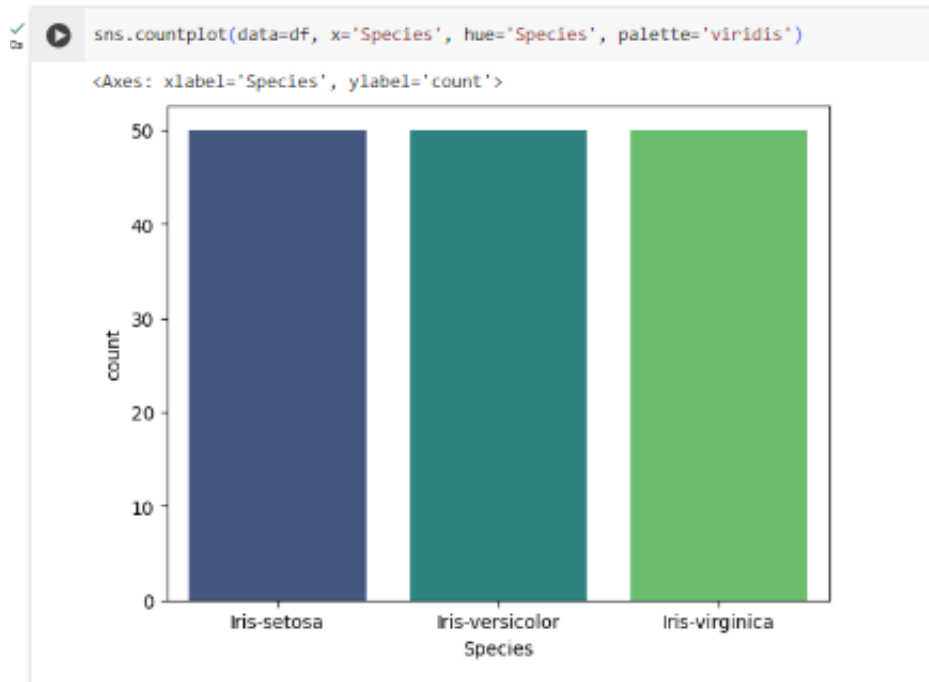
```
df.columns
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species'], dtype='object')
```

ההוראה df.columns מאפשרת העתקה קלה של שמות העמודות בטבלה.

על מנת לספור את כמות הפרחים מכל סוג בטבלה:



ניתן לשנות את מפת הצבעים באמצעות בחירת ה-palette:



## גרסיה לינארית

### מבוא ודוגמאות

מתחילים בהדגמה בתחום למידת מכונה הנקרא גרסיה לינארית, כי התהליך פשוט יחסית להבנה ולהוראה. גרסיה – חיזוי משתנה רציף בעל מספר אינסופי של ערכים. למשל, משקל, אורך, מחיר וכו'. וזאת לעומת משתנה בדיד בעל מספר סופי של ערכים כמו מספר ילדים במשפחה, יום בשבוע, ערים במדינה וכו'. להלן הדגמת התהליך של חיזוי מחיר (=מטרה) של בית לפי המאפיינים שלו שמשפיעים על המחיר:

### גרסיה: חיזוי משתנה רציף

מאפיינים

מטרה

bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	price
3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	221900
3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	538000
2	1	770	10000	1	0	0	3	6	770	0	1933	0	98028	180000
4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	98136	604000

חיזוי מחיר בית

3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	98074	
---	---	------	------	---	---	---	---	---	------	---	------	---	-------	--



דוגמאות → אימון → מודל → חיזוי מחיר בית לא ידוע



הטכניון  
מכון טכנולוגי  
לישראל

<https://www.kaggle.com/burhanykiyagolu/pr edicting-house-prices>

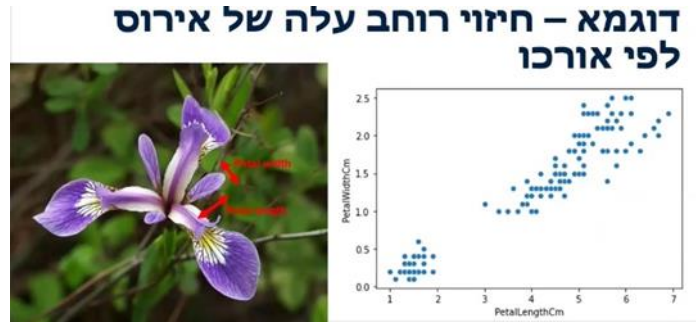
חיזוי באמצעות רגרסיה לינארית עשוי להתאים בהם הקשר בין המטרה והמאפיינים הוא קשר בעל אופי לינארי.

המשוואה החד ממדית  $\hat{y} = w * x + b$  מתארת את נוסחת החישוב של חיזוי (Predict) משתנה מטרה בודד  $\hat{y}$  (מסמן גודל מוערך) לפי מאפיין בודד  $x$  לאחר תהליך אימון (אותו נתאר בהמשך) בו חושבו המקדם  $w$  (משקל, weight) והקבוע  $b$  האופטימליים. המשוואה  $\hat{y} = w * x + b$  ניתנת כמובן לתיאור במערכת הצירים  $x, \hat{y}$  כקו ישר בעל שיפוע  $w$  החותך את ציר ה- $\hat{y}$  בקודה  $b$ .

את המשוואה החד ממדית ניתן להרחיב למשוואה רב ממדית  $\hat{y} = w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n + b$

בדוגמה שלהלן רואים שיש קשר בעל אופי לינארי עולה בין אורך העלה ורוחבו. הקורלציה היא מדד המתאר את מידת הלינאריות של הקשר הלינארי והיא יכולה לנוע בין 1 במצב של לינאריות עולה מושלמת (כל הנקודות מונחות על אותו ישר עולה) לבין -1 במצב של לינאריות יורדת מושלמת (כל הנקודות מונחות על אותו ישר יורד). הערכים בין 1 ל-1 הם

קורלציה חלקית בהן הנקודות מפוזרות סביב הישר המחושב. קורלציה 0 משמעותה חוסר קשר לינארי מוחלט (למשל, נקודות המונחות על מלבן).



ניתן לשער מהדוגמא לחיזוי רוח עלה של האירוס שבאזור, שהקורלציה היא בערך 0.7-0.8. וזה אומר שניתן לייצר תחזית של רוחב העלה בהינתן אורך העלה.

כאשר יש מספר משתנים, כמו בדוגמת מחירי הבתים ורוצים לבצע חיזוי לינארי, צריך שלפחות לחלק מהמשתנים שיהיה קשר בעל אופי לינארי עם המשתנה אותו רוצים לחזות.

### הדגמת חישוב קורלציה

פתחו את המחברת של ב-[Iris dataset](#) שמרו אותה בכונן שלכם Save Copy In drive (כי רוצים לבצע בה שינויים). השמירה יוצרת מחברת חדשה בשם Copy of Iris dataset עם [הקישור הזה](#). ניתן לשנות את השם (ניתן לשנות מ- Iris dataset ל-Copy of Iris dataset).

הפעילו פעולת `df.corr()` המחשבת ומציגה את הקורלציה בין המשתנים שבטבלה `df`. את הפעולה `corr`. יש להפעיל רק על עמודות עם ערכים מספריים ולכן יש לסלק את עמודת ה-`Species` וגם את עמודת ה-`Id` שלא מעניינת אותכם לצורך קורלציה:

```
df.drop(['Species', 'Id'], axis=1)
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows x 4 columns

axis=1 מכוון לעמודה, axis=0 מכוון לשורה. פעולת המחיקה הזו היא למעשה מחיקה זמנית לצורך תצוגה בלבד ואם מציגים שוב את df מקבלים:

```
[7] df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

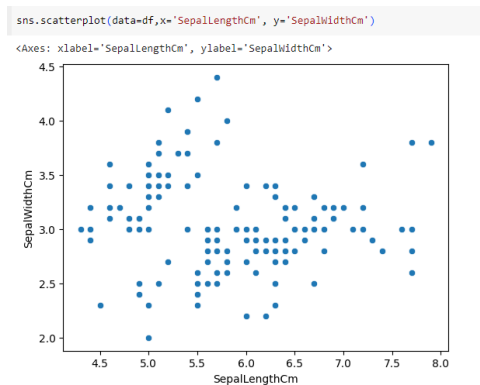
150 rows x 6 columns

אם רוצים למחוק לגמרי, יש להוסיף לפקודה inplace=True. מחקו לגמרי והפעילו את df.corr() מקבלים את מטריצת הקורלציה בין 4 המשתנים שבטבלה:

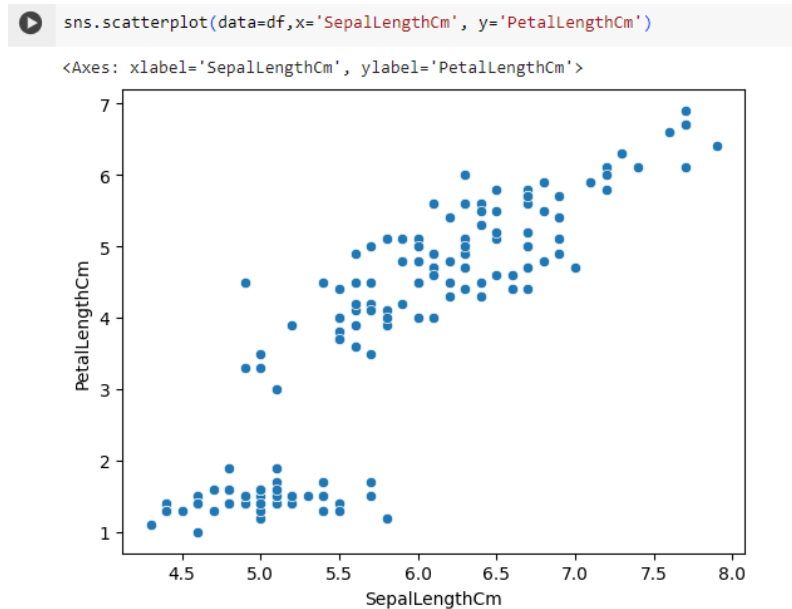
```
[11] df.corr()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

אלכסון המטריצה מתאר את הקורלציה של כל אחד מ-4 המשתנים עם עצמו וכולם כמובן 1. ניתן לראות שהקורלציה בין SepalLengthCm ובין SepalWidthCm היא לא טובה (-0.109369). ואכן, אם מציגים את גרף הפיזור שלהם מקבלים פיזור עם לינאריות נמוכה (הנקודות רחוקות מישר כלשהו) ומגמה שלילית (SepalWidthCm גבוה עבור ערכי SepalLengthCm נמוכים ולהיפך):

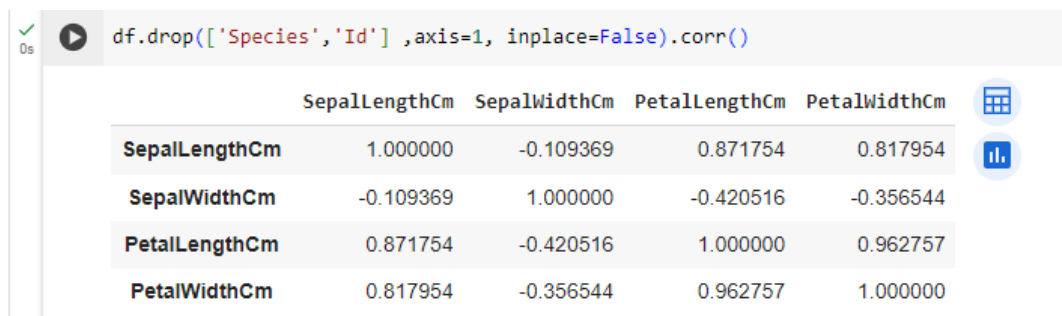


לעומת זאת ניתן לראות שהקורלציה בין **SepalLengthCm** ובין **PetalLengthCm** היא טובה (0.871754). ואכן, אם מציגים את גרף הפיזור שלהם מקבלים פיזור עם לינאריות די טובה (הנקודות די קרובות לישר העובר בתוך הפיזור) ומגמה חיובית (PetalLengthCm גבוה עובר ערכי SepalLengthCm גבוהים ולהיפך):



מטבלת הקורלציה ניתן ללמוד אילו משתנים קל יותר לחזות (למשל, PetalLengthCm) ואילו קשה יותר (למשל, SepalWidthCm).

אפשר לבצע חישוב קורלציה גם עם "מחיקה זמנית" של העמודות הלא רלוונטיות באמצעות הצמדת פעולת ה-corr. בשורת המחיקה הזמנית:





## חיזוי באמצעות רגרסיה ליניארית תוך שימוש ב- scikit-learn

scikit-learn היא ספרייה פופולרית של בשפת Python למימוש יישומים בלמידת מכונה הודות למחלקות רבות המשמשות לסיווג ורגרסיה והודות לקלות השימוש בהן.

יש לייבא את מחלקת LinearRegression מהספרייה sklearn.linear\_model:

```
from sklearn.linear_model import LinearRegression
```

מבצעים עכשיו משימה של חיזוי משתנה אחד (PetalWidthCm) מתוך משתנה אחד (PetalLengthCm):  
 הגדרת המשתנה ממנו אנו רוצים לחזות:

```
X=df[['PetalLengthCm']].to_numpy()
```

שם המשתנה (X) הוא אות גדולה היות ועבור המקרה הכללי של חיזוי ממספר משתנים זו תהיה מטריצה שמייצגת את טבלת המשתנים מהם רוצים לחזות ומטריצה נהוג לסמן באות גדולה. גם הסוגריים הכפולים ([]) הם הכנה למקרה הכללי של חיזוי ממספר משתנים.

הגדרת המשתנה אותו אנו רוצים לחזות:

```
y=df['PetalWidthCm'].to_numpy()
```

שם המשתנה (y) הוא אות קטנה היות וגם עבור המקרה הכללי של חיזוי במספר דוגמאות של פרחים זה יהיה ווקטור (ובמקרה הפרטי שלנו רק מספר בודד) ווקטור (או מספר) נהוג לסמן באות קטנה. כאן אין צורך להכין סוגריים כפולים היות והחיזוי במחלקת LinearRegression מתבצע תמיד על משתנה בודד.

```
X=df[['PetalLengthCm']]
X
```

0	1.4
1	1.4
2	1.3
3	1.5
4	1.4
...	...
145	5.2
146	5.0
147	5.2
148	5.4
149	5.1

Name: PetalLengthCm, Length: 150, dtype: float64

הפעולה to\_numpy():



ה-X הוא מערך של עמודה אחת ו-150 שורות וה-y הוא מערך ווקטורי נטול כיוון עם 150 איברים. עבודה עם מערכי numpy היא יעילה יותר (מ-Pandas Series ו-Pandas DataFrame) ובחלק מהמקרים הספריות דורשות עבודה עם מערכי numpy ולכן רצוי להתרגל תמיד להגדיר את המשתנים כמערכי numpy.

יש להקים אובייקט ממחלקת **LinearRegression**: `reg = LinearRegression()`  
 לאובייקטים יש תכונות מסוימות ופעולות מסוימות. אחת הפעולות של אובייקט ממחלקה זו היא `fit` שמחשבת את מקדמי הרגרסיה ( $w, b$ ) היעילים ביותר:

```
reg = LinearRegression()
reg.fit(X,y)

print ('w=',reg.coef_)
print ('b=',reg.intercept_)

w= [0.41641913]
b= -0.3665140452167275
```

כאשר הפרמטר הראשון (X) הוא המשתנה ממנו יש לחזות והפרמטר השני (y) הוא המשתנה אותו יש לחזות.

על מנת לראות את תוצאת החישוב, יש להדפיס את תכונת `reg.coef_` (מקדם) לקבלת  $w$  ואת תכונת `reg.intercept_` (חותך) לקבלת  $b$ . בהדפסה ניתן לראות ש- $w$  הוא ווקטור ו- $b$  הוא מספר. לאחר שמבצעים שלב האימון, ניתן לעבור לשלב החיזוי:

```
reg.predict([[3],[5],[7]])

array([0.88274335, 1.71558162, 2.54841988])
```

הפעולה `predict`. מבצעת את החיזוי. בדוגמא כאן מבצעים 3 תחזיות של `PetalWidthCm` לפי תכונה אחת של `PetalLengthCm` כפי שאימנתם אותו בפעולת ה-`fit`. פעולת החיזוי היא פשוט הכפלת כל אחד מהמספרים `[3],[5],[7]` ב- $w$  שחושב באימון והוספת ה- $b$  שחושב באימון.

בצעו את פעולת `score`. שנותנת הערכת ביצועים לתהליך הלמידה לפי מדד  $r^2$ . תחום התוצאות של הביצועים הוא  $0 \div 1$  (לא כמו מדד הקורלציה שהתחום שלו הוא  $-1 \div 1$ ) כאשר 0 הוא הביצוע הכי גרוע (תיאורטית, אם יש טעות באלגוריתם, התוצאה יכולה להיות גם שלילית, אבל לא נכנס לזה עכשיו) ו-1 הוא ביצוע מושלם:

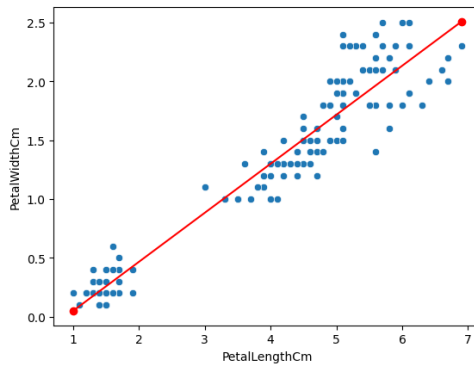
```
reg.score(X,y)

0.9269012279220037
```

ניתן לראות שביצועי החיזוי כאן הם די טובים.

בצעו המחשה גרפית של פיזור נקודות האמון והעברת הישר שיצר תהליך האימון:

```
[49] sns.scatterplot(data=df,x='PetalLengthCm',y='PetalWidthCm');
r = np.array([[X.min()],X.max()]])
plt.plot(r,reg.predict(r),color='red');
plt.plot(r,reg.predict(r),'ro');
```



השורה הראשונה משרטטת את הפיזור של שתי התכונות.

השורה השנייה מוצאת את שני ערכי הקצה של הציר האופקי (PetalLengthCm).

השורה השלישית (או הרביעית) מחשבת את הערך האנכי של שני ערכי הקצה (באמצעות הפעולה predict(r) ומשרטטת את הישר האדום בין שתי נקודות הקצה.

השורה הרביעית משרטטת את שתי נקודות הקצה האדומות.

## חיזוי הכולל מאפיינים מרובים

לאחר שהערכתם את התשתית של אימון וחיזוי על בסיס מאפייני בודד, תוכלו להרחיבו לאימון וחיזוי על בסיס מספר מאפיינים.

בצעו אימון על 'SepalLengthCm', 'PetalLengthCm', 'SepalWidthCm' במטרה לחזות את 'PetalWidthCm'.

```
x=df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']].to_numpy()
x
array([[5.1, 3.5, 1.4],
       [4.9, 3. , 1.4],
       [4.7, 3.2, 1.3],
       [4.6, 3.1, 1.5],
       [5. , 3.6, 1.4],
       ...,
       [5.4, 4. , 1.7]])

y=df['PetalWidthCm'].to_numpy()
y
array([[0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
       0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
       0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.1, 0.2, 0.2, 0.1, 0.2,
       0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5,
       1.5, 1.3, 1.5, 1.3, 1.6, 1. , 1.3, 1.4, 1. , 1.5, 1. , 1.4, 1.3,
       1.4, 1.5, 1. , 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7,
       1.5, 1. , 1.4, 1. , 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2,
       1.4, 1.2, 1. , 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8,
       2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2. , 1.9, 2.1, 2. , 2.4, 2.3, 1.8,
       2.2, 2.3, 1.5, 2.3, 2. , 2. , 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6,
       1.9, 2. , 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9,
       2.3, 2.5, 2.5, 1.9, 2. , 2.3, 1.8])
```

בצעו את שלב האימון:

```
reg = LinearRegression()
reg.fit(X,y)

print ('w=',reg.coef_)
print ('b=',reg.intercept_)

w= [-0.21027133  0.22877721  0.52608818]
b= -0.2487235860244532
```

בהדפסה ניתן לראות שהפעם  $w$  הוא ווקטור עם 3 רכיבים ו- $b$  הוא גם כן מספר.

עברו לשלב החיזוי עבור 3 דוגמאות:

```
reg.predict([[1,2,3],[3,4,5],[5,6,7]])

array([1.57682405, 2.66601218, 3.75520031])
```

בצעו הערכת ביצועים לתהליך הלמידה לפי מדד  $r^2$ :

```
reg.score(X,y)

0.9380481344518986
```

ניתן לראות שביצועי החיזוי כאן הם קצת יותר טובים מאשר בחיזוי הקודם (0.9269012279220037) לפי מאפיין בודד.

## תרגיל השלמת קוד Linear regression

בצעו את התרגיל [Linear regression ex1](#). יש להשלים את הקוד המתאים במקומות שרשום "?".

פתחו את הקישור למחברת ובצעו File/Save a copy in drive (על מנת לשמור אצלכם את המחברת עם השינויים שלכם).

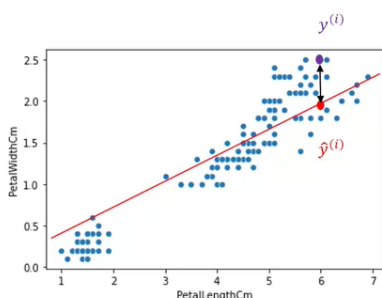
הצעה לפתרון התרגיל נמצא במחברת [Linear regression ex1-didi.ipynb](#).

הוראת `X.shape==(150, 1)` בודקת האם ה-`X` שנוצר קודם הוא בעל מבנה `(150,1)`. אם לא, המערכת תודיע על שגיאה.

ההוראה `X[:5]` היא הוראת slicing - לקחת את 5 האיברים הראשונים מהמערך `X`. הצורה הכללית של ההוראה היא `var(i1:i2)` כאשר `i1` הוא אינדקס האיבר הראשון שיש להציג ו-`i2` הוא אינדקס האיבר שעד אליו, **לא כולל** אותו, יש לקחת. כאשר `i1=0` (האיבר הראשון במערך) ניתן להשאיר אותו ריק (כמו בדוגמא כאן). כאשר רוצים לקחת עד סוף המערך, יש להשאיר את `i2` ריק.

## פונקציית המחיר ו- gradient descent

בשלב זה נלמד כיצד לנתח באופן אנליטי "מהו הישר הטוב ביותר" לחיזוי לינארי לפי פיזור של נתוני אימון ולמצוא את הפרמטרים של הישר הזה בשיטה הנקראת gradient descent.



בהינתן סט של  $m$  דוגמאות  $(x^{(i)}, y^{(i)})$   
 $x^{(i)}$  - ערכי דגימות של המשתנה הבלתי תלוי  
 $y^{(i)}$  - ערכי דגימות המשתנה התלוי בהתאמה

$$\hat{y} = w \cdot x + b$$

נגדיר את שגיאת החיזוי על הדוגמה ה- $(i)$ :

$$l^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 = (w \cdot x^{(i)} + b - y^{(i)})^2$$

שגיאת החיזוי מסומנת ב- $l$  (Loss).

הסיבות להגדרת שגיאת חיזוי כריבוע הפרש:

- התייחסות לערך המוחלט של השגיאה תוך התעלמות מכיוונה, וזאת על מנת למנוע מצב בו חלק מהשגיאות יבוטלו ע"י שגיאות אחרות בעלות סימן הפוך.
- מתן משקל רב יותר (לעומת, למשל הגדרת השגיאה כערך מוחלט של הפרש) לשגיאות גדולות.

פונקציית המחיר  $J$  (Cost) מוגדרת כממוצע של שגיאות החיזוי של כל  $m$  הדוגמאות באופן הבא:

ובהינתן מודל רגרסיה  $\hat{y} = w \cdot x + b$   
 נגדיר את שגיאת החיזוי על כל סט נתוני האימון:

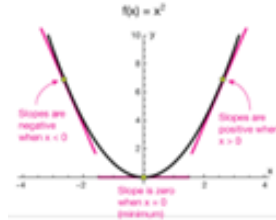
$$J = \frac{1}{m} \sum_{i=1}^m l^{(i)} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m (w \cdot x^{(i)} + b - y^{(i)})^2$$

$j$  כמובן לא יכול להיות מספר שלילי. המטרה היא למזער את  $J$ . הישר הכי טוב הוא זה שייתן את ערך ה- $j$  הכי נמוך והערך הכי נמוך שיכול להיות הוא 0.

מהגדרת פונקציית המחיר נובע שפונקציית המחיר  $J(w, b) = \frac{1}{m} \sum_{i=1}^m (w \cdot x^{(i)} + b - y^{(i)})^2$  היא פונקציה ריבועית של  $w$  ו- $b$ . באופן כללי, לכל פונקציה ריבועית, מציאת המינימום בדרך של איטרטיבית מתבצעת באופן הבא:

## הכללה - התקדמות בכיוון הנגזרת

בחרנו  $x$  התחלתי. אנחנו רוצים לקדם את  $x$  לכיוון נקודת המינימום.



כדאי לעדכן את  $x$  (להגדיל או להקטין) נגד כיוון הנגזרת

$$-\frac{df(x)}{dx}$$

## אלגוריתם gradient descent

- על מנת למצוא את נקודת המינימום של פונקציה  $J(w)$ :
- נגדיל נקודת התחלה  $w_0$
- נקדם את ערך  $w$  לפי הנוסחה:

$$w \leftarrow w - \alpha \frac{\partial J(w)}{\partial w}$$

•  $\alpha$  - קצב הלמידה

•  $-\frac{\partial J(w)}{\partial w}$  - נגזרת הפונקציה

יש להיזהר בבחירת קצב הלמידה  $\alpha$ . בחירת קצב נמוך מידי עלול לדרוש מספר גבוה מידי של איטרציות (=משך זמן חישוב גבוה מידי) וקצב גבוה מידי עלול לגרום לקפיצה ל"צד השני" של הפרבולה והתרחקות מנקודת המינימום. לאחר כל איטרציה של חישוב נגזרת לפי  $w$  וקידום  $w$ , יש לבצע איטרציה של חישוב נגזרת לפי  $b$  וקידום  $b$ .

בצעו את התרגיל [Linear regression ex2](#) (cost function & Pandas DF) שנמצא ב-קישורים יחידה/1 יחידה/1/classroom. יש להשלים את הקוד המתאים במקומות שרשום "?".

פתחו את הקישור למחברת ובצעו File/Save a copy in drive (על מנת לשמור אצלכם את המחברת עם השינויים שלכם). הצעה לפתרון התרגיל במחברת כאן: [Linear regression ex2-didi.ipynb](#).

ההוראה `df.head(n)` מציגה את  $n$  השורות הראשונות של טבלת המשתנים `df`. עבור  $n=0$  תוצג רק שורת הכותרות של הטבלה. כאשר לא מצוין  $n$  (`df.head()`), תוצגנה 5 השורות הראשונות של הטבלה.



הוספת ממד למערך קיים:

```
print(X.shape)
X = np.expand_dims(X,axis=1) #expand dimensions
X.shape

(150,)
(150, 1)
```

יצירת טבלה עם כותרות עמודות ושורת ממוצע ממערכי ווקטורים בעלי אורך זהה באמצעות pandas

```
[ ] pd.options.display.float_format = '{:,.5f}'.format
df1 = pd.DataFrame(data={'x': X, 'y':y, 'y_hat':y_hat, 'dy':dy, 'dy_sq':dy_sq})
df1.loc["Mean"] = df1.mean()
df1
```

	x	y	y_hat	dy	dy_sq
0	1.40000	0.20000	0.21647	-0.01647	0.00027
1	1.40000	0.20000	0.21647	-0.01647	0.00027
2	1.30000	0.20000	0.17483	0.02517	0.00063
3	1.50000	0.20000	0.25811	-0.05811	0.00338
4	1.40000	0.20000	0.21647	-0.01647	0.00027
...	...	...	...	...	...
146	5.00000	1.90000	1.71558	0.18442	0.03401
147	5.20000	2.00000	1.79887	0.20113	0.04046
148	5.40000	2.30000	1.88215	0.41785	0.17460
149	5.10000	1.80000	1.75722	0.04278	0.00183
Mean	3.75867	1.19867	1.19867	-0.00000	0.04229

151 rows x 5 columns

## רגרסיה ליניארית חד/רב ממדית

יש לטעון את המחברת [KC house data](#) ולבצע את התרגיל הבא:

- חשבו רגרסיה ליניארית חד ממדית של מחיר הדירה ביחס לשטחה
- חשבו רגרסיה ליניארית חד ממדית של מחיר הדירה ביחס למספר החדרים
- חשבו רגרסיה ליניארית דו ממדית של מחיר הדירה ביחס לשטח ולמספר החדרים
- חשבו רגרסיה ליניארית רב ממדית של מחיר הדירה ביחס לכל המשתנים פרט ל:

```
['id', 'date', 'zipcode']
```

פתחו את הקישור למחברת [KC house data](#) ובצעו File/Save a copy in drive (על מנת לשמור אצלכם את המחברת עם השינויים שלכם).

הצעה לפתרון התרגיל מוצג במחברת: [KC House data-didi Targil1.ipynb](#)

שימו לב ל-score (הערכת ביצועים של הרגרסיה לפי מדד  $r^2$ ) שהתקבל עבור כל אחת מהרגרסיות.

- בסעיף האחרון, בפעולת ה-df.drop לפני הגדרת ה-X בחיזוי המחיר ביחס לכל המשתנים פרט ל-, id, date, zipcode יש לזכור להסיר גם את עמודת ה-price.

## רגרסיה לוגיסטית

### הערכת ביצועים של מסווג

להלן קישור לסרטון הסבר בנושא של הערכת ביצועים של מסווג.

MNIST database – (Modified National Institute of Standards and Technology) בסיס נתונים של הספרות 0-9 בכתב יד. בסיס הנתונים מכיל 70,000 דוגמאות של ספרות מסווגות - 60,000 לאימון ו-10,000 לבדיקה.

בסיס הנתונים מכיל תמונות של הספרות בגודל 28x28 פיקסלים grey scale.

כל פיקסל הוא מספר בתחום 0-255 שמתאר את רמת ה"לבן" של הפיקסל.

על מנת לזהות את הספרה, מספקים למחשב מערך של 28x28 (או וקטור של 1x784) מספרים בתחום 0-255.

ברגרסיה הלינארית מנסים לחזות משתנה רציף (מחירים, לחץ דם, משקל...). בזיהוי ספרות מנסים לזהות class/מחלקה ולכן הרגרסיה הלינארית לא מתאימה לשימוש כאן ויש להשתמש בתהליך אחר, רגרסיה לוגיסטית, שמתאים לבעיות של סיווג למחלקות ולא של תחזית משתנה רציף.

לפני שלומדים על רגרסיה לוגיסטית, יש להבין וללמוד את מושגי סיווג והערכת ביצועים של מסווג מאומן.

תקציר מהסרטון:

הערכת ביצועים של מסווג מאומן – השוואת תוצאות החיזוי של דוגמאות שלא השתתפו בתהליך האימון עם תגיות האמת שבבסיס הנתונים.

מטריצת בלבול – תיאור ביצועי המסווג באמצעות מטריצה עם התגיות האמיתיות של כל הדוגמאות שנבדקו בציר אחד (מקובל, האנכי) והתגיות שחזה המסווג בציר השני (האופקי).

- על האלכסון הראשי של המטריצה נרשמים כמות הסיווגים הנכונים לכל תגית.
- מחוץ לאלכסון הראשי נרשמים כמויות הסיווגים הלא נכונים לכל הצירופים של תגית אמיתית/תגית חזויה.

Accuracy – מספר הדוגמאות שנחזו נכון (סכום האברים באלכסון הראשי) חלקי מספר הדוגמאות הכולל (סכום האברים בכל המטריצה).

ההגדרה הזו של Accuracy היא בעייתית - כאשר הבדיקה מתבצעת על סט לא מאוזן, כמו בדוגמה שבסרטון של "זיהוי אריות" (בסט של 1 אריה + 1000 חיות אחרות) שבסרטון, מקבלים דיוק גבוה של 95% עם חיזוי של 51 אריות כאשר רק אחד מהם נכון. זה גם מה שיקרה אם בודקים מחלה על מדגם של מעט מאוד אנשים חולים והרוב בריאים.

מקרה פרטי של מסווג בעל שתי אפשרויות (=תגיות) בלבד P (Positive) ו-N (Negative) נקרא גלאי. בדרך כלל, מגדירים את האפשרות שחשוב לגלות אותה (נשא קורונה, אור אדום ברמזור...) כ-Positive ואת האפשרות האחרת כ-Negative). מטריצת הבלבול של מסווג בעל שתי אפשרויות כוללת 4 תאים:

TP – TruePositive – חיזוי P כ-P.

TN – TrueNegative – חיזוי N כ-N.

FN – FalseNegative (זה למעשה Miss Detection) – חיזוי P כ-N.

FP – FalsePositive – חיזוי N כ-P.

Accuracy – מספר החיזויים הנכונים חלקי מספר החיזויים הכולל  $\frac{TP+TN}{TP+TN+FN+FP}$

Precision – מספר חיזוי החיובים כחיוביים חלקי מספר חיזוי החיוביים הכולל  $\frac{TP}{TP+FP}$

Recall – מספר חיזוי החיובים כחיוביים חלקי מספר החיוביים הכולל  $\frac{TP}{TP+FN}$

עבור הדוגמא של זיהוי אריות שבסרטון:

$$TP=1, TN=950, FN=0, FP=50$$

$$\frac{TP+TN}{TP+TN+FN+FP} = \frac{1+950}{1+950+0+50} = \frac{951}{1000} = 95.1\% \text{ – Accuracy}$$

$$\frac{TP}{TP+FP} = \frac{1}{1+50} = \frac{1}{51} \approx 2\% \text{ – Precision}$$

$$\frac{1}{1+0} = 100\% \text{ – Recall}$$

מהגדרות ודוגמת האריה אנו למדים שהשימוש ב-Precision/Recall עדיף על השימוש ב-Accuracy. לפי אופי האפליקציה ניתן לקבוע מהו המדד החשוב יותר Precision או Recall.

מדד F1 הוא מדד של ממוצע הרמוני של Precision ו-Recall:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

מדד F1 עבור זיהוי אריות שבסרטון:

$$F1 = \frac{2 * \frac{1}{51} * 1}{\frac{1}{51} + 1} = \frac{2/51}{52/51} = 2/52 \approx 3.8\%$$

מדידת ביצועים של מסוג – תרגיל

[הישור](#) לדף העבודה [ההישור](#) לפתרון.

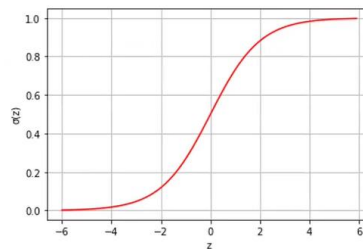
הגרסיה לוגיסטית – הגדרה

הגרסיה לוגיסטית היא אלגוריתם סיווג המחשב את ההסתברות שדוגמא לא ידועה שייכת לכל אחד מהסוגים. הרגרסיה הלוגיסטית מסווגת לשתי מחלקות בלבד (0/1, אריה/לא אריה...).

## הפונקציה הלוגיסטית - סיגמואיד

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

```
def sigmoid(z):
    return 1/(1+np.exp(-z))
```



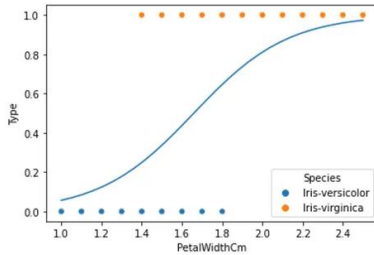
תכונות רלוונטיות לצורך מטרת הרגרסיה הלוגיסטית:

- קיימת עבור כל הערכים של Z.
- רציפה עם נזרות רציפות (לצורך אלגוריתם Gradient Descent)

- תחומה בין 0 ל-1 (לביטוי הסתברות).

הסיווג מתבצע על ידי התאמת קו רגרסיה לוגיסטית:

## אלגוריתם רגרסיה לוגיסטית



הסיווג מבוצע על ידי התאמת קו רגרסיה לוגיסטית (סיגמואיד)

$$\hat{y} = \sigma(wx + b)$$

נסווג את x לסוג 0 אם  $\hat{y} < 0.5$   
 ולסוג 1 אם  $\hat{y} \geq 0.5$

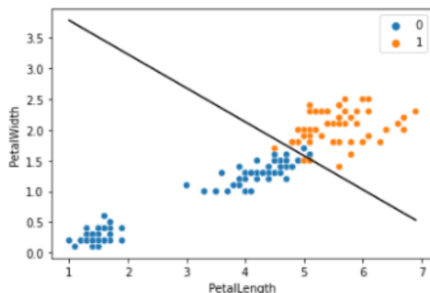
שימו לב :

x – הוא גודל רציף בדרך כלל (מידות עלי הפרח, שטח הדירה...) . לעיתים הוא בדיד (מספר החדרים בדירה, ציון במבחן).

w, b – גדלים רציפים. ולכן (גם עבור x בדיד), גם Z – גודל רציף. ולכן, גם  $\hat{y}$  הוא גודל רציף (בין 0 ל-1).

ה-predicted y הוא גודל בדיד (ברגרסיה לוגיסטית - של 2 מצבים בלבד). יש דרך לסווג גם ליותר מ-2 מחלקות ע"י ביצוע מספר רגרסיות לוגיסטיות אבל לא נכנס לזה במסמך זה.

בחיזוי עם מאפיין בודד (PetalWidthCm בשקף 4), התהליך ייתן נקודת הפרדה בה ה-z=0 (הסתברות 50%).



ריבוי מאפיינים:

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

נסווג את x לסוג 0 אם  $\hat{y} < 0.5$   
 ולסוג 1 אם  $\hat{y} \geq 0.5$

בחיזוי עם 2 מאפיינים (PetalLengthCm ו-PetalWidthCm בשקף 5), התהליך ייתן ישר הפרדה עליו ה-  
 $z=0$  (הסתברות 50%).

בחיזוי עם 3 משתנים בלתי תלויים, התהליך ייתן מישור הפרדה עליו ה- $z=0$  (הסתברות 50%).

הסבר כיצד נוצר קו ישר (בשקף 5) למרות שהסיגמואיד אינו פונקציה ליניארית:

$$PetalWidthCm - x_1$$

$$PetalLengthCm - x_2$$

$$z = w_1 * x_1 + w_2 * x_2 + b$$





המשוואה האחרונה היא משוואה של מישור במרחב  $(x_1; x_2; z)$  קו הפרדה הוא המקום בו  $z = 0$   
 (הסתברות 0.5) כלומר, ישר החיתוך של המישור הזה עם מערכת הצירים המישורית  $(x_1; x_2)$ :

$$w_1 * x_1 + w_2 * x_2 + b = 0$$

$$w_1 * x_1 = -w_2 * x_2 - b$$

$$x_1 = -\frac{w_2}{w_1} * x_2 - \frac{b}{w_1}$$

### סיווג על ידי רגרסיה לוגיסטית – תרגיל

Image	X <sub>1</sub> (red)	X <sub>2</sub> (blue)	$z = w_1 \cdot x_1 + w_2 \cdot x_2 + b$	$\hat{y} = \sigma(z)$	סוג יער=0 יער=1
	153	173			
	93	83			
	81	124			
	90	18			

נתונות תמונות בעלות שני מאפיינים:

- x1 ממוצע האדום בתמונה
- x2 ממוצע הכחול בתמונה

נתון מסווג מסוג רגרסיה לוגיסטית  
בעל הפרמטרים הבאים:

- w1=0.06
- w2=0.04
- b=-10

סווגו את התמונות הבאות:

[קישור](#) לקובץ הפתרון ב-excel:

w1=	0.06
w2=	0.04
b=	-10

x1(RED)	x2(BLUE)	$z=w_1 \cdot x_1 + w_2 \cdot x_2 + b$	$y^{\wedge} = \sigma(Z)$	סוג יער=0 יער=1
153	173	6.1	0.997762151	1
93	83	-1.1	0.249739894	0
81	124	-0.18	0.455121108	0
90	18	-3.88	0.020232997	0

תמונות 1 ו-4 זוהו נכון ברמת וודאות גבוהה (ה- $y^{\wedge}$  קרובים מאוד ל-1 או ל-0).

תמונה 2 זוהתה נכון כיער ברמת וודאות בינונית (0.25 קרוב יותר ל-0).

תמונה 3 זוהתה לא נכון כיער ברמת וודאות מאוד נמוכה (0.455 קרוב מאוד ל-0.5).

[קישור](#) לקובץ הפתרון בפייטון.

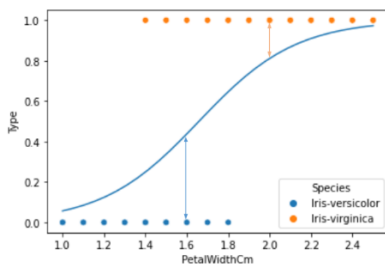


## תהליך האימון ברגרסיה לוגיסטית

תהליך האימון ברגרסיה לוגיסטית דומה לזה ברגרסיה לינארית:

- רושמים פונקציית מחיר.
- גוזרים את פונקציית המחיר.
- מוצאים את נקודת המינימום בשיטת gradient descent.

## פונקציית המחיר - binary cross entropy



$$l^{(i)} = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

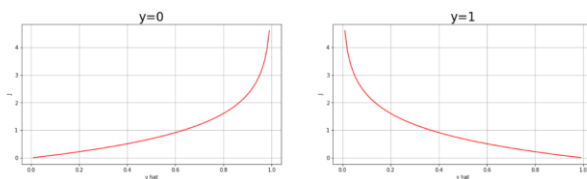
$$J = \frac{1}{m} \sum_{i=1}^m l^{(i)}$$

פונקציית המחיר הזו נבחרה כי משיקולים סטטיסטיים של most likelihood (מחוץ לתחום הלמידה שלנו כאן) היא עובדת הכי טוב עבור בעיות החלטה בין שני class.

### האם המופעים של $(i)$ בפונקציית המחיר הזו מציינים אינדקס או חזקה?

שימו לב! – ה-log בנוסחאות כאן הוא לפי בסיס הלוגריתם הטבעי (ולא בבסיס 10).

### יזואליזציה של פונקציית המחיר



ה- $y^{(i)}$  יכול לקבל רק את הערכים 0 או 1 ולכן:

$$l^{(i)} = -\log(1 - \hat{y}^{(i)}) : y^{(i)} = 0 \text{ כאשר}$$

$$l^{(i)} = -\log \hat{y}^{(i)} : y^{(i)} = 1 \text{ כאשר}$$

גזירת פונקציית המחיר ומציאת נקודת המינימום מתבצעת בשיטת gradient descent בצורה דומה לרגרסיה לינארית כאשר ההבדל הוא בנגזרות עצמן עקב הסיגמואיד ב-  

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$
**וגם ה- $l^{(i)}$**   
 **$l^{(i)} = -\log(\hat{y}^{(i)})$  או  $l^{(i)} = -\log(1 - \hat{y}^{(i)})$  ?**  
 גם כאם חשוב לבחור מקדם קצב למידה ( $\alpha$ ) עם ערך לא גבוה מידי על מנת למנוע "התבדרות לצד השני".

## אלגוריתם gradient descent

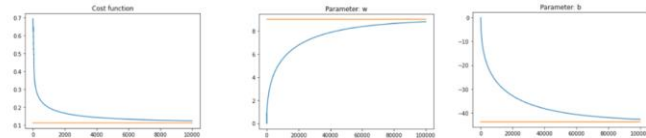
הגדרנו את פונקציית המחיר  
 נאתחל:  
 $w \leftarrow 0, b \leftarrow 0$   
 נתקדם לכיוון המינימום:  

$$w \leftarrow w - \alpha \frac{\partial J(w, b)}{\partial w}$$

$$b \leftarrow b - \alpha \frac{\partial J(w, b)}{\partial b}$$

## התכנסות בתהליך האימון

בתהליך האימון ניתן לראות התכנסות של w-ה ו-b-ה ושאיפה למינימום של פונקציית המחיר.



## רגרסיה לוגיסטית ב-sklearn

- יש לייבא ספריית LogisticRegression.
- יש לבחור את העמודות הרלוונטיות בטבלת הנתונים עבור המאפיינים עליהם אנו רוצים לבצע קלסיפיקציה – למשל, 4 עמודות האורך/רוחב של העלים.
- בבחירת התנאי הלוגי יש לבחור את עמודת הסיווג בטבלת הנתונים ולבצע תנאי השוואה בוליאנית של ערכה לסוג שמוגדר כ"1" לוגי. התוצאה y יכולה להיות 0 או 1 (לכל אחת מהדוגמאות/שורות שבטבלת הנתונים).

## רגרסיה לוגיסטית ב-sklearn

```
# ייבוא ספרייה
from sklearn.linear_model import LogisticRegression
# בחירת מאפיינים
X=df[['...']].to_numpy()
# בחירת תנאי לוגי
y=1*(df[['...']]==...).to_numpy()
# הגדרת מסווג
clf = LogisticRegression()
# אימון
clf.fit(X, y)
```

- מגדירים מסווג ומבצעים פעול fit שמחשבת את וקטור ה-w והמספר b.

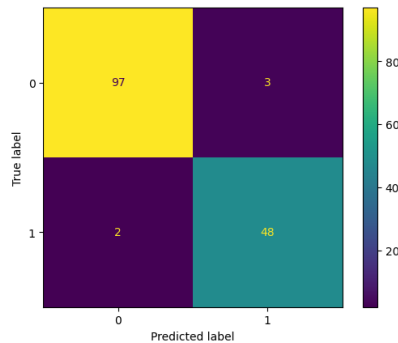
**קישור** לדוגמה לרגרסיה לוגיסטית ב-sklearn עם בסיס הנתונים של הפרחים לחיזוי האם סוג הפרח הוא "Iris-virginica" לפי המאפיינים 'PetalLengthCm', 'PetalWidthCm'.

- הפקודה `df['Species'].unique()` מציגה את הסוגים הקיימים.
- הפקודה `df['Species'].nunique()` מציגה מספר הסוגים הקיימים.
- הפקודה `df['Species'].value_counts()` מציגה כמה יש מכל סוג.
- שימו לב! ברגרסיה לוגיסטית יש להוסיף 0 בתוך הסוגים של חילוף המקדמים  $w_{sk}$
- `reg.coef_[0]`. הסיבה לכך היא שברגרסיה לוגיסטית ניתן לבצע מספר סיווגים ברגרסיה אחת שמחזירה מערך של מקדמים עבור כל מאפיין. אנחנו לומדים בשלב זה סיווג בינרי בלבד.

```
reg.coef_  
array([[2.77759875, 2.38552765]])
```

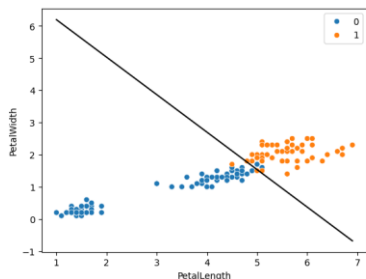
באת `reg.coef_` ניתן לראות (לפי ה-`reg.coef_` שהיא) הוא מערך של מערכים.

- ה-score נותן את ה-Accuracy.
- ההוראה `ConfusionMatrixDisplay.from_estimator(reg, X, y)` מציגה את מטריצת הבלבול של



המסווג reg המאפיינים X והסוג y.

- ניתן לראות 3 אירועי FN ו-2 אירועי FP.



- הצגה ויזואלית של פיזור המאפיינים וישר הסיווג.  
○ `mg` ו-`bg` הם השיפוע והחיתוך של ישר הסיווג.

- ניתן להבחין בשני אירועי FN כנקודות כחולות (0) באזור הכתום (1) ו-2-4 אירועי FP כנקודות כתומות (1) באזור הכחול (0).

## רגרסיה לוגיסטית ב-sklearn – תרגיל

- טענו את המחברת של [Penguin dataset](#).
- נסו לחזות 'df['species']' על פי 'bill\_length\_mm', 'bill\_depth\_mm'.
- חשבו את מדד accuracy.
- חשבו את מטריצת הבלבול.
- שרטטו את קו ההפרדה.

הצעה לפתרון מוצג במחברת: [Penguin dataset Targil 2.ipynb](#).

```
df=pd.read_csv('https://raw.githubusercontent.com/ariel38/datasets/main/penguins.csv')
df.head()

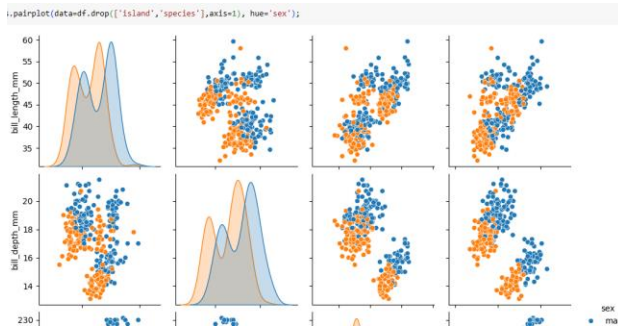
species  island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen         39.1             18.7             181.0         3750.0  male
1  Adelie  Torgersen         39.5             17.4             186.0         3800.0  female
2  Adelie  Torgersen         40.3             18.0             195.0         3250.0  female
3  Adelie  Torgersen         NaN             NaN             NaN             NaN     NaN
4  Adelie  Torgersen         36.7             19.3             193.0         3450.0  female

[ ] df.dropna(inplace=True) #Delete missing data
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 333 entries, 0 to 343
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  --
 0   species         333 non-null    object
 1   island          333 non-null    object
 2   bill_length_mm  333 non-null    float64
 3   bill_depth_mm  333 non-null    float64
 4   flipper_length_mm  333 non-null  float64
 5   body_mass_g    333 non-null    float64
 6   sex             333 non-null    object
dtypes: float64(4), object(3)
memory usage: 28.4+ KB
```

פקודת `df.dropna()` מסירה את כל השורות בהם חסרים נתונים (NaN).

בבסיס הנתונים הזה יש 4 עמודות של מאפיינים (2 אורכים, עומק, מסה) ו-3 עמודות של תגיות (אי(3), מין(2)).



הפקודה

`sns.pairplot(data=df.drop(['island', 'species'], axis=1), hue='sex');`

מסירה מבסיס הנתונים את עמודות תגיות "אי"

ו"סוג" ומציגה את מטריצת גרפי הפיזור של 16

צירופי זוגות המאפיינים (4 מאפיינים=16 תאים).

באלכסון הראשי (גרף הפיזור של מאפיין עם עצמו) מוצגת

היסטוגרמה של כמות המאפיין בחלוקה למין

(זכר=כחול, נקבה=כתום).

```
X=df[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']].to_numpy()
y= df['species'].to_numpy()
print ("X.shape=",X.shape)
print ("y.shape=",y.shape)

X.shape= (333, 4)
y.shape= (333,)
```

```
x
array([[ 39.1,  18.7, 181. , 3750. ],
       [ 39.5,  17.4, 186. , 3800. ],
       [ 40.3,  18. , 195. , 3250. ],
       ...,
       [ 49.6,  18.2, 193. , 3775. ],
       [ 50.8,  19. , 210. , 4100. ],
       [ 50.2,  18.7, 198. , 3775. ]])
```

הפקודות:  
 scaler = MinMaxScaler()  
 X=scaler.fit\_transform(X)  
 מנרמלות את תחומי ערכי המאפיינים לתחום 0-1.  
 ראו את ערכי X לפני ואחרי הנרמול.

### Using MinMaxScaler

```
[23] print('Max: ', X.max())
print('Min: ', X.min())
scaler = MinMaxScaler()
X=scaler.fit_transform(X)
print('Max: ', X.max())
print('Min: ', X.min())

Max: 6300.0
Min: 13.1
Max: 1.0
Min: 0.0
```

```
[24] X
array([[0.25454545, 0.66666667, 0.15254237, 0.29166667],
       [0.26909091, 0.51190476, 0.23728814, 0.30555556],
       [0.29818182, 0.58333333, 0.38983051, 0.15277778],
       ...,
       [0.63636364, 0.60714286, 0.3559322 , 0.29861111],
       [0.68 , 0.70238095, 0.6440678 , 0.38888889],
       [0.65818182, 0.66666667, 0.44067797, 0.29861111]])
```

### Logistic regression with SKLearn

הגדרת המאפיינים (X) והתווית (y) לאימון.  
 ביצוע רגרסיה לוגיסטית.  
 הדפסת המקדמים (w\_sk) והחותך (b\_sk).

```
[109] X=df[['bill_length_mm', 'bill_depth_mm']].to_numpy()
y=1*(df['species']=='Adelie').to_numpy()
print(X.shape,y.shape)

(333, 2) (333,)
```

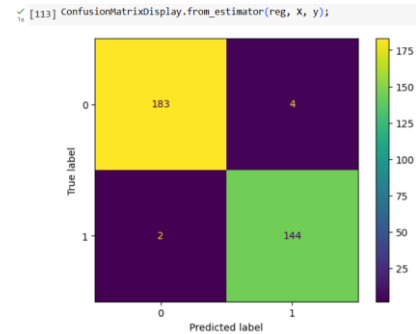
```
reg = LogisticRegression()
reg.fit(X,y)
w_sk = reg.coef_[0]
b_sk = reg.intercept_[0]
print(w_sk,b_sk)

[-1.37826428  1.98506245] 24.297550666550585
```

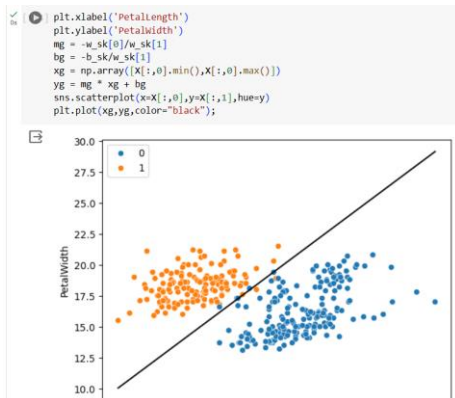
### חישוב מדד Accuracy

```
[112] reg.score(X,y) #compute accuracy
0.9819819819819819
```

### מטריצת הבלבול



### קו ההפרדה על גרף הפיזור



**בקישה** מודגמת הבעיה של הפרדה של המין Iris-versicolor לפי המאפיינים `PetalLengthCm,PetalWidthCm` שמפוזר בתוך שני המינים האחרים ולכן קשה/לא ניתן לחזות אותו באמצעות ישר חיתוך בודד. מודגם גם הפתרון באמצעות שתי גרסיות שיוצרות שני ישרים המפרידים בין שלשת המינים.

הדוגמה הזו מהווה מוטיבציה לפתרון בעיות הפרדה באמצעות למידה עמוקה / רשת נוירונים שהיא למעשה ריבוי מובנה של גרסיות לוגיסטיות שיוצר גבולות שמחלקות את מרחב הפיזור של המאפיינים לתת-מרחבים נפרדים עבור כל תגית שנדרש לחזות כאשר מרחב הפיזור ותתי המרחבים הנפרדים הם בעלי ממד זהה. וזאת, לעומת הרגרסיה הלוגיסטית הבודדת שיוצרת את הפרדה לשני תתי מרחב בלבד.

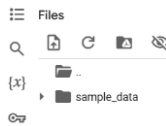
## למידה עמוקה - רשת נוירונים

### טעינת dataset לגוגל Colab

להלן [קישור](#) לרשימה של 3 סרטונים על דרכים שונות לטעינת dataset לגוגל Colab. ניתן להשתמש בידע הזה במסגרת ביצוע פרויקט.

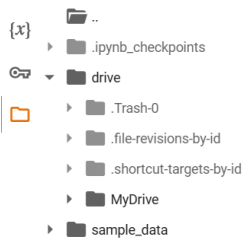
- מומלץ לבצע את החיפוש וההעלאה לפי ההנחיות ב-"from google drive..." (הקישור הראשון).  
 הסרטון "from github..." הוא לבעלי github. הסרטון "directly from Kaggle..." הוא למתקדמים.
- הורידו ה-dataset ממאגר בסיסי הנתונים (למשל, Kaggle) לכוון המקומי, העבירו ל-google-drive וטענו ל-Colab.
- הרשמו וכנסו ל-Kaggle עם חשבון גוגל: [xxx@gmail.com](mailto:xxx@gmail.com).
- חפשו diamonds, בדקו שזה אכן dataset ע"י בחירת לשונית Datasets והורידו למחשב האישי. לחיצה כפולה על קובץ ה-zip אמנם מראה את קובץ ה-diamonds.csv אבל העתקתו ל-google-drive מסתיימת בתוצאה של קובץ ריק ולכן יש לבצע extract to archive ולהעתיק את diamonds.csv אל ה-Drive.
- העלו ה-dataset למחברת – פותחים מחברת ונותנים לה שם/קישור [MyDiamonds](#).

לוחצים בסרגל הכלים משמאל על צלמית הקבצים, מקבלים את מבנה התיקיות של



שרת הלינוקס עליו רץ ה-colab, בחרים בצלמית העלאת ה-drive וזה

יוצר במחברת את ההוראה `from google.colab import drive` `drive.mount('/content/drive')` שהרצתה עם מתן כל ההרשאות



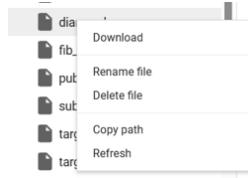
מודיעה Mounted at /content/drive ומוסיפה למבנה הקבצים את ה-drive

- הקובץ diamonds.csv נמצא במחיצת MyDrive. על מנת לייבא אותו יש להוסיף את הקוד הבא:

```
import pandas as pd
```

df=pd.read\_csv("/content/drive/MyDrive/diamonds.csv")

את הלינק ניתן לקבל ב-copy path שבתפריט של לחיצה ימנית על diamonds.csv:



הרצת ה-df:

```
import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/diamonds.csv")
df
```

Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z	
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows x 11 columns

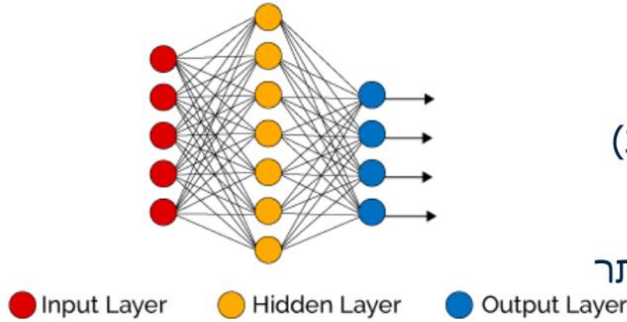
## למידה עמוקה

בסוף הפרק הקודם ראינו שסיווג (החלטה בינרית) באמצעות רגרסיה לוגיסטית בודדת (=נוירון בודד) היא בעייתית במקרים המסובכים בהם ערכי המאפיינים של ה-classes השונים מפוזרים זה בתוך זה ולא נותנת תוצאות טובות. וזו המוטיבציה לפתור את המקרים המסובכים הללו באמצעות רשת נוירונים.

רשת נוירונים יכולה להפיק קו/משטח/מרחב הפרדה מורכבים (בעלי פונקציה אוניברסלית כלשהי) לעומת נוירון (מסווג) בודד שמפריד בין הסיווגים רק באמצעות ישר/מישור/מרחב לינאריים. למעשה, רשת נוירונים מספיק עמוקה יכולה להגיע לאחוז הצלחה של קרוב מאוד ל-100% על סט נתוני אימון, אבל זה ממש לא מבטיח הצלחה של דומה גם על נתון אמת!!! ויותר מזה, הפתרון עלול להיות פחות טוב מפתרון אופטימלי יותר שנותן אחוז הצלחה נמוך יותר על סט הנתונים אבל ייתן אחוז הצלחה גבוה יותר על נתוני אמת – הסיבה לכך היא שהרשת למעשה "תפרה" פתרון ספציפי שמתאים לסט הנתונים שקיבלה בלי להשאיר "אזורי משחק" לנתונים שמתנהגים בצורה קצת שונה.



### Simple Neural Network



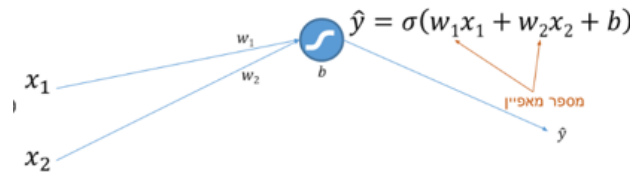
- רשת היא אוסף של שכבות, כל שכבה מורכבת מאוסף של נוירונים
- לרשת שכבת כניסה, שכבת יציאה ושכבות פנימיות (חבויות)
- הנתונים עוברים ומעובדים משכבה לשכבה, כל שכבה מבצעת עיבוד ברמה גבוהה יותר

שכבת קלט (Input Layer) – מאפיינים. לדוגמא: אורך ורוחב של עלים ב-dataset של הפרחים. מסת גוף, אורך מקור, מין בפינגווינים.

שכבת פלט (Output Layer) – נתחיל את הלימוד ברשתות עם פלט מסווג בינרי (=נוירון בודד בשכבת הפלט).

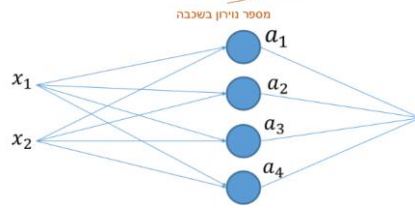
שכבות ביניים (Hidden Layer) – שכבה אחת ברשת רדודה ויותר משכבה אחת ברשת עמוקה.

### נוירון



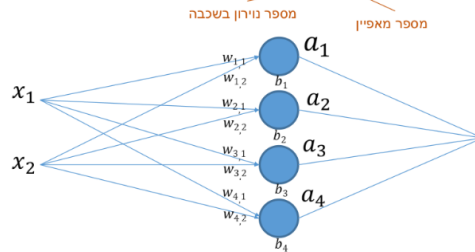
כל נוירון בשכבות הביניים הוא כמו רגרסיה לוגיסטית – יש לו קלטים, משקולות, ופונקציית אקטיבציה. פונקציית האקטיבציה היחידה שלמדנו עד כה היא סיגמואיד, אבל יש סוגים נוספים שנלמד בהמשך.

## נירון בתוך שכבה $\hat{y} \rightarrow a_i$



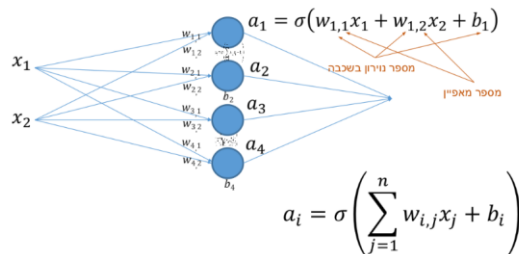
שיטת המספור של רשת ניורונים: את המוצא ( $\hat{y}$ ) של הנירון נסמן ב- $a_i$  ואת מקומו בשכבה נסמן ב- $i$  ולכן המוצא של הנירון ה- $i$  יסומן כ- $a_i$ . והחותך של הנירון הזה יסומן כ- $b_i$ .

## נירון בתוך שכבה $w_i \rightarrow w_{i,j}$

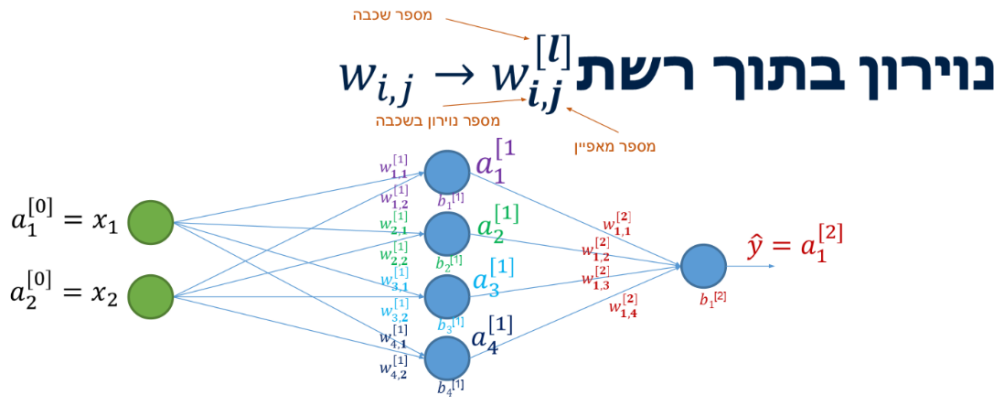


את מספרו של המאפיין בכניסה לנירון נסמן ב- $j$  ולכן המקדם של מאפיין זה בנירון במקום ה- $i$  יסומן כ- $w_{i,j}$ .

## נירון בתוך שכבה – חישוב הפלט

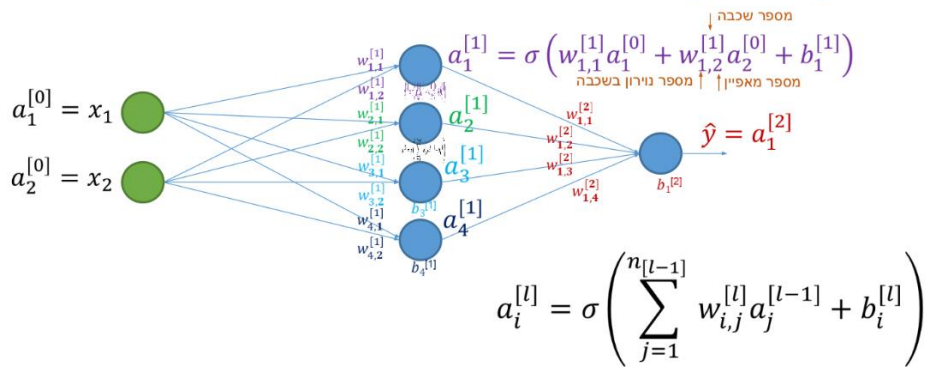


לפי שיטת הסימון שתוארה לעיל, הנוסחה לחישוב הפלט של נירון בתוך שכבה היא  $a_i = \sigma(\sum_{j=1}^n w_{i,j} * x_j + b_i)$  כאשר  $n$  היא כמות המאפיינים בכניסה לנירון שזה למעשה מספר הניורונים בשכבה הקודמת.



על מנת להתייחס לנוירון בתוך רשת בעלת יותר משכבת ביניים אחת, מוסיפים בסוגריים מרובעים מעל לסימונים של הנוירון את מספר השכבה  $[l]$ , כאשר שכבת הקלט מסומנת כ- $l = 0$ .

## נוירון בתוך רשת – חישוב הפלט

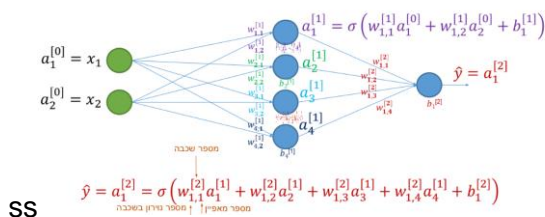


ולכן, הנוסחה לחישוב הפלט של נוירון בתוך שכבה  $l$  היא:

$$a_i^{[l]} = \sigma\left(\sum_{j=1}^{n^{[l-1]}} w_{i,j}^{[l]} * a_j^{[l-1]} + b_i^{[l]}\right)$$

כאשר  $n$  היא כמות הנוירונים בשכבה. הביטוי  $n_{[l-1]}$  שרושמים כאן הוא למעשה כמות המאפיינים בכניסה לנוירון בשכבה  $l$ . שימו לב, הביטוי הזה נכון לשכבה 1 ומעלה ולא נכון לשכבה 0 שאין לה כניסות ולכן אין לה משקלות.

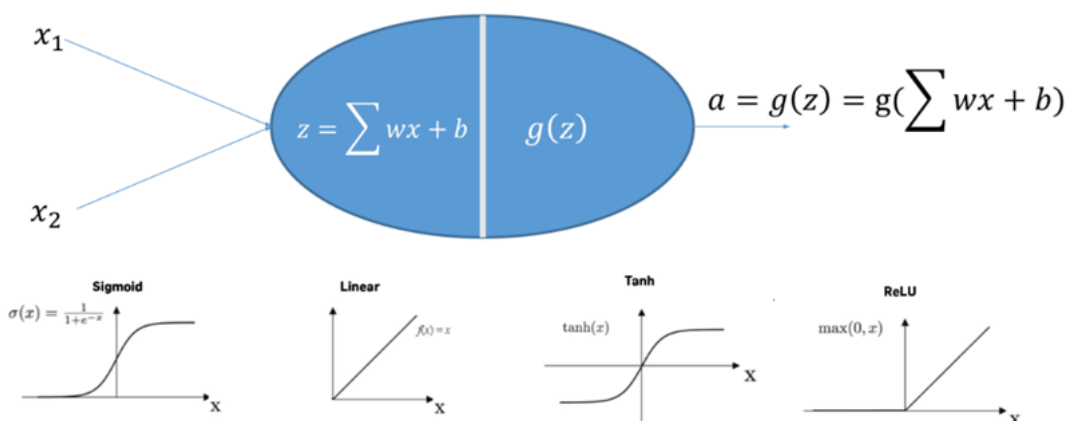
### חישוב מוצא הרשת



תרגיל חישוב הפלט  $a_1^{[2]}$ :

$$a_1^{[2]} = \sigma\left(\sum_{j=1}^{n_1^{[1]}} w_{1,j}^{[2]} * a_j^{[1]} + b_1^{[2]}\right) = \sigma\left(\sum_{j=1}^4 w_{1,j}^{[2]} * a_j^{[1]} + b_1^{[2]}\right) = \sigma(w_{1,1}^{[2]} * a_1^{[1]} + w_{1,2}^{[2]} * a_2^{[1]} + w_{1,3}^{[2]} * a_3^{[1]} + w_{1,4}^{[2]} * a_4^{[1]} + b_1^{[2]})$$

## הפרדת תפקידים בניירון – חלק לינארי ואקטיבציה



עד כה למדנו לבצע את האקטיבציה באמצעות פונקציית סיגמואיד שהיא מאוד שימושית בשכבת המוצא ובמיוחד בבעיות של סיווג לשני classes. אבל, קיימות פונקציות אקטיבציה נוספות רבות ובשקף ניתן לראות 4 דוגמאות מהן.

הפונקציה הכי שימושית היא Relu (Rectified Linear Unit).

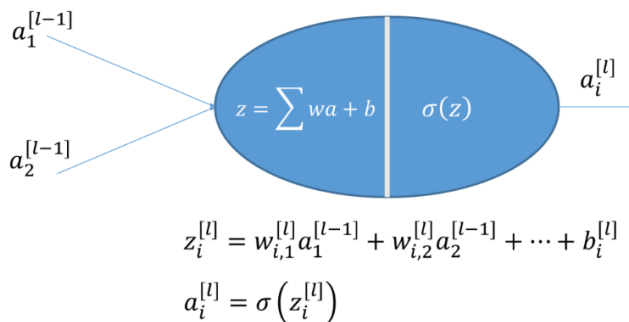
- מאוד קלה לחישוב  $(\max(0,x))$ .
- הנגזרת מאוד פשוטה – 0 עבור x שלילי ו-1 עבור x חיובי.

פונקציית linear היא למעשה "אין אקטיבציה". שימושית בבעיות רגרסיה בהן נדרש לחזות במוצא כל ערך שהוא (ולא לסווג) שיכול להיות גם חיובי וגם שלילי.

הדרישות מפונקציית אקטיבציה:

- חייבת להיות לא לינארית כי אם היא לינארית ניתן להוכיח שהיא מיותרת וניתן לאחדה עם שכבה סמוכה שגם היא לינארית.
- גזירה עבור כל x. הדרישה הזו לכאורה לא מתקיימת ב-Relu אבל ניתן להחליט שרירותית שהנגזרת עבור x=0 היא 0. תשובה נוספת היא שהסיכוי ש-x יהיה בדיוק 0 בתהליך למידת מכונה הוא אפסי עקב פעולות ה-float המתבצעות בתהליך.

## חלק לינארי ואקטיבציה בנוירון – נוסחה כללית



בתמונה רשום סיגמואיד באקטיבציה, אבל, כאמור לעיל, יכולה להיות גם פונקציית אקטיבציה אחרת.

### Sklearn MLPClassifier

MLPClassifier (Multi-Layer Perceptron) מבוסס על רגרסיה לינארית ורגרסיה לוגיסטית של Sklearn. בתרגיל שתבצעו בהמשך על סט הנתונים של הפינגווינים נדרש לחזות גם משתנה בדיד Dual Class (sex) וגם משתנה בדיד שהוא Multi Class (island ו-species).

הורידו ל-drive את מחברת ה-Dual Class ותנו לה את השם/קישור <https://neural-network-sklearn-dual-class-iris-didi.ipynb>.

במקטע ייבוא הספריות נוספה השורה:

`from sklearn.neural_network import MLPClassifier`

- כפי שראיתם בסוף הפרק הקודם, לא ניתן היה לחזות את "האם versicolor" באמצעות רגרסיה לוגיסטית בודדת. כשמבצעים זאת בשתי רגרסיות הייתם צריכים "להנדס" את התהליך ולשאול איזה חיתוכים נדרשים לבצע? בעבודה עם רשת הנוירונים החיזוי מבוצע בהצלחה באופן אוטומטי.
- מקטע קביעת ה-Hyper Parameters עבור רשת רדודה (רשת עם שכבת hidden אחת):
  - `hidden_layer_size = 15 #must be large enough`

- learning\_rate = 0.1
- epochs = 20000

ה-sklearn רגיש למספר נירונים קטן ולכן עלול שלא לעבוד עבור hidden\_layer\_size נמוך מידי.

במידה ורוצים לעבוד עם רשת עמוקה (יותר משכבת ביניים אחת) יש להכניס בהשמה hidden\_layer\_size רשימה של מספר הנירונים בכל אחת משכבות הביניים. למשל עבור, שתי שכבות ביניים עם 15 נירונים בראשונה ו-4 בשנייה יש לרשום hidden\_layer\_size=[15,4] כפי שתראו בהמשך המחרת.

ה-epochs הוא כמות המעברים שרוצים לבצע על ה-Data Set.

- מספר השכבות ברשת (כולל שכבות הכניסה והיציאה):

```
[7] mlp.n_layers_
3
```

- מטריצות ה-W:

```
[8] W1 = mlp.coefs_[0]
W2 = mlp.coefs_[1]
print(W1.shape, W2.shape)
(2, 15) (15, 1)
```

W1 היא מטריצת המקדמים בין שכבת הכניסה לשכבת הביניים ו-W2 היא מטריצת המקדמים בין שכבת הביניים לשכבת היציאה.

ה-(2,15) הם הממדים של W1 שהיא מטריצה של 2 שורות עם 15 עמודות והיא מכילה את המקדמים בין 2 "הנירונים" (=המאפיינים 'PetalLengthCm','PetalWidthCm') שבשכבת הכניסה ו-15 נירונים שבשכבת הביניים.

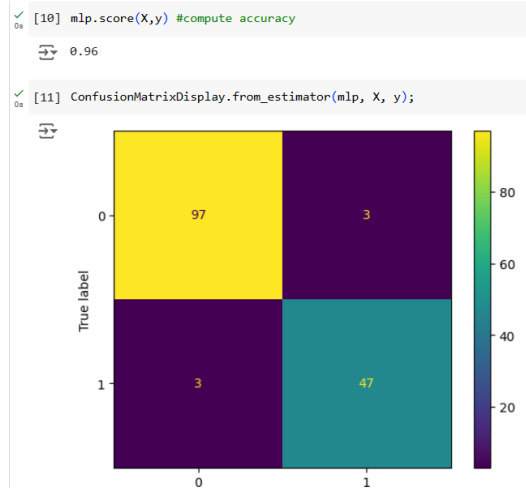
ה-(15,1) הם הממדים של W2 שהיא מטריצה של 15 שורות עם עמודה אחת והיא מכילה את המקדמים בין 15 הנירונים שבשכבת הביניים והנירון הבודד ("האם versicolor") שבשכבת היציאה.

- וקטורי ה-b:

```
[9] b1 = mlp.intercepts_[0]
b2 = mlp.intercepts_[1]
print(b1.shape, b2.shape)
(15,) (1,)
```

b1 הוא ווקטור החותכים של 15 הנורונים שבשכבת הביניים ו-b2 הוא "ווקטור" (בעל אורך 1 = מספר) החותך של הנורון הבודד שבשכבת היציאה.

- חישוב accuracy ומטריצת הבלבול:



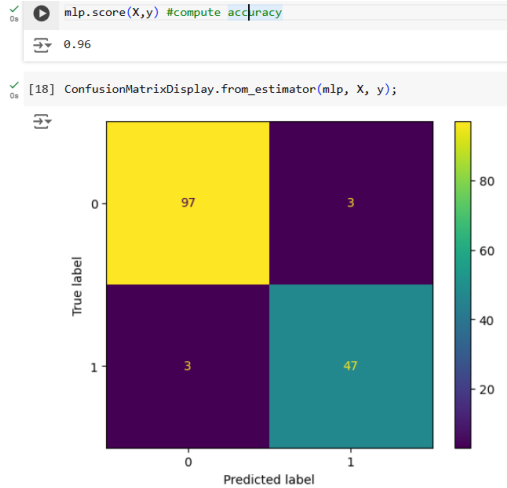
החישוב מתבצע ע"י הפעלת המשקלים על סט נתוני האימון והשוואה לתגיות הנתונות.

- מקטע קביעת ה-Hyper Parameters עבור רשת עמוקה (רשת עם 2 שכבות hidden):
  - hidden\_layer\_size = [15,20] #must be large enough
  - learning\_rate = 0.1
  - epochs = 20000
- זהו נותן רשת של 4 שכבות (כולל שכבות הכניסה והיציאה) עם 3 מטריצות מקדמים הקושרים בין 4 השכבות ו-3 ווקטורים של החותכים של הנורונים בשכבות הביניים ושכבת היציאה:

```
[15] W1 = mlp.coefs_[0]
      W2 = mlp.coefs_[1]
      W3 = mlp.coefs_[2]
      print(W1.shape, W2.shape, W3.shape)
      (2, 15) (15, 20) (20, 1)

[16] b1 = mlp.intercepts_[0]
      b2 = mlp.intercepts_[1]
      b3 = mlp.intercepts_[2]
      print(b1.shape, b2.shape, b3.shape)
      (15,) (20,) (1,)
```

- ניתן לראות שבמקרה הזה העמקת הרשת לא שיפרה את הדיוק ולא שינתה את מטריצת הבלבול:



הורידו ל-drive את מחברת ה-Multi Class ותנו לה את השם/קישור [Neural network sklearn multi class Iris didi.ipynb](#). המחברת הזו דומה מאוד לזו שלמדתם ב-Dual-Class עם השינויים הבאים:

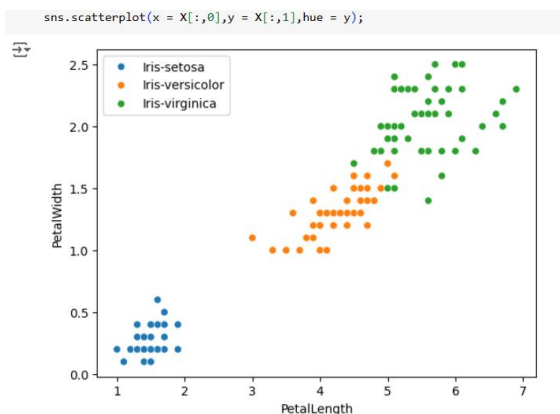
- במקום להגדיר את ה-y כסיווג בינרי של "האם 'versicolor'":

```
y=1*(df['Species']=='Iris-versicolor').to_numpy()
```

הגדרתם את ה-y כטקסט של סיווג של אחד מתוך כל (3) הסיווגים האפשריים:

```
y=df['Species'].to_numpy()
```

- ולכן, גרף הפיזור של המאפיינים נצבע עכשיו ב-3 צבעים (לעומת 2 צבעים ב-Dual-Class):



- ממדי מטריצה W2 משתנים מ-(15,1) ל-(15,3) וממד הווקטור b2 משתנים מ-(1) ל-(3).



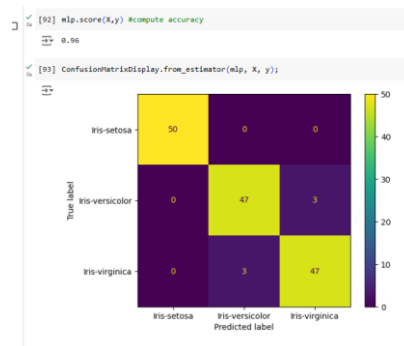
```
[8] W1 = mlp.coefs_[0]
W2 = mlp.coefs_[1]
print(W1.shape, W2.shape)

(2, 15) (15, 3)

[9] b1 = mlp.intercepts_[0]
b2 = mlp.intercepts_[1]
print(b1.shape, b2.shape)

(15,) (3,)
```

- אחוזי הדיוק נשארו כמו ב-Dual, אבל המשימה הייתה מורכבת יותר (סיווג של 3 במקום של 2) ומטריצת הבלבול היא 3x3 (לעומת 2x2 ב-Dual):



- העמקת הרשת (הוספת שכבה פנימית נוספת של 10 ניוונים) לא שיפרה את הדיוק ולא שינתה את מטריצת הבלבול.

## MLP Classifier – תרגיל

טענו את המחברת [Penguin Dataset](#), השתמשו ב-MLPClassifier:

- נסו לחזות את אחד המשתנים הבדידים (species, island, sex) על פי משתני ה-float. חשבו מדד accuracy. ציירו מטריצת הבלבול.
- בצעו את סעיף א' על משנה בדיד נוסף.

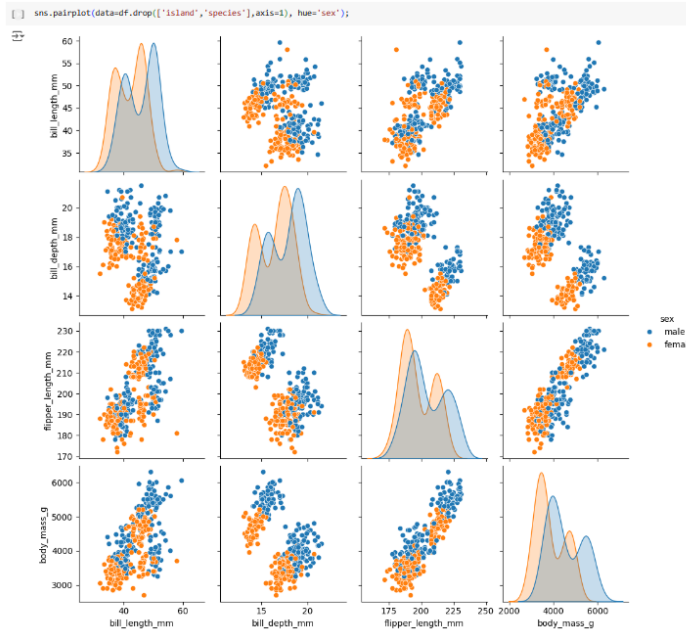
הורידו ל-drive את מחברת ה-[Penguin Dataset](#) ולתת לה את השם/קישור [Penguin dataset\\_didi.ipynb](#).

- ההוראה `df.dropna(inplace=True)` מוחקת מה-dataset את כל השורות בהן יש ערך ריק באיזושהי עמודה. הוסיפו הוראות `df.info()` ו-`df.head()` לפני ואחרי ה-`df.dropna()` על מנת לראות את השינוי.

מחיקה כזו גורמת לאיבוד המידע הנתון באותה שורה בשאר העמודות/מאפיינים ולכן בפועל לא מקובל לבצע מחיקה גורפת כזו אלא להשלים את החסר עם איזשהו ערך (ממוצע העמודה/ערך קצה/...). אבל היות ואנו לא עוסקים כאן בטיפול ב-dataset אלא ברשת עמוקה, הכי פשוט הוא למחוק את כל השורה.

- ההוראה: `sns.countplot(data=df,x='species');` מציגה מספר שורות/דוגמאות יש מכל סוג של פינגווין. אפשר לשנות על מנת לספור כמות כל מין או כל אי.
- ההוראה: `sns.pairplot(data=df.drop(['island','species'],axis=1), hue='sex');` מציגה את מטריצת גרפי הפיזור של צירופי זוגות של כל עמודות המאפיינים המספריים תוך צביעה לפי אחת העמודות של מאפיין סיווגי (sex) ו-df.drop של כל שאר העמודות המסווגות (island ו-species). ניתן לראות כאן שרגרסיה על הצירוף של bill\_depth עם body\_mass קל להפרדה וייתן סיווג ברמת דיוק גבוהה.

באלכסון המטריצה מופיעות הסטוגרמות של כמויות ה-sex (זכר/נקבה) על התחום של המאפיין המספרי וניתן לראות שכל הסטוגרמות די חופפות ולכן יהיה שקשה לחזות את ה-sex לפי מאפיין אחד בלבד.

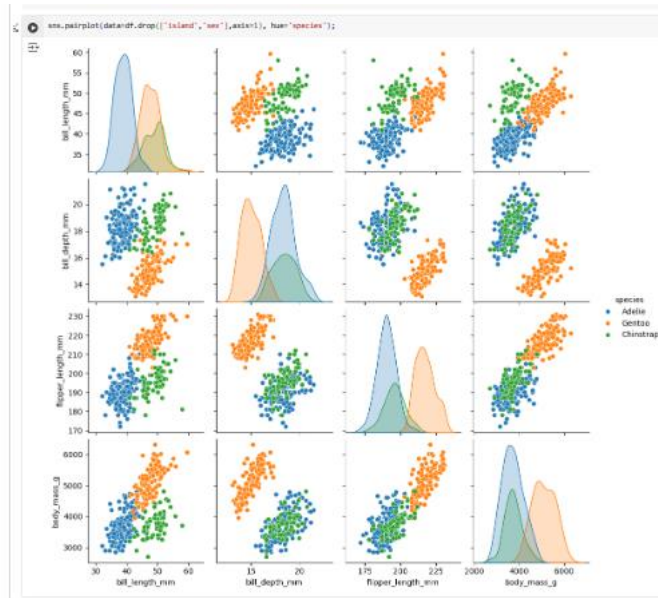


להוסיף הוראה בה מחליפים את sex ב-species:

`sns.pairplot(data=df.drop(['island','sex'],axis=1), hue='species');`

מקבלים את מטריצת גרפי הפיזור של צירופי זוגות של כל עמודות המאפיינים המספריים תוך צביעה של הנקודות לפי סוג הפינגווין והסטוגרמות של כמויות כל אחד מ-3 הסוגים על התחום של המאפיין המספרי. מגרפי

הפיזור ניתן לראות שניתן לבצע חיזוי Gento בסבירות גבוהה מאוד לפי מספר זוגות של מאפיינים ולעומת זאת קשה לחזות Chinstrap לפי שני מאפיינים.  
 ניתן לראות שבחלק מהסטוגרמות יש חפיפה מעטה (למשל Gento לפי flipper\_length) ולכן יהיה ניתן לחזות את סוג הפינגווין בהסתברות טובה גם לפי מאפיין אחד בלבד.



- הגדירו מטריצת הכניסות X (עם כל המאפיינים המספריים) ווקטור היציאה y (species) עבור ה-train:

```
[ ] X=df[['bill_length_mm', 'bill_depth_mm','flipper_length_mm', 'body_mass_g']].to_numpy()
y= df['species'].to_numpy()
print ("X.shape=",X.shape)
print ("y.shape=",y.shape)

X.shape= (333, 4)
y.shape= (333,)

X
array([[ 39.1,  18.7, 181. , 3750. ],
       [ 39.5,  17.4, 186. , 3800. ],
       [ 40.3,  18. , 195. , 3250. ],
       ...,
       [ 49.6,  18.2, 193. , 3775. ],
       [ 50.8,  19. , 210. , 4100. ],
       [ 50.2,  18.7, 198. , 3775. ]])
```

- נרמלו ערכים לתחום 0-1 במטריצת הכניסה:

#### Using MinMaxScaler

```
[ ] print('Max: ', X.max())  
print('Min: ', X.min())  
scaler = MinMaxScaler()  
X=scaler.fit_transform(X)  
print('Max: ', X.max())  
print('Min: ', X.min())
```

```
Max: 6300.0  
Min: 13.1  
Max: 1.0  
Min: 0.0
```

[ ] X

```
array([[0.25454545, 0.66666667, 0.15254237, 0.29166667],  
 [0.26909091, 0.51190476, 0.23728814, 0.30555556],  
 [0.29818182, 0.58333333, 0.38983051, 0.15277778],  
 ...,  
 [0.63636364, 0.60714286, 0.3559322 , 0.29861111],  
 [0.68 , 0.70238095, 0.6440678 , 0.38888889],  
 [0.65818182, 0.66666667, 0.44067797, 0.29861111]])
```

הפעלת הרשת על dataset בהם המאפיינים המספריים השונים הם בעלי סדר גודל שונה זה מזה עלולה לגרום לכל מיני תופעות לא טובות (למשל, תחזיות שונות בהרצות חוזרות על אותו dataset של הפינגווינים עקב בסקלה השונה ב-3 סדרי גודל של המשקל בגרמים לעומת המידות בס"מ בודדים) בהן העמודה בעלת הערכים הגבוהים משפיעה על החיזוי באופן הרבה יותר גבוה מאשר "מגיע לה" וזה קורה במיוחד ברשתות KNN (שלא נתייחס אליהן במסמך זה).

למשל: זה לא נכון שמשקל של 3500 גרם ישפיע יותר מאשר משקל של 3.5 ק"ג.

סיבה נוספת לצורך בנרמול: בגלל הצורה שהרשת עובדת, היא יותר יציבה ו"אוהבת" מבחינה מתמטית מספרים קטנים ומספרים גדולים נוטים להתנפח ולגרום ל-exploding gradient.

ה-Scaler ממפה את הערכים של כל העמודות לתחום 0-1. לכל עמודה יש פרמטרי מיפוי משלה אותם יש לשמור ולהפעיל גם על ה-test dataset וגם על ה-dataset האמיתיים בשלב המבצע.

#### תרגיל:

צרו העתק של המחברת Penguin dataset\_didi.ipynb עליה עבדתם קודם וקראו לה בשם/קישור [MLP Classifier Targil1.ipynb](#) וכאן יש את פתרון התרגיל.

ה-X וה-y שהגדרתם קודם מתאים לחיזוי species לפי כל 4 המאפיינים. את תהליך ה-train מבצעים בצורה דומה לזו שביצעתם במחברת sklearn multi class\_Iris\_didi.ipynb. בהרצה עם ההיפר פרמטרים:

hidden\_layer\_size = 15 #must be large enough

learning\_rate = 0.1

epochs = 20000

חיזוי של 100% ובריאות חוזרות התנדנדה התוצאה בין 99.7% עם שגיאה אחת של חיזוי Adelie כ-Chinstrap לבין 100%.

בהורדה ל-10 hidden\_layer\_size = 10 התקבל חיזוי של 99.7% עם שגיאה אחת של חיזוי Adelie כ-Chinstrap אבל בהרצות חוזרות התוצאה חזרה ל-100%. בהורדה ל-5 hidden\_layer\_size = 5 התקבל חיזוי של 99.7% עם שגיאה אחת של חיזוי Adelie כ-Chinstrap ובריאות חוזרות הדיוק ירד ל-95.8% (14 שגיאות) עלה ל-100%, ירד ל-96.1% (13 שגיאות)....

אם מחזירים ל-10 hidden\_layer\_size = 10 ומציבים learning\_rate = 0.2 מקבלים 97% ובהרצות חוזרות 100%. אם מורידים ל-10000 epochs, מתקבל חיזוי של 99.7% עם שגיאה אחת של חיזוי Chinstrap כ-Adelei ובהרצה חוזרות התוצאה נעה בין 98.8% ל-99.7% ו-100%.

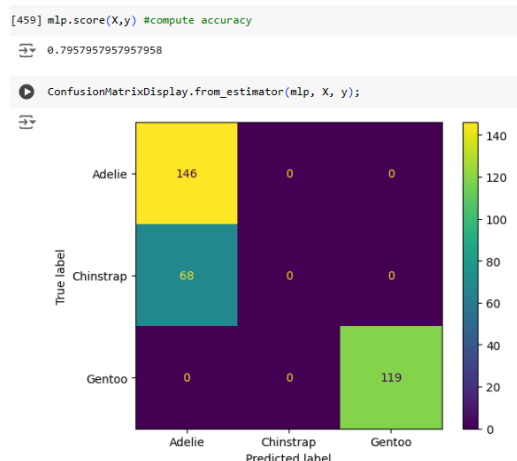
לפי הגרפים של פיזור ה-dataset בתחילת המחברת, ניתן לעבור לבעיה מאתגרת יותר של חיזוי species לפי כל 2 המאפיינים bill\_depth ו-body\_mass. בהרצה עם ההיפר פרמטרים:

hidden\_layer\_size = 15 #must be large enough

learning\_rate = 0.1

epochs = 20000

מהאיור הבא ניתן לראות שיש כשלון מוחלט בחיזוי ה-chinstrap



50

בכדי לשפר את המודל ניתן להציב ערכים שונים ולבדוק אם חל שיפור בביצועים, לדוגמה:

```
hidden_layer_size = 1000 #must be large enough  
learning_rate = 0.01  
epochs = 100000
```

(תוצאה: לא חל שיפור בביצועים של חיזוי ה-chinstrap)

שיפור באמצעות רשת עמוקה:

```
hidden_layer_sizes = [150,100,100] #must be large enough  
learning_rate = 0.05  
epochs = 50000
```

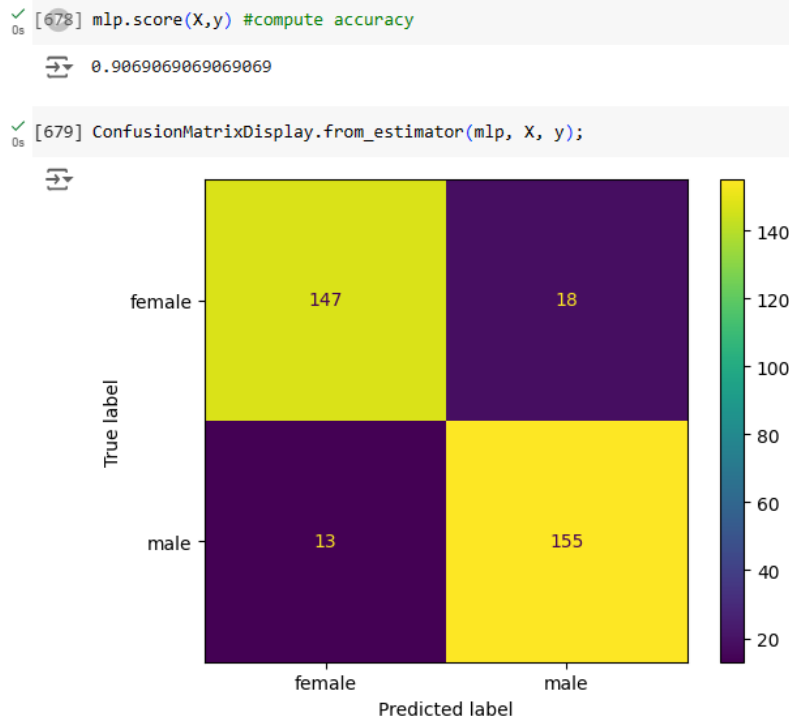
(התוצאה – לא חל שיפור בביצועים של חיזוי ה-chinstrap).

לפי הגרפים של פיזור ה-dataset בתחילת המחברת, ובמעבר לבעיה פחות מאתגרת יותר של חיזוי sex לפי 2 המאפיינים bill\_depth ו-body\_mass.

בהרצה עם ההיפר פרמטרים:

```
hidden_layer_size = 15 #must be large enough  
learning_rate = 0.1  
epochs = 20000
```

מקבלים:



### הפרדת ה-Dataset ל-Train ו-Test

עד כה בצעתם אימון על סט דוגמאות ולאחר מכן בדיקה על אותו סט. צורה בדיקה כזו אינה באמת חוזה ביצועים על סט חדש שלא השתתף בתהליך הבדיקה.

ולכן, נהוג לחלק את כמות השורות/דוגמאות של ה-dataset הנתון לשני datasets נפרדים, train ו-test. למשל: 90% מהשורות יהיו ל-train ו-10% ל-test. את הנרמול (Scale) רצוי לבצע לאחר החלוקה ובכל מקרה, יש לבצע את הנרמול של ה-test עם אותם פרמטרים של ה-train.

לצורך ביצוע החלוקה קיימת פונקציה בשם `train_test_split`.

הורידו ל-drive שלכם את המחברת [sklearn multi class with train-test split](#) Neural network שעושה שימוש בפונקציה `train_test_split` ותנו לה את השם/קישור [sklearn multi class with train-test split Didi](#).

```
X=df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].to_numpy()
y=df['Species'].to_numpy()
print ("X.shape=",X.shape)
print ("y.shape=",y.shape)
```

```
X.shape= (150, 4)
y.shape= (150,)
```

לאחר שמגדירים את ה-X וה-y של ה-dataset, מפעילים את הוראת הפיצול באופן הבא:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

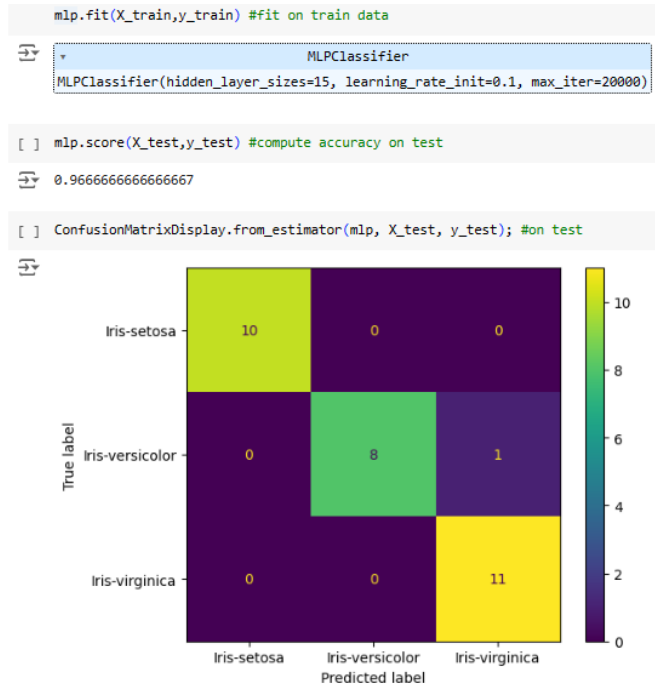
```
(120, 4) (30, 4)
(120,) (30,)
```

- ה-X,y מפוצלים ל-X\_train,y\_train ו-X\_test,y\_test.
- גודל ה-test הוא 20% (0.2) מה-X,y המקורי (ולכן, גודל ה-train הוא 80% (0.8) מה-X,y המקורי).
- בהדפסות ה-shape רואים שהגודל המקורי של הוא 150 וזה פוצל ל-120 (80%) ל-train ו-30 (20%) ל-test.
- בחירת השורות בחלוקה היא אקראית ותיתן בסופו של דבר, בכל הפעלה של המזהה, תוצאות סיווג שונות במקצת. במידה ורוצים לקבוע חלוקה זהה (ותוצאת סיווג זהה) בכל הפעלה, יש להוסיף את הפרמטר random\_state= ולתת לו ערך שלם כלשהו שיקבע את צורת בחירת השורות.



(train\_test\_split(X, y, train\_size=0.80, random\_state=42) הפוכה

- את הוראת ה-fit (ביצוע האמון) יש להפעיל על X\_train,y\_train ואת ה-score וה-ConfusionMatrix על X\_test,y\_test:



### חלוקה ל-Train, Test – תרגיל

צרו עותק של המחברת MLP Classifier on Penguin dataset\_Targil1 וקראו לה בשם/קישור:

[MLP Classifier on Penguin dataset with split train-test\\_Targil2](#)

ובצעו את התרגיל הבא:  
 הוסיפו חלוקה ל-train, test  
 נסו להקטין ולהגדיל את ה-test\_size  
 נסו לשנות את ה-random\_state  
 נסו לשנות פרמטרים נוספים

עבור:

```
[9] X=df[['bill_depth_mm','body_mass_g']].to_numpy()
y= df['sex'].to_numpy()
print ("X.shape=",X.shape)
print ("y.shape=",y.shape)
```

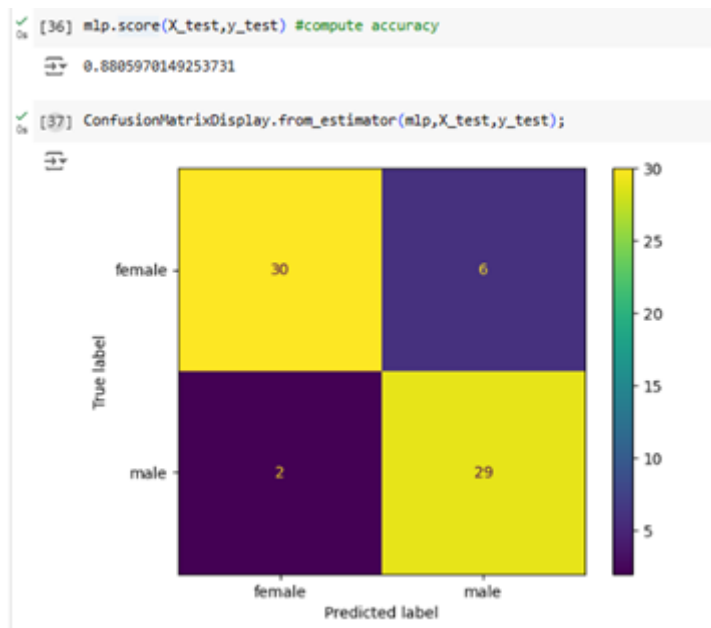
```
X.shape= (333, 2)
y.shape= (333,)
```

```
[27] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

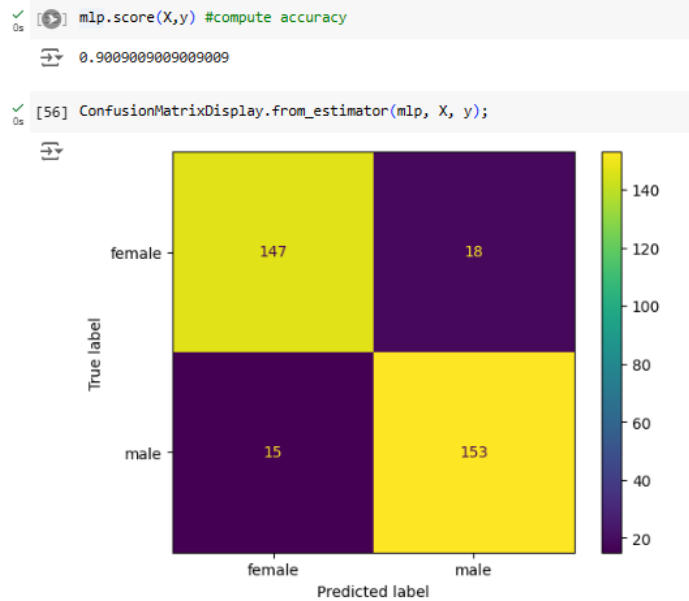
```
(266, 2) (67, 2)
(266,) (67,)
```

```
[28] hidden_layer_size = 15 #must be large enough
learning_rate = 0.1
epochs = 20000
```

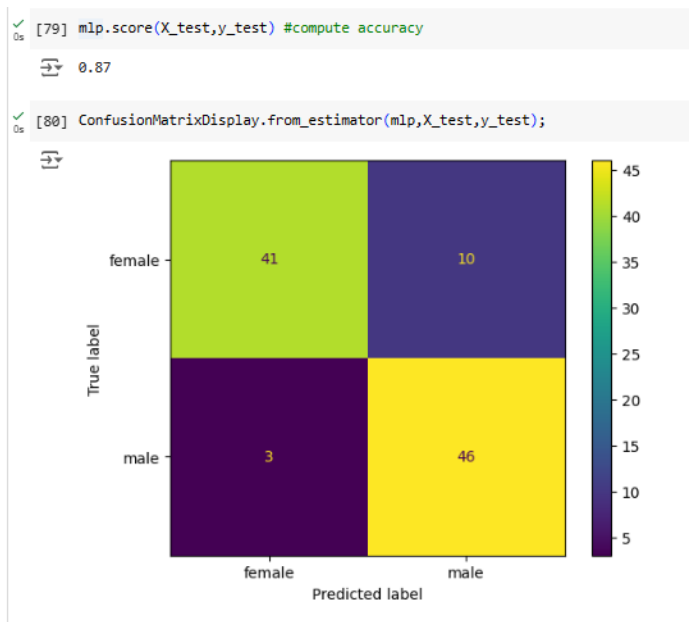
מקבלים את התוצאה של המצב ההתחלתי (אותם היפר פרמטרים של הרשת וללא חלוקה ל-train-test)



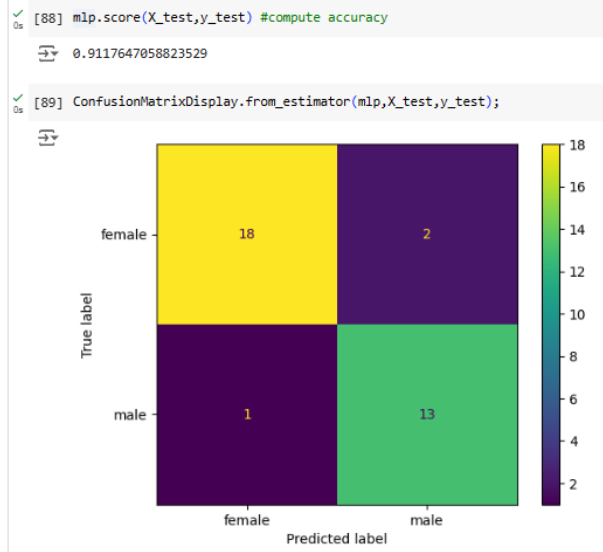
הייתה בחלוקה ל-train-test אחוז הדיוק ירד מ-90% ל-88%.



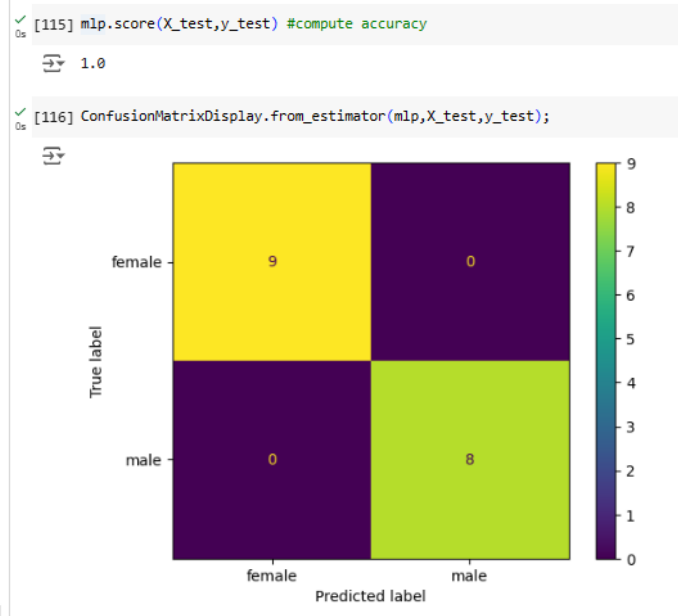
הגדלת ה-`test_size` מ-20% ל-30% נותנת : ירידה מ-88% ל-87%.



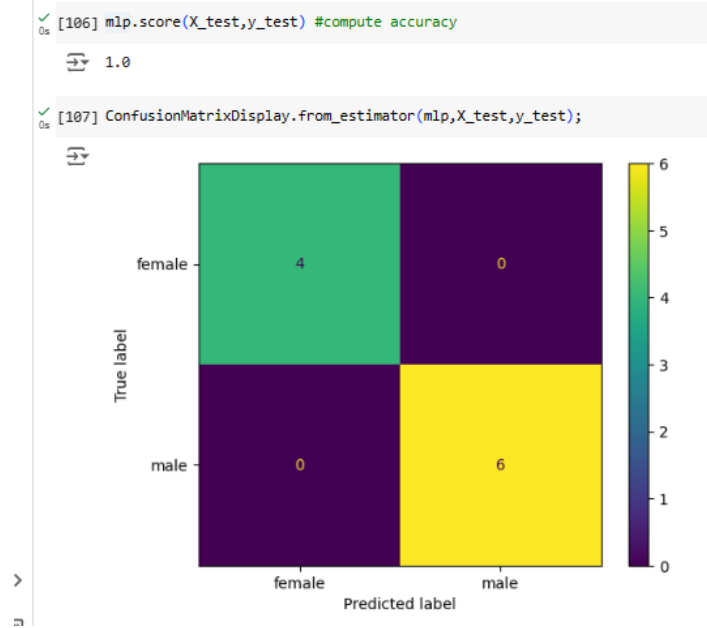
הקטנת ה-`test_size` מ-20% ל-10% נותנת עלייה מ-88% ל-91%.



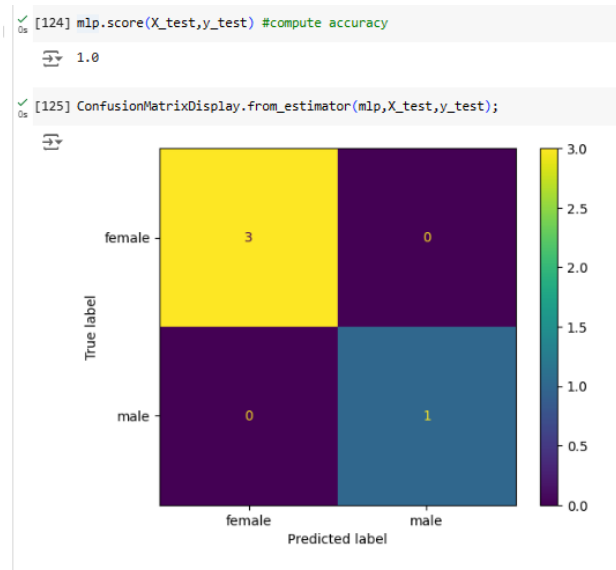
הקטנת ה-`test_size` מ-20% ל-5% נותנת עלייה מ-88% ל-100%.



הקטנת ה-`test_size` מ-20% ל-3% נותנת גם כאן עלייה מ-88% ל-100%.



הקטנת ה-`test_size` מ-20% ל-1% נותנת גם כאן עלייה מ-88% ל-100%.



שינוי של ה-`random_state` מ-42 ל-3 עם המצב ההתחלתי של ההיפר פרמטרים גורם לעליה בביצועים עם תוצאות שונות בהרצות עוקבות:

