Contents

1 covid19.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os


def fit_linear_regression(design_mat, response_vec):
    singular_values = np.linalg.svd(design_mat, compute_uv=False)
    coefficients_vec = np.linalg.pinv(design_mat.T) @ response_vec
    return coefficients_vec, singular_values


def load_data(path):
    os.chdir(os.path.dirname(path))
    return pd.read_csv(os.path.basename(path))


def add_log_detected_col(path):
    data_df = load_data(path)
    data_df['log_detected'] = np.log(data_df['detected'])
    return data_df


def fit_covid(data_df):
    column_length = len(data_df)
    ones_col = np.ones(column_length)
    data_df.insert(loc=0, column='ones_col', value=ones_col)

    design_mat = data_df[['ones_col', 'day_num']].values.T
    response_vec = data_df['log_detected'].T
    coefficients_vec = fit_linear_regression(design_mat, response_vec)[0]
    day_num_vec = np.array(data_df['day_num'])
    linear_prediction_vec = design_mat.T @ coefficients_vec

    plt.plot(day_num_vec, linear_prediction_vec)
    plt.scatter(day_num_vec, response_vec)
    plt.title('log of the number of detected cases\n'
              'with the corona virus in Israel\n'
              'as a function of the number of days\n'
              'since the first case detected')
    plt.xlabel('number of days')
    plt.ylabel('log of the number of detected')
    plt.legend(['prediction for the log of the number of detected',
                'actual log of the number of detected'])
    plt.show()

    zero_coef = coefficients_vec[0]
    first_coef = coefficients_vec[1]
    exponential_prediction_vec = np.exp(first_coef * day_num_vec) * np.exp(
        zero_coef)

    plt.plot(day_num_vec, exponential_prediction_vec)
    plt.scatter(day_num_vec, np.exp(response_vec))
    plt.title('the number of detected cases\n'
              'with the corona virus in Israel\n'
              'as a function of the number of days\n'
              'since the first case detected')
    plt.xlabel('number of days')
    plt.ylabel('number of detected')
```

```python
    plt.legend(['prediction for the number of detected',
                'actual number of detected'])
    plt.show()
```

מערכות לומדות תרגיל 2

עמית בסקין 312259013

1. צ.להוכיח ש־ $\ker\left(X^{t}\right)=\ker\left(XX^{t}\right)$. הוכחה:

כיוון ראשון: $\ker\left(X^{t}\right)\subseteq\ker\left(XX^{t}\right)$: יהא $v\in\mathbb{R}^{p}$ כך ש־ $X^{t}v=\mathbf{0}_{\mathbb{R}^{n}}$, אז בפרט מתקיים ש־ $XX^{t}v=X\mathbf{0}_{\mathbb{R}^{n}}=\mathbf{0}_{\mathbb{R}^{p}}$.

כיוון שני: $\ker\left(X^{t}\right)\supseteq\ker\left(XX^{t}\right)$: יהא $v\in\mathbb{R}^{p}$ כך ש־ $XX^{t}v=\mathbf{0}_{\mathbb{R}^{p}}$. אז:

$$v^{t}XX^{t}v=0$$

$$\left(X^{t}v\right)^{t}\left(X^{t}v\right)=0$$

$$\left\langle X^{t}v,X^{t}v\right\rangle =0$$

קרי:

$$X^{t}v=\mathbf{0}_{\mathbb{R}^{n}}$$

$$v\in\ker\left(X^{t}\right)$$

מש"ל.

2. צ.להוכיח עבור מטריצה ריבועית A ש־ $\mathrm{Im}\left(A^{t}\right)=\left(\ker\left(A\right)\right)^{\perp}$. הוכחה:

כיוון ראשון: $\mathrm{Im}\left(A^{t}\right)\subseteq\left(\ker\left(A\right)\right)^{\perp}$: יהא $v\in\mathrm{Im}\left(A^{t}\right)$. אז קיים u כך ש־ $v=A^{t}u$. יהא $x\in\ker\left(A\right)$ קרי $Ax=0$, ונראה ש־ $\langle v,x\rangle=0$. אכן:

$$A^{t}u=v$$

$$u^tA=v^t$$

$$u^tAx=v^tx$$

$$0=v^tx$$

$$(v,x)=0$$

כיוון שני: $\mathrm{Im}\left(A^t\right)\supseteq\left(\ker\left(A\right)\right)^{\perp}$: יהא $v\notin\left(\ker\left(A\right)\right)^{\perp}$ ונראה ש־ $v\notin\mathrm{Im}\left(A^t\right)$. אז קיים $x\in\ker\left(A\right)$ כך ש־ $\langle v,x\rangle\neq0$. נניח בשלילה שיש u כך ש־ $A^tu=v$, קרי:

$$u^tA=v^t$$

$$u^tAx=v^tx$$

$$0=v^tx$$

$$\langle v,x\rangle=0$$

סתירה.

מש"ל.

3. יהא $y=X^tw$ מערכת לינארית אי־הומוגנית. נניח ש־ X^t ריבועית ולא הפיכה. צ.להראות שלמערכת יש אינסוף פתרונות אםם y מאונך ל־ $\ker\left(X\right)$. הוכחה:

כיוון ראשון: נניח שלמערכת יש אינסוף פתרונות. נניח בשלילה שיש $x\in\ker\left(X\right)$ כך ש־ $\langle y,x\rangle\neq0$. אז:

$$y^t=w^tX$$

$$y^tx=w^tXx$$

$$y^t x = 0$$

$$\langle y, x \rangle = 0$$

סתירה.

