داده‌کاوی

دانشگاه آزاد شیراز — دانشکده مهندسی کامپیوتر

ترم اول سال تحصیلی ۱۴۰۶-۱۴۰۵

فصل دوم: Data, measurements, and data preprocessing

استاد: دکتر امین اسکندری

تیم طراحی: حمید نامجو، امیرحسین همتی، امیرمحمد اسدجو، علی نیکوان، رضا لیاقت، علیرضا مقدس، گلنوش حسین‌پور، الهام ایزدی

فهرست مطالب

2.1 Data types

2.1.1 Nominal attributes
2.1.2 Binary attributes
2.1.3 Ordinal attributes
2.1.4 Numeric attributes
2.1.5 Discrete vs. continuous attributes

2.2 Statistics of data

2.2.1 Measuring the central tendency
2.2.2 Measuring the dispersion of data
2.2.3 Covariance and correlation analysis
2.2.4 Graphic displays of basic statistics of data

2.3 Similarity and distance measures

2.3.1 Datamatrix vs. dissimilaritymatrix
2.3.2 Proximity measures for nominal attributes
2.3.3 Proximity measures for binary attributes
2.3.4 Dissimilarity of numeric data: Minkowski distance
2.3.5 Proximity measures for ordinal attributes
2.3.6 Dissimilarity for attributes of mixed types
2.3.7 Cosine similarity
2.3.8 Measuring similar distributions: the Kullback-Leibler divergence
2.3.9 Capturing hidden semantics in similarity measures

2.4 Data quality, data cleaning, and data integration

2.4.1 Data quality measures
2.4.2 Data cleaning
2.4.3 Data integration

2.5 Data transformation

2.5.1 Normalization
2.5.2 Discretization
2.5.3 Data compression
2.5.4 Sampling

2.6 Dimensionality reduction

2.6.1 Principal components analysis
2.6.2 Attribute subset selection

2.6.3 Nonlinear dimensionality reduction methods

📌 2.1 Data types

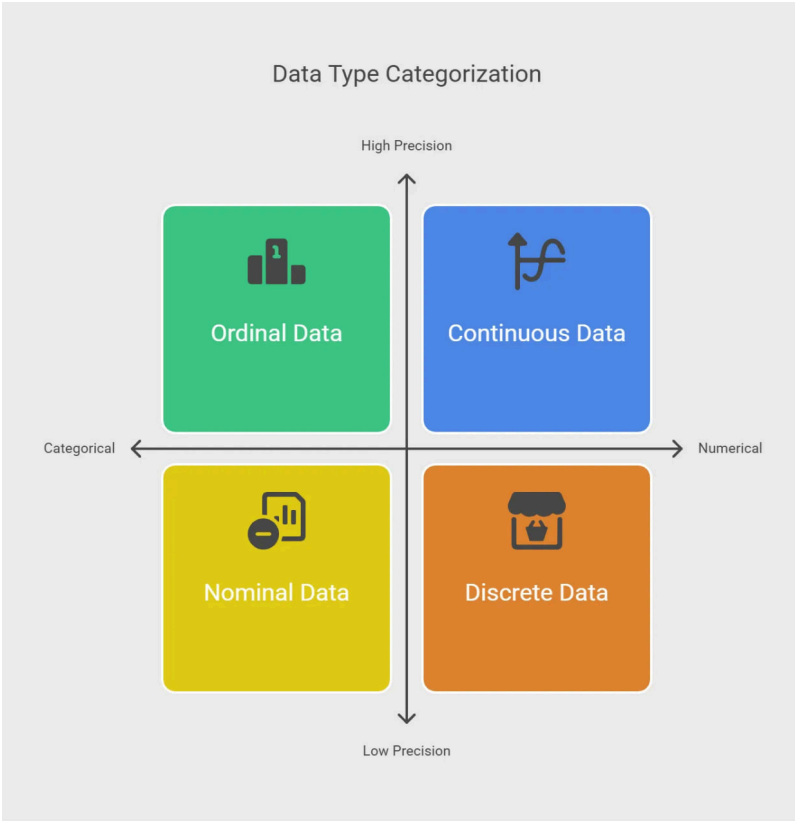
تحلیل بدون شناخت داده = رانندگی با چشمان بسته ⚠️

دست‌یابی به نتایج دقیق و قابل‌اعتماد در داده‌کاوی، **نیاز به شناخت کاملِ داده‌های مورد استفاده دارد**. این شناخت شامل آگاهی از **نوع ویژگی‌ها**، **مقیاس اندازه‌گیری**، **ساختار درونی** و **توزیع داده‌ها** می‌شود. انتخاب الگوریتم مناسب برای هر مسئله، مستقیماً به این شناخت وابسته است؛ به‌گونه‌ای که هرگونه سهل‌انگاری در این مرحله، معمولاً به بروز خطاهای چشمگیر در نتایج نهایی می‌انجامد.

پیش از آن‌که به دسته‌بندی دقیق‌تر ویژگی‌ها بپردازیم، لازم است داده‌ها را در یک سطح کلی‌تر نیز بررسی کنیم: **داده‌های کیفی (Qualitative)** در برابر **داده‌های کمّی (Quantitative)**. داده‌های کیفی، ویژگی‌هایی هستند که مقادیرشان بر پایهٔ توصیف یا دسته‌بندی استوار است و ماهیت عددی ندارند؛ برای نمونه، رنگ چشم یا جنسیت را می‌توان در این گروه جای داد. در سوی دیگر، داده‌های کمّی مقادیری عددی دارند که امکان انجام عملیات ریاضی مانند جمع، تفریق و میانگین‌گیری روی آن‌ها وجود دارد؛ مانند سن، درآمد یا دما.

> آیا متغیر «**رتبهٔ کنکور**» یک دادهٔ کیفی است یا کمّی؟ پاسخ خود را با ذکر دلیل توضیح دهید.

بر اساس این تفاوت بنیادین، ویژگی‌ها در چهار دستهٔ اصلی طبقه‌بندی می‌شوند: **اسمی (Nominal)**، **دودویی (Binary)**، **ترتیبی (Ordinal)** و **عددی (Numeric)**. در بخش‌های بعدی، هریک از این دسته‌ها را بررسی خواهیم کرد.



📐 ریاضیات — نمایش ریاضی داده

هر شیء داده را می‌توان به‌صورت یک بردار ویژگی نمایش داد.

$$x_i = (x_{i1}, x_{i2}, \dots, x_{ip}) \tag{2.1}$$

نمادها

- x_i: بردار ویژگی شیء iام.
- x_{if}: مقدار ویژگی fام.
- i: اندیس شیء.
- f: اندیس ویژگی.
- p: تعداد ویژگی‌ها.

مثال / تمرین حل‌شده

صورت سؤال: یک شیء با سه ویژگی به‌صورت بردار $x_1 = (25, 40, 3)$ داده شده است. این نمایش را تفسیر کنید.

حل: این بردار سه مؤلفه دارد؛ بنابراین شیء یکم سه ویژگی دارد: $x_{11} = 25$، $x_{12} = 40$ و $x_{13} = 3$.

توضیح: اندیس اول مشخص می‌کند داده مربوط به کدام شیء است و اندیس دوم مشخص می‌کند کدام ویژگی را نشان می‌دهد. این نمایش پایه‌ای برای ساخت ماتریس داده است.

2.1.1 Nominal attributes

ویژگی‌های اسمی

ویژگی‌های اسمی (Nominal) متغیرهای کیفی هستند که فقط نشان‌دهندهٔ **دسته‌ها** یا **برچسب‌های متفاوت اند** و هیچ‌گونه **رتبه‌بندی**، **اولویت** یا **ترتیب ذاتی** در آن‌ها وجود ندارد. به زبان ساده، مقادیر این ویژگی‌ها فقط نشان می‌دهند که یک داده به چه **طبقه‌ای** تعلق دارد. برای مثال، وقتی از نوع حیوانات حرف می‌زنیم، **خرگوش** نه از **گربه** برتر است و نه کمتر؛ آن‌ها فقط دو **دستهٔ متفاوت**اند.

از نمونه‌های رایج ویژگی‌های اسمی می‌توان به **رنگ**، **جنسیت**، **گروه خونی**، **نوع حیوان**، **شغل** و **ملیت** اشاره کرد. در تمامی این موارد، مقادیر صرفاً معرف **طبقات مستقل**اند و هیچ رابطهٔ **برتری** یا **توالی منطقی** میان آن‌ها وجود ندارد.

ریاضیات — ویژگی‌های اسمی

در ویژگی اسمی، برچسب‌ها ترتیب عددی ندارند؛ بنابراین فقط برابر یا نابرابر بودن آن‌ها اهمیت دارد.

$$\text{match}(x_i, x_j) = \begin{cases} 1 & x_i = x_j \\ 0 & x_i \neq x_j \end{cases} \tag{2.2}$$

نمادها

- x_i, x_j: دو مقدار اسمی.
- match: تابع بررسی تطابق.

مثال / تمرین حل‌شده

صورت سؤال: دو مقدار اسمی «قرمز» و «قرمز» را با هم مقایسه کنید؛ سپس «قرمز» و «آبی» را مقایسه کنید.

حل: در حالت اول دو مقدار برابرند، پس مقدار Match برابر 1 است. در حالت دوم مقادیر متفاوت‌اند، بنابراین Match برابر 0 می‌شود.

توضیح: در ویژگی اسمی فقط یکسان یا متفاوت بودن اهمیت دارد؛ فاصله عددی بین برچسب‌ها معنایی ندارد.

مثال عملی: رمزگشایی سلیقه غذایی در استارتاپ «فودباکس»

فرض کنید «فودباکس» یک سرویس اشتراک هوشمند غذای رژیمی است که بر اساس مشخصات و سلیقه کاربران، بسته‌های غذایی هفتگی ارسال می‌کند. یکی از مهم‌ترین متغیرهای این سیستم، نوع رژیم غذایی کاربر است؛ متغیری از جنس **ویژگی اسمی (Nominal Attribute)** که هیچ ترتیب یا رتبه‌ای بین گزینه‌هایش وجود ندارد (کیتو نه بالاتر است، نه پایین‌تر از گیاه‌خواری).

گام اول: مواجهه با چالش

تیم داده‌کاوی فودباکس با ستونی از جنس متن روبه‌رو است:

- Diet_Type: کیتو، گیاه‌خواری، پروتئین بالا، عادی

مشکل اینجاست که الگوریتم‌های یادگیری ماشین، متن نمی‌فهمند و فقط با عدد کار می‌کنند. اگر بی‌گدار به آب بزنیم و مثلاً به کیتو عدد ۱، گیاه‌خواری عدد ۲ و پروتئین بالا عدد ۳ بدهیم، به مدل دروغ گفته‌ایم؛ چون این کار به مدل القا می‌کند که «پروتئین بالا» سه برابر «کیتو» ارزش دارد؛ فرضی کاملاً نادرست و گمراه‌کننده.

گام دوم: راه‌حل — One-Hot Encoding

برای حل این مشکل، به‌جای یک ستون عددی گمراه‌کننده، برای هر رژیم غذایی یک ستون جداگانه صفر و یکی می‌سازیم. مثلاً کاربری که رژیم کیتو دارد، در ستون Diet_Type_Keto عدد ۱ می‌گیرد و در بقیه ستون‌ها عدد ۰؛ به این ترتیب هیچ رتبه یا ارجحیت کاذبی بین گزینه‌ها شکل نمی‌گیرد.

گام سوم: پیاده‌سازی با پایتون

با یک خط کد ساده در کتابخانه pandas، این تبدیل به‌سادگی انجام می‌شود:

```
encoded_df = pd.get_dummies(df, columns=['Diet_Type'])
```

نتیجه: حالا مدل داده‌کاوی، بدون هیچ سوءتفاهمی درباره ارزش نسبی رژیم‌ها، می‌تواند الگوی تغذیه کاربران را یاد بگیرد و پیشنهادهای دقیق‌تری ارائه دهد.

خروجی: چهار ستون مستقل صفر و یک به‌جای یک ستون متنی گمراه‌کننده

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display, HTML

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 120

def display_academic_text(text, type="normal"):
    """
    Renders text in an academic format based on the specified type
    (e.g., title, paragraph, insight).
    Uses strict typography and styling to avoid informal elements.
    """
    styles = {
        "main_title": (
            "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 24px; "
            "font-weight: bold; text-align: center; border-bottom: 2px solid #34495e; "
            "padding-bottom: 15px; margin: 30px 0;"
        ),
        "sub_title": (
            "font-family: Tahoma, sans-serif; color: #34495e; font-size: 19px; "
            "font-weight: bold; border-right: 5px solid #2980b9; padding-right: 15px; "
```

```
        "margin: 40px 0px 20px 0px; background-color: #ecf0f1; padding: 12px; "
        "border-radius: 4px;"
    ),
    "paragraph": (
        "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 16px; "
        "margin-bottom: 20px; line-height: 1.8; text-align: justify;"
    ),
    "insight": (
        "font-family: Tahoma, sans-serif; color: #16a085; font-size: 16px; "
        "font-weight: bold; background-color: #e8f8f5; padding: 20px; "
        "border-left: 5px solid #1abc9c; margin: 30px 0; line-height: 1.8; "
        "text-align: justify;"
    )
}
display(HTML(f"<div dir='rtl' style='text-align: right; {styles[type]}'>{text}</div>"))

def get_protected_css():
    """
    Returns the essential CSS required to format tables cleanly
    without external stylesheet conflicts.
    """
    return """
    <style>
    .academic-table-wrapper {
        direction: ltr !important;
        overflow-x: auto !important;
        width: 100% !important;
        background-color: #fff;
        border: 1px solid #bdc3c7;
        border-radius: 4px;
        box-shadow: 0 3px 6px rgba(0,0,0,0.08);
    }
    table.strict-tbl {
        border-collapse: collapse !important;
        width: 100% !important;
        margin: 0 !important;
        font-family: inherit !important;
        font-size: 14px !important;
        table-layout: auto !important;
        border: none !important;
    }
    table.strict-tbl thead th {
        background-color: #2c3e50 !important;
        color: #ffffff !important;
        padding: 12px 18px !important;
        text-align: center !important;
        border: 1px solid #34495e !important;
        white-space: nowrap !important;
        font-weight: bold !important;
    }
    table.strict-tbl tbody td {
        padding: 10px 18px !important;
        text-align: center !important;
        border: 1px solid #bdc3c7 !important;
        white-space: nowrap !important;
        color: #2c3e50 !important;
    }
    table.strict-tbl tbody tr:nth-child(even) td {
        background-color: #f8f9fa !important;
    }
    table.strict-tbl tbody tr:hover td {
        background-color: #e8ecef !important;
    }
    </style>
    """

def display_side_by_side_datasets(dfs, titles, *args):
    """
    Renders one or multiple pandas DataFrames.

    - If only one dataframe: displays it full-width.
    - If multiple dataframes: displays them side-by-side.

    Parameters:
    -----------
    dfs : list or DataFrame
        List of dataframes, or a single dataframe
    titles : list or str
        List of titles, or a single title string
    *args : (optional) for backward compatibility with old four-argument calls
    """
    if not isinstance(dfs, list) and not isinstance(titles, list):
        df1, title1, df2, title2 = dfs, titles, args[0], args[1]
        dfs = [df1, df2]
        titles = [title1, title2]
    else:
        if not isinstance(dfs, list) or not isinstance(titles, list):
            raise ValueError("(یا چهار آرگومان جداگانه) دو آرگومان اول باید لیست باشند.")
        if len(dfs) != len(titles):
            raise ValueError("تعداد دیتافریم‌ها و عنوان‌ها باید برابر باشد.")
        if len(dfs) == 0:
            raise ValueError("حداقل یک دیتافریم برای نمایش نیاز است.")

    if len(dfs) == 1:
        df = dfs[0]
        title = titles[0]
        table_html = df.to_html(index=False, classes=['strict-tbl'], border=0)
        html = f"""
        <div style="width: 100%; margin: 20px 0 35px 0;">
            <div dir="rtl" style="background-color: #2980b9; color: white; padding: 12px;
                                  text-align: center; font-family: Tahoma, sans-serif;
                                  font-weight: bold; font-size: 15px; border-radius: 4px 4px 0 0;">
                {title}
```

```
        </div>
        <div class="academic-table-wrapper" style="margin-top: 0; border-top: none;">
            {table_html}
        </div>
    </div>
    """
    display(HTML(get_protected_css() + html))
    return

num_cols = len(dfs)
col_html_list = []
for i, (df, title) in enumerate(zip(dfs, titles)):
    table_html = df.to_html(index=False, classes=['strict-tbl'], border=0)
    bg_color = "#2980b9" if i % 2 == 0 else "#27ae60"
    col_html = f"""
    <div class="flex-column" style="flex: 1 1 0; min-width: 0; display: flex; flex-direction: column;">
        <div style="background-color: {bg_color}; color: white; padding: 12px;
                    text-align: center; font-family: Tahoma, sans-serif;
                    font-weight: bold; font-size: 15px; border-radius: 4px 4px 0 0;">
            {title}
        </div>
        <div class="academic-table-wrapper" style="margin-top: 0; border-top: none;">
            {table_html}
        </div>
    </div>
    """
    col_html_list.append(col_html)

columns_joined = "".join(col_html_list)
html = f"""
<div style="display: flex; flex-direction: row; justify-content: space-between;
            gap: 20px; width: 100%; max-width: 100%; margin: 20px 0 35px 0;
            box-sizing: border-box;">
    {columns_joined}
</div>
    """
    display(HTML(get_protected_css() + html))


display(HTML("""
<style>
@import url('https://cdn.jsdelivr.net/gh/rastikerdar/vazirmatn@v33.003/Vazirmatn-font-face.css');

* {
    font-family: 'Vazirmatn', 'Vazir', Tahoma, sans-serif !important;
}
div, p, h1, h2, h3, h4, td, th, span, li, .strict-tbl, .academic-table-wrapper {
    font-family: 'Vazirmatn', 'Vazir', Tahoma, sans-serif !important;
}
</style>
"""))



np.random.seed(42)
```

In [2]:

```
display_academic_text(
    "پیش‌پردازش داده‌ها: مدیریت ویژگی‌های اسمی با استفاده از تکنیک رمزگذاری یک‌باره (One-Hot Encoding)",
    "main_title"
)

display_academic_text(
    "نمایانگر مقادیر یا حالت‌های فاقد ترتیب هستند (Nominal Features) بر اساس مبانی فصل دوم، ویژگی‌های اسمی. "
    "ندارد. تیم کاوش دانش (Ordinal) در یک سیستم اشتراک غذای رژیمی، متغیر نوع رژیم غذایی هیچ‌گونه ساختار ترتیبی "
    "،در مواجهه با این پایگاه داده متوجه می‌شود که تخصیص مقادیر عددی پیوسته (مانند ۱، ۲ و ۳) به این متغیرها "
    "منجر به القای ارزش‌های نسبیِ کاذب (مثلاً فرض اینکه رژیم کتو دو برابرِ رژیم عادی ارزش دارد!) و بروز سوءگیری "
    "محاسباتی در الگوریتم‌های فاصله‌محورِ یادگیری ماشین خواهد شد.",
    "paragraph"
)

# ==========================================================
# Section 2: Data Simulation & Preprocessing
# ==========================================================
display_academic_text(
    "بازنمایی معماری داده‌ها پیش و پس از اعمال تحولات ساختاری",
    "sub_title"
)

display_academic_text(
    "بهره‌برداری می‌گردد. <b>رمزگذاری یک‌باره (One-Hot Encoding)</b> جهت رفع ناهنجاری حاصل از مقادیر متنی، از مکانیزم "
    "در فضای ویژگی تولید می‌شود. <i>بُعدِ باینریِ مجزا</i> در این روش، به جای تخصیص یک شناسهٔ عددیِ گمراه‌کننده، به ازای هر دستهٔ مستقل، یک "
    "جداول زیر ساختار خام و ساختارِ متعامدِ نهایی را در کنار یکدیگر مقایسه می‌کنند.",
    "paragraph"
)

# Step 1: Simulate the raw dataset containing a nominal attribute (Diet_Type)
raw_data_dict = {
    'User_ID': [1, 2, 3, 4, 5],
    'Diet_Type': ['Keto', 'Vegetarian', 'High-Protein', 'Normal', 'Keto']
}
df_raw = pd.DataFrame(raw_data_dict)

# Step 2: Apply One-Hot Encoding to prevent ordinal bias.
# The 'get_dummies' function pivots the nominal categories into independent binary dimensions.
df_encoded = pd.get_dummies(df_raw, columns=['Diet_Type'], dtype=int)

# Clean up column names for better display
df_encoded.columns = [col.replace('Diet_Type_', '') if 'Diet_Type' in col else col for col in df_encoded.columns]

# Step 3: Render both raw and encoded DataFrames side-by-side using the list format
display_side_by_side_datasets(
    [df_raw, df_encoded],
    [
```

```
        "وضعیت خام: ویژگی اسمی فاقد ترتیب (Nominal Attribute)",
        "وضعیت نهایی: ویژگی‌های باینری و متعامد (One-Hot Encoded)"
    ]
)

# ==========================================================
# Section 3: Compact Visual Analytics
# ==========================================================
# Step 4: Isolate the binary feature matrix for visualization
heatmap_matrix = df_encoded.drop('User_ID', axis=1)

# Modify the index labels to represent actual users in the visualization
heatmap_matrix.index = [f"User Instance {i}" for i in df_encoded['User_ID']]

# Configure a customized matplotlib figure
plt.figure(figsize=(14, 5), dpi=135)

# Step 5: Render a heatmap to visually prove the orthogonality of the new features.
# We use a custom color map: White for 0, Navy/Academic Blue for 1
from matplotlib.colors import ListedColormap
cmap_binary = ListedColormap(['#f8f9fa', '#2c3e50'])

sns.heatmap(
    heatmap_matrix,
    annot=True,
    cmap=cmap_binary,
    cbar=False,
    linewidths=2.0,
    linecolor='#bdc3c7',
    annot_kws={"size": 16, "weight": "bold"}
)

# Add an explanatory annotation indicating the mutual exclusivity
plt.annotate(
    "Mutual Exclusivity (Orthogonality):\nNotice how every row contains exactly one '1'.\nThe machine no longer compares 'Keto' vs 'Normal' as numbers,\nbut treats them as independent dimensi
    xy=(4, 1.5), xytext=(5.5, 2.5),
    fontsize=11, fontweight='bold', color='#2c3e50',
    arrowprops=dict(arrowstyle="->", color="#34495e", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.5", fc="#ecf0f1", ec="#34495e", lw=1.5),
    clip_on=False
)

# Apply formal academic labeling
plt.title(
    'Figure 1: Orthogonal Representation of One-Hot Encoded Feature Space',
    fontdict={'fontsize': 14, 'fontweight': 'bold'},
    pad=15
)
plt.ylabel('User Records', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.xlabel('Binary Encoded Diet Dimensions (Features)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

plt.tight_layout()
plt.show()

# ==========================================================
# Section 4: Final Strategic Conclusion
# ==========================================================
display_academic_text(
    r"<b>استنتاج راهبردیِ عملیات پیش‌پردازش:</b><br><br>"
    r"نمودار بالا (Figure 1) جادویِ ریاضیاتیِ One-Hot Encoding را به نحوی سازمان‌دهی می‌کند که الگوریتم‌های داده‌کاوی (مانند فاصله‌محور و الگوریتم‌های مبتنی بر درخت) را نشان می‌دهد. اعمال این مکانیزمِ رمزگذاری، ماتریس ویژگی‌ها را به
    r"بدون مواجهه با سوءتفاهم‌های ناشی از «مقیاس‌گذاری کاذب»، الگوهای تغذیه‌ای را به صورت <b>عینی و متعامد (Orthogonal)</b> تفسیر نموده و پیشنهادات شخصی‌سازی‌شده را با بالاترین سطح دقت استخراج نمایند.",
    "insight"
)
```

پیش‌پردازش داده‌ها: مدیریت ویژگی‌های اسمی با استفاده از تکنیک رمزگذاری یک‌باره (One-Hot Encoding)

بر اساس مبانی فصل دوم، ویژگی‌های اسمی (Nominal Features) نمایانگر مقادیر یا حالت‌های فاقد ترتیب هستند. در یک سیستم اشتراک غذای رژیمی، متغیر نوع رژیم غذایی هیچ‌گونه ساختار ترتیبی (Ordinal) ندارد. تیم کاوش دانش در مواجهه با این پایگاه داده متوجه می‌شود که تخصیص مقادیر عددی پیوسته (مانند ۱، ۲ و ۳) به این متغیرها، منجر به القای ارزش‌های نسبیِ کاذب (مثلاً فرض اینکه رژیم کتو دو برابرِ رژیم عادی ارزش دارد!) و بروز سوءگیری محاسباتی در الگوریتم‌های فاصله‌محورِ یادگیری ماشین خواهد شد.

بازنمایی معماری داده‌ها پیش و پس از اعمال تحولات ساختاری

جهت رفع ناهنجاری حاصل از مقادیر متنی، از مکانیزم **رمزگذاری یک‌باره (One-Hot Encoding)** بهره‌برداری می‌گردد. در این روش، به جای تخصیص یک شناسهٔ عددیِ گمراه‌کننده، به ازای هر دستهٔ مستقل، یک *بُعدِ باینریِ مجزا* در فضای ویژگی تولید می‌شود. جداول زیر ساختار خام و ساختارِ متعامدِ نهایی را در کنار یکدیگر مقایسه می‌کنند.

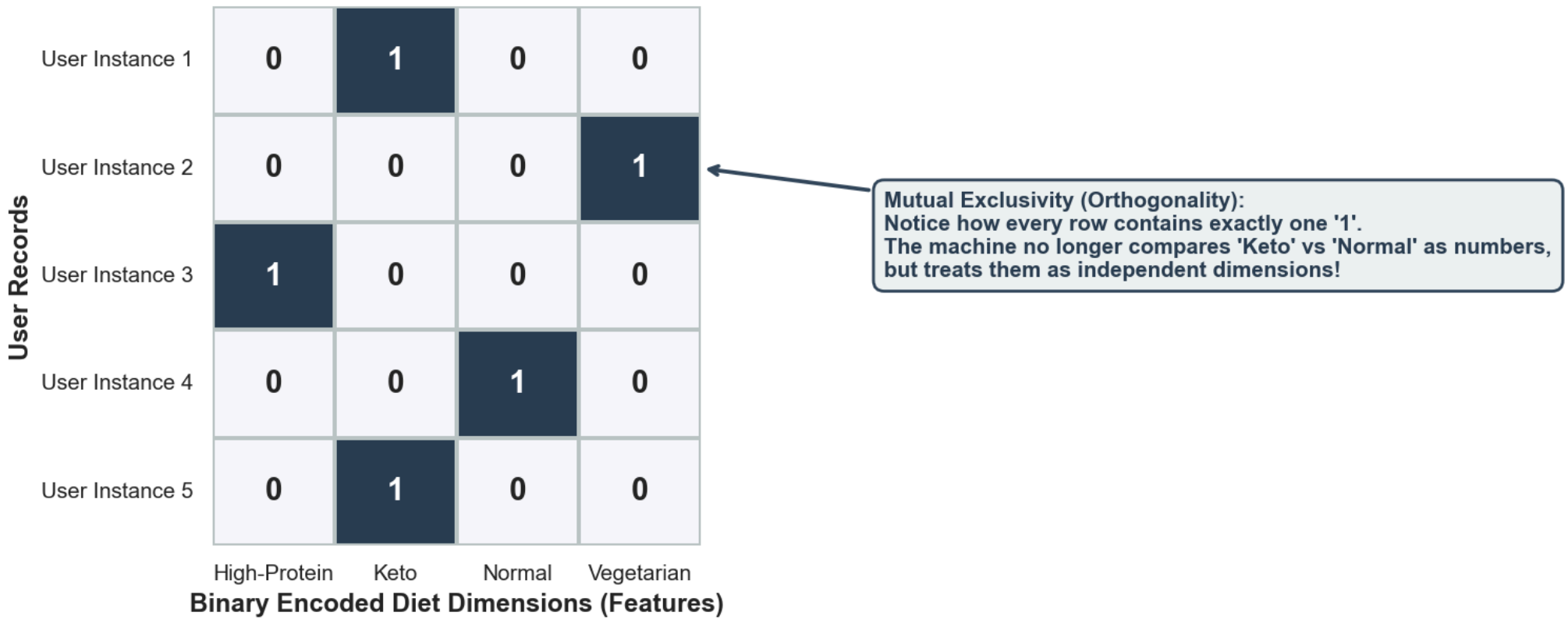
وضعیت خام: ویژگی اسمی فاقد ترتیب (Nominal Attribute)

User_ID	Diet_Type
1	Keto
2	Vegetarian
3	High-Protein
4	Normal
5	Keto

وضعیت نهایی: ویژگی‌های باینری و متعامد (One-Hot Encoded)

User_ID	High-Protein	Keto	Normal	Vegetarian
1	0	1	0	0
2	0	0	0	1
3	1	0	0	0
4	0	0	1	0
5	0	1	0	0

Figure 1: Orthogonal Representation of One-Hot Encoded Feature Space



استنتاج راهبردیِ عملیات پیش‌پردازش:

نمودار بالا (Figure 1) جادویِ ریاضیاتیِ One-Hot Encoding را نشان می‌دهد. اعمال این مکانیزمِ رمزگذاری، ماتریس ویژگی‌ها را به نحوی سازمان‌دهی می‌کند که الگوریتم‌های داده‌کاوی (به ویژه شبکه‌های عصبی و الگوریتم‌های فاصله‌محور مانند KNN) قادر خواهند بود بدون مواجهه با سوءتفاهم‌های ناشی از «مقیاس‌گذاری کاذب»، الگوهای تغذیه‌ای را به صورت عینی و متعامد (Orthogonal) تفسیر نموده و پیشنهادات شخصی‌سازی‌شده را با بالاترین سطح دقت استخراج نمایند.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم (DataFrame) پانداس با دو ویژگی: **'User_ID'** (شناسه کاربر) و **'Diet_Type'** (نوع رژیم غذایی) ایجاد می‌کنیم که ویژگی دوم از جنس **اسمی (Nominal)** است.
- مقادیر این ویژگی اسمی شامل چهار دسته‌ی **'High-Protein'**، **'Vegetarian'**، **'Keto'** و **'Normal'** هستند که هیچ ترتیب یا رتبه‌ای بین آن‌ها وجود ندارد.
- دیتافریم اصلی را با تابع **print()** نمایش می‌دهیم تا ساختار خام داده پیش از تبدیل مشخص شود.
- از تابع **pd.get_dummies()** برای انجام **one-hot encoding** استفاده می‌کنیم و ستون **'Diet_Type'** را به چهار ستون دودویی (Binary) مستقل تبدیل می‌کنیم.
- دیتافریم نهایی (**encoded_df**) پس از عملیات **one-hot encoding** نمایش داده می‌شود؛ جایی که هر رژیم غذایی با مقدار **True** یا **False** در ستون مخصوص به خودش مشخص شده است.

2.1.2 Binary attributes

ویژگی‌های دودویی 1️⃣ 0️⃣

ویژگی‌های دودویی (Binary Attributes) متغیرهای کیفی هستند که تنها **دو مقدار** یا **دو حالت مجزا** را می‌توانند بپذیرند. این دو حالت معمولاً بیانگر **وقوع یا عدم وقوع** یک رویداد، **وجود یا نبود** یک ویژگی، یا پاسخ **مثبت و منفی** به یک پرسش هستند. به همین دلیل، این نوع ویژگی‌ها در نمایش متغیرهای کیفی با دو وضعیت مشخص، کاربرد گسترده‌ای در **داده‌کاوی**، **یادگیری ماشین**، **آمار** و سایر حوزه‌های تحلیل داده دارند.

در اغلب کاربردها، مقادیر ویژگی‌های دودویی به‌صورت **۰ و ۱** نمایش داده می‌شوند؛ به‌گونه‌ای که مقدار **۱** معمولاً نشان‌دهندهٔ **وجود** یک ویژگی یا **وقوع** یک رویداد و مقدار **۰** بیانگر **عدم وجود** آن است. همچنین، این مقادیر می‌توانند با برچسب‌هایی نظیر **بله/خیر (Yes/No)**، **درست/نادرست (True/False)**، **مثبت/منفی** یا سایر زوج‌های متقابل نمایش داده شوند. با وجود آنکه این ویژگی‌ها ممکن است به‌صورت عددی کدگذاری شوند، مقادیر آن‌ها صرفاً نقش **شناسه** یا **برچسب** را ایفا می‌کنند و دارای **مفهوم کمّی** نیستند؛ بنابراین، انجام عملیات ریاضی بر روی آن‌ها فاقد تفسیر آماری خواهد بود.

از جمله نمونه‌های رایج ویژگی‌های دودویی می‌توان به **وضعیت تأهل (مجرد/متأهل)**، **وضعیت اشتغال (شاغل/بیکار)**، **نتایج آزمون (قبول/رد)** و **وضعیت خرید (خرید کرده/نکرده)** اشاره کرد. در تمامی این موارد، مقادیر صرفاً بیانگر **دو حالت متقابل** هستند و هیچ‌گونه **ترتیب**، **اولویت** یا **رابطهٔ کمّی** میان آن‌ها وجود ندارد.

📊 **کاربرد عملی:** در سیستم‌های تشخیص پزشکی، بسیاری از ویژگی‌ها به‌صورت دودویی تعریف می‌شوند؛ برای نمونه، «آیا بیمار سابقهٔ دیابت دارد؟» (بله/خیر) یا «نتیجهٔ آزمایش مثبت است؟» (مثبت/منفی). این ویژگی‌ها با وجود سادگی، نقشی تعیین‌کننده در فرایند تشخیص و تصمیم‌گیری پزشکی ایفا می‌کنند.

📐 ریاضیات – ویژگی‌های دودویی

مقدار یک ویژگی دودویی معمولاً با صفر و یک نمایش داده می‌شود.

$$x_i \in \{0, 1\} \tag{2.3}$$

نمادها

- x_i: مقدار ویژگی دودویی.
- 0: یک حالت، معمولاً عدم وجود.
- 1: حالت دیگر، معمولاً وجود.

مثال / تمرین حل‌شده

صورت سؤال: برای ویژگی «خرید»، دو حالت «خرید انجام شده» و «خرید انجام نشده» را به شکل دودویی کدگذاری کنید.

حل: قرارداد می‌گذاریم 1 به معنی «خرید انجام شده» و 0 به معنی «خرید انجام نشده» باشد. در نتیجه رکوردی که خرید کرده مقدار 1 و رکوردی که خرید نکرده مقدار 0 خواهد داشت.

توضیح: معنی صفر و یک باید از قبل مشخص باشد؛ خود عددها به‌تنهایی معنای «خوب» یا «بد» ندارند.

مثال عملی: مدیریت سلامت و امنیت غذایی در «فودباکس» 1️⃣ 0️⃣

در استارتاپ «فودباکس»