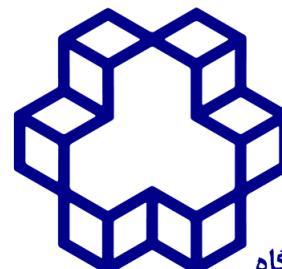


بسم الله الرحمن الرحيم



دانشگاه  
خواجہ نصیرالدین طوسی  
K. N. Toosi University  
of Technology

پاسخ تمرین اول یادگیری ماشین

Google Colab  
GitHub

نگارش: علی شعبانپور مقدم - هدیه شوشیان

شماره دانشجویی: ۴۰۲۰۷۳۰۴-۴۰۳۰۸۰۵۴

استاد درس: دکتر مهدی علیاری شوره دلی

زمستان ۱۴۰۳

## فهرست مطالب

۳	۱	مقداری جبر خطی
۳	۱.۱	ضرب ابعاد
۴	۲.۱	ضرب و ابعاد دشوارتر
۴	۲	کمی آمار و احتمالات
۴	۱.۲	مقدمه
۵	۲.۲	سناریو
۵	۳.۲	آ) پرده اول
۵	۴.۲	ب) پرده دوم
۶	۵.۲	ج) پرده سوم
۷	۳	پردازش داده
۷	۱.۳	CWRU Dataset
۱۵	۲.۳	بخش دوم : داده گل زنبق



## ۱ مقداری جبر خطی

### ۱.۱ ضرب ابعاد

آ) در صورتی که ضرب دو ماتریس امکان پذیر باشد، ابعاد ماتریس حاصل ضرب برابر تعداد سطر های ماتریس اول در تعداد ستون های ماتریس دوم می باشد.  
همچنین با ترانهاده کردن ماتریس جای سطر و ستون های ماتریس تغییر میابد پس ابعاد ماتریس های ترانهاده به طوری است که سطرهای ماتریس ترانهاده برابر تعداد ستون های ماتریس قبلی و تعداد ستون های آن برابر با تعداد سطر های ماتریس قبلی است. اگر تعداد ستون های ماتریس اول با تعداد سطر های ماتریس دوم برابر نباشد ضرب ماتریس ها امکان پذیر نخواهد بود.

$$A_{p \times q} * B_{q \times l} = AB_{p \times l}$$

$$B_{q \times l} * A_{p \times m} = \text{error}$$

با فرض ماتریس های  $A$  و  $B$  به صورت زیر:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}_{2 \times 3}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix}_{4 \times 2}$$

آنگاه خواهیم داشت:

$$BA : 4 \times 3$$

$$B^T : 2 \times 4$$

$$B^T A : \text{error}$$

$$A^T B : \text{error}$$

(ب)

$$BA = \begin{bmatrix} a_{11}b_{11} + a_{21}b_{12} & a_{12}b_{11} + a_{22}b_{12} & a_{13}b_{11} + a_{23}b_{12} \\ a_{11}b_{21} + a_{21}b_{22} & a_{12}b_{21} + a_{22}b_{22} & a_{13}b_{21} + a_{23}b_{22} \\ a_{11}b_{31} + a_{21}b_{32} & a_{12}b_{31} + a_{22}b_{32} & a_{13}b_{31} + a_{23}b_{32} \\ a_{11}b_{41} + a_{21}b_{42} & a_{12}b_{41} + a_{22}b_{42} & a_{13}b_{41} + a_{23}b_{42} \end{bmatrix}$$

$$B^T = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} \\ b_{12} & b_{22} & b_{32} & b_{42} \end{bmatrix} \quad B^T A = \text{error} \quad A^T B = \text{error}$$



## ۲.۱ ضرب و ابعاد دشوارتر

آ) در صورتی که  $x$  توصیفی مطابق صورت سوال داشته باشد :

$$x \in \mathbb{R}^{1 \times 2}$$

و  $\theta$  نیز بصورت زیر تعریف گردد:

$$\theta \in \mathbb{R}^{2 \times 1}$$

حاصل ضرب آن ها بصورت زیر خواهد بود:

$$x\theta \in \mathbb{R}^{1 \times 1}$$

که برابر با یک مقدار عددی می باشد. اگر فرض کنیم ماتریس  $X$  تعریف زیر را داشته باشد:

$$X \in \mathbb{R}^{n \times 2}$$

حاصل ضرب  $X\theta$  به صورت زیر تعریف می شود:

$$X\theta \in \mathbb{R}^{n \times 1}$$

ب) مربع بردار بصورت ضرب خودش در ترانهاده اش در نظر گرفته می شود پس ضرب پرانتز ها داریم:

$$(X\theta - Y)^T(X\theta - Y) = (\theta^T X^T - Y^T)(X\theta - Y) = \theta^T X^T X\theta - \theta^T X^T Y - Y^T X\theta + Y^T Y$$

وارد کردن ترانهاده در پرانتز باعث تعویض جای  $x$  و  $\theta$  می گردد. از عبارت فوق بر حسب  $\theta$  مشتق می گیریم بدست می آید:

$$\nabla J_\theta = 2X^T X\theta - 2X^T Y$$

## ۲ کمی آمار و احتمالات

### ۱.۲ مقدمه

قضیه بیز ، احتمال اتفاق افتادن یک پیشامد را به شرط وقوع یک پیشامد دیگر با احتمالی غیر صفر از طریق فرمول زیر محاسبه می کند. در یادگیری ماشین معمولاً بدنبال بهترین فرضیه‌ای هستیم که در مورد داده‌های آموزشی صدق کند. یک راه تعیین بهترین فرضیه، این است که بدنبال محتمل‌ترین فرضیه‌ای باشیم که با داشتن داده‌های آموزشی و احتمال قبلی در مورد فرضیه‌های مختلف می‌توان انتظار داشت تئوری بیز چنین راه حلی را ارائه می‌دهد. این فرمول از قانون احتمال کل نتیجه گیری می شود. در فرمول زیر احتمال وقوع پیشامد اولی به شرط وقوع پیشامد دومی برابر است با نسبت احتمال وقوع آن پیشامد دوم به شرط وقوع پیشامد اول ضرب در احتمال وقوع پیشامد اول به احتمال وقوع پیشامد دوم:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

ارزش قضیه بیز زمانی خود را نشان می‌دهد که محاسبه احتمال  $P(A|B)$  از محاسبه  $P(B|A)$  راحت‌تر باشد و احتمالات  $P(A)$  و  $P(B)$  در دسترس باشند. آنگاه احتمال  $P(A|B)$  به سادگی قابل محاسبه است. احتمال  $P(B)$  معمولاً از قانون احتمال کل بدست می‌آید.



## ۲.۲ سناریو

## ۳.۲ آ) پرده اول

با توجه به قضیه بیز می‌توان رابطه زیر را نوشت:

$$P(sick|+) = \frac{P(+|sick)P(sick)}{P(+)}$$

در معادله بالا  $P(+)$  در دسترس نمی‌باشد. این احتمال مثبت آمدن جواب آزمایش است چه واقعاً فرد بیمار باشد چه به اشتباه بیمار تشخیص داده شده باشد. باید از قانون احتمال کل برای بدست آوردن آن استفاده کنیم.

$$P(+) = P(+|sick)P(sick) + P(+|not sick)P(not sick) = \frac{99}{100} \frac{1}{10^4} + \frac{1}{100} \frac{9999}{10^4} = 0.010098$$

حال معادله قبل را تکمیل می‌کنیم:

$$P(sick|+) = \frac{P(+|sick)P(sick)}{P(+)} = \frac{\frac{99}{100} \frac{1}{10^4}}{0.010098} = 0.00980392$$

عمو محمد توضیح می‌دهد چون آزمایش علی مثبت آمده پس برای بررسی آن که آیا او واقعاً این بیماری را دارد یا نه باید احتمال وقوع مريض بودن آن در احتمال مثبت آمدن آزمایش زمانی که علی واقعاً مريض است ضرب و بر احتمال مثبت آمدن آزمایش تقسیم گردد. این عدد بسیار کوچک است زیرا نه تنها بیماری علی بسیار نادر است بلکه احتمال آنکه آزمایش نیز درست تشخیص داده باشد بسیار کم است.

## ۴.۲ ب) پرده دوم

عمو محمد برای تشخیص قطعی تر آزمایشگاه دیگری که صحت بالاتر دارد توصیه می‌کند. این کار باعث کاهش شدی احتمال تشخیص خطای آزمایش علی می‌گردد. از قضیه بیز مجدداً داریم:

$$P(sick|++) = \frac{P(+|sick)P(sick|+)}{P(++)}$$

از قبل داریم:

$$P(sick|+) = 0.00980392$$

برای تست دوم داریم:

$$P(+|Sick) = 0.9999$$

$$P(+|not Sick) = 1 - 0.9999 = 0.0001$$

$$P(not sick|+) = 1 - 0.00980392 = 0.99019608$$

حال احتمال  $P(+)|(+)$ 

$$P(++) = P(+)|sick)P(sick|+) + P(+)|notsick)P(not sick|+)$$

مطابق قانون احتمال کل، احتمال اینکه نتیجه تست دوم مثبت باشد در حالی که می‌دانیم نتیجه تست اول مثبت شده است دو حالت دارد: یا شخص مريض بوده و تست مثبت شده است که در اين صورت احتمال مريض بودن شخص در صورت مثبت بودن نتیجه تست اول باید در يك احتمال شرطی ضرب شود. در حالت دوم شخص مريض نبوده و تست دوم او مثبت شده است: در اين حالت نيز احتمال مريض نبودن او در صورت مثبت بودن نتیجه تست اول باید در يك احتمال شرطی ضرب شود نتیجه معادله بالا را می‌نویسیم:

$$P(++) = 0.9999 \times 0.009803 + 0.0001 \times 0.990196 = 0.0099019592$$

در نهايّت احتمال مطلوب سوال را بررسی می‌کنيم:

$$P(sick|++) = \frac{0.9999 \times 0.009803}{0.0099019592} = 0.9899999 \approx 0.99$$

## ۵.۲ ج) پرده سوم

با استفاده مجدد از قانون بیز داریم

$$P(sick|++-) = \frac{P(-|sick)P(sick|++)}{P(-|++)}$$

$$P(-|sick) = 0.999999$$

$$P(sick|++) = 0.99$$

احتمال  $P(-|++)$  باید با استفاده از قانون احتمال کل بدست آید.

$$P(-|++) = P(-|sick)P(sick|++) + P(-|not sick)P(not sick|++)$$

$$P(-|++) = (1 - 0.999999)(0.9899999) + (0.999999)(1 - 0.9899999) = 0.0100009819999959$$

$$P(sick|++-) = \frac{0.000001(0.989999998)}{0.010000981999995981} = 9.8990278954646e - 05$$

مشاهده می‌شود از آنجایی که دوبار آزمایش صورت گرفته است و هر کدام با صحت بالایی بیمار بودن علی را تایید کرده اند حتی با وجود آنکه آزمایش سوم بیمار نبودن علی را با صحتی دقیق تر از دو صحت آزمایش قبلی نشان می‌دهد اما در مجموع احتمال بیمار نبودن آن تقریباً به صفر میل می‌کند.



### ۳ پردازش داده

#### CWRU Dataset ۱.۲

(آ) دریافت داده ۱. به لینک داده شده در صورت سوال مراجعه کرده و دیتابست ۱۰۹ با کد ۰ – *IR007* را انتخاب می نماییم. فرمت این فایل *.mat* است. این فایل در نرم افزار متلب ایجاد گردیده است برای فرآخوانی آن در پایتون از کتابخانه *Scipy* و دستور *scipy.io.loadmat* استفاده می کنیم. برای گوگل کولب با توجه به آنکه فایل ها را از خارج از لپتاپ وارد می کنیم از دستور *gdown* بهره می بریم و آن را از گوگل درایو فرآخوانی می کنیم. ۲. فایل را خوانده و در متغیری به نام *data* ذخیره می کنیم. سپس با دستور *type* نوع متغیر را بررسی می کنیم. آیتم های دیکشنری را در جدولی ذخیره می کنیم. نوع داده *dict* است و دارای کلید های دیکشنری نام برده است. *header*, *version*, *globals*, *X109-DE-time*, *X109-FE-time*, *X109RPM* این دیتابست به نوعی یک کلاس است و دارای دو سیگنال از نوع آرایه که هر کدام شامل ۲۴۳۹۳۸ عدد اعشاری است و یک متغیر دیگر با نام *values* با نوع *unit16* است. نوع داده در متلب *structure* نمایش داده شده است. یک متال برای داده با نوع *dict* را می توان اسم، سن، قد و جنسیت را نام برد و کلید های آن را محسن، ۲۵، ۱۷۰ در نظر گرفت.

```
[ ] import scipy
data = scipy.io.loadmat('109.mat')

from IPython.display import display, HTML
# Get the data type
data_type = type(data).__name__

# Convert dictionary items to an HTML table
table_rows = "".join([f"<tr><td>{key}</td><td>{value}</td></tr>" for key, value in data.items()])

# Display everything in a structured format
display(HTML(f"""
<p style="color:red; font-size:20px;"><b>Data Type:</b> {data_type}</p>
<p style="color:lime; font-size:20px;"><b>Dictionary Items:</b></p>
<table border="1" style="border-collapse: collapse; font-size:18px;">
    <tr style="background-color:green;"><th>Key</th><th>Value</th></tr>
    {table_rows}
</table>
"""))

```

شکل ۱: خواندن فایل

Data Type: dict	
Dictionary Items:	
Key	Value
<i>_header_</i>	b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Mon Jan 31 15:28:48 2000'
<i>_version_</i>	1.0
<i>_globals_</i>	[]
<i>X109_DE_time</i>	[[ 0.010016 ] [-0.023788] [-0.00792933] ... [-0.085136] [-0.05800933] [ 0.00542533]]
<i>X109_FE_time</i>	[[ -0.35757879] [-0.24455101] [-0.1870096] ... [-1.17137879] [-1.11630343] [-0.98930131]]
<i>X109RPM</i>	[[1796]]

شکل ۲: نوع فایل و نمایش آن



۳. سیگنال  $X109 - DE - time$  که یکی از سیگنال های در دیتابست بود را به عنوان متغیر  $vib - DE$  انتخاب می کنیم. این سیگنال سرعت داخلی است که برای اندازه گیری لرزش استفاده می شود.

```
[ ] import numpy as np
      import pandas as pd
      # Extract the vibration signal
      vib_DE = data['X109_DE_time']
```

شکل ۳: انتخاب سیگنال لرزش

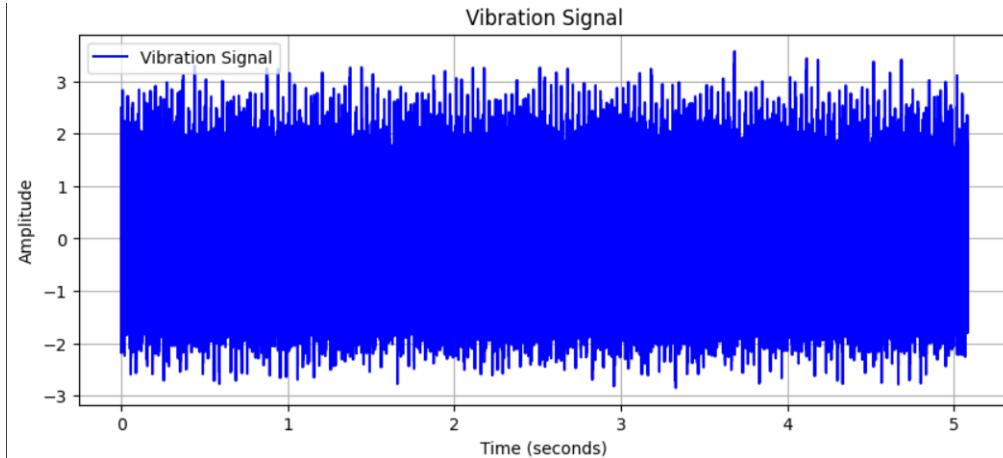
## ب) نمایش سیگنال

۱. نرخ نمونه برداری 48000 هرتز است. تعداد نمونه ها 243938 تا است. زمان نمونه برداری برابر عکس مقدار فرکانس برابر با 0.02083 میلی ثانیه است. با دستور *matplotlib* نمودار افقی را زمان و نمودار عمودی را دامنه سیگنال تعریف می کنیم. با توجه به آنکه تعداد نمونه داریم زمان کل نمونه برداری 0.082 ثانیه است. شکل خروجی را با دستور *plt.plot* ابتدا آن را تعریف و با دستور *plt.show* آن را نمایش می دهیم.

```
[ ] import matplotlib.pyplot as plt
# Define time axis (full signal)
sampling_rate = 48000 # Given sampling frequency (48 kHz)
time_full = np.arange(0, len(vib_DE)) / sampling_rate # Convert samples to time in seconds
fs=48000 #hz
ts=1/fs

# Plot the full signal
plt.figure(figsize=(10, 4))
plt.plot(time_full, vib_DE, label="Vibration Signal", color='b')
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.title("Vibration Signal")
plt.legend()
plt.grid()
plt.show()
```

شکل ۴: کد نمایش سیگنال



شکل ۵: سیگنال لرزش



۲. برای ترسیم سیگنال از زمان ۰.۰۱ تا ۰.۰۲ ثانیه درواقع باید نمونه های ۹۶۰۰۰ تا ۹۶۴۸۰ را ترسیم نماییم. برای این کار محور افقی زمان ۰.۰۱ تا ۰.۰۲ ثانیه خواهد بود و نمودار عمودی دامنه های سیگنال به ازای این زمان ها است.

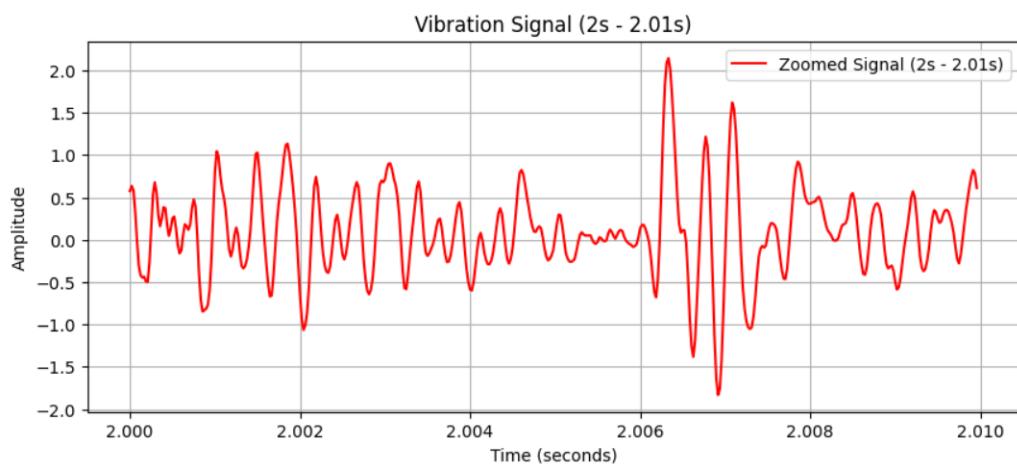
```
# Define time range for zoom-in (2s to 2.01s)
start_time = 2 # Start at 2s
end_time = 2.01 # End at 2.01s

# Find the corresponding indices
start_index = int(start_time * sampling_rate)
end_index = int(end_time * sampling_rate)

# Extract the portion of signal within the given time range
time_zoom = time_full[start_index:end_index]
vib_zoom = vib_DE[start_index:end_index]

# Plot the zoomed-in signal
plt.figure(figsize=(10, 4))
plt.plot(time_zoom, vib_zoom, label="Zoomed Signal (2s - 2.01s)", color='r')
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.title("Vibration Signal (2s - 2.01s)")
plt.legend()
plt.grid()
plt.show()
```

شکل ۶: کد نمایش سیگنال



شکل ۷: سیگنال لرزش در بازه مطلوب



ج) تحلیل فرکانسی ۱. با دستور `fft` طیف سیگنال فرکانسی را به ازای نرخ نمونه برداری بدهست می‌آوریم. دقت داریم که طیف فرکانسی قرینه است. و پدیده‌ی الیاسینگ نیز رخ می‌دهد و این پدیده به علت نمونه برداری رخ می‌دهد و کیفیت سیگنال را کاهش می‌دهد. در اینجا با دستور `abs` اندازه‌ی طیف فرکانسی را درنظر گرفته‌یم می‌توان فاز آن را نیز در نظر گرفت.

```
def frequency_spectrum(signal, fs):
    N = len(signal) # Number of samples

    # Compute FFT
    fft_values = np.fft.fft(signal)
    fft_magnitude = np.abs(fft_values) # Magnitude of FFT
    # Extract only a single value from the array

    # Print with proper formatting
    freqs = np.fft.fftfreq(N, d=1/fs) # Frequency bins (including negative freqs)
    dominant_magnitude = float(fft_magnitude[np.argmax(fft_magnitude)])
    # Find dominant frequency (ignoring negative sign)
    dominant_freq = freqs[np.argmax(fft_magnitude)]

    # Plot the full FFT spectrum
    plt.figure(figsize=(8, 4))
    plt.plot(freqs, fft_magnitude, label="FFT Spectrum")
    plt.axvline(dominant_freq, color='r', linestyle='--', label=f"Dominant Freq: {dominant_freq:.2f} Hz")
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Magnitude")
    plt.title("Full Frequency Spectrum")
    plt.legend(loc='lower right')
    plt.grid()
    plt.show()

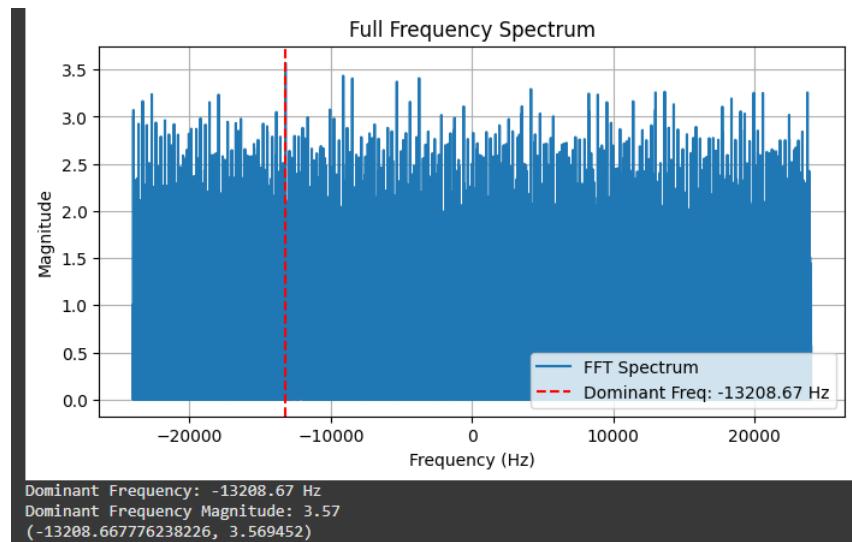
    # Print dominant frequency and its magnitude
    print(f"Dominant Frequency: {dominant_freq:.2f} Hz")
    print(f"Dominant Frequency Magnitude: {dominant_magnitude:.2f}")

    return dominant_freq, dominant_magnitude #you can add these too: freqs, fft_magnitude

# Run the function with a real signal
frequency_spectrum(vib_DE, fs)
```

شکل ۸: تابع نمایش طیف فرکانسی

۲. فرکانس غالب را می‌توان با دستور `fft.freq` می‌یابیم. این مقدار برابر با ۱۳۲۰۸.۶۷ و ۳.۵۷ هرتز است.



شکل ۹: فرکانس غالب روی طیف



۵) تقسیم بندی سیگنال با فراخوانی کتابخانه *numpy* و تعریفتابع *segment – signal* سیگنال دیتاست را به قطعه هایی با ۱۲۸ قسمت تقسیم می کنیم.

با استفاده از کد زیر داده ها را به چند مجموعه ۱۰ تایی با همپوشانی ۱۰ سیگنال تقسیم می کنیم، سپس در حلقه ای ابتدا اعداد ۱ تا ۱۰ را انتخاب و مضارب ۱۳ ان ها را انتخاب و به عنوان قطعه های منتخب نمایش میدهیم، اما برای اینکه منظور سوال اشتباہ برداشت نشده باشد یکبار نیز با اعداد تصادفی که مضارب ۱۳ هستند، ۱۰ نمونه انتخاب و نمایش میدهیم، در کد زیر این مراحل طی شده است:

```
[ ] def segment_signal(signal, segment_size=128, overlap=10):
    """
    Splits the signal into overlapping segments and stores them in a NumPy array.

    Parameters:
        signal (numpy array): Input signal.
        segment_size (int): Size of each segment (default 128).
        overlap (int): Overlap between segments (default 64).

    Returns:
        numpy array: Segmented signal (each row is a segment).
    """
    step = segment_size - overlap # Compute the step size
    num_segments = (len(signal)) - overlap // step # Calculate number of segments

    # Create an array to store segments
    segments = np.array([signal[i : i + segment_size] for i in range(0, len(signal) - segment_size + 1, step)])
    segmented_signal = np.squeeze(segments) # Removes extra dimensions
    return segmented_signal

segmented_signal = segment_signal(vib_DE, segment_size=128, overlap=10)
print(segmented_signal.shape) # (number_of_segments, 128)
```

شکل ۱۰: قطعه بندی سیگنال و تشکیل تابع

```
[ ] ##e.1
# Convert to DataFrame
df = pd.DataFrame(segmented_signal)
df

##e.2
plt.figure(figsize=(10, 5))

for idx in [i * 13 for i in range(1, 11) if i * 13 < len(df)]: # Select & filter indices in one step
    plt.plot(df.iloc[idx], label=f'Segment {idx}', linewidth=1.5)

# Final formatting
plt.xlabel("Sample Index"), plt.ylabel("Amplitude"), plt.title("Signal Segments (Multiples of 13)")
plt.legend(loc="upper right"), plt.grid(), plt.show()

# Randomly choose 10 unique numbers to multiply by 13
num_segments = 10
max_index = len(segmented_signal) // 13 # Ensure we stay within bounds
random_numbers = np.random.choice(range(1, max_index), num_segments, replace=False) # Unique numbers
selected_indices = [13 * num for num in random_numbers] # Multiply by 13

# Ensure indices are within range
selected_indices = [idx for idx in selected_indices if idx < len(segmented_signal)]

# Plot the selected segments
plt.figure(figsize=(10, 5))

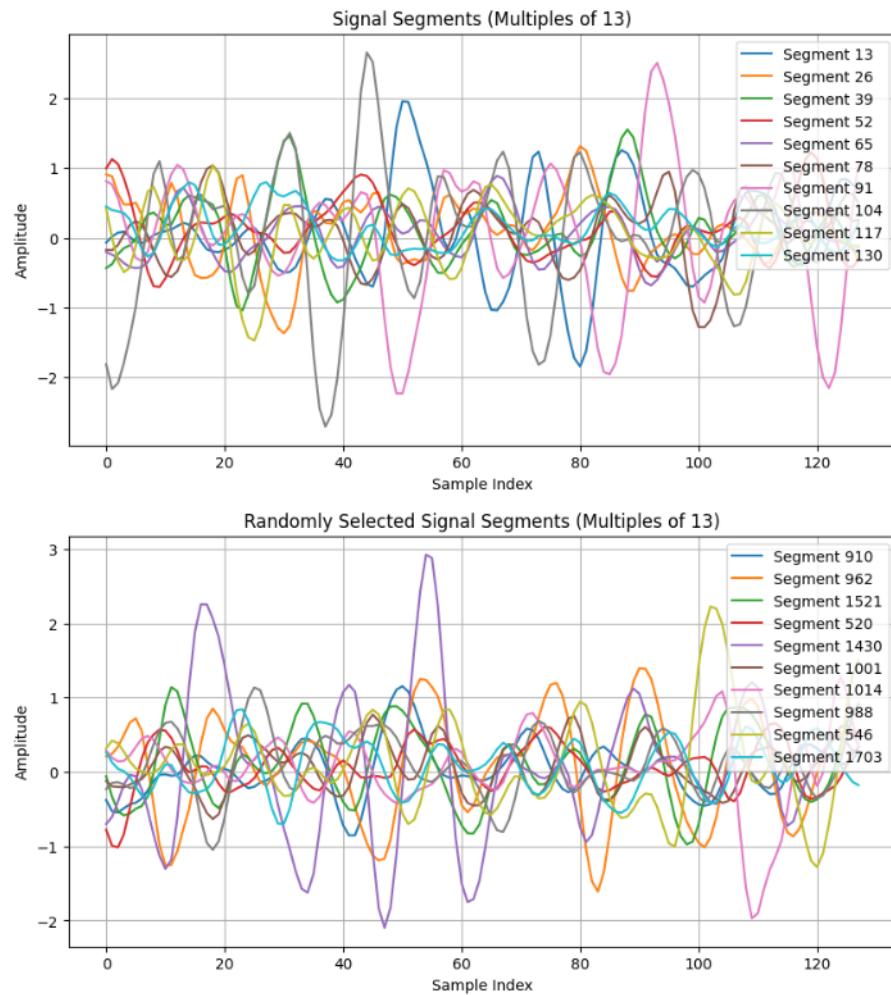
for i, idx in enumerate(selected_indices):
    plt.plot(segmented_signal[idx], label=f'Segment {idx}', linewidth=1.5)

plt.xlabel("Sample Index")
plt.ylabel("Amplitude")
plt.title("Randomly Selected Signal Segments (Multiples of 13)")
plt.legend(loc="upper right")
plt.grid()
plt.show()
```

شکل ۱۱: ذخیره در دیتا فریم و نمایش مضارب ۱۳



نتیجه عملیات های بالا به صورت تصویر به شکل زیر در می آید.



شکل ۱۲: نمایش سگمنت های مضرب ۱۳



## و) استخراج ویژگی

۱.

در این بخش برای استخراج ویژگی ابتدا یک تابع کلی نوشته شده است که میانگین، انحراف معیار و مقدار RMS هر سیگنال را بدست می‌آورد.

۲ و ۳.

سپس این تابع را به مجموع داده‌ها اعمال می‌کنیم تا این معیارها را برای آنها بدست آوریم. در خط اول کد زیر برای هر یک از سگمنت‌های ۱۲۸ تابی هر کدام از ویژگی‌های آماری خواسته شده در صورت سوال محاسبه شده است و با حفظ نام ستون‌ها در یک دیتابریم جدید قرار گرفته است. سپس با استفاده از دستور `to_csv` به فایل CSV شده است.

```
[ ] #f.1
# Step 1: Define a function to compute signal statistics
def compute_signal_features(signal):
    """
    Computes mean, standard deviation, and root mean square (RMS) for a given signal.
    """
    mean = np.mean(signal)                      # Compute Mean
    std_dev = np.std(signal)                    # Compute Standard Deviation
    rms = np.sqrt(np.mean(signal ** 2))        # Compute Root Mean Square (RMS)

    return mean, std_dev, rms
#f.2
# Step 2: Compute features for each segment
results = [compute_signal_features(segment) for segment in segmented_signal]

# Step 3: Convert results into a DataFrame
df_stat = pd.DataFrame(results, columns=["Mean", "Standard Deviation", "RMS"])

#f.3
# Step 4: Save DataFrame to csv
csv_filename = "segmented_signal_statistics.csv"
df_stat.to_csv(csv_filename, index=False)

# Step 5: Display DataFrame in Colab
display(df_stat)
print(f"CSV file '{csv_filename}' saved successfully!")
# Step 6: Provide download link
from google.colab import files
files.download(csv_filename)
```

شکل ۱۳: تابع استخراج ویژگی‌ها و ذخیره آن

	Mean	Standard Deviation	RMS
0	0.093287	0.679569	0.685942
1	0.057207	0.697064	0.699408
2	0.080653	0.366588	0.375356
3	0.057540	0.872134	0.874030
4	0.089198	0.335796	0.347441
...	...	...	...
2062	0.052388	0.811131	0.812821
2063	0.032542	0.354657	0.356147
2064	0.077001	0.528279	0.533861
2065	0.039924	0.439399	0.441209
2066	0.050080	0.701509	0.703294
2067 rows × 3 columns			
CSV file 'segmented_signal_statistics.csv' saved successfully!			

شکل ۱۴: نتیجه مختصر فایل حاصله



## ۲.۳ بخش دوم : داده گل زنبق

(۱)

این داده‌ها برای دسته‌بندی سه گونه مختلف از گل Iris (زنبق) استفاده می‌شود.

### اجزای مجموعه داده Iris

مجموعه داده زنبق شامل ۱۵۰ نمونه از گل‌های زنبق است که در سه کلاس زیر دسته‌بندی شده‌اند:

Iris Setosa

Iris Versicolor

Iris Virginica

هر نمونه دارای ۴ ویژگی عددی است که اندازه‌های مختلف گل را نشان می‌دهند:

petal width (cm) petal length (cm) sepal width (cm) sepal length (cm)

همچنین یک ستون برچسب (کلاس) وجود دارد که نشان می‌دهد نمونه مربوط به کدام یک از سه گونه‌ی زنبق است.



در کد زیر ابتدا داده از کتابخانه مربوطه فراخوانی می شود سپس با دستورات مقتضی تبدیل به داده های آموزش و تست می گردد، سپس مطابق صورت این داده ها در یک دیتابریم با هم ادغام می شوند هر سطر شامل ویژگی هایی از این گلبرگ هاست.

تمام بند های این بخش در مجموعه کد های زیر اجام شده است:

```
[ ] import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[ ] dataset = datasets.load_iris()
Iris_data = pd.DataFrame(dataset.data, columns=dataset.feature_names)
Iris_data['species'] = dataset.target
#Dividing Data to train and test:
train_data, test_data = train_test_split(Iris_data, test_size=0.2, random_state=4)
#Adding Label:
train_data['dataset'] = 'train'
test_data['dataset'] = 'test'
#Concatenating data:
df = pd.concat([train_data, test_data])
df.head()
```

شکل ۱۵: فراخوانی، تقسیم داده، بر جسب گذاری و ادغام داده زنبق

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	dataset
84	5.4	3.0	4.5	1.5	1 train
47	4.6	3.2	1.4	0.2	0 train
108	6.7	2.5	5.8	1.8	2 train
1	4.9	3.0	1.4	0.2	0 train
93	5.0	2.3	3.3	1.0	1 train

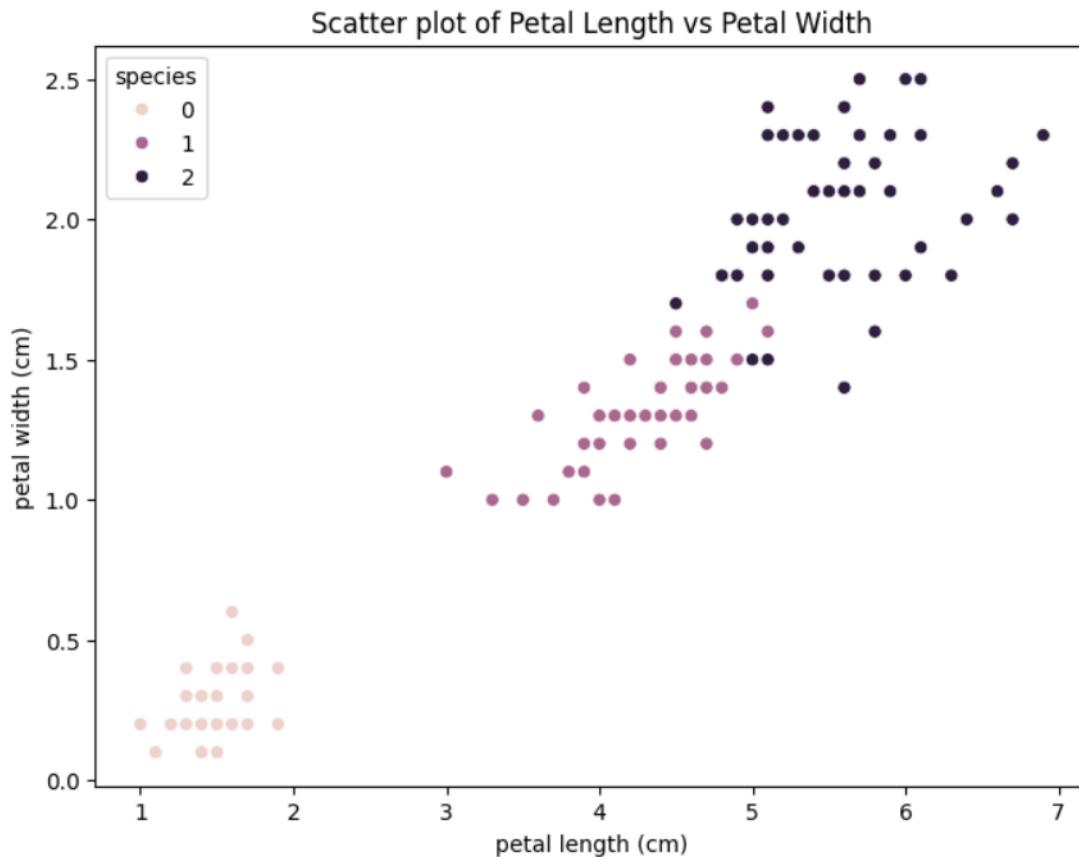
شکل ۱۶: نتیجه مختصر فایل حاصله



ب) در بخش بعد ابتدا پرآندگی دو بعدی (به ازای دو ویژگی طول گلبرگ و عرض گلبرگ) را بررسی کرده ایم. کد های زیر این موضوع را بررسی می کنند:

```
1 plt.figure(figsize=(8, 6))
2 sns.scatterplot(x=df['petal length (cm)'], y=df['petal width (cm)'], hue=df['species'])
3 plt.title('Scatter plot of Petal Length vs Petal Width')
4 plt.show()
```

در شکل زیر این پرآندگی نمایش داده شده است.

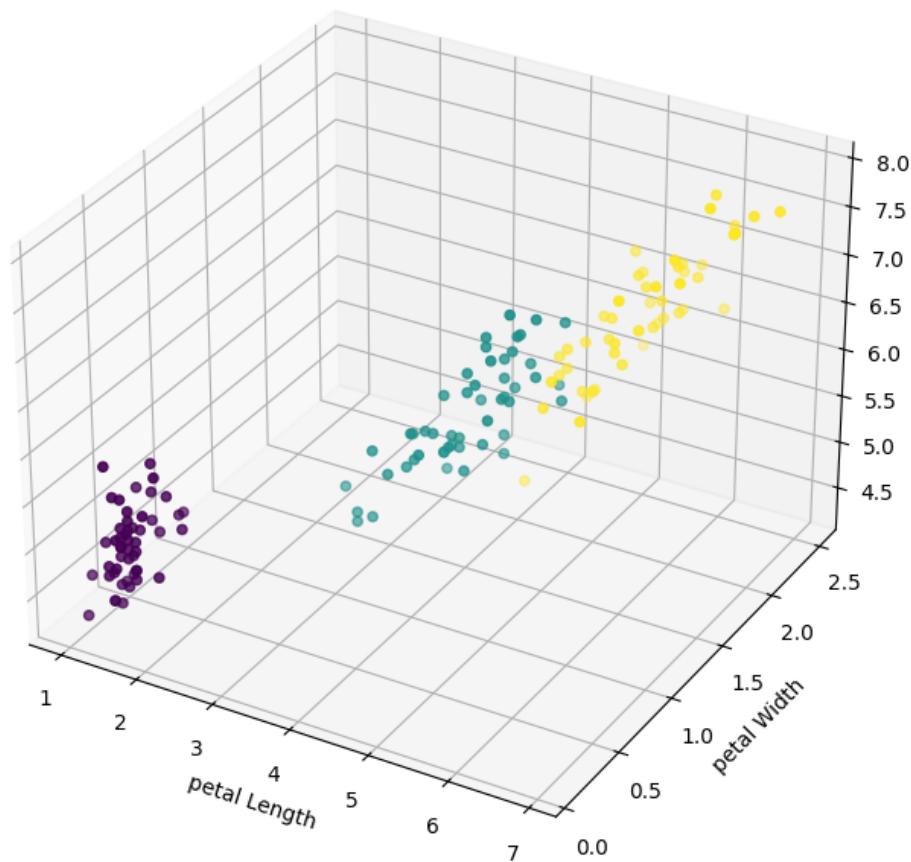


شکل ۱۷: نمایش پرآندگی دو بعدی

در بخش بعدی پرآندگی سه بعدی (برای سه ویژگی طول گلبرگ، عرض گلبرگ و طول کاسبرگ) ترسیم گردیده است. کد زیر نمایش دهنده این تقسیم بندی سه بعدی است.

```
1 fig = plt.figure(figsize=(8, 8))
2 ax = fig.add_subplot(111, projection='3d')
3 ax.scatter(df['petal length (cm)'], df['petal width (cm)'], df['sepal length (cm)'], c=df['species'])
4 ax.set_xlabel('petal Length')
5 ax.set_ylabel('petal Width')
6 ax.set_zlabel('sepal Length')
7 plt.title('3D Scatter plot of three features')
8 plt.show()
```

3D Scatter plot of three features



شکل ۱۸: نمایش پرآندازی سه بعدی



در ادامه با استفاده از متدهای آماده کتابخانه seaborn نقشه حرارتی ترسیم شده است وتابع چگالی احتمال تخمین زده شده است.  
کد های مربوطه در زیر نمایش داده شده است:

```
[ ] # Compute correlation matrix
corr_matrix = df.drop(columns=['species', 'dataset']).corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

plt.title("Heatmap of Feature Correlations")
plt.show()
```

شکل ۱۹: کد نقشه حرارتی دادگان گل زنبق

```
[ ] import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import pandas as pd

# Load Iris dataset from scikit-learn
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names) # Ensure proper feature names
df["species"] = iris.target

# Split dataset into train and test sets
train_df, test_df = train_test_split(df, test_size=0.2, random_state=4, stratify=df["species"])

# Correctly extract feature names
features = df.columns[:-1] # Exclude 'species' column

# Plot PDFs for all four features
plt.figure(figsize=(12, 10)) # Set overall figure size

for i, feature in enumerate(features, 1):
    plt.subplot(2, 2, i) # Create a 2x2 grid of subplots
    sns.kdeplot(train_df[feature], label="Train Data", fill=True, )
    sns.kdeplot(test_df[feature], label="Test Data", fill=True, )

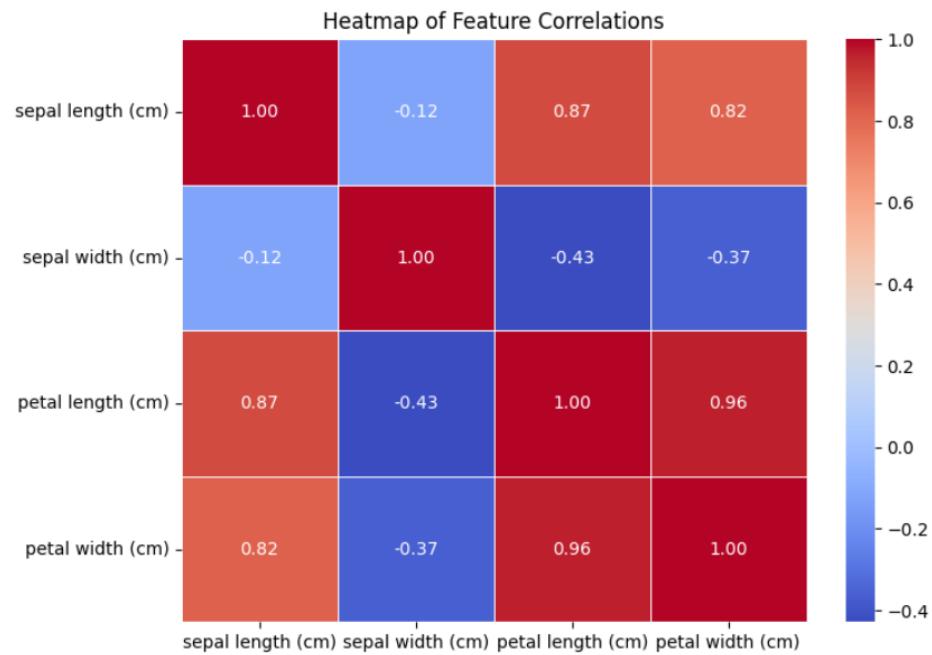
    plt.title(f"Probability Density Function of {feature}")
    plt.xlabel(feature)
    plt.ylabel("Density")
    plt.legend()

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```

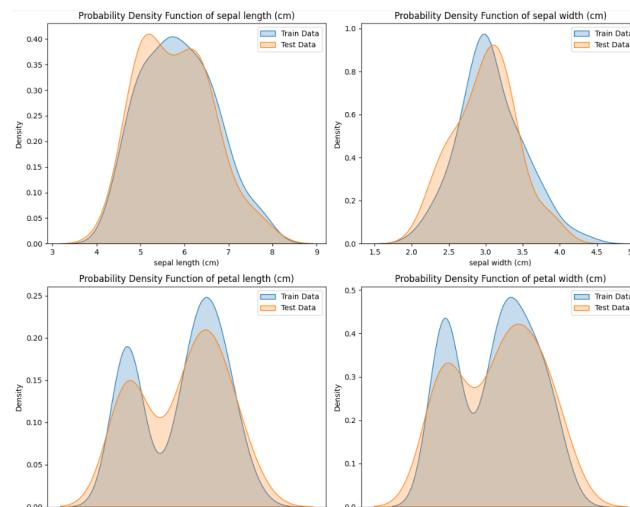
شکل ۲۰: کد نمایشتابع چگالی احتمال دادگان گل زنبق به تفکیک آموزش و تست



نتایج کد های مربوطه در زیر نمایش داده شده اند.



شکل ۲۱: نقشه حرارتی دادگان گل زنبق



شکل ۲۲: تابع چگالی احتمال دادگان گل زنبق به تفکیک آموزش و تست



در قسمت بعد با انتخاب چند مقدار عددی دلخواه که در لیست bins قرار می‌گیرند داده‌ها را گسسته سازی کرده‌ایم. کد زیر گویای این مطلب می‌باشد.

```
# Choose the numerical feature for discretization
feature = "petal length (cm)"
# Define bin edges and labels for discretization
def discretize_feature(df, feature, bins, labels):
    df[f'{feature}_category'] = pd.cut(df[feature], bins=bins, labels=labels,)

discretize_feature(df, feature, bins=[df[feature].min(),2.5, 5,df[feature].max()], labels=['Short', 'Medium', 'Long'])

# Display first few rows
df.head()
```

شکل ۲۳: گسسته سازی داده

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	petal length (cm)_category
0	5.1	3.5	1.4	0.2	0	Short
1	4.9	3.0	1.4	0.2	0	Short
2	4.7	3.2	1.3	0.2	0	Short
3	4.6	3.1	1.5	0.2	0	Short
4	5.0	3.6	1.4	0.2	0	Short

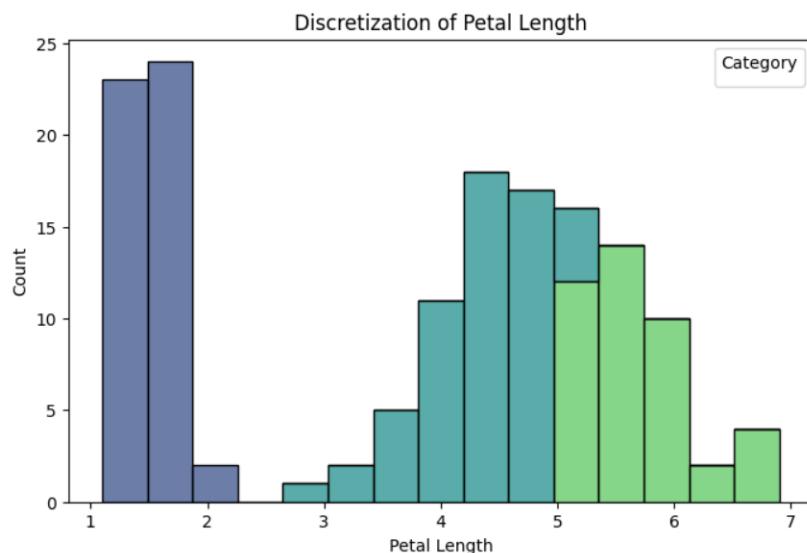
شکل ۲۴: نمایش بخشی از دیتای خاصل



همچنین هیستوگرامی برای نشان دادن دسته بندی های جدید نمایش داده شده:

```
plt.figure(figsize=(8, 5))
sns.histplot(df, x="petal length (cm)", hue="petal length (cm)_category", multiple="stack", bins=15, palette="viridis")
plt.title("Discretization of Petal Length")
plt.xlabel("Petal Length")
plt.ylabel("Count")
plt.legend(title="Category")
plt.show()
```

شکل ۲۵: کد هیستوگرام



شکل ۲۶: هیستوگرام تعداد گل ها بر اساس برحسب اندازه



در بخش بعدی ویژگی های آماری یک خانواده از گل های زنبق (Setosa) را با متدها describe() و describe() که در زیر نمایش داده شده اند.

```
[ ] setosa_stats = df[df['species'] == 0].describe()  
print(setosa_stats)
```

شکل ۲۷: کد نمایش تحلیل آماری

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
count	50.00000	50.000000	50.000000	
mean	5.00600	3.428000	1.462000	
std	0.35249	0.379064	0.173664	
min	4.30000	2.300000	1.000000	
25%	4.80000	3.200000	1.400000	
50%	5.00000	3.400000	1.500000	
75%	5.20000	3.675000	1.575000	
max	5.80000	4.400000	1.900000	

	petal width (cm)	species
count	50.000000	50.0
mean	0.246000	0.0
std	0.105386	0.0
min	0.100000	0.0
25%	0.200000	0.0
50%	0.200000	0.0
75%	0.300000	0.0
max	0.600000	0.0

شکل ۲۸: توصیف ویژگی های آماری Setosa

# THE END