כיוון שני: נניח ש־ y מאונך ל־ $\ker(X)$. אז מהסעיף הקודם מתקיים ש־ $y \in \mathrm{Im}(X^t)$, קרי יש w כך ש־ $X^t w = y$. נניח בשלילה שזה הפיתרון היחיד ונקבל ש־ X מדרגה מלאה ולכן הפיכה בסתירה לנתון.

מש"ל

4. נתבונן במערכת הנורמלית הלינארת $XX^t w = Xy$. צ.להוכיח שלמערכת הנורמלית יש פיתרון יחיד אם XX^t הפיכה ואינסוף פתרונות אחרת. הוכחה:

מצד אחד אם XX^t הפיכה אז w חייב לקיים:

$$w = \left(XX^t\right)^{-1} Xy$$

קרי יש פיתרון יחיד.

מצד שני נניח ש־ XX^t לא הפיכה. לפי סעיף 3 מספיק להראות ש־ Xy מאונך ל־ $\ker(XX^t)$ ונקבל שיש אינסוף פתרונות.

כמו כן הראנו בסעיף 1 שלהיות מאונך ל־ $\ker(XX^t)$ זה שקול להיות מאונך ל־ $\ker(X^t)$, ולפי סעיף 2 זה שקול להיות ב־ $\mathrm{Im}(X)$, ואכן, $Xy \in \mathrm{Im}(X)$, כנדרש.

מש"ל

.5

א. צ.להוכיח ש־ P סימטרית. הוכחה: לכל i נסמן:

$$v_i = (v_{i1}, \ldots, v_{in})$$

ונקבל:

$$P = \sum_{i=1}^{k} v_i \otimes v_i^t =$$

$$= \sum_{i=1}^{k} \begin{bmatrix} v_{i1}^2 & v_{i1}v_{i2} & \cdots & v_{i1}v_{ik} \\ v_{i2}v_{i1} & v_{i2}^2 & \cdots & v_{i2}v_{ik} \\ \vdots & \vdots & \ddots & \vdots \\ v_{ik}v_{i1} & v_{ik}v_{i2} & \cdots & v_{ik}^2 \end{bmatrix} =$$

$$= \begin{bmatrix} \sum_{i=1}^k v_{i1}^2 & \sum_{i=1}^k v_{i1}v_{i2} & \cdots & \sum_{i=1}^k v_{i1}v_{ik} \\ \sum_{i=1}^k v_{i2}v_{i1} & \sum_{i=1}^k v_{i2}^2 & \cdots & \sum_{i=1}^k v_{i2}v_{ik} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^k v_{ik}v_{i1} & \sum_{i=1}^k v_{ik}v_{i2} & \cdots & \sum_{i=1}^k v_{ik}^2 \end{bmatrix}$$

ואכן עבור אינדקס j כלשהו, השורה ה־ j היא:

$$\begin{pmatrix} \sum_{i=1}^k v_{ij}v_{i1} \cdots & \sum_{i=1}^k v_{ij}^2 & \cdots & \sum_{i=1}^k v_{ij}v_{ik} \end{pmatrix}$$

והעמודה ה־ j:

$$\begin{pmatrix} \sum_{i=1}^k v_{i1}v_{ij} \\ \vdots \\ \sum_{i=1}^k v_{ij}^2 \\ \vdots \\ \sum_{i=1}^k v_{ik}v_{ij} \end{pmatrix}$$

אז אם נסמן את וקטור העמודה ה־ j ב־ c_j ואת וקטור השורה ה־ j ב־ r_j, נקבל שמתקיים:

$$c_j = r_j^t$$

כלומר, P סימטרית.

מש"ל

ב. צ.להוכיח שהוקטורים העצמיים של P הם 0 או 1 וש־ $v_1, \dots, v_k$ הם הוקטורים העצמיים המתאימים לערכך העצמי 1. הוכחה: נתבונן במטריצה הסימטרית האורתונורמלית:

$$Q = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1k} & v_{2k} & \cdots & v_{kk} \end{bmatrix}$$

ונשים לב כי:

$$QIQ^t =$$

$$= \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1k} & v_{2k} & \cdots & v_{kk} \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kk} \end{bmatrix} =$$

$$= \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1k} & v_{2k} & \cdots & v_{kk} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kk} \end{bmatrix} =$$

$$= \begin{bmatrix} \sum_{i=1}^{k} v_{i1}^2 & \sum_{i=1}^{k} v_{i1}v_{i2} & \cdots & \sum_{i=1}^{k} v_{i1}v_{ik} \\ \sum_{i=1}^{k} v_{i2}v_{i1} & \sum_{i=1}^{k} v_{i2}^2 & \cdots & \sum_{i=1}^{k} v_{i2}v_{ik} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} v_{ik}v_{i1} & \sum_{i=1}^{k} v_{ik}v_{i2} & \cdots & \sum_{i=1}^{k} v_{ik}^2 \end{bmatrix} = P$$

כלומר, מדובר בפירוק ספקטרלי של P ולכן האיברים באלכסון של I הם הערכים העצמיים, קרי הערך העצמי 1 וכן הוקטורים העצמיים הם העמודות של Q, קרי וקטורי הבסיס האורתונורמלי של המרחב V, כנדרש.

