

تمرین شماره 2

علیرضا حسینی

شماره دانشجویی : ۸۱۰۱۰۱۱۴۲

یادگیری ماشین

دکتر ابوالقاسمی و دکتر توسلی پور

۱۴۰۲ بهار

فهرست مطالب

5 سوال 1 : linear Regression
8 سوال 2 : Lasso & Ridge Regretion
11 سوال 3 : Extention of Logestic Regression
11 سوال 4 : تخمین پنجره پارزن با h_n کوچک
14 سوال 5 : فاصله استاندارد
16 سوال 6 : بررسی under fit و overfit
16 1-6-1 : اضافه کردن نویز گوسی
22 1-6-2 : اضافه کردن نویز پواسن
24 7- Logistic Regretion
24 1-7-1 : تولید دیتا ها
27 1-7-2 : پیاده سازی L2 Regression بدون استفاده از کتابخانه آماده
29 1-7-3 : تخمین مرز در فضای ویژگی دو بعدی
31 1-7-4 : بررسی بهترین درجه
34 1-7-5 : نتیجه گیری
34 8- سوال : Parzen

فهرست اشکال

11 شکل (۱-۱) پاسخ بخش الف سوال ۳
12 شکل (۲-۱) پاسخ الف سوال ۴
13 شکل (۳-۱) بخش ب سوال ۴.
16 شکل (۴-۱) کد پایتون تولید داده ها و اضافه کردن نویز گوسی
17 شکل (۵-۱) کد پایتون رسم داده های اصلی و نویزی با نویز گوسی
17 شکل (۶-۱) منحنی داده های اصلی و نویزی (گوسی)
18 شکل (۷-۱) کد پایتون فیت کردن درجات مختلف بر روی منحنی نویزی
18 شکل (۸-۱) منحنی MSE بر حسب درجات (دیتای نویز گوسی).....
19 شکل (۹-۱) مقادیر MSE برای منحنی های درجه های ۱ تا ۱۵ (اضافه کردن نویز گوسی)
19 شکل (۱۰-۱) تعریفتابع برای رسم منحنی های فیت شده بر روی داده اصلی
20 شکل (۱۱-۱) کد پایتون رسم منحنی های فیت شده بر روی داده های اصلی (فیت روی داده های نویزی بوده)
20 شکل (۱۲-۱) رسم داده های فیت شده روی داده های اصلی (فیت روی داده های نویزی)
22 شکل (۱۳-۱) اضافه کردن نویز پواسن به Y
22 شکل (۱۴-۱) داده های اصلی و نویزی با پواسن
23 شکل (۱۵-۱) منحنی مقادیر MSE بر حسب درجه (اضافه کردن نویز پواسن)
23 شکل (۱۶-۱) مقادیر MSE برای درجه های مختلف در های مختلف
24 شکل (۱۷-۱) کد پایتون تولید دیتای حالت اول
25 شکل (۱۸-۱) دیتا های تولید شده حالت اول
25 شکل (۱۹-۱) کد پایتون دیتای حالت دوم
26 شکل (۲۰-۱) دیتای حالت دوم
27 شکل (۲۱-۱) (کد پایتون پیاده سازی L2 Regression
28 شکل (۲۲-۱) کد پایتون آموزش و پیشینی و محاسبه دقت L2 Regression
28 شکل (۲۳-۱) کد پایتون رسم مرز تصمیم گیری
29 شکل (۲۴-۱) بردن فضای ویژگی به فضای ۲
29 شکل (۲۵-۱) تجمعی داده ها در بردار X (حالت اول)
29 شکل (۲۶-۱) تجمعی داده ها در بردار X و y
30 شکل (۲۷-۱) مرز تصمیم در حالت اول
30 شکل (۲۸-۱) مرز تصمیم در حالت دوم
31 شکل (۲۹-۱) کد پایتون بررسی بهترین درجه بدون استفاده از کتابخانه آماده
32 شکل (۳۰-۱) کد پایتون استفاده از کدهای آماده و رسم و بررسی بهترین درجه
33 شکل (۳۱-۱) مرز تصمیم درجه ۶ برای حالت اول
33 شکل (۳۲-۱) مرز تصمیم برای بهترین درجه - حالت دوم

35.....	شکل (۱-۳۳) فراخوانی دیتاست Ted
35.....	شکل (۱-۳۴) جداسازی ستون duration
35.....	شکل (۱-۳۵) تعریف کرnel گوسی
36.....	شکل (۱-۳۶) تنظیم پنجره و تعریف رنج مقادیر برای تخمین density
36.....	شکل (۱-۳۷) توزیع دیتای ستون duration با پنجره به طول 10
37.....	شکل (۱-۳۸) رسم توزیع برای پنجره های مختلف
37.....	شکل (۱-۳۹) منحنی های توزیع به ازای پنجره های مختلف
38.....	شکل (۱-۴۰) کد پایتون تخمین به ازای پنجره های مختلف با کتابخانه های آماده
39.....	شکل (۱-۴۱) منحنی همگرایی توزیع با افزایش sample size

1-1- سوال 1 : linear Regression

رگرسیون خطی یک روش آماری است که هدف آن برقراری رابطه بین دو متغیر پیوسته است که در آن یک متغیر به نام متغیر دیگر به نام متغیر مستقل پیش بینی می شود. هدف رگرسیون خطی یافتن بهترین خطی است که بتواند تغییرات متغیر وابسته را بر اساس مقادیر متغیر مستقل توضیح دهد.

دو نوع رگرسیون خطی وجود دارد: رگرسیون خطی ساده و رگرسیون خطی چندگانه. در رگرسیون خطی ساده، تنها یک متغیر مستقل وجود دارد، در حالی که در رگرسیون خطی چندگانه، دو یا چند متغیر مستقل وجود دارد.

معادله یک مدل رگرسیون خطی ساده به شکل زیر است:

$$y = B_0 + B_1 \cdot x + e$$

که در آن y متغیر وابسته، x متغیر مستقل، B_0 عرض از مبدا، B_1 ضریب شیب، و e جمله خطأ است. ضریب شیب نشان دهنده تغییر در y برای تغییر واحد در x است، در حالی که عبارت عرض از مبدا نشان دهنده مقدار y زمانی است که x برابر با صفر است.

یکی از مفروضات رگرسیون خطی این است که رابطه بین متغیر مستقل و متغیر وابسته خطی است. بنابراین، رگرسیون خطی زمانی مناسب نیست که رابطه غیرخطی باشد.

فرض مهم دیگر رگرسیون خطی این است که خطاهای طور معمول توزیع شده و دارای واریانس ثابت هستند. نقض این فرض می تواند منجر به تخمین های مغرضانه و ناکارآمد ضرایب رگرسیون شود.

در برخی موارد، ممکن است نادیده گرفتن ضریب عرض از مبدا یا شیب در مدل رگرسیون خطی مناسب

باشد. نادیده گرفتن عبارت رهگیری به این معنی است که خط رگرسیون از مبدأ عبور می کند، در حالی که

نادیده گرفتن ضریب شیب به این معنی است که خط رگرسیون افقی است.

نادیده گرفتن زمانی می تواند مفید باشد که متغیر مستقل حول محور صفر باشد، و زمانی که متغیر مستقل

صفر است، منطقی نیست که متغیر وابسته مقداری غیر صفر داشته باشد.

نادیده گرفتن ضریب شیب زمانی می تواند مفید باشد که متغیر مستقل تأثیر معنی داری بر متغیر وابسته نداشته

باشد. در این حالت، خط رگرسیون به سادگی یک خط افقی است و متغیر وابسته با مقدار میانگین آن پیش‌بینی

می‌شود. مزیت نادیده گرفتن شیب یا عبارت وقفه این است که مدل رگرسیون ساده‌تر و تفسیر آن آسان‌تر می‌

شود. با این حال، توجه به این نکته مهم است که نادیده گرفتن عبارت وقفه یا شیب تنها زمانی باید انجام شود که

بر اساس ماهیت متغیرهای مورد تجزیه و تحلیل، توجیه قوی برای آن وجود داشته باشد.

الف) نادیده گرفتن ضریب شیب :

$$Y_{hat} = B0 + e$$

تابع هدف در اینجا (تابع هدف f)

$$f : \text{Arg min} (Y - B0)^2$$

$$f : \sum yi^2 - B0^2 - 2 B0 \sum yi$$

$$\frac{df}{dB0} = 0 \implies B0 = \sum yi = 59.6$$

ب) نادیده گرفتن ضریب عرض از مبدا :

$$Y_{hat} = B1 X + e$$

تابع هدف در اینجا (تابع هدف f)

$$f : \text{Arg min} (Y - B1 X)^2$$

$$f : \sum yi^2 - 2B1 \sum xi yi + B1^2 \sum xi^2$$

$$\frac{df}{dB0} = 0 \implies 0 = -2 \sum xi yi + 2 B1 \sum xi^2$$

$$B1 = \frac{Cov(X, Y)}{Var(X)}$$

$$Cov(X, Y) = 107.24$$

$$Var(X) = 36.64$$

$$B1 = 2.9269$$

- ج) تفاوت 2 معادله داده شده در این میباشد که اولی تخمینی است که میزیم ولی دومی مقدار

واقعی y میباشد و e همان خطای تخمین میباشد.

- بهترین تخمین واریانس :

برای محاسبه ابتدا مدل تخمین را به صورت زیر تعریف میکنیم :

$$Y_{\text{hat}} = 2.9269 X + 59.6$$

$$SSE = \sum (Y - Y_{\text{hat}})^2$$

$$SSE = 1.8441 * 10^4$$

برای محاسبه بهترین تخمین واریانس باید مقدار $SEE/(n-k)$ کنیم که n تعداد داده ها (10) و k

تعداد متغیر محاسبه شده در تخمین $(B0, B1)$ میباشد.

$$\frac{SSE}{8} = 2.3051 * 10^3$$

- ه) به منظور تعیین اینکه آیا می توانیم با استفاده از مدل رگرسیون خطی توسعه یافته، نمره دانش آموز

را بر اساس میزان مطالعه اش به طور دقیق پیش بینی کرد، باید عملکرد مدل روی مجموعه داده

جدیدی که برای آموزش مدل استفاده نشده است، ارزیابی شود.

اگر مدل با استفاده از یک مجموعه داده به اندازه کافی بزرگ و معروف توسعه یافته باشد، و

مفروضات رگرسیون خطی (به عنوان مثال، خطی بودن، همسانی، نرمال بودن، استقلال) برآورده

شود، آنگاه مدل باید پیش‌بینی‌های منطقی دقیقی را در مورد داده‌های جدید ارائه دهد. با این حال،

مهم است که در نظر داشته باشید که همیشه درجاتی از عدم قطعیت و خطأ در هر مدل آماری وجود

دارد، بنابراین پیش‌بینی‌ها باید با احتیاط تفسیر شوند.

یکی دیگر از موارد مورد نیاز برای پیش‌بینی دقیق تر تعداد ویژگی‌های بیشتری می‌باشد چراکه

صرف میزان مطالعه بر نمره کافی نمی‌باشد و عوامل دیگری نیز وجود دارد.

به طور کلی برای ارزیابی عملکرد مدل بر روی داده‌های جدید، یک رویکرد این است که مجموعه

داده را به یک مجموعه آموزشی و یک مجموعه آزمایشی تقسیم کنیم، مدل را در مجموعه

آموزشی قرار دهیم و سپس عملکرد مدل را در مجموعه آزمایشی ارزیابی کنیم. روش دیگر استفاده

از اعتبارسنجی متقابل برای ارزیابی عملکرد مدل در زیر مجموعه‌های متعدد داده است.

اگر مدل در داده‌های آزمون یا در اعتبارسنجی متقابل عملکرد خوبی داشته باشد، این شواهدی را

ارائه می‌کند که نشان می‌دهد می‌تواند به طور دقیق نمره دانش‌آموز را بر اساس میزان مطالعه آنها

پیش‌بینی کند. با این حال، اگر مدل در داده‌های آزمون یا در اعتبارسنجی متقابل ضعیف عمل کند،

این نشان می‌دهد که ممکن است پیش‌بینی کننده خوبی برای نمره دانش‌آموز نباشد. در هر صورت،

مهم است که پیش از نتیجه گیری، مفروضات مدل و کیفیت داده‌ها را به دقت ارزیابی کنید.

2-1- سوال 2 : Lasso & Ridge Regretion

الف) L_1 و L_2 تکنیک‌هایی هستند که برای جلوگیری از برآذش بیش از حد در

مدل‌های یادگیری ماشین با افزودن یک عبارت جریمه به تابع ضرر استفاده می‌شوند.

که با نام L1 که نیز شناخته می‌شود، یک عبارت جریمه به تابع ضرر اضافه می‌کند که متناسب با قدر مطلق وزن‌های مدل است. فرمول تنظیم L1 را می‌توان به صورت زیر نوشت:

$$\sum_{i=1}^n (y_i - \sum_j x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

که در آن λ پارامتر تنظیم می‌باشد.

تفاوت اصلی بین منظم‌سازی L1 و L2 این است که تنظیم L1 می‌تواند منجر به مدل‌های پراکنده شود، که در آن برخی از وزن‌ها دقیقاً صفر هستند (لازم به ذکر است در مواقعي که نیاز به sparse کردن می‌باشد این مورد مزیت محسوب می‌شود)، در حالی که منظم‌سازی L2 معمولاً مدل‌هایی با وزن‌های غیرصفر برای همه ویژگی‌ها تولید می‌کند. این باعث می‌شود که تنظیم L1 برای انتخاب ویژگی مفید باشد، زیرا می‌توان از آن برای حذف ویژگی‌های نامریوط از مدل استفاده کرد. از طرف دیگر، منظم‌سازی L2 برای جلوگیری از اضافه شدن به طور کلی بهتر است و معمولاً در بسیاری از روش‌های یادگیری ماشین استفاده می‌شود.

منظم‌سازی L2 که به نام ridge (ridge) نیز شناخته می‌شود، یک عبارت جریمه به تابع ضرر اضافه می‌کند که متناسب با محدود وزن‌های مدل است. فرمول تنظیم L2 را می‌توان به صورت زیر نوشت:

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p w_j^2$$

که در آن λ پارامتر تنظیم می‌باشد.

به طور خلاصه، L1 زمانی استفاده می‌شود که انتخاب ویژگی مهم است، و مدل باید وزن‌های کمی داشته باشد، در حالی که L2 زمانی استفاده می‌شود که جلوگیری از برازش

بیش از حد نگرانی اصلی است.

ب) فرم بسته bridge regularization :

$$B_{ha} = \arg \min_B \sum (Y_i - X_i B)^2 + \lambda \|B\|_2^2$$

به صورت برداری بازنویسی کرده و حل میکنیم (برای محاسبه مشتق های برداری هم از trace استفاده کرده

و به کمک قانون trace مشتق بردارها محاسبه میشود :

$$B_{hat} = \min(Y - XB)^T(Y - XB) + \lambda \sum B_i^2$$
$$0 = \frac{dB_{hat}}{dB}$$

$$-2 X^T(Y - B^T X) + 2 \lambda B = 0$$

$$X^T Y = X^T X B + \lambda B$$

$$X^T Y = (X^T X + \lambda I) B$$

$$B = (X^T X + \lambda I)^{-1} X^T Y$$

Extention of Logestic Regression : 3 سوال ۱-۳

$$P(Y=y_k | X) \sim \exp(w_{k_0} + \sum_{i=1}^d w_{ki} x_i) \quad \text{for } k=1, \dots, k-1$$

نحویم به تحریر حاکم و اینکه جو احتمالات باید ؟ بسودارم :

$$P(Y=y_m|X) + \sum_{k=1}^{k-1} P(Y=y_k|X) = 1$$

$$\Rightarrow P(Y=y_k | X) = \frac{1}{1 + \sum_{i=1}^{k-1} \exp(w_{k,i} + \sum_{j=1}^d w_{k,j} x_j)}$$

$$P(Y=y_k|X) = \frac{\exp(w_{y_k} + \sum_{i=1}^d w_i x_i)}{1 + \sum_{y_k} \exp(w_{y_k} + \sum_{i=1}^d w_i x_i)} \quad : \begin{matrix} y_k = 1, \dots, k-1 \\ \vdots \\ y_k = 0 \end{matrix} \quad \text{SVM}$$

نیز ب دل اینگی درود خورد.

شكل (١-١) پاسخ بخش الف سوال ٣

ب) در این حالت قانون طبقه بندي به اين صورت است که ليلي که پيشترین احتمال را دارد انتخاب ميکند.

$$k^* = \arg \min P(Y = y_k | X)$$

$$Y^{hat} = y_{k^*}$$

سوال 4: تخمین پنجره پارزن با h_n کوچک

پاسخ این سوال در صفحه بعدی آمده است.

حل 4:

الف)

$$\begin{aligned}
 \bar{P}_n(n) &= \frac{1}{n h_n} \sum_{i=1}^n \mathbb{E} \left[\Phi \left(\frac{n - n_i}{h_n} \right) \right] \\
 &= \frac{1}{h_n} \int_{-\infty}^{+\infty} \Phi \left(\frac{n - v}{h_n} \right) \rho(v) dv \\
 &= \frac{1}{h_n} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{n - v}{h_n} \right)^2 \right] \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{1}{2} \left(\frac{v - \alpha}{\sigma} \right)^2} dv \\
 &= \frac{1}{\sqrt{2\pi} h_n \sigma} \exp \left[\frac{1}{2} \left(\frac{n}{h_n} + \frac{\alpha^2}{\sigma^2} \right) + \frac{\alpha^2}{2\sigma^2} \right] \int_{-\infty}^{+\infty} \exp \left[-\frac{1}{2} \left(\frac{v - \alpha}{\sigma} \right)^2 \right] dv
 \end{aligned}$$

$\left. \begin{array}{l} \theta^2 = \frac{h_n^2 \sigma^2}{h_n^2 + \sigma^2} \\ \alpha = \theta^2 \left(\frac{n}{h_n} + \frac{\alpha^2}{\sigma^2} \right) \end{array} \right\}$

$$\begin{aligned}
 \Rightarrow \bar{P}_n(n) &= \frac{1}{\sqrt{2\pi} h_n \sigma} \frac{h_n \sigma}{\sqrt{h_n^2 + \sigma^2}} \exp \left[-\frac{1}{2} \underbrace{\left(\frac{n^2}{h_n^2} + \frac{\alpha^2}{\sigma^2} - \frac{\alpha^2}{\theta^2} \right)}_A \right] \\
 A &= \frac{n^2}{h_n^2} + \frac{\alpha^2}{\sigma^2} - \frac{(\theta^2)^2}{\theta^2} \left(\frac{n}{h_n} + \frac{\alpha^2}{\sigma^2} \right)^2 \\
 &= \frac{n^2 h_n^2}{(h_n^2 + \sigma^2) h_n^2} + \frac{\alpha^2 \sigma^2}{(h_n^2 + \sigma^2) \sigma^2} - \frac{2 n \alpha}{h_n^2 + \sigma^2} = \frac{(n - \alpha)^2}{h_n^2 + \sigma^2}
 \end{aligned}$$

$$\begin{aligned}
 \Rightarrow \bar{P}_n(n) &= \frac{1}{\sqrt{2\pi} \sqrt{h_n^2 + \sigma^2}} \exp \left[-\frac{1}{2} \frac{(n - \alpha)^2}{h_n^2 + \sigma^2} \right] \\
 &= \underbrace{\sim N(\alpha, h_n^2, \sigma^2)}_{\text{أ}}
 \end{aligned}$$

شكل (٤-١) پاسخ الف سوال 4

$$P(n) - \bar{P}_n(n) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{n-\mu}{\sigma}\right)^2\right] - \frac{1}{\sqrt{2\pi}\sqrt{h_n+\sigma^2}} e^{-\frac{1}{2}\left(\frac{n-\mu}{\frac{h_n}{\sigma}+\sigma}\right)^2}$$

$$= P(n) \left\{ 1 - \frac{1}{\sqrt{1+\left(\frac{h_n}{\sigma}\right)^2}} \exp\left[-\frac{(n-\mu)^2}{2}\left(\frac{1}{h_n+\sigma^2} - \frac{1}{\sigma^2}\right)\right] \right\}$$

$$= P(n) \left\{ 1 - \frac{1}{\sqrt{1+\left(\frac{h_n}{\sigma}\right)^2}} \exp\left[\underbrace{\frac{1}{2} \frac{h_n^2 (n-\mu)^2}{h_n^2 + \sigma^2}}_B\right] \right\}$$

$h_n \rightarrow \text{small}$

$$\frac{1}{\sqrt{1+\left(\frac{h_n}{\sigma}\right)^2}} \sim 1 - \frac{1}{2} \left(\frac{h_n}{\sigma}\right)^2, \quad e^B = 1 + \frac{h_n^2}{2\sigma^2} \frac{(n-\mu)^2}{h_n^2 + \sigma^2}$$

$$\Rightarrow P(n) - \bar{P}_n(n) \leq P(n) \left\{ 1 - 1 + \frac{1}{2} \frac{h_n^2}{\sigma^2} - \frac{h_n^2}{2\sigma^2} \frac{(n-\mu)^2}{h_n^2 + \sigma^2} \right\}$$

$$= \frac{1}{2} \left(\frac{h_n}{\sigma}\right)^2 \left[1 - \frac{(n-\mu)^2}{h_n^2 + \sigma^2} \right] P(n)$$

$$\xrightarrow{h_n \rightarrow \text{small}} \underline{\underline{\frac{1}{2} \left(\frac{h_n}{\sigma}\right)^2 \left[1 - \left(\frac{n-\mu}{\sigma}\right)^2 \right] P(n)}}$$

شكل (١-٣) بخش ب سوال 4

۱-۵-سوال ۵ : فاصله استاندارد

فرض کنید در فضای n بعدی دو نقطه A و B داریم که مختصات $A (a_1, a_2, \dots, a_n)$ و مختصات $B (b_1, b_2, \dots, b_n)$

هستند. فاصله اقلیدسی بین A و B به صورت زیر تعریف می شود:

$$D(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

حال، اگر هر مختصات را در یک مقدار واقعی غیر صفر ضرب کنیم، مثلاً k . مختصات A و B را A' و B'

بگیریم:

فاصله اقلیدسی بین A' و B' برابر است با:

$$D(A', B') = \sqrt{(ka_1 - kb_1)^2 + (ka_2 - kb_2)^2 + \dots + (ka_n - kb_n)^2}$$

$$D(A', B') = |k| \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

$$D(A', B') = |k| D(A, B)$$

اما در مورد فاصله استاندارد داریم :

در ریاضیات، متریک فاصله استاندارد یا متریک تابعی است که فاصله بین هر دو نقطه در یک مجموعه معین

را مشخص می کند. برای در نظر گرفتن معیار فاصله استاندارد، یک تابع باید شرایط خاصی را داشته باشد:

- غیر منفی: فاصله بین هر دو نقطه باید غیر منفی یا صفر باشد.
- تقارن: فاصله بین دو نقطه بدون توجه به ترتیبی که در نظر گرفته می شوند باید یکسان باشد.
- نابرابری مثلث: فاصله بین هر دو نقطه همیشه باید کمتر یا مساوی با مجموع فواصل بین نقاط و نقطه

سوم باشد

- هویت غیر قابل تشخیص: فاصله بین هر دو نقطه متمایز نباید به هویت نقاط بستگی داشته باشد. به عبارت دیگر، اگر x و y نقاط متمایز در مجموعه باشند، $d(x, y)$ باید بدون توجه به هر ویژگی دیگر نقاط یکسان باشد.

این شرایط تضمین می کند که متریک فاصله به خوبی تعریف شده است و در عملیات ریاضی و محاسباتی به طور مداوم رفتار می کند. فاصله اقلیدسی یک متریک فاصله استاندارد است که این شرایط را در فضای n بعدی برآورده می کند.

از آنجایی که k غیر صفر است، قدر مطلق آن $|k|$ نیز غیر صفر است. بنابراین، نشان دادیم که ضرب عناصر هر بعد در یک مقدار واقعی غیر صفر، فاصله بین دو نقطه در فضای اقلیدسی n بعدی را تغییر نمی دهد و از این رو فاصله استاندارد باقی می ماند.

الگوریتم K-nearest (KNN) یک الگوریتم یادگیری ماشینی محبوب است که برای کارهای طبقه‌بندی و رگرسیون استفاده می شود. این کار با یافتن k نزدیکترین نقطه (همسایه ها) به یک نقطه داده داده شده، بر اساس متریک فاصله، و استفاده از کلاس یا مقدار آن همسایگان برای پیش‌بینی کلاس یا مقدار نقطه داده داده شده، کار می کند.

در الگوریتم KNN، انتخاب متریک فاصله نقش مهمی در تعیین نزدیک‌ترین همسایگان دارد. هنگامی که عناصر هر بعد را در یک مقدار واقعی غیر صفر ضرب می کنیم، همانطور که در پاسخ قبلی نشان دادیم، متریک فاصله همچنان استاندارد است. با این حال، این ضرب می تواند بر روی الگوریتم KNN تأثیر بگذارد.

هنگامی که عناصر هر بعد را در یک مقدار واقعی غیر صفر ضرب می کنیم، اساساً داده ها را مقیاس بندی می کنیم. این می تواند بر فاصله بین نقاط و بنابراین نزدیکترین همسایگان تأثیر بگذارد. به عنوان مثال، مجموعه داده ای را با دو بعد در نظر بگیرید: قد و وزن. اگر مقادیر قد را در 2 ضرب کنیم، فاصله بین دو نقطه تحت سلطه اختلاف مقادیر وزنی خواهد بود و مقادیر ارتفاع تأثیر کمتری بر متریک فاصله خواهند داشت. این می تواند منجر به انتخاب نزدیکترین همسایگان مختلف شود که به نوبه خود می تواند بر روی نتایج طبقه‌بندی یا رگرسیون الگوریتم KNN تأثیر بگذارد.

برای کاهش اثر مقیاس گذاری بر روی الگوریتم KNN، اغلب توصیه می شود که داده ها را نرمال سازی کنید، به این معنی که هر بعد را برای داشتن میانگین و واریانس واحد صفر مقیاس کنید. این تضمین می کند که همه ابعاد در متریک فاصله به یک اندازه مهم هستند و الگوریتم KNN نسبت به هیچ بعد خاصی تعصب ندارد. عادی سازی همچنین به جلوگیری از مشکلات مربوط به بی ثباتی عددی که می تواند هنگام برخورد با مقادیر بسیار بزرگ یا بسیار کوچک ایجاد شود، کمک می کند.

به طور خلاصه، مقیاس بندی داده ها می تواند بر عملکرد الگوریتم KNN با تغییر متریک فاصله و انتخاب نزدیکترین همسایگان تأثیر بگذارد. برای جلوگیری از این مسائل، توصیه می شود قبل از اعمال الگوریتم KNN، داده ها نرمال شود.

۱-۶-سوال ۶ : بررسی under fit و overfit

۱-۶-۱- حالت ۱ : اضافه کردن نویز گوسی

ابتدا مطابق توضیحات داده شده داده ها و داده های نویزی را رسم میکنیم.

```
# Generate data
X = np.arange(-10, 10, 0.2)
Y = 2*np.cos(X)/(-np.pi) + (2*X)/(2*np.pi) + 2*np.cos(3*X)/(-3*np.pi)
noise = np.random.randn(len(X)) * 0.1
Y_noisy = Y + noise
```

شکل (۱-۴) کد پایتون تولید داده ها و اضافه کردن نویز گوسی

در ادامه نیز داده ها را برای درک بهتر از موضوع رسم میکنیم.

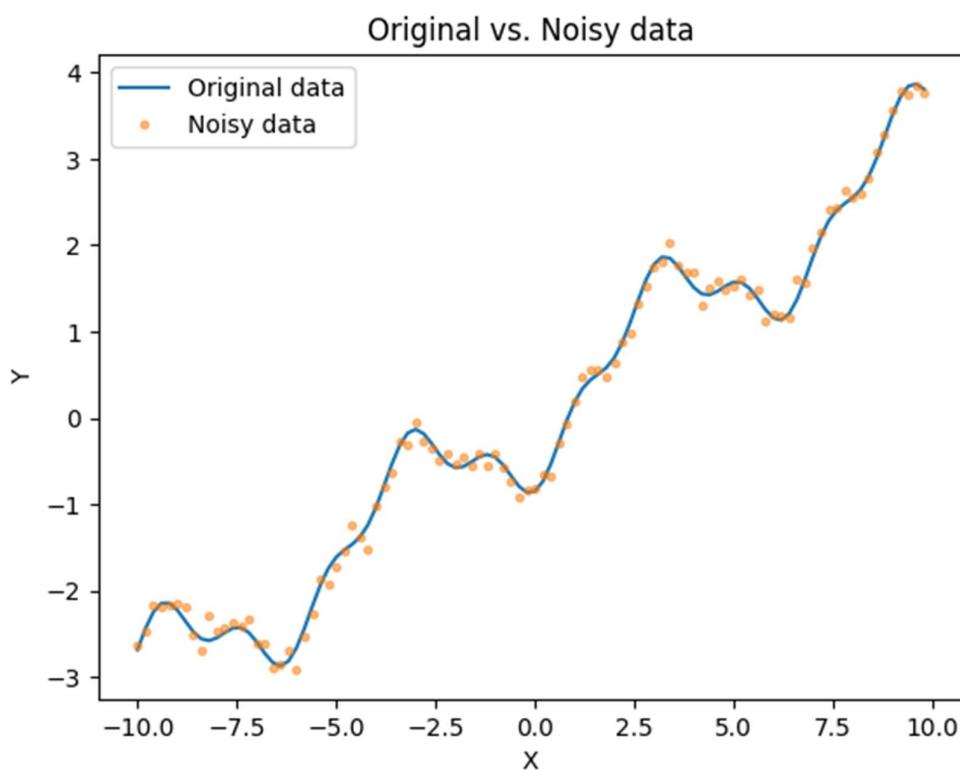
```

# Plot X, Y, and Y_noisy
fig, ax = plt.subplots()
ax.plot(X, Y, label='Original data')
ax.plot(X, Y_noisy, '.', alpha=0.5, label='Noisy data')
ax.legend()
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title('Original vs. Noisy data')
plt.show()

```

شکل (۱-۵) کد پایتون رسم داده های اصلی و نویزی با نویز گوسی

شکل زیر نمای کلی از داده های مساله در حالت ۱ را نشان میدهد. که قرار است منحنی هایی با درجه های مختلف را بر آن fit کرده و نتیجه را بررسی کنیم.



شکل (۱-۶) منحنی داده های اصلی و نویزی (گوسی)

با توجه به عدم ذکر این موضوع که الزاماً باید از scratch کد ها را نوشت از کتابخانه های آماده استفاده

میشود. اما این موضوع بدون استفاده از کتابخانه های آماده نیز چندان کار سختی نمیباشد.

نکته حائز اهمیت بررسی و محاسبه MSE با Y میباشد نه Y های نویزی

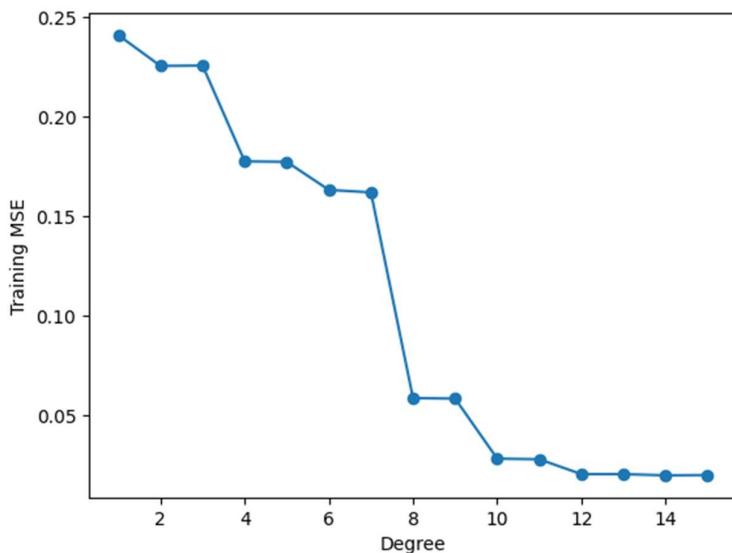
```
# Fit polynomial regression models of degrees 1 to 15
degrees = range(1, 16)
train_mses = []

for degree in degrees:
    # Fit polynomial regression model
    poly_features = PolynomialFeatures(degree=degree, include_bias=False)
    X_poly = poly_features.fit_transform(X.reshape(-1, 1))
    model = LinearRegression()
    model.fit(X_poly, Y_noisy)

    # Evaluate on data
    y_train_pred = model.predict(X_poly)
    train_mse = mean_squared_error(Y, y_train_pred)
    train_mses.append(train_mse)
```

شکل (۱-۷) کد پایتون فیت کردن درجات مختلف بر روی منحنی نویزی

منحنی زیر مقادیر MSE را بر حسب درجات مختلف نمایش میدهد.



شکل (۱-۸) منحنی MSE بر حسب درجات (دیتای نویز گوسی)

مقادیر MSE ها به ترتیب در زیر آمده است :

```
[0.24094146713216863,
 0.22548730186203966,
 0.2256829720441145,
 0.17761945868109347,
 0.17736706482625528,
 0.1632835939099453,
 0.16204838156711862,
 0.05879331628130377,
 0.05857517694947249,
 0.028440081934079812,
 0.028107727497808793,
 0.02065735120258373,
 0.02065001246347317,
 0.020050182962398896,
 0.020196263815617947]
```

شکل (1-۹) مقادیر MSE برای منحنی های درجه های 1 تا 15 (اضافه کردن نویز گوسی)

با توجه به مقادیر به دست آمده بدترین حالت بر اساس MSE درجه 1 و بهترین حالت (کمترین MSE) نیز درجه 14 میباشد (البته اگر بخواهیم دستی و چشمی بررسی کنیم MSE در درجه 14 اندکی کمتر از 12 میباشد و اگر یک threshold ای داشتیم درجه 12 مقدار مناسبی میباشد) مقدار MSE برای درجه 1 برابر 0.24 و برای درجه 14 نیز برابر 0.0200 میباشد.

برای رسم تمام منحنی های درجه های مختلف روی یک منحنی یک تابع نوشته و از آن استفاده میکنیم.

```
# Define function for plotting the fitted line on the original data
def plot_fitted_line(degree, X, Y, color):
    poly_features = PolynomialFeatures(degree=degree, include_bias=False)
    X_poly = poly_features.fit_transform(X.reshape(-1, 1))
    model = LinearRegression()
    model.fit(X_poly, Y)
    x_plot = np.linspace(-10, 10, 200)
    X_plot_poly = poly_features.transform(x_plot.reshape(-1, 1))
    y_plot = model.predict(X_plot_poly)
    plt.plot(x_plot, y_plot, color=color, label=f"Degree {degree} fit")
```

شکل (1-۱۰) تعریف تابع برای رسم منحنی های فیت شده بر روی داده اصلی

```

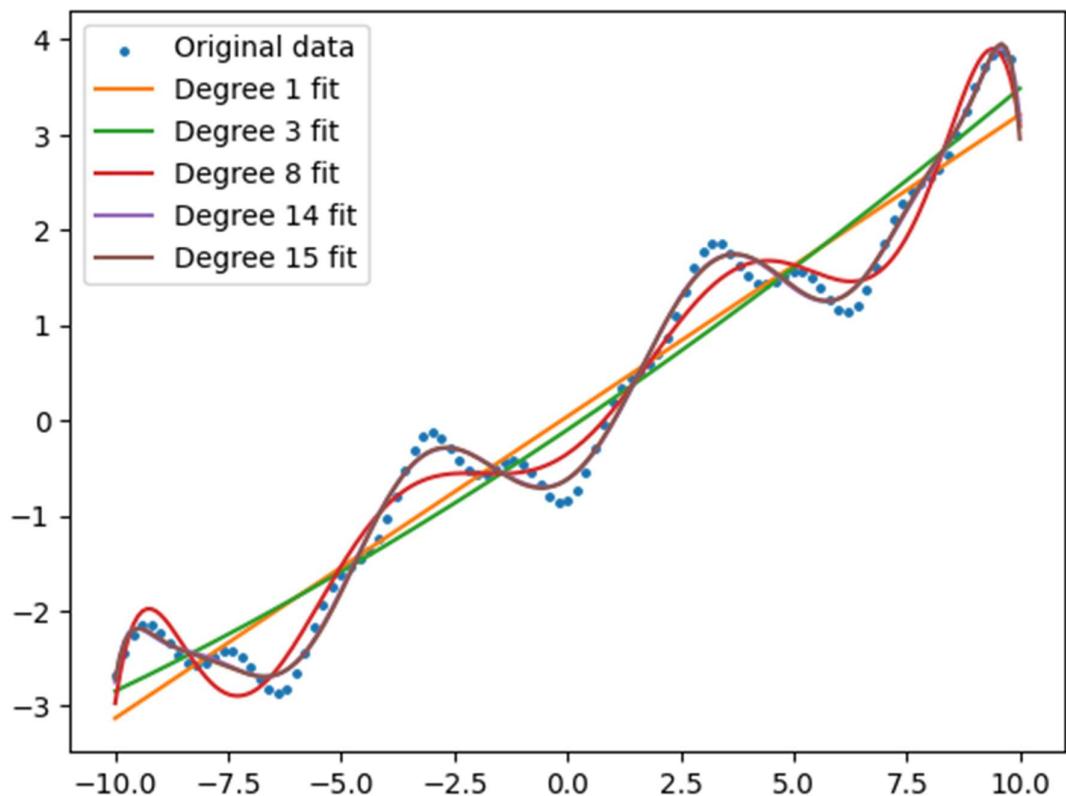
# Plot original data
plt.scatter(X, Y, s=6, label='Original data')

# Plot fitted lines of degree 1, 3, 8, 14 and 15
plot_fitted_line(1, X, Y, 'C1')
plot_fitted_line(3, X, Y, 'C2')
plot_fitted_line(8, X, Y, 'C3')
plot_fitted_line(14, X, Y, 'C4')
plot_fitted_line(15, X, Y, 'C5')

plt.legend()
plt.show()

```

شکل (۱-۱۱) کد پایتون رسم منحنی های فیت شده بر روی داده های اصلی (فیت روی داده های نویزی بوده)



شکل (۱-۱۲) رسم داده های فیت شده روی داده های اصلی (فیت روی داده های نویزی)

در ادامه نیز Best Model و Worst Model را لود کرده و مقادیر Bias و واریانس را محاسبه میکنیم

برای این محاسبه 2 رویکرد داریم :

یکی استفاده از کتابخانه های آماده و دیگری الگوریتم bootstrap که در کد آمده در فایل ارسالی از روش دوم (به جهت پیاده سازی از scratch) استفاده شده است ، که مراحل این الگوریتم محاسبه به شرح زیر است :

- با توجه به یک مجموعه داده با N نمونه ، به طور تصادفی از N نمونه (با جایگزینی) از مجموعه داده

برای ایجاد یک مجموعه داده جدید "bootstrap" نمونه برداری میشود

- مدل روی مجموعه داده بوت استرپ آموزش داده شده و عملکرد آن را روی مجموعه داده اصلی

ارزیابی میشود. خطای پیش بینی (MSE) برای هر نقطه داده محاسبه میشود

- مراحل 1 و 2 به تعداد B بار برای ایجاد مجموعه داده های بوت استرپ B و مدل های مربوطه

تکرار میشود.

- با ایاس مدل را به عنوان تفاوت بین میانگین پیش بینی مدل ها و مقادیر واقعی نقاط داده محاسبه

میشود.

- واریانس مدل به عنوان میانگین اختلاف مجدور بین پیش بینی های هر مدل و پیش بینی میانگین

محاسبه میشود

- در پیاده سازی آمده شده در کد ، از نمونه های بوت استرپ $B=100$ برای تخمین بایاس و واریانس

هر مدل استفاده شده است.

نتیجه نهایی برای حالت 1 در زیر آمده است :

Best model: Bias = 0.0188, Variance = 0.0015

Worst model: Bias = 0.2409, Variance = 0.0002

با توجه به مقادیر فوق در Worst Model در undefit هستیم.

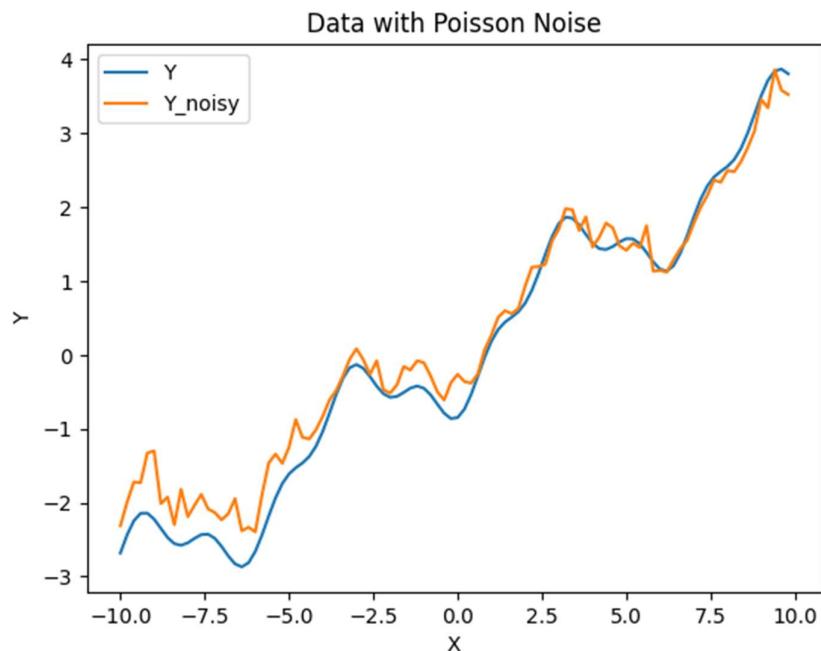
1-6-2- حالت 2: اضافه کردن نویز پواسن

همان مراحل قبل با تقریبا همان کد ها را همین جا تکرار میکنیم که برای آنکه توضیح اضافاتی نداده باشیم

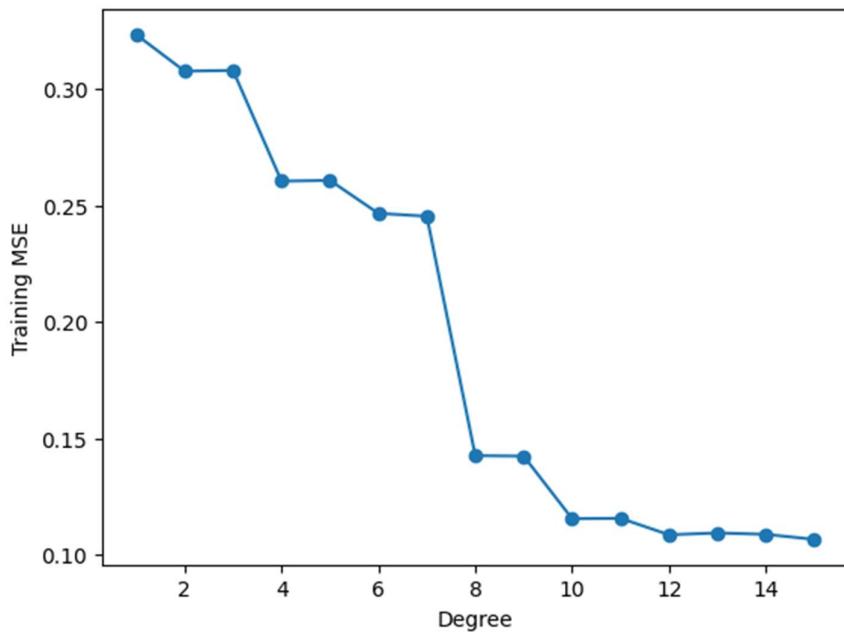
در اینجا فقط نتایج را می آوریم.

```
# Add Poisson noise to Y
lam = np.maximum(np.exp(-Y/10), 1e-8)
Y_noisy = np.random.poisson(lam=lam*2, size=Y.shape)
Y_noisy = Y + 0.1 * (Y_noisy - Y)
```

شکل (1-13) اضافه کردن نویز پواسن به Y



شکل (1-14) داده های اصلی و نویزی با پواسن



شکل (۱-۱۵) منحنی مقادیر MSE بر حسب درجه (اضافه کردن نویز پواسن)

```
[0.3233210164952722,
 0.3076027083510651,
 0.3079054510485156,
 0.26035798291645007,
 0.2606997589317259,
 0.24652698677650825,
 0.2452908514090677,
 0.1424692844261482,
 0.14225851937974926,
 0.11538570286398185,
 0.11554755629298429,
 0.10847996049002254,
 0.10927131819627477,
 0.10866252959300944,
 0.10649986470412624]
```

شکل (۱-۱۶) مقادیر MSE برای درجه های مختلف در های مختلف

با توجه به مقادیر خوب کمترین MSE برای درجه ۱۵ میباشد.

طبق همان الگوریتم هم مقادیر bias و variance به صورت زیر محاسبه میشود.

Best model: Bias = 0.0188, Variance = 0.0017

Worst model: Bias = 0.2409, Variance = 0.0000

به طور کلی، مشاهدات ما اهمیت انتخاب یک پیچیدگی مدل مناسب برای متعادل کردن بایاس و واریانس، و همچنین تاثیر نویز بر عملکرد مدل را نشان می‌دهد. ما همچنین نحوه محاسبه بایاس و واریانس را با استفاده از روش بوت استرپ نشان دادیم، که یک تکیک مفید برای ارزیابی عملکرد مدل‌های یادگیری ماشین است.

1-7-بررسی Logistic Regretion

1-7-1- تولید دیتا ها

ابتدا بر اساس سناریو های داده شده داده ها را تولید می‌کنیم.

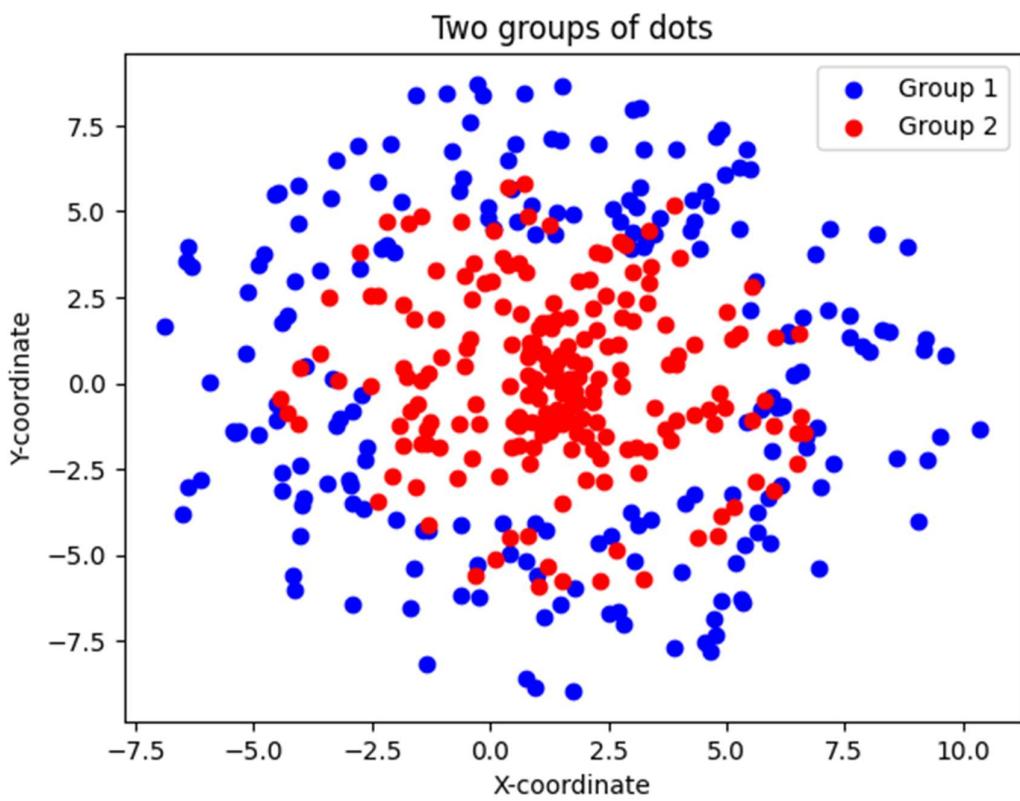
```
# Group 1
center1 = (1.5, 0)
r_min1 = 4
r_max1 = 9
n_points1 = 200

theta1 = np.random.uniform(0, 2*np.pi, n_points1)
r1 = np.random.uniform(r_min1, r_max1, n_points1)
x1 = center1[0] + r1 * np.cos(theta1)
y1 = center1[1] + r1 * np.sin(theta1)

# Group 2
center2 = (1.5, 0)
r_min2 = 0
r_max2 = 6
n_points2 = 200

theta2 = np.random.uniform(0, 2*np.pi, n_points2)
r2 = np.random.uniform(r_min2, r_max2, n_points2)
x2 = center2[0] + r2 * np.cos(theta2)
y2 = center2[1] + r2 * np.sin(theta2)
```

شکل (1-17) کد پایتون تولید دیتا ها



شکل (۱-۱۸) دیتا های تولید شده حالت اول

```

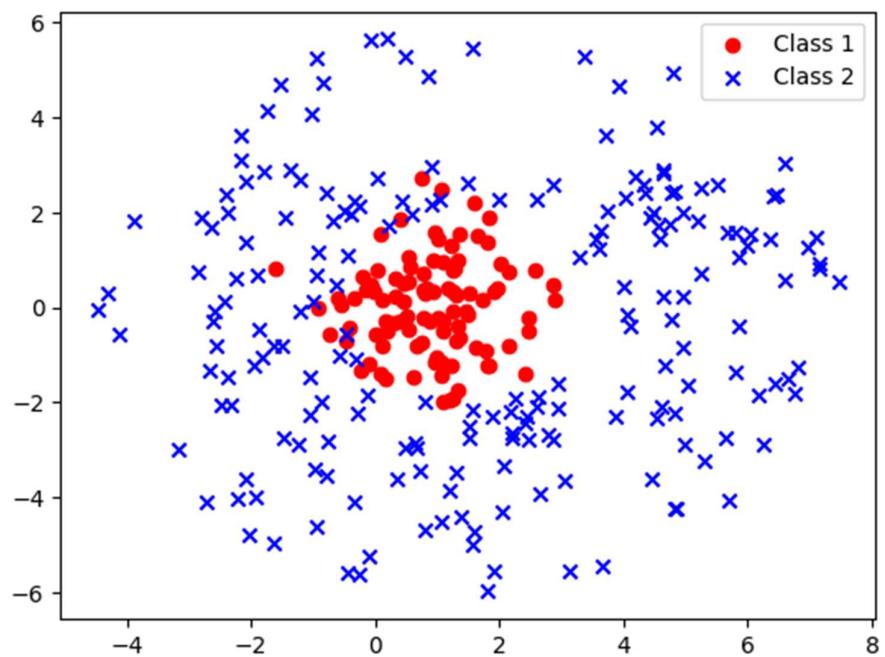
np.random.seed(42)
mean = [1, 0]
cov = [[1, 0], [0, 1]]
x1, y1 = np.random.multivariate_normal(mean, cov, 100).T

# Generate data for the second category
theta = np.random.uniform(0, 2*np.pi, 200)
r = np.random.uniform(2, 6, 200)
x2 = r*np.cos(theta) + 1.5
y2 = r*np.sin(theta)

# Plot the data
fig, ax = plt.subplots()
ax.scatter(x1, y1, c='r', marker='o', label='Class 1')
ax.scatter(x2, y2, c='b', marker='x', label='Class 2')
ax.legend()
plt.show()

```

شکل (۱-۱۹) کد پایتون دیتای حالت دوم



شکل (۱-۲۰) دیتای حالت دوم

1-7-2- پیاده سازی L2 Regression بدون استفاده از کتابخانه آماده

```
# Define logistic regression with L2 regularization
def logistic_regression_l2(X, y, alpha, num_iterations):
    # Initialize weights
    w = np.zeros(X.shape[1])

    # Define sigmoid function
    def sigmoid(z):
        return 1 / (1 + np.exp(-z))

    # Define cost function with L2 regularization
    def cost_function(w, X, y, alpha):
        m = X.shape[0]
        z = X.dot(w)
        J = -1/m * (y.dot(np.log(sigmoid(z))) + (1-y).dot(np.log(1-sigmoid(z)))) + alpha/(2*m) * np.sum(w[1:]**2)
        return J

    # Define gradient function with L2 regularization
    def gradient(w, X, y, alpha):
        m = X.shape[0]
        z = X.dot(w)
        grad = 1/m * X.T.dot(sigmoid(z) - y) + alpha/m * np.concatenate(([0], w[1:]))
        return grad

    # Run gradient descent
    for i in range(num_iterations):
        J = cost_function(w, X, y, alpha)
        grad = gradient(w, X, y, alpha)
        w = w - 0.01 * grad
```

شکل (1-۲۱) کد پایتون پیاده سازی L2 Regression

در این قسمت از کد یک مدل رگرسیون لجستیک با تنظیم L2 از ابتدا تعریف شده است. پارامترهای ورودی

این تابع عبارتند از:

X: ماتریس ویژگی های ورودی

y: بردار برچسب های خروجی

آلفا: پارامتر تنظیم

num_iterations: تعداد تکرارها برای بهینه سازی گرادیان نزول

این تابع با مقداردهی اولیه بردار وزن "w" به بردار صفر با همان تعداد ستون X شروع می شود. سپس تابع

سیگموید را تعریف می کند که برای ترسیم ویژگی های ورودی به یک مقدار احتمال بین 0 و 1 استفاده می

شود. تابع هزینه را تعریف می کند که اتلاف آنتروپی متقابل را با جریمه تنظیم L2 محاسبه می کند. تابع گرادیان

نیز تعریف شده است که گرادیان تابع هزینه را با توجه به وزن‌های "w" و همچنین با جریمه تنظیم L2 محاسبه می‌کند.

در نهایت، این تابع از بهینه‌سازی gradient decent برای بهروزرسانی بردار وزن به صورت تکراری برای تعداد دفعات «num_iterations» استفاده می‌کند. در هر تکرار، تابع هزینه و گرادیان را محاسبه می‌کند و بردار وزن را با کم کردن گرادیان ضرب در نرخ یادگیری 0.01 به روز می‌کند. تابع پس از فرآیند بهینه‌سازی، بردار وزن "w" را برمی‌گرداند.

در نهایت نیز برای ترین و پیشینی مدل و محاسبه دقت از کد زیر استفاده می‌شود.

```
# Train logistic regression with L2 regularization
w = logistic_regression_l2(X, y, alpha=0.1, num_iterations=1000)

# Predict class labels for all data points
y_pred = sigmoid(X.dot(w)) >= 0.5

# Compute accuracy
accuracy = np.mean(y_pred == y)
print("Accuracy:", accuracy)
```

شکل (۱-۲۲) کد پایتون آموزش و پیشینی و محاسبه دقت L2 Regression

در نهایت نیز از کد زیر برای رسم مرز تصمیم‌گیری استفاده می‌شود.

```
# Plot decision boundary
fig, ax = plt.subplots()
x_min, x_max = X[:, 1].min() - 1, X[:, 1].max() + 1
y_min, y_max = X[:, 2].min() - 1, X[:, 2].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                      np.arange(y_min, y_max, 0.1))
Z = np.column_stack((np.ones(xx.ravel().shape), xx.ravel(), yy.ravel(),
                      (xx.ravel())**2, xx.ravel()**yy.ravel(), (yy.ravel())**2)).dot(w)
Z = sigmoid(Z)
Z = Z.reshape(xx.shape)
cmap = ListedColormap(['#FF0000', '#0000FF'])
ax.contourf(xx, yy, Z, alpha=0.4, cmap=cmap)
ax.scatter(x1, y1, c='r', marker='o', label='Class 1')
ax.scatter(x2, y2, c='b', marker='x', label='Class 2')
ax.legend()
plt.show()
```

شکل (۱-۲۳) کد پایتون رسم مرز تصمیم‌گیری

1-7-3- تخمین مرز در فضای ویژگی دو بعدی

با توجه به داده های ترسیم شده در بخش 1 ، به نظر میرشد با درجه 2 و دایره هایی میتوان با دقت خوبی داده های 2 کلاس را تفکیک کرد ابتدا این بخش را تست میکنیم.

قبل از آموزش باید یک سری تغییرات بر روی داده های y_1 و y_2 و x_1 و x_2 بدهیم. تمام مراحل زیر برای هر 2 حالت باید انجام شود.

```
# Increase feature space to degree 2
X1 = np.column_stack((x1, y1, x1**2, x1*y1, y1**2))
X2 = np.column_stack((x2, y2, x2**2, x2*y2, y2**2))
```

شکل (1-۲۴) بردن فضای ویژگی به فضای 2

```
# Concatenate data from both classes
num_samples = 200
X = np.vstack((X1, X2))
y = np.concatenate((np.zeros(num_samples), np.ones(num_samples)))

# Add intercept term to feature matrix
X = np.column_stack((np.ones(2*num_samples), X))
```

شکل (1-۲۵) تجمعی داده ها در بردار X (حالت اول)

در حالت دوم به دلیل آنکه یکی از داده ها 100 تایی و یکی 200 تایی است به صورت زیر میشود.

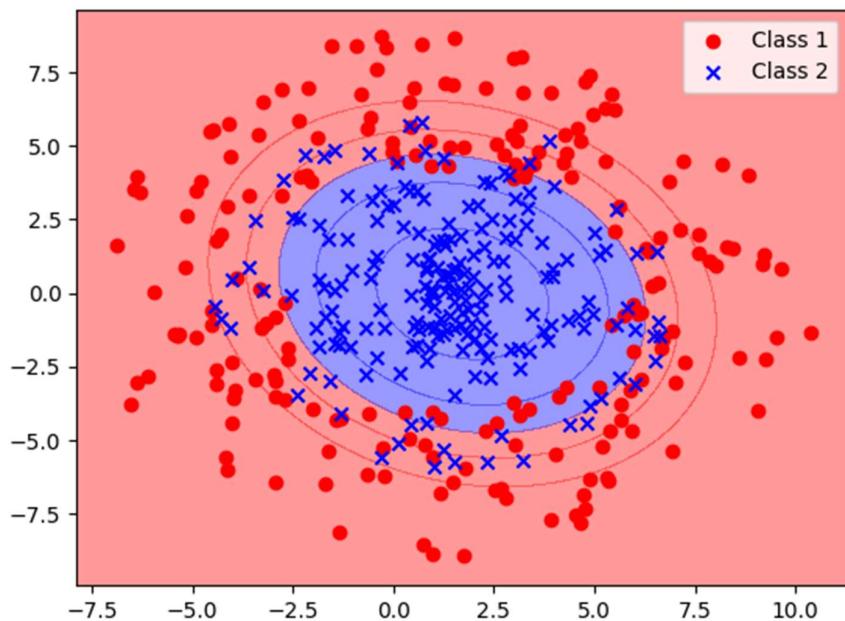
```
# Concatenate data from both classes
X = np.vstack((X1, X2))
y = np.concatenate((np.zeros(100), np.ones(200)))

# Add intercept term to feature matrix
X = np.column_stack((np.ones(300), X))
```

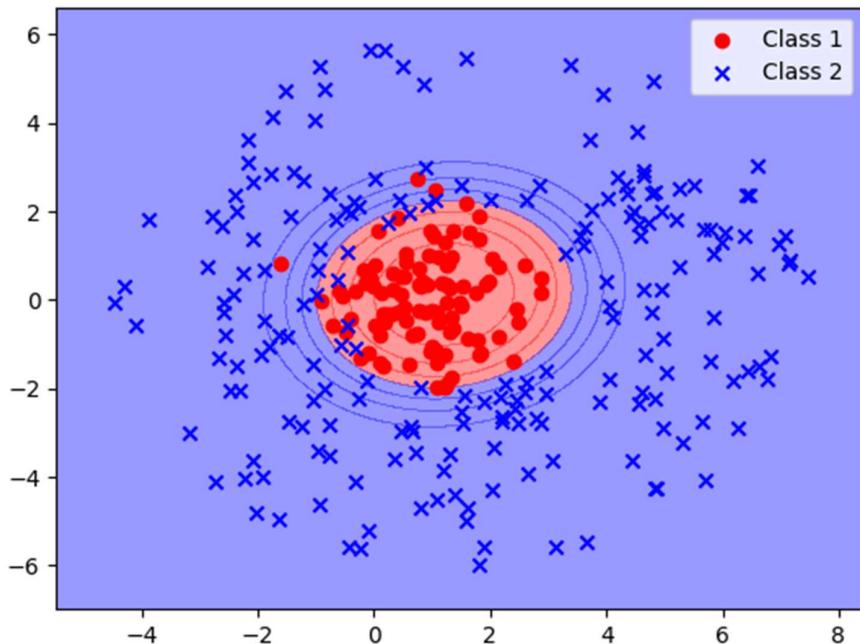
شکل (1-۲۶) تجمعی داده ها در بردار X و y

در نهایت نیز پس از آموزش دقت در حالت اول 0.83 و در حالت دوم 0.9433 میباشد که شکل زیر

مرز تصمیم در هر حالت را در 2 بعد نشان میدهد.



شکل (۱-۲۷) مرز تصمیم در حالت اول



شکل (۱-۲۸) مرز تصمیم در حالت دوم

1-7-4- بررسی بهترین درجه

همانطور که در بخش قبل دیدیم محاسبات و کد ها برای 2 بعد انجام شد.

در اینجا به کمک یک لوپ همان محاسبات را برای هر حالت برای درجات مختلف بررسی کرده و تا بهترین

درجه را بدست بیاوریم.

برای آنکه استفاده از کتابخانه آماده هم انجام شود در این حالت به جای استفاده از تابع نوشته شده در بخش

2 از کتابخانه های آماده استفاده میکیم (لازم به ذکر است در صورتی که نخواهیم از کتاب خانه های آماده

استفاده کنیم میتوانیم از همان تابع به راحتی استفاده کرده و تفاوتی ایجاد نمیشود که کد آن در زیر آمده است)

```
# Define degree range to test
degree_range = range(1, 11)

# Loop over degrees to find best model
best_degree = None
best_accuracy = 0

for degree in degree_range:
    # Create polynomial features up to the specified degree
    poly_features = np.ones((X.shape[0], 1))
    for d in range(1, degree+1):
        poly_features = np.hstack((poly_features, X**d))

    # Standardize features
    poly_features = (poly_features - np.mean(poly_features, axis=0)) / np.std(poly_features, axis=0)

    # Fit logistic regression model with L2 regularization
    w = logistic_regression_l2(poly_features, y, alpha=0.1, num_iterations=1000)

    # Calculate accuracy on training data
    y_pred = sigmoid(poly_features.dot(w)) >= 0.5
    accuracy = np.mean(y_pred == y)

    # Update best model if necessary
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_degree = degree
        best_w = w
```

شکل (1-۲۹) کد پایتون بررسی بهترین درجه بدون استفاده از کتابخانه آماده

```

# Fit logistic regression models with polynomial features of degree 1 to 10
acc = []
for deg in range(1, 11):
    model = make_pipeline(PolynomialFeatures(deg), LogisticRegression(penalty='l2'))
    model.fit(X, y)
    acc.append(model.score(X, y))

# Print best degree and accuracy
best_degree = np.argmax(acc) + 1
best_acc = np.max(acc)
print(f"Best degree: {best_degree}")
print(f"Best accuracy: {best_acc}")

# Plot decision boundary of best model
model = make_pipeline(PolynomialFeatures(best_degree), LogisticRegression(penalty='l2'))
model.fit(X, y)

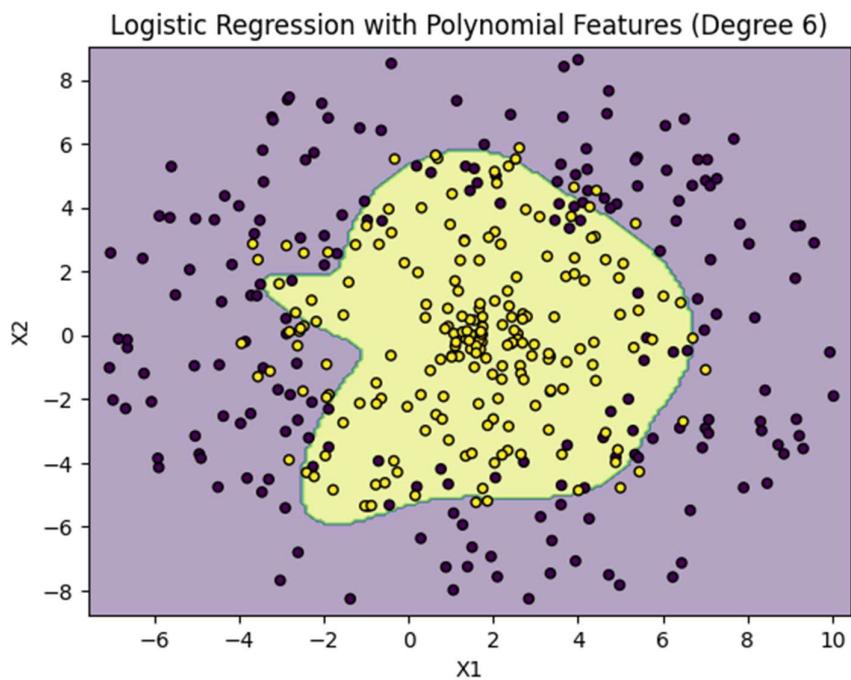
x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                      np.arange(y_min, y_max, 0.1))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolor='k')
plt.title(f"Logistic Regression with Polynomial Features (Degree {best_degree})")
plt.xlabel("X1")
plt.ylabel("X2")
plt.show()

```

شکل (1-۳۰) کد پایتون استفاده از کدهای آماده و رسم و بررسی بهترین درجه

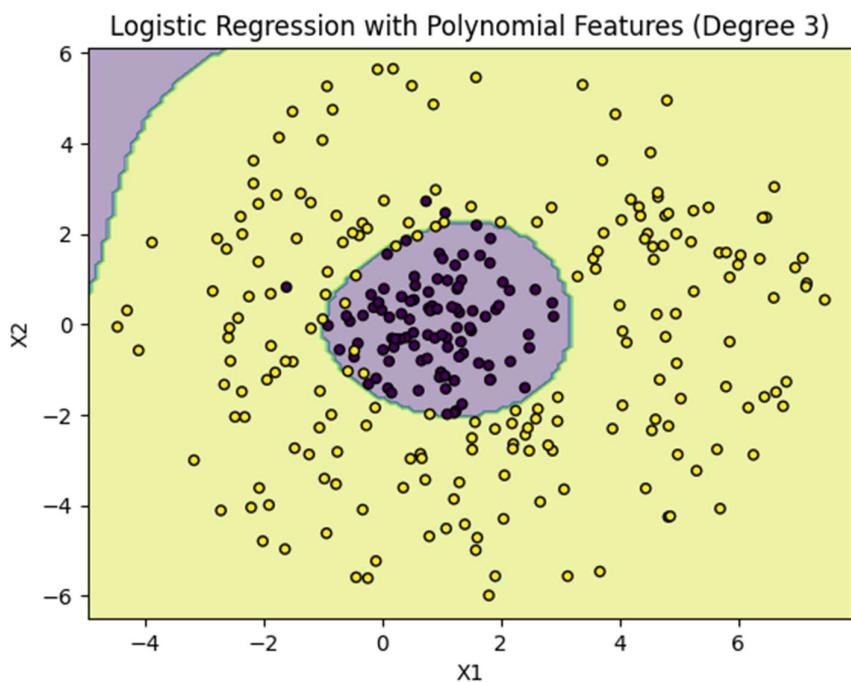
در حالت اول بهترین درجه 6 بوده و مرز تصمیم به صورت زیر میباشد (لازم به ذکر است چون دقت خیلی

بالا نرفته است همان درجه 2 هم کافی بود)



شکل (۱-۳۱) مرز تصمیم درجه ۶ برای حالت اول

در حالت دوم بهترین درجه ۳ بوده و دقت برای آن ۹۶.۶۶ درصد میباشد که مرز آن به صورت زیر میباشد.



شکل (۱-۳۲) مرز تصمیم برای بهترین درجه - حالت دوم

5-7-1- نتیجه گیری

در این کار، ما دو مجموعه داده تولید کردیم که هر کدام دارای دو دسته بودند. برای اولین مجموعه داده‌ها، توزیع‌های گاوی دو بعدی را برای یک دسته تولید کردیم و از نقاط تصادفی روی یک دایره برای دسته دیگر استفاده کردیم. برای مجموعه دوم داده‌ها، توزیع‌های گاوی دو بعدی را برای هر دو دسته ایجاد کردیم. سپس از رگرسیون لجستیک با تنظیم L_2 برای طبقه‌بندی داده‌ها استفاده کردیم. ما درجات مختلفی را برای برآش چند جمله‌ای، از 1 تا 10 امتحان کردیم و بهترین درجه را یافیم که منجر به بالاترین دقت شد. برای اولین مجموعه داده‌ها، متوجه شدیم که درجه 6 برای توزیع گاوی و دایره‌ای بهترین است که منجر به دقت 0.94 شد. برای مجموعه دوم داده‌ها، متوجه شدیم که درجه 3 برای توزیع‌های گاوی بهترین بود، که منجر به دقت 0.96 شد.

به طور کلی، ما دریافتیم که درجه 2 برای هر دو مجموعه داده کافی است، اما استفاده از درجات بالاتر منجر به دقت بالاتری شد. با این حال، توجه به این نکته ضروری است که استفاده از درجات بالاتر نیز می‌تواند منجر به بیش از حد برآش شود، بنابراین باید در انتخاب درجه مناسب برای مشکل مورد نظر دقت کرد.

8-1- سوال : Parzen

الف) مراحل برای این بخش به صورت زیر می‌باشد:

- مجموعه داده TED_talks را از فایل "ted_main.csv" می‌خوانیم
- ستون "مدت" را از مجموعه داده استخراج می‌کنیم
- تابع کرنل گاوی را تعریف می‌کنیم
- تابع پنجره Parzen را تعریف می‌کنیم

• محدوده مقادیر را برای تخمین چگالی تعریف میکنیم

• اندازه پنجره را روی 10 قرار میدهیم.

• از تابع پنجره Parzen برای تخمین چگالی برای هر مقدار در محدوده استفاده میکنیم.

• چگالی تخمینی را با استفاده از matplotlib رسم میکنیم.

ابتدا دیتاست را خوانده و ستون duration را جدا میکنیم.

# Load the TED_talks dataset												
ted_data = pd.read_csv("ted_main.csv")												
ted_data.head()												
	comments	description	duration	event	film_date	languages	main_speaker	name	num_speaker	published_date	ratings	related_talks
0	4553	Sir Ken Robinson makes an entertaining and pro...	1164	TED2006	1140825600	60	Ken Robinson	Ken Robinson: Do schools kill creativity?	1	1151367060	[{"id": 7, "name": "Funny", "count": 19645}, ...]	[{"id": 865, "hero": "https://pe.tedcdn.com/im...}
1	265	With the same humor and humanity he exuded in ...	977	TED2006	1140825600	43	Al Gore	Al Gore: Averting the climate crisis	1	1151367060	[{"id": 7, "name": "Funny", "count": 544}, {"...	[{"id": 243, "hero": "https://pe.tedcdn.com/im...}
		New York Times						David			[{"id": 7, "name": "Funny", "count": 1725, "hero": "https://pe.tedcdn.com/im...}]]	

شکل (۱-۳۳) فرآخوانی دیتاست Ted

```
# Extract the "duration" column
duration = ted_data["duration"].values
```

شکل (۱-۳۴) جداسازی ستون duration

```
# Define the Gaussian kernel function
def gaussian_kernel(u):
    return (1/np.sqrt(2*np.pi))*np.exp(-0.5*u**2)

# Define the Parzen window function
def parzen_window(x, data, h, kernel):
    n = len(data)
    density = 0
    for i in range(n):
        u = (x - data[i])/h
        density += kernel(u)
    density /= (n*h)
    return density
```

شکل (۱-۳۵) تعریف کرنل گوسی

```

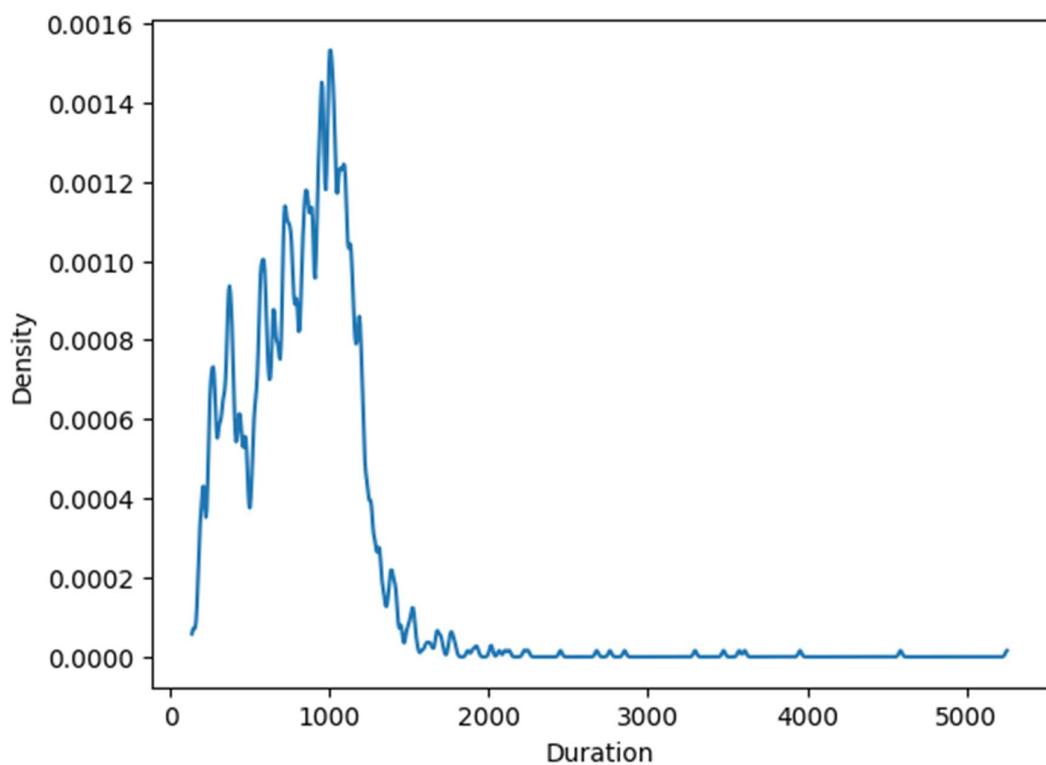
#Define the range of values to estimate the density
x_range = np.linspace(np.min(duration), np.max(duration), num=1000)

# Set the window size to 10
h = 10

# Estimate the density using the Parzen window method with the Gaussian kernel
density = [parzen_window(x, duration, h, gaussian_kernel) for x in x_range]

```

شکل (۱-۳۶) تنظیم پنجره و تعریف رنج مقادیر برای تخمین density



شکل (۱-۳۷) توزیع دیتای ستون duration با پنجره به طول 10

ب) برای این کار از یک loop استفاده میکنیم.

```

# Define the range of values to estimate the density
x_range = np.linspace(np.min(duration), np.max(duration), num=1000)

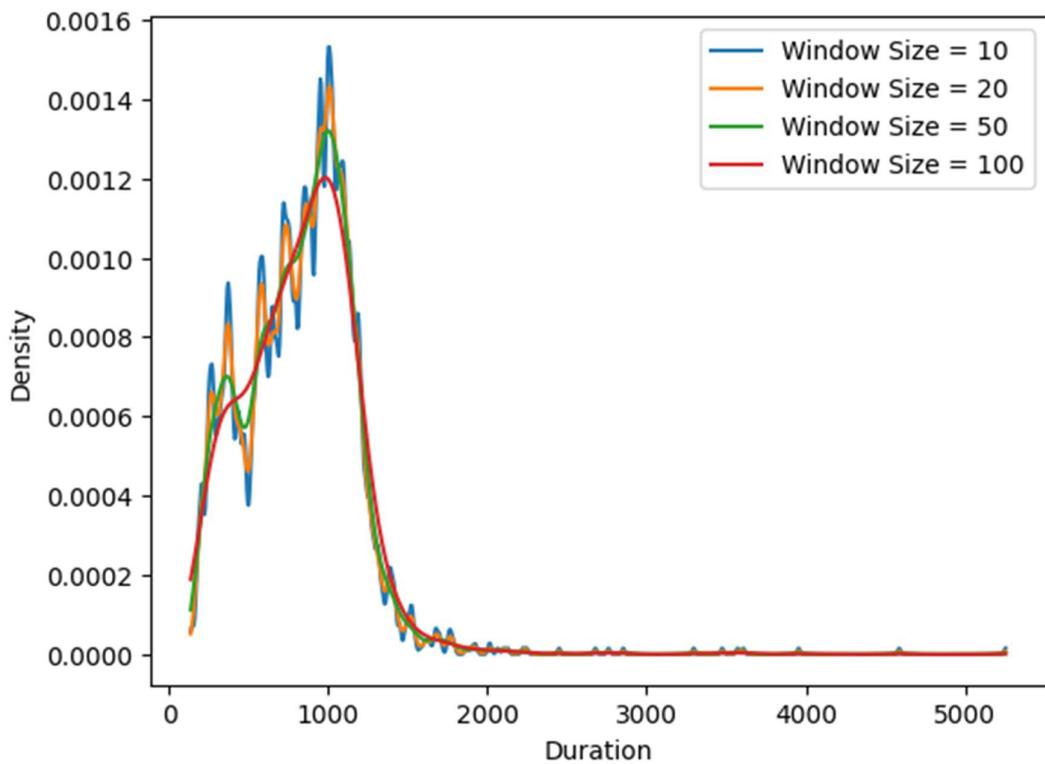
# Set the window sizes to test
h_values = [10, 20, 50, 100]

# Estimate the density using the Parzen window method with the Gaussian kernel
densities = []
for h in h_values:
    density = [parzen_window(x, duration, h, gaussian_kernel) for x in x_range]
    densities.append(density)

# Plot the estimated densities for each window size
plt.plot(x_range, densities[0], label="Window Size = 10")
plt.plot(x_range, densities[1], label="Window Size = 20")
plt.plot(x_range, densities[2], label="Window Size = 50")
plt.plot(x_range, densities[3], label="Window Size = 100")
plt.xlabel("Duration")
plt.ylabel("Density")
plt.legend()
plt.show()

```

شکل (۱-۳۸) رسم توزیع برای پنجره های مختلف



شکل (۱-۳۹) منحنی های توزیع به ازای پنجره های مختلف

نمودار حاصل نشان می دهد که چگونه چگالی تخمینی با اندازه های مختلف پنجره تغییر می کند. ما می توانیم مشاهده کنیم که اندازه های کوچکتر پنجره ها معمولاً تخمین های تراکم تیز تر و پر نوسان تری تولید می کنند، در حالی که اندازه های بزرگتر تمایل دارند تخمین های چگالی نرم تر اما با جزئیات کمتری تولید کنند.

ج) با توجه به توضیحات صورت مساله و کد زیر و استفاده از gaussian_kde مساله را مجدد با کتابخانه های آماده حل می کنیم.

```

# Set the number of data points to use for estimation
n_values = np.linspace(250, len(duration), num=250, dtype=int)

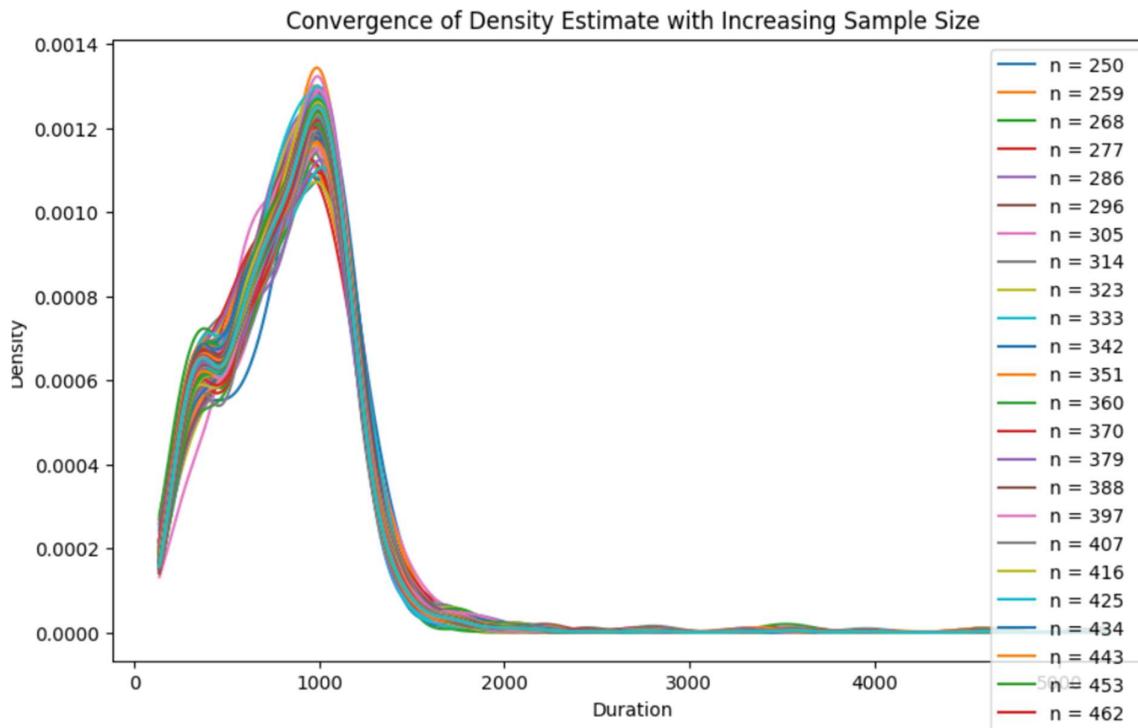
# Initialize an array to store the estimated densities
density_estimates = np.zeros((len(n_values), len(x_range)))

# Estimate the density for each number of data points
for i, n in enumerate(n_values):
    # Select a random sample of n data points from the duration column
    sample = np.random.choice(duration, size=n, replace=False)
    # Calculate the density estimate using Gaussian kernel density estimation
    density_estimates[i] = gaussian_kde(sample)(x_range)

# Plot the estimated densities for each number of data points
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlabel("Duration")
ax.set_ylabel("Density")
ax.set_title("Convergence of Density Estimate with Increasing Sample Size")
for i, n in enumerate(n_values):
    ax.plot(x_range, density_estimates[i], label=f"n = {n}")
ax.legend()
plt.show()

```

شکل (۱-۴۰) کد پایتون تخمین به ازای پنجره های مختلف با کتابخانه های آماده



شکل (۱-۴۱) منحنی همگرایی توزیع با افزایش sample size

نتیجه گیری:

در این بخش ، ما دو روش مختلف را برای تخمین چگالی مجموعه داده های مدت زمان گفتگوی TED

مقایسه کردیم: روش پنجره Parzen با کرnel گاوی و روش تخمین چگالی کرnel گاوی (KDE) از کتابخانه

`scipy.stats`

ابتدا روش پنجره Parzen را با اندازه پنجره 10 پیاده سازی کردیم و چگالی مجموعه داده های مدت زمان

را تخمین زدیم. ما دریافتیم که با افزایش اندازه پنجره، چگالی تخمینی صاف تر و گسترده تر می شود و با کاهش

اندازه پنجره، چگالی تخمینی سیخ و باریک تر می شود. ما همچنین تأثیر تعداد نقاط داده مورد استفاده برای

تخمین را بر روی همگرایی چگالی برآورد شده به چگالی واقعی بررسی کردیم. با این حال، این روش نیاز به

تنظیم دستی اندازه پنجره دارد و عملکرد آن به شدت به مقدار انتخاب شده بستگی دارد.

در مرحله بعد، روش تخمین چگالی هسته گاوی را از کتابخانه `scipy.stats` پیاده سازی کردیم. این روش نیازی به تنظیم دستی اندازه پنجره ندارد و می تواند به طور خودکار با داده ها سازگار شود. ما همچنین تأثیر تعداد نقاط داده مورد استفاده برای تخمین را بروی همگرایی چگالی برآورد شده به چگالی واقعی بررسی کردیم. ما دریافیم که با افزایش تعداد نقاط داده مورد استفاده برای تخمین، چگالی برآورد شده به چگالی واقعی همگرا می شود.

برای مقایسه عملکرد دو روش، نمودارهایی از چگالی های تخمین زده شده را برای اندازه های مختلف نمونه تولید کردیم. ما متوجه شدیم که روش KDE نسبت به روش پنجره Parzen قادر به تخمین چگالی دقیق تر و روان تر است. روش KDE همچنین انعطاف پذیرتر و استفاده آسان تر بود، زیرا نیازی به تنظیم دستی اندازه پنجره نداشت.