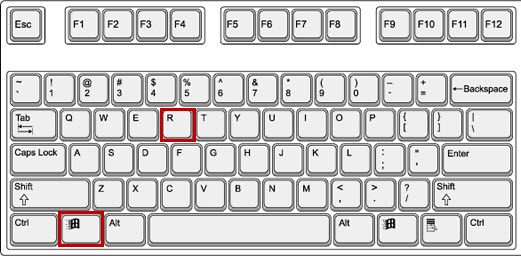
**תרגיל ניתוח מחירי בתים עם Data Science**

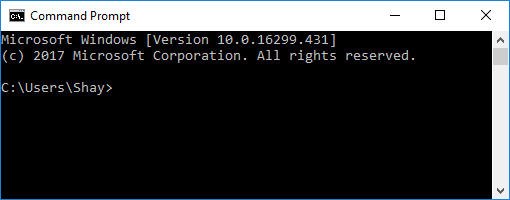
זמן הגשה – שבועיים מרגע פרסום התרגיל – עד ב 30 לאפריל 2019. מאושר קבוצות של 2 סטודנטים.

בתרגיל זה עליכם לנתח את מחירי הבתים בעיר **בוסטון**, בירתה של מדינת **מסצ'וסטס** (נראה אתכם אומרים את זה 3 פעמים ברצף בלי שגיאה 😉...) . שימו לב שהתרגיל מבוסס על מה שנעשה בכיתה – פלוס מספר תוספות קלות – ומטרתו לאפשר לכם לשחזר באופן עצמאי את תהליך החקירה של המחירים, ולהתמודד לבד עם כתיבת המחברת.

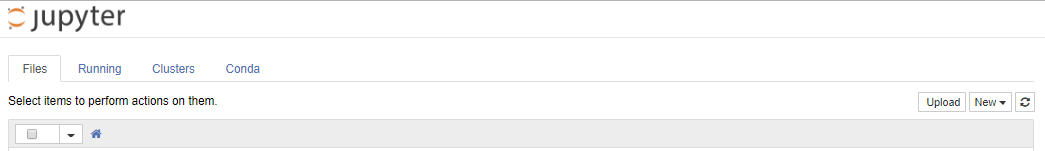
התרגיל בנוי באופן שעליכם לבצע שלב אחרי שלב. אין לדלג על שלבים. בסיום עליכם להעלות את קובץ המחברת (*בשם המכיל את הת.ז. שלכם , עם Ex1, וסיומת ipynb*) לתיבה שהוגדרה לשם כל ב Moodle.

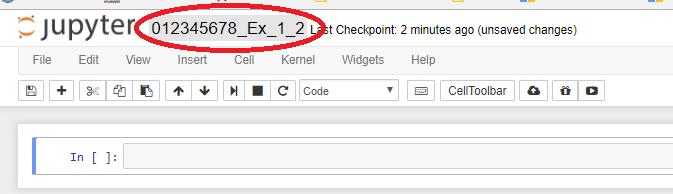
1. פתחו את סביבת Jupyter באמצעות כניסה ל Command Line
   1. במערכת ההפעלה Windows, לחצו על Win+R :



* 1. הקישו cmd ואז לחצו על מקש ה Enter. אמור לעלות המסך הבא: 
  2. במסך ה command prompt הקישו:

**jupyter notebook**

* 1. בשלב זה אמור להיפתח הדפדפן עם הממשק שאתם מכירים מהכיתה המאפשר לכתוב קוד פייתון. אם זה לא קרה... אוי לא, בהחלט מבאס!!.... ובכל זאת - עליכם לוודא שמותקנת אצלכם סביבת Anaconda. אם לא התקנתם אותה עד היום, עליכם לבצע את הבא:
     1. פתחו את Google Chrome או כל דפדפן אחר
     2. הקישו את הכתובת: <https://www.anaconda.com/download/#windows>
     3. לחצו על כפתור ה Download עבור גירסה 3.6 (או חדשה יותר במידה וקיימת): 
     4. הפעילו את קובץ ההתקנה שירד ולחצו על Next, Next, …, Ok עד שהושלמה ההתקנה.
  2. אם הגעתם בשלום לשלב הזה, זה אומר שנפתח לכם הדפדפן במסך שנראה כך (בערך): 
  3. בחרו מצד ימין את **new** ואז **Python (default)** או **Python 3** או מה שתראו ברשימה שהכי קרוב לזה. אמורה להיפתח לשונית חדשה עם מחברת ריקה, המחכה לכם על מנת לממש את התרגיל
  4. שנו את שם המחברת כך שתכיל את ת.ז. שלכם יחד עם Ex1\_2 כך:



* 1. עד לכאן כל מה שעשינו זה ליצור קובץ מחברת עבור התרגיל (והרבה רוח...). הגיע הזמן להתחיל לעבוד! בעצם עבדתם קשה עד עכשיו, אז צאו להפסקת קפה וחיזרו עם כוחות מחודשים!
  2. אם קראתם את מה שכתוב כאן – זה אומר שחזרתם! (לא טריוויאלי בכלל)
  3. לפני שנתחיל עם הקוד, במידה ושכחתם את הסינטקס תוכלו לצפות במחברות הקודמות שכתבנו בשיעורים הקודמים. אם זה לא מספיק, תוכלו להשתמש ב Cheat Sheet-ים (בעברית פשוטה – "שליפים") המופיעים בלינק הבא:

<https://mfaizmzaki.com/2017/08/25/python4datascience/>

במיוחד אלו הנוגעים בספריות Numpy, Pandas, SciPi, Matplotlib שעבדנו איתן עד היום

* 1. נתחיל בסדרה של הצהרות על ספריות שנראה להשתמש בהן במהלך התרגיל לניתוח מחירי הבתים (עליכם כמובן להכניס זאת למחברת, וזיכרו שלאחר כל פיסקה במחברת יש ללחוץ על Shift+Enter על מנת שזה יתבצע, בדיוק כמו בשיעורים...)

ב numpy שנשתמש בה עבור חישובים מתמטיים בסיסיים, בעיקר מעל ייצוג ווקטורי (אוסף של ערכים):

**import** **numpy** **as** **np**

למי שנרדם... הכוונה היתה שתעתיקו את השורה הצבעונית למחברת ותפעילו אותה...

נמשיך עם pandas שכפי שלמדנו בכיתה, מאפשרת עבודה עם טבלאות בפייתון – באופן שמזכיר את אקסל:

**import** **pandas** **as** **pd**

על מנת שיהיה לנו נוח ולא נצטרך בכל פעם שנרצה לעבוד עם Series (המייצג מערכים ) ועם DataFrame (המייצג טבלאות), נייבא אותן לשימוש ישיר:

**from** **pandas** **import** Series, DataFrame

נרצה גם להציג גרפים בתרגיל זה, אז נייבא את ספריית matplotlib

**import** **matplotlib.pyplot** **as** **plt**

וכפי שראינו בשיעורים האחרונים, במקרים בהם נרצה גרפים יותר מתקדמים המכילים גם תוצרים סטטיסטיים (כמו pearson למשל) – נייבא את ספריית seaborn, עם רקע לבן:

**import** **seaborn** **as** **sns**

sns.set\_style('whitegrid')

ולגבי Matplotlib נרצה שהתוצרים (גרפים) יודפסו ויישמרו במחברת עצמה:

%matplotlib inline

שימוב לב לשימוש בסימן ה % - זה נקרא ipython magic function – סוג של "פונקציית קסם" או מאקרו שנועד להנחות את המנגנון האחראי על המחברת להריץ פקודה שהוגדרה מראש. במקרה שלנו מדובר כפי שנכתב- בהנחיה להדפיס את הגרפים בתוך המחברת (ולשמור אותם במחברת בנוסף).

* 1. עד לכאן הצהרנו על ספריות שונות, כל אחת ומטרתה הראויה. כעת נראה לייבא את הנתונים של מחירי הבתים בבוסטון. לפני שנייבא, אם תרצו לקבל קצת יותר אינפורמציה על הנתונים שאנחנו הולכים לעבוד איתם, תוכלו לעשות זאת כאן: <https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>

מומלץ כי תבינו את משמעות כל אחד מה feature-ים (עמודות) בטבלה של בוסטון, עוד לפני הניתוח עצמו.

* 1. אוקי, מקווה שנהניתם... עכשיו הגיע הזמן לייבא את הנתונים לתוך המחברת בפייתון. בעיקרון אפשר היה להוריד את קובץ הנתונים ולהשתמש בפונקציית ייבוא לתוך מבנה Data Frame , באמצעות הפקודה pd.read\_csv(some\_file.csv'), אבל במקרה שלנו לא צריך, כי הנתונים של הבתים בבוסטון נכללים כבר במאגרי הנתונים של ספריית Scikit Learn. נוריד אותם באמצעות הספריה אם כך:

ראשית נצהיר על שימוש בספריה:

**from** **sklearn.datasets** **import** load\_boston

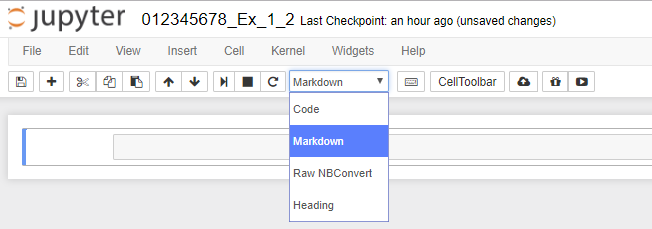
ואז פשוט נטען את הנתונים:

boston = load\_boston()

אם תבדקו מה קיבלתם (פשוט על ידי הקלדת שם המשתנה boston ולחיצה על Shift+Enter) תוכלו לראות שקיבלנו נתונים בפורמט המכיל סוגריים מסולסלות וסוגריים מרובעות... הפורמט הזה נקרא JSON – פירוש – JavaScript Object Notation שהוא פשוט פורמט מקובל בעולם להעברת נתונים בין מערכות, ובא במקור משפת התכנות Javascript. שימו לב שבתחילת הנתון שקיבלתם ישנו משתנה בשם DESCR המכיל תיאור של הקובץ, נוכל לצפות בו פשוט באמצעות:

**print** boston.DESCR

1. מעבר לכך, שימו לב ששמות העמודות נמצאים תחת boston.feature\_names , כתבו במחברת הסבר בעברית לגבי המשמעות של כל אחת מהשדות שב boston.feature\_names . (שימו לב שאת התשובות לשאלות המילוליות - כמו בסעיף זה, יש לכתוב למחברת באמצעות שורה חדשה במחברת מסוג Markdown:



1. כעת שיש לנו את הנתונים, נרצה להכיר אותם קצת יותר מקרוב. שימו לב שהנתונים של מחירי הבתים נמצאים תחת המשתנה boston.target . נתחיל בהיסטוגרמה של מחירי הבתים.

plt.hist(boston.target,bins = **5**)

plt.xlabel('Prices in $1000s')

plt.ylabel('Number of houses')

נקבל היסטוגרמה המחלקת את המחירים (באלפי דולרים) של הבתים ל 5 תאים השווים בגודלם.

1. כעת הוסיפו למחברת גם היסטוגרמות של מחירי הבתים עם 10 תאים, 100 תאים, 500 תאים, 1000 תאים.
2. האם מבין 4 ההיסטוגרמות החדשות שהוספתם ישנן היסטוגרמות זהות? מי מביניהן זהות? מדוע זה כך? (כתבו זאת בתוך שורת Markdown)
3. אנו מעונינים למצוא קשר בין הנתונים המופיעים בעמודות השונות (תחת boston.data) לבין מחירי הבתים (boston.target). על מנת לבצע זאת, ננסה קודם ידנית למצוא שדות הנמצאים בקורלציה עם ה target (מחיר), באמצעות הדפסת גרף נקודות (scatter) כך שציר Y יכיל את מחיר הבתים וציר X יכיל את היחס בין מספר התלמידים למורה בבית הספר. השדה של מספר התלמידים למורה נקרא PTRATIO והאינדקס שלו (בהנחה שהשדה הראשון הוא אינדקס 0) ברשימת השדות הוא 10. לנוחיותכם, הרי רשימת כל השדות עם האינדקסים בהתאמה (לפנק לפנק לפנק...):

0 - CRIM

1 - ZN

2 - INDUS

3 - CHAS

4 - NOX

5 - RM

6 - AGE

7 - DIS

8 - RAD

9 - TAX

**10 - PTRATIO**

11 - B

12 - LSTAT

13 - MEDV

אם כן, על מנת להציג את גרף הנקודות בין מחירי הבתים (הנתונים באלפי דולרים) לבין היחס בין מספר התלמידים למורה, נשתמש בפקודה הבאה:

plt.scatter(boston.data[:, **10**],boston.target)

plt.ylabel('Price in $1000s')

plt.xlabel('Pupil-teacher ratio')

שימו לב ש boston.data הוא משתנה מהסוג numpy.ndarray (מערך רב מימדי( בעל 506 שורות ו 13 עמודות (ניתן לבדוק זאת על ידי הפקודה boston.data.shape ). המשמעות של boston.data[:, **10**] היא לשלוף את כל השורות של המערך (נקודותיים : זה אומר לשלוף הכל), ומבין העמודות רק את העמודה בעלת האינדקס 10 – של שדה ה PTRATIO.

1. כתבו, מהתרשמותכם מהגרף שיצרתם, האם לדעתכם ישנה קורלציה חזקה או חלשה בין מחירי הבתים לבין היחס בין מספר התלמידים למורה?
2. הציגו גרפים המראים את היחס בין מחירי הבתים לבין השדות (גרף נפרד לכל מקרה): AGE, RM, INDUS,TAX,LSTAT. התאימו את הכיתוב המתאר בצירים של הגרפים בהתאם לשדות כמובן.
3. על פי התבוננות בגרפים שיצרתם, למי מבין השדות יש התאמה/קורלציה חזקה למחירי הבתים ולמי יש קורלציה חלשה? נסו להסביר מדוע זה כך.
4. על מנת לצפות בנתונים כטבלה, נרצה להמיר את המערך הרב מימדי boston.data לטיפוס מסוג DataFrame (השייך לספריית Pandas) כך:

boston\_df = DataFrame(boston.data)

נסו לראות את התוכן של המשתנה החדש boston\_df. נראה שחסרות לנו כותרות העמודות – וזה הגיוני כי הפכנו מערך רב מימדי וטבלה, אבל במערך הרב מימדי לא קיימות שמות העמודות. על מנת שנוכל גם להינות מטבלה בעלת שמות עמודות בעלי משמעות, נוכל לצרף את שמות העמודות מתוך boston.feature\_names

boston\_df.columns = boston.feature\_names

כעת נוכל פשוט לראות את התוצאה על ידי מבט מהיר ב 5 השורות הראשונות של ה DataFrame החדש שלנו:

boston\_df.head()

1. בנתונים שקיבלנו, הפרידו בין העמודות הרגילות (boston.data) לבין העמודה של מחירי הבתים (boston.target). נרצה לראות את כל העמודות (כולל מחירי הבתים) בטבלה אחת. על מנת לבצע זאת, נוסיף את מחירי הבתים לטבלה שיצרנו, תחת עמודה חדשה בשם Price:

boston\_df['Price'] = boston.target

1. הדפיסו את הטבלה boston\_df עם 10 השורות הראשונות (כן כן... 10, לא 5...)
2. כעת נרצה להשתמש בספריית Seaborn על מנת להדפיס גרף נקודות (scatter בלשון העם..) ולהוסיף לו קו לינארי המייצג את הקשר בין מספר החדרים לבין היחס בין מספר התלמידים למורה:

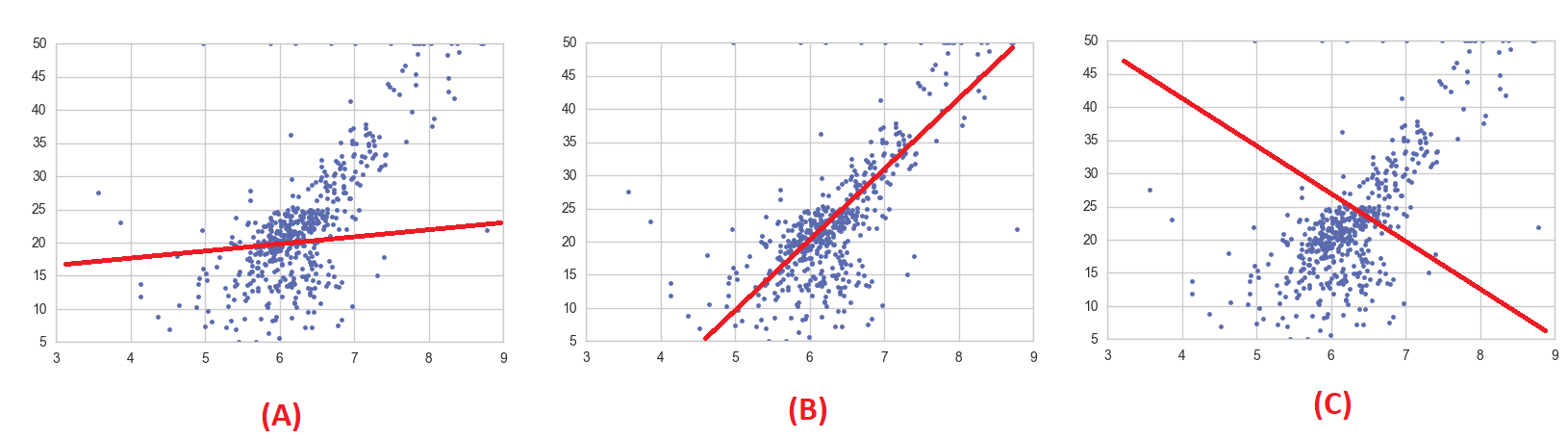
sns.lmplot('PTRATIO','Price',data = boston\_df)

plt.title('Pupil-teacher-ratio and Price Relationship')

1. הסבירו מה המשמעות של הרקע הבהיר בתכלת אשר עוטף את הקו הלינארי. תוכלו לקרוא הסברים על הפקודה lmplot כאן: <https://seaborn.pydata.org/generated/seaborn.lmplot.html>
2. הוציאו גרפים מסוג lmplot גם לקשר של **המחיר** אל מול השדה של **RM** ואל מול **LSTAT** בהתאמה
3. [קריאה בלבד]

הגרף שקיבלנו באמצעות הפקודה lmplot מייצג מודל של רגרסיה לינארית בין המחיר לבין הנתון שביקשנו לבדוק עד כמה הוא קשור ברגרסיה למחיר. על מנת להבין את המשמעות של הישר בצורה טובה יותר, למעשה נוכל לייצר את הישר הזה בעצמנו, בלי lmplot.

אבל איזה ישר נרצה לצייר בעצם? את זה שנמצא "הכי קרוב" לנקודות של הגרף הבנוי מ2 המשתנים שאנחנו משווים (למשל מחיר, ומספר חדרים). אז אם למשל הגרף שמראה נקודות (scatter) של המחיר ומספר החדרים נראה כך:

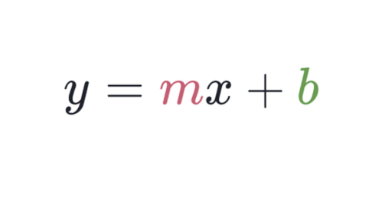


האופציה האמצעית (B) מראה ישר שנמצא הכי קרוב לנקודות עצמן – ולכן הוא מייצג את המודל הלינארי הטוב ביותר מבין ה 3 המוצגים כאן. את הקו הקרוב ביותר ניתן לאמוד בשיטת הריבועים הפחותים (Least Squares) ולמעוניינים בפרטים נוספים ניתן לקרוא כאן:

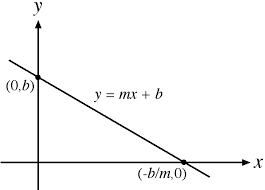
<http://mathworld.wolfram.com/LeastSquaresFitting.html>

כדי לבצע זאת, מכיוון שמדובר רק ב 2 משתנים (מחיר, מספר חדרים למשל) אפשר להשתמש בפעולות של אלגברה לינארית – או במילים פשוטות יותר – מציאת משוואת הישר בדיוק כמו שלמדנו בתיכון.. וחשבנו שלא נפגוש עוד... 😊

כזכור לכם, משוואת הישר ניתנת לתיאור על ידי:



כך ש m מייצגת את השיפוע של הישר ו b מייצגת את הערך של ציר y בו הישר פוגש את הציר y. כמו בציור הבא:

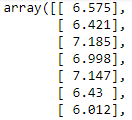
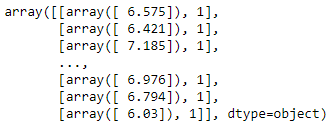


אז אם נרצה לחשב את הקו המייצג את הקשר בין מחירי הבתים לבין מספר החדרים, נוכל לספק 2 וקטורים – בציר X את מספר החדרים למשל, ובציר Y את המחיר, ונרצה שהקו יעבור כמה שיותר קרוב לנקודות שלנו (Least Squares אמרנו כבר?...)

1. נתחיל בהכנת הוקטור של ציר ה X עם הנתונים של מספר החדרים בבתים:

X = np.vstack(boston\_df.RM)

מסיבות טכניות נבצע טרספוזיציה ל X:

*(על ידי העברת X מפורמט: לפורמט: ):*

באמצעות הפקודה:

X = np.array([np.concatenate((v, [1])) for v in X])

כעת נכין את הוקטור Y המייצג את מחירי הבתים:

Y = boston\_df.Price

עכשיו צריך להריץ את המודל המוצא את משוואת הישר הקרוב ביותר לנקודות המיוצגות על ידי X,Y (הישר יתקבל באמצעות השיפוע ונקודת המפגש עם הציר y באמצעות הפרמטרים: m,b) :

s, total\_error, \_, \_ = np.linalg.lstsq(X,Y)

m,b = s

כל מה שנותר לעשות כעת זה לקבוע את המשתנה x (קטן) שאצלנו זה מספר החדרים – ובאמצעות נוסחת הישר mx+b נוכל לצייר את היישר מעל הגרף של הנקודות.

x = boston\_df.RM

נתחיל בציור הגרף של הנקודות ומייד אחריו את גרף הישר (באדום):

plt.plot(boston\_df.RM, boston\_df.Price,'.')

plt.plot(x, m\*x + b, 'red')

על מנת לחשב את השגיאה משתמשים ב RMSE = Root Mean Square Error, על ידי הפקודות הבאות:

result = np.linalg.lstsq(X,Y)

error\_tot = result[1]

rmse = np.sqrt(error\_tot/len(X))

1. כתוב במחברת כמה יצא ה RMSE והסבר מה המשמעות של השגיאה מבחינת המחיר של הבתים. למשל אם יצא לנו RMSE של 4 (מכל אחד מהצדדים של הקו הלינארי שהתאמנו לבעיה שלנו) – לשני הצדדים זה פי 2 – כלומר 8 ואז זה אומר שהשגיאה שלנו היא של 8000 דולר במחיר. כמה יצא לכם? כתבו במחברת
2. לסיום, נרצה להריץ אלגוריתם שמחשב רגרסיה לינארית (שתומך ביותר מ 2 משתנים), כך שנוכל לספק לאלגוריתם לא רק עמודה אחת, אלא את כל העמודות, והוא ימצא קשר בין העמודות הרלוונטיות (אחת או יותר) לבין המחיר. נתחיל עם הצהרה על ספריית sklearn ופקודת הרגרסיה הלינארית שלה:

**import** **sklearn**

**from** **sklearn.linear\_model** **import** LinearRegression

כעת נוכל להשתמש בפקודה. ראשית נגדיר אובייקט לרגרסיה הלינארית, ונגדיר את הנתונים לעבודה, כך שציר X יהיה כל השדות של הטבלה (מלבד השדה Price כמובן – שאותו נרצה בציר Y. את זאת נבצע על ידי בקשת הטבלה פחות העמודה Price), ובציר Y את המחיר בלבד:

lreg = LinearRegression()

X\_multi = boston\_df.drop('Price',**1**)

Y\_target = boston\_df.Price

מה שנותר לבצע זה פשוט להריץ את אלגוריתם הרגרסיה הלינארית:

lreg.fit(X\_multi,Y\_target)

המימד של הישר הלינארי שנמצא על ידי האלגוריתם עשוי להיות גדול יותר מ 2 כמובן, בואו נבדוק מאיזה מימד הוא על ידי:

len(lreg.coef\_)

למעשה lreg.coef\_ מכיל את המקדמים של כל אחד מהצירים במימד הגבוה של הישר שהתקבל. את ה intercept (מה שהיה נקרא b בישר של 2 מימדים) ניתן לקבל על ידי lreg.intercept\_

1. הדפיסו לתוך המחברת את המקדמים ואת ה intercept של הישר שהתקבל
2. כעת נייצר טבלה (DataFrame) המכילה מיפוי בין ה features – כלומר העמודות של הטבלה המקורית לבין המקדמים של כל פרמטר/feature. ראשית ניצור טבלה המכילה את שמות העמודות בלבד, ואז נוסיף אליה את המקדמים:

coef\_df = DataFrame(boston\_df.columns)

coef\_df.columns = ['Features']

coef\_df['Coefficient Estimate'] = Series(lreg.coef\_)

1. כעת הדפיסו לתוך המחברת את הטבלה של המקדמים וציינו מי מבין ה features מכיל את המקדם בעל הערך הגבוה ביותר. מה המשמעות לכך?
2. כעת נרצה לבחון עד כמה מודל הרגרסיה שירצנו עובד טוב. כיצד נבצע זאת? נבצע למעשה חלוקה של הנתונים שלנו ל רבע ושלושה רבעים מהנתונים. על הרוב (שלושה רבעים מהנתונים) נלמד את מודל הלמידה שלנו (חלק זה ייקרא train מלשון training set)– זה עם הרגרסיה הלינארית. על הרבע הנותר – נבדוק עד כמה המודל שלנו צדק או טעה בחיזוי שלו (חלק זה ייקרא test – מלשון test set כי עם החלק הזה בודקים את המודל).

נתחיל בחלוקה ל רבע ו שלושה רבעים:

X\_train, X\_test, Y\_train, Y\_test = sklearn.model\_selection.train\_test\_split(X, boston\_df.Price)

שימו לב שלא כתוב כאן במפורש שחילקנו לרבע ו שלושה רבעים – אך זוהי ברירת המחדל כפי שתוכלו לבדוק כאן: <http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

תוכלו לבדוק את גדלי החלוקה ולוודא שנעשתה כמו שצריך על ידי הדפסת הגדלים:

print X\_train.shape, X\_test.shape, Y\_train.shape, Y\_test.shape

עכשיו נריץ את אלגוריתם הרגרסיה הלינארית – כפי שהרצנו קודם, אך הפעם רק על החלק של שלושת הרבעים – הנקרא גם train אצלנו:

lreg = LinearRegression()

lreg.fit(X\_train, Y\_train)

על החלק הקטן יותר של הרבע (test) נריץ את הפרדיקציה של המודל שיצרנו באמצעות הפקודה predict (למעשה כדי להשוות נריץ גם על ה train)

pred\_train = lreg.predict(X\_train)

pred\_test = lreg.predict(X\_test)

כדי לחשב את השגיאה, נחשב את ממוצע ריבועי ההפרשים בין הפרדיקציה לנתוני האמת על ידי:

np.mean((Y\_test - pred\_test)\*\*2)

כתבו במחברת כמה יצאה השגיאה

1. בשלב האחרון, נרצה פשוט לצייר את ה residuals (ההפרשים בין הפרדיקציה למציאות) אל מול ערכי הפרדיקציה. בעיקרון אם הצלחנו ללמוד כמו שצריך, ה residuals אמורים להתפזר סביב קו ה 0 של ציר Y, ולא אמורה להיות תבנית המראה קשר לינארי בין הנקודות על הגרף. אם קיימת תבנית לינארית של רגרסיה על הנקודות בגרף ה residuals זה אומר שצריך מודל לא לינארי על מנת לפתור את הבעיה:

test = plt.scatter(pred\_test,(pred\_test - Y\_test), color = 'blue', alpha = 0.5)

plt.hlines(y=0, xmin = -10, xmax = 40, color = 'red')

plt.title('Residual Error Plot')

1. הצג את הגרף במחברת והסבר האם המודל הלינארי שהתקבל טוב או לא? האם לדעתכם צריך מודל לא לינארי?
2. אם הגעתם לכאן, מזל טוב! כל שעליכם לעשות הוא להעלות את המחברת לתיבה שנפתחה לשם כך במודל.

**בהצלחה!**