מש"ל

ג. ישירות מהסעיף הקודם מתקבל, לכל $v \in V$ מתקיים ש- $Pv = v$, כי v הוא וקטור עצמי עם ערך עצמי 1.

מש"ל

ד.

$$P^2 = QQ^tQQ^t$$

אבל:

$$Q^tQ =$$

$$= \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kk} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1k} & v_{2k} & \cdots & v_{kk} \end{bmatrix} =$$

$$= \begin{bmatrix} v_1^t v_1 & v_1^t v_2 & \cdots & v_1^t v_k \\ v_2^t v_1 & v_2^t v_2 & \cdots & v_2^t v_k \\ \vdots & \vdots & \ddots & \vdots \\ v_k^t v_1 & v_k^t v_2 & \cdots & v_k^t v_k \end{bmatrix}$$

ומכך שמדובר בוקטורים של בסיס אורתונורמלי:

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = I$$

ולכן:

$$P^2 = QQ^tQQ^t = P^2 = QIQ^t = QQ^t = P$$

מש"ל.

ד. ישירות מהסעיף הקודם:

$$P^2 - P = 0$$

$$P\left(I - P\right) = 0$$

מש"ל.

.6

א. יהא $X = U\Sigma V^t$ פירוק SVD של X. אז:

$$XX^t = U\Sigma V^t V\Sigma^t U^t$$

ומכך שההפכית של מטריצה אורתוגונלית היא השחלוף שלה:

$$= U\Sigma I\Sigma^t U^t = U\Sigma\Sigma^t U^t$$

ולפי הסימון בתרגיל:

$$= UDU^t$$

ומאחר שאנחנו מניחים ש- XX^t הפיכה, אזי שגם כל אחת מ- UDU^t הפיכה כי הדטרמיננטה של XX^t היא מכפלת הדטרמיננטות של המכפלה הנ"ל. מכאן שקיימת D^{-1}, ובפרט:

$$\left(UDU^t\right)^{-1} = \left(U^t\right)^{-1} D^{-1}U^{-1}$$

ושוב מאחר שההפכית של אורתונורמלית היא השחלוף:

$$= UD^{-1}U^t$$

מש"ל.

ב. מצד אחד:

$$\left(XX^t\right)^{-1}X=UD^{-1}U^tU\Sigma V^t=$$

$$=UD^{-1}\Sigma V^t=U\left(\Sigma\Sigma^t\right)^{-1}\Sigma V^t$$

אבל $\Sigma\Sigma^t$ אלכסונית וסימטרית ולכן:

$$\left(\Sigma\Sigma^t\right)^{-1}=\left(\Sigma\Sigma^t\right)^{\dagger}=\Sigma^{t\dagger}\Sigma^{\dagger}$$

קרי:

$$U\left(\Sigma\Sigma^t\right)^{-1}\Sigma V^t=U\Sigma^{t\dagger}\Sigma^{\dagger}\Sigma V^t=$$

$$=U\Sigma^{t\dagger}IV^t=U\Sigma^{t\dagger}V^t$$

ומצד שני:

$$X^{t\dagger}=\left(V\Sigma^tU^t\right)^{\dagger}=U\Sigma^{t\dagger}V^t$$

מש"ל.

7. בשאלה 1 ראינו ש- $\ker\left(X^t\right)=\ker\left(XX^t\right)$. עתה XX^t הפיכה אסם $\ker\left(XX^t\right)=\{\mathbf{0}_{\mathbb{R}^d}\}$ אסם $\ker\left(X^t\right)=\{\mathbf{0}_{\mathbb{R}^m}\}$. אבל $X:\mathbb{R}^m\to\mathbb{R}^d$, אז לפי משפט המימדים השני, הנ"ל מתקיים אסם מימד מרחב העמודות שהוא מימד התמונה של X, שווה ל- d, אבל מרחב העמודות הוא בדיוק $\operatorname{span}\{x_1,\dots x_m\}$, קרי הנ"ל קורה אסם $\operatorname{span}\{x_1,\dots,x_m\}=\mathbb{R}^d$, כנדרש.

מש"ל

8. נסמן $\widehat{w}=X^{t\dagger}y$. צ.להראות שלכל פיתרון $\overline{w}$ של $XX^tw=Xy$, מתקיים ש- $\|\widehat{w}\|_2\le\|\overline{w}\|_2$.

הוכחה: יהא $X=U\Sigma V^t$ פירוק SVD של X. ניזכר כי:

$$XX^t=U\Sigma V^tV\Sigma^tU^t=U\Sigma\Sigma^tU^t$$

ונתבונן במשוואה שלנו:

$$XX^tw=Xy$$

כלומר:

$$U\Sigma\Sigma^tU^tw = U\Sigma V^ty$$

נסמן ב־ Σ_1 את הבלוק של Σ שאינו אפסים ונקבל:

$$U\begin{bmatrix}\Sigma_1^2 & O\\ O & O\end{bmatrix}U^tw = U\begin{bmatrix}\Sigma_1 & O\\ O & O\end{bmatrix}V^ty$$

נכפיל ב־ U^t משמאל:

$$\begin{bmatrix}\Sigma_1^2 & O\\ O & O\end{bmatrix}U^tw = \begin{bmatrix}\Sigma_1 & O\\ O & O\end{bmatrix}V^ty$$

נחלק לבלוקים את U^tw ו־ V^ty בהתאם לחלוקה הקודמת:

$$\begin{bmatrix}\Sigma_1^2 & O\\ O & O\end{bmatrix}\begin{bmatrix}U_1^tw\\ U_2^tw\end{bmatrix} = \begin{bmatrix}\Sigma_1 & O\\ O & O\end{bmatrix}\begin{bmatrix}V_1^ty\\ V_1^ty\end{bmatrix}$$

נכפיל משמאל ב־ $\begin{bmatrix}\left(\Sigma_1^\dagger\right)^2 & O\\ O & O\end{bmatrix}$:

$$\begin{bmatrix}\left(\Sigma_1^\dagger\right)^2 & O\\ O & O\end{bmatrix}\begin{bmatrix}\Sigma_1^2 & O\\ O & O\end{bmatrix}\begin{bmatrix}U_1^tw\\ U_2^tw\end{bmatrix} = \begin{bmatrix}\left(\Sigma_1^\dagger\right)^2 & O\\ O & O\end{bmatrix}\begin{bmatrix}\Sigma_1 & O\\ O & O\end{bmatrix}\begin{bmatrix}V_1^ty\\ V_1^ty_2\end{bmatrix}$$

ונקבל:

$$\begin{bmatrix}U_1^tw\\ O\end{bmatrix} = \begin{bmatrix}I_1 & O\\ O & O\end{bmatrix}\begin{bmatrix}U_1^tw\\ U_2^tw\end{bmatrix} = \begin{bmatrix}\Sigma_1^\dagger & O\\ O & O\end{bmatrix}\begin{bmatrix}V_1^ty\\ V_2^ty\end{bmatrix} = \begin{bmatrix}\Sigma_1^{-1}V_1^ty\\ \mathbf{0}_{d-r}\end{bmatrix}$$

ובה"כ קיבלנו:

$$U_1^tw = \Sigma_1^\dagger V_1^ty$$

קרי:

$$w_1 = U_1\Sigma_1^\dagger V_1^ty$$

ו־ U_2^tw חופשי. אז נסמן:

$$U_2^tw = z$$

קרי:

$$w_2 = U_2 z$$

ובסה"כ:

$$w = \begin{bmatrix} U_1 \Sigma_1^{\dagger} V_1^t y \\ U_2 z \end{bmatrix}$$

וזאת עבור פיתרון כללי כאשר דרגת החופש היא ב־ z.

אז:

$$\|w\|_2^2 = \left\| \begin{bmatrix} U_1 \Sigma_1^{\dagger} V_1^t y \\ U_2 z \end{bmatrix} \right\|_2^2 =$$

$$= \left\| U_1 \Sigma_1^{\dagger} V_1^t y \right\|_2^2 + \|U_2 z\|_2^2$$

ונורמה מינימלית מתקבלת עבור $z = 0$, קרי:

$$\left\| U_1 \Sigma_1^{\dagger} V_1^t y \right\|_2^2 = \left\| U \Sigma^{\dagger} V^t y \right\|_2^2$$

אבל:

$$\widehat{w} = X^{t\dagger} y = U \Sigma^{t\dagger} V^t y$$

כלומר קיבלנו את הנורמה של $\widehat{w}$, כנדרש.

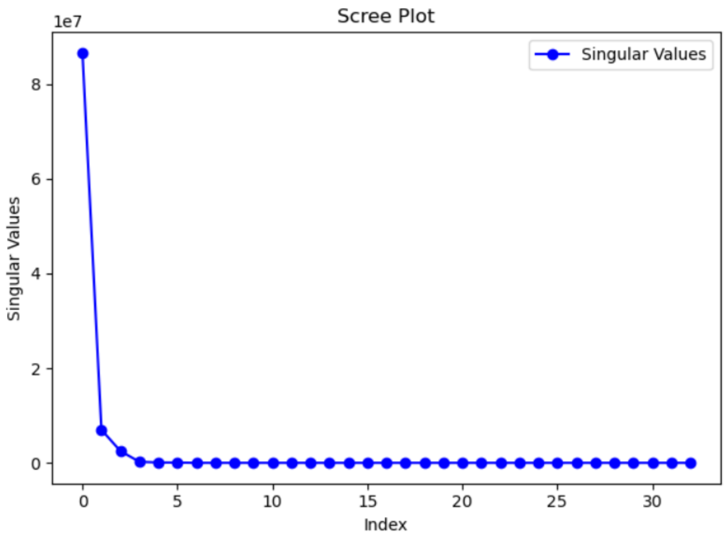
מש"ל

.13

- באשר לעמודה של התאריך: החלטתי לפצל לפי שנה וחודש, על מנת לקבל אינדיקציה על השפעה של החודש והשנה שבהם בית נמכר, וזאת בניגוד לתאריכים ספציפיים מדי, לפי ימים, ללא משמעות כמותית שאותה ניתן לכלול ברגרסיה הלינארית.
- כמו כן, החלטתי להתבונן במחיר הממוצע פר מיקוד במקום על המיקוד עצמו, על מנת לתת איפיון נוסף לטיב המיקום של הבית, שכן, המיקוד לבדו לא חושף מידע רלוונטי עבור הרגרסיה הלינארית שהרי אין לו משמעות כמותית בפני עצמו.
- את העמודה של תעודת הזהות מצאתי כלא רלוונטית ללמידה שנפעיל על המידע ולכן הסרתי את העמודה.

- באשר לעמודות שמכילות טווח ערכים מצומצם שאיננו מנורמל, טווח שנקודות הקצה שלו קשורים במשמעות של המידע, כמו קו רוחב, קו גובה, שנת בנייה, שנת שיפוץ, את כל אלה החלטתי לנרמל, שהרי המידע החשוב הוא מיקום הערך ביחס לקצוות של הטווח, וכך ניתן לראות את המידע בצורה ברורה יותר ביחס למה שמעניין אותנו לגביו, וכן המספרים יותר נוחים לעיבוד.

.15



- המטריצה X היא בקירוב מסדר $20,000 \times 30$, קרי יש לנו כ־ $20,000$ דגימות עם כ־ 30 תכונות לכל דגימה. אז X לא ולכן לא הפיכה, ובפרט מימד הגרעין של X הינו גדול מאפס.

- נתבונן ב־ X כהעתקה מ־ $\mathbb{R}^{20,000}$ ל־ $\mathbb{R}^{30}$. לפי משפט המימדים השני, מתקיים כי:

$$\dim\left(\operatorname{Im}\left(X\right)\right)+\dim\left(\ker\left(X\right)\right)=\dim\left(\mathbb{R}^{20,000}\right)$$

ובכן, הדרגה של המטריצה היא לכל היותר המינימום שבין מימד מרחב העמודות למימד מרחב השורות, קרי $.30$

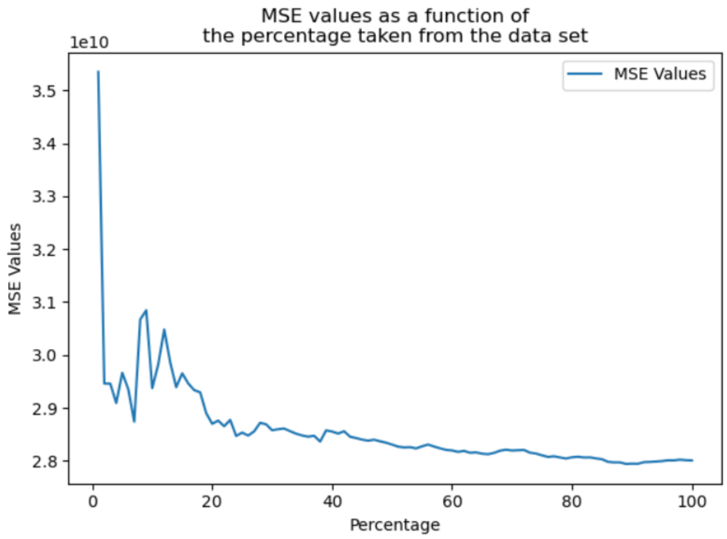
- אבל מימד התמונה הוא מימד מרחב העמודות. לכן: $\dim\left(\operatorname{Im}\left(X\right)\right)\leq 30$, ומכאן ש־ $\dim\left(\ker\left(X\right)\right)\geq$ $.19,970$

- נתבונן בפירוק SVD של X, קרי $X=U\Sigma V^{t}$. הערכים על האלכסון של Σ הם הערכים הסינגולריים של X שמקיימים $Xv=\sigma u$ עבור u,v וקטורי העמודות של U,V בהתאמה. אז לכל v עם ערך סינגולרי $\sigma=0$, מתקיים ש־ $v\in\ker\left(X\right)$. ונשים לב שכל שורה של X היא שורה של פיצ'ר. אז אם $\sigma=0$, זאת אומרת שלכל שורה x_i ב־ X מתקיים ש־ $x_iv=0$, קרי יש תלות לינארית בין הדגימות ביחס לפיצ'ר ה־ i, וזאת לכל i. אז

ככל שיש לנו יותר ערכים סינגולריים אפסיים כך התלות בין עמודות המטריצה גדולה יותר (מה שצפוי שיקרה מאחר שיש לנו סדר גודל של 30 משתנים לעומת כ־ $20,00$ משוואות).

- אם כן, הנ"ל מסתדר עם הגרף המוצג: העובדה שרוב הערכים הסינגולריים הם אפס או מאוד קרובים לאפס, עולה בקנה אחד עם העובדה שמימד הגרעין של X גדול.
- בשאלה 1 ראינו ש־ $\ker(X) = \ker(XX^t)$. מכאן שגם XX^t לא הפיכה, שהרי לכל מטריצה מתקיים שהיא הפיכה אםם מימד הגרעין שלה הוא אפס.
- כמו כן, בשאלה 4 ראינו שלמשוואה הנורמלית יש פיתרון יחיד אםם XX^t הפיכה, ואינסוף פתרונות אחרת.
- ואכן, יש לנו הרבה מאוד דגימות ביחס למספר המשתנים, ולאור הניתוח הנ"ל, ניתן להיווכח בתרומה של מספר גדול של דגימות, למציאת פתרון לבעיית הרגרסיה הלינארית שהמטריצה X היא מטריצת העיצוב שלה.
- אז לסיכום, ניתן לומר שגרף הערכים הסינגולריים מאשרר את הסינגולריות של X, ואכן, כאמור בניתוח הנ"ל, X רחוקה מאוד מלהיות הפיכה.

.16



הסבר:

w הוא וקטור המקדמים שמתאר את הקשר הלינארי בין התכונות של כל דגימה ובין המחיר של אותו בית שעליו נעשתה הדגימה.

$\hat{y}$ הוא וקטור המחירים שאנו מנחשים באמצעות w ו־ y הוא וקטורים המחירים האמיתי.

ככל שלוקחים יותר דגימות, כך w נעשה מדוייק יותר, ולכן, כלומר $\hat{y}$ קרוב יותר ל־ y, ולכן ממוצע סכום ההפרשים ביניהם דועך ככל שמספר הדגימות גדל.

.17

- תכונה שתורמת למודל: **שטח הבית**

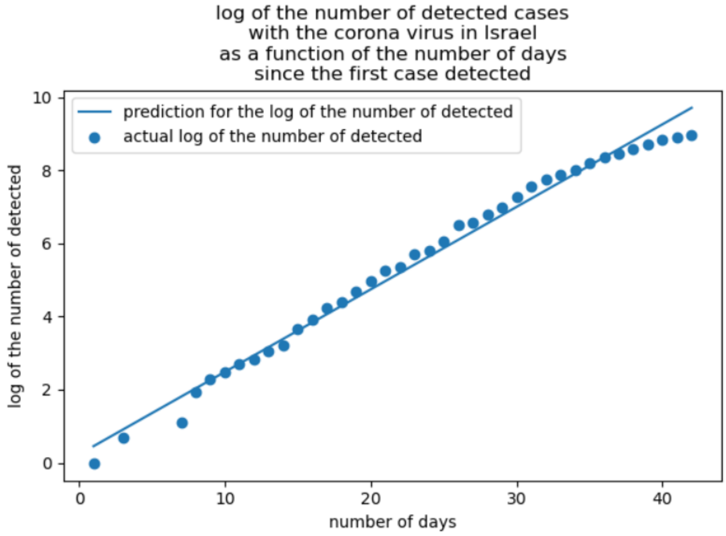
אכן, ניתן לראות בגרף שיש קשר בין השטח של הבית ובין המחיר שבו הוא נמכר: ככל שהשטח גדול יותר כך מחיר הבית נוטה להיות גבוה יותר, ואכן הקורולציה המחושבת היא הגבוהה ביותר מבין הקורולציות המחושבות.

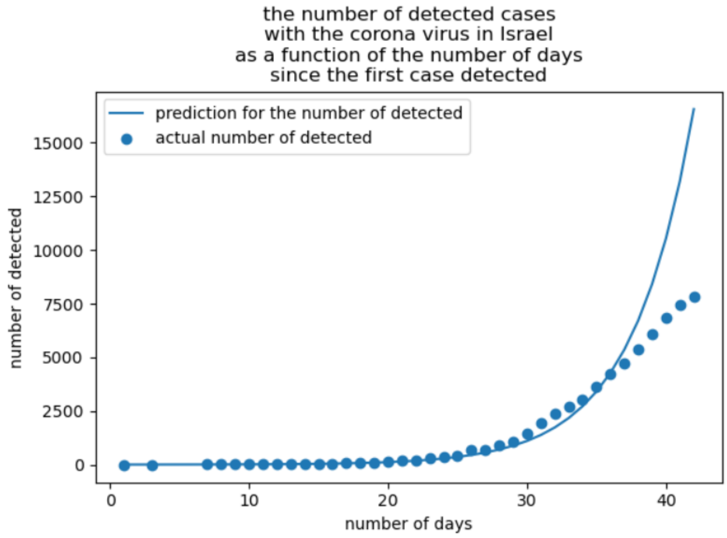
מכאן שזוהי תכונה שעוזרת למודל להתאים בין התכונות של הבית ובין המחיר שלו, ולכן היא תורמת.

- תכונה שאינה תורמת למודל: **מצב הבית**

ניתן לראות שדווקא כאשר מצב הבית הינו באמצע בין הגרוע ביותר לטוב ביותר, יש כמות גבוהה יותר של מחירים גבוהים. כמו כן, בכל אחד מהדירוגים יש בתים במחירים נמוכים. אמנם המחירים מטפסים קצת יותר כאשר מצב הבית טוב יותר, אבל עצם זה שהרוב מרוכז באמצע, מותיר עמימות בפני המודל, שכן זה לא נותן אינדיקציה לינארית לגבי הקשר בין מצב הבית למחיר. אכן, מדד הקורולציה הוא גם הנמוך ביותר. מכאן שתכונה זו לא תורמת למודל.

.21





אופן החישוב של הגרף האקספוננציאלי:

נסמן את המשתנה של prediction_detected ב־ $\hat{y}$ ואת המשתנה של day_num ב־ x. נניח שוקטור

המקדמים שמצאנו הוא (β_0, β_1) כאשר β_1 הוא המקדם של x ו־ β_0 הוא הגורם החופשי. אז:

$$\log \hat{y} = \beta_1 x + \beta_0$$

$$e^{\log \hat{y}} = e^{\beta_1 x + \beta_0}$$

$$\hat{y} = e^{\beta_1 x} e^{\beta_0}$$

22. בהתבסס על הנימוק שסיפקתי בשאלה הקודמת, במקרה האקספוננציאלי, עבור $\mathrm{x} = (x_0, x_1, \ldots, x_n)$ ו־ $\mathrm{w} = (w_0, w_1, \ldots w_n)$, כאשר $x_0 = 1$, פונקציית ההפסד צריכה להיות:

$$\left(y - \prod_{i=1}^{n} e^{w_i x_i}\right)^2 = \left(y - e^{\mathrm{x}^t \mathrm{w}}\right)^2$$

ואז כדי למצוא את w, הפיתרון ל־ ERM, נפעל בדומה להרצאה: נרצה למזער את:

$$\sum_{i=1}^{m} \left(y_i - e^{\mathrm{x}_i^t \mathrm{w}}\right)^2$$

אז נגזור לפי w ונשווה לאפס. נפתור את המשוואה שמתקבלת ונקבל את w.

3 linear model.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dateutil.parser import parse
import re
import os


def fit_linear_regression(design_mat, response_vec):
    singular_values = np.linalg.svd(design_mat, compute_uv=False)
    coefficients_vec = np.linalg.pinv(design_mat.T) @ response_vec
    return coefficients_vec, singular_values


def predict(design_mat, coefficients_vec):
    return np.matmul(design_mat.T, coefficients_vec)


def mse(prediction_vec, response_vec):
    diff = prediction_vec - response_vec
    vec_len = len(response_vec)
    return np.sum(diff ** 2) / vec_len


def is_int_and_not_negative(num):
    if isinstance(num, int) or (isinstance(num, float) and num % 1 == float(0)):
        if num >= 0:
            return True
    return False


def is_float_and_not_negative(num):
    if isinstance(num, float) or isinstance(num, int):
        if num >= 0:
            return True
    return False


def load_data(path):
    data_df = load_valid_data(path)
    data_df = data_df.iloc[:, 1:21]

    data_price_by_zip_code = data_df[['price', 'zipcode']]
    mean_price = data_price_by_zip_code.groupby('zipcode').mean()
    mean_price = mean_price.rename(
        columns={'zipdoe': 'zipcode', 'price': 'mean_price_by_zipcode'})

    data_df = data_df.merge(mean_price, on='zipcode')
    data_df = data_df.drop(['zipcode'], axis=1)

    data_df[['year', 'month']] = data_df.date.str.split(expand=True)
    data_df = pd.get_dummies(data_df, columns=['year', 'month'])
    data_df = data_df.drop(['date'], axis=1)

    cols = data_df.columns.tolist()

    def convert_to_int(string):
        return int(string) if string.isdigit() else string

```

```
    def keys(string):
        return [convert_to_int(component)
                for component in re.split('(\d+)', string)]

    cols.sort(key=keys)
    data_df = data_df[cols]

    return data_df


def load_valid_data(path):
    os.chdir(os.path.dirname(path))
    data_df = pd.read_csv(os.path.basename(path))
    for row in data_df.iterrows():
        if not is_id(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_date(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue
        date = row[1]['date']
        actual_date = parse(date, fuzzy=True)
        year = actual_date.year
        month = actual_date.month
        data_df.at[row[0], 'date'] = str(year) + ' ' + str(month)

        if not is_price(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_bedrooms_num(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_bathrooms_num(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_sqft_living(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_sqft_lot(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_floors(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_waterfront(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_view(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_condition(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_grade(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_sqft_above(row):
```

```
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_sqft_basement(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_yr_built(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue
        data_df.at[row[0], 'yr_built'] /= 2015

        if not is_yr_renovated(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue
        data_df.at[row[0], 'yr_renovated'] /= 2015

        if not is_zipcode(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_lat(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue
        data_df.at[row[0], 'lat'] /= 47.8

        if not is_long(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue
        data_df.at[row[0], 'long'] /= -123

        if not is_sqft_living15(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

        if not is_sqft_lot15(row):
            data_df = data_df.drop([row[0]], axis=0)
            continue

    return data_df


def is_id(row):
    id_num = row[1]['id']
    if is_int_and_not_negative(id_num):
        return 10 ** 6 < id_num < 10 ** 10


def is_date(row):
    date = row[1]['date']
    if isinstance(date, str):
        nums = list(map(int, re.findall(r'\d+', date)))
    else:
        return False
    is_date = None
    date_str = ''
    if len(nums) > 0:
        date_str = str(nums[0])
        try:
            actual_date = parse(date, fuzzy=True)
            year = actual_date.year
            if year == 2014 or year == 2015:
                is_date = True

        except ValueError:
            is_date = False
    if is_date:
        date_format = date_str + 'T000000'
```

```
        if date_format.__eq__(date):
            return True
    return False


def is_price(row):
    price = row[1]['price']
    if is_float_and_not_negative(price):
        return 75 * 10 ** 3 <= price <= 8 * 10 ** 6


def is_bedrooms_num(row):
    num = row[1]['bedrooms']
    if is_int_and_not_negative(num):
        return num <= 33


def is_bathrooms_num(row):
    num = row[1]['bathrooms']
    if is_float_and_not_negative(num):
        return num <= 8


def is_sqft_living(row):
    num = row[1]['sqft_living']
    if is_float_and_not_negative(num):
        return 200 <= num <= 14 * 10 ** 3


def is_sqft_lot(row):
    num = row[1]['sqft_lot']
    if is_float_and_not_negative(num):
        return 520 <= num <= 1.7 * 10 ** 6


def is_floors(row):
    num = row[1]['floors']
    if is_float_and_not_negative(num):
        return 1 <= num <= 5


def is_waterfront(row):
    num = row[1]['waterfront']
    if is_int_and_not_negative(num):
        return num <= 1


def is_view(row):
    num = row[1]['view']
    if is_int_and_not_negative(num):
        return num <= 4


def is_condition(row):
    num = row[1]['condition']
    if is_float_and_not_negative(num):
        return num <= 5


def is_grade(row):
    num = row[1]['grade']
    if is_float_and_not_negative(num):
        return 1 <= num <= 13


def is_sqft_above(row):
    num = row[1]['sqft_above']
    if is_float_and_not_negative(num):
```

```python
        return 200 <= num <= 10 ** 4


def is_sqft_basement(row):
    num = row[1]['sqft_basement']
    if is_float_and_not_negative(num):
        return num <= 4820


def is_yr_built(row):
    num = row[1]['yr_built']
    if is_int_and_not_negative(num):
        return 1900 <= num <= 2015


def is_yr_renovated(row):
    num = row[1]['yr_renovated']
    if is_int_and_not_negative(num):
        return num == 0 or 1900 <= num <= 2015


def is_zipcode(row):
    num = row[1]['zipcode']
    if is_int_and_not_negative(num):
        return 98 * 10 * 3 <= num <= 98.2 * 10 ** 3


def is_lat(row):
    num = row[1]['lat']
    if is_float_and_not_negative(num):
        to_return = 47 <= num <= 48
        return to_return


def is_long(row):
    num = row[1]['long']
    return -123 <= num <= -121


def is_sqft_living15(row):
    num = row[1]['sqft_living15']
    if is_float_and_not_negative(num):
        return 300 <= num <= 7 * 10 ** 3


def is_sqft_lot15(row):
    num = row[1]['sqft_lot15']
    if is_float_and_not_negative(num):
        return 600 <= num <= 880 * 10 ** 3


def plot_singular_values(singular_values):
    singular_values.sort()
    singular_values = singular_values[::-1]
    plt.plot(np.arange(singular_values.shape[0]), singular_values, 'bo-')
    plt.title('Scree Plot')
    plt.xlabel('Index')
    plt.ylabel('Singular Values')
    plt.legend(['Singular Values'])
    plt.show()


def add_ones_col(data_df):
    column_length = len(data_df)
    ones_col = np.ones(column_length)
    data_df.insert(loc=0, column='ones_col', value=ones_col)


```

```python
def q15(path):
    data_df = load_data(path)
    add_ones_col(data_df)
    data_df = data_df.drop('price', axis=1)
    data_mat = data_df.values.T
    singular_values = np.linalg.svd(data_mat, compute_uv=False)
    plot_singular_values(np.array(singular_values))


def q17(path):
    data_df = load_data(path)
    add_ones_col(data_df)
    data_df = data_df.sample(frac=1)
    response_vec = np.array(data_df['price'])
    data_df = data_df.drop('price', axis=1)
    desin_mat = data_df.values

    rows_amount = desin_mat.shape[0]
    test_mat_size = int(0.25 * rows_amount)
    test_mat = desin_mat[:test_mat_size]
    test_response_vec = response_vec[:test_mat_size]
    train_mat = desin_mat[test_mat_size:]
    train_response_vec = response_vec[test_mat_size:]

    coefficients_vecs_lst = []
    train_mat_rows_amount = train_mat.shape[0]
    for i in range(1, 101):
        current_rows_amount = int(i / 100 * train_mat_rows_amount)
        current_train_mat = train_mat[:current_rows_amount]
        current_response_vec = train_response_vec[:current_rows_amount]
        current_coefficients_vec = \
            fit_linear_regression(current_train_mat.T, current_response_vec)[0]
        coefficients_vecs_lst.append(current_coefficients_vec)

    mse_lst = []
    for i in range(0, 100):
        current_coefficients_vec = coefficients_vecs_lst[i]
        current_prediction_vec = predict(test_mat.T, current_coefficients_vec)
        mse_lst.append(mse(current_prediction_vec, test_response_vec))

    mse_array = np.array(mse_lst)
    percentages_array = np.array(range(1, 101))
    plt.plot(percentages_array, mse_array)
    plt.title('MSE values as a function of\nthe percentage taken from the '
              'data set')
    plt.xlabel('Percentage')
    plt.ylabel('MSE Values')
    plt.legend(['MSE Values'])
    plt.show()


def feature_evaluation(path):
    data_df = load_data(path)
    response_vec = np.array(data_df['price'])
    response_vec_std = np.std(response_vec)
    non_categorical_features = ['lat', 'long', 'price', 'year_2014',
                                'year_2015', 'yr_built', 'yr_renovated']
    months_lst = ['month_' + str(i) for i in range(1, 13)]
    features_to_drop = non_categorical_features + months_lst
    data_df = data_df.drop(features_to_drop, axis=1)
    features = data_df.columns.to_list()
    for feature in features:
        current_feature = np.array(data_df[feature])
        current_feature_std = np.std(current_feature)
        feature_mean = np.mean(current_feature)
        response_mean = np.mean(response_vec)
        feature_from_mean = current_feature - feature_mean
        response_from_mean = response_vec - response_mean
```

```
        mul = feature_from_mean * response_from_mean
        cor_numerator = np.mean(mul)
        cor_denominator = current_feature_std * response_vec_std
        current_pearson_correlation = cor_numerator / cor_denominator

        plt.scatter(current_feature, response_vec)
        plt.title('house prices as a function of ' + feature + ' values\n\n'
                  + 'Pearson Correlation = ' + str(current_pearson_correlation))
        plt.xlabel(feature + ' values')
        plt.ylabel('house prices')
        plt.legend(['house prices'])
        plt.show()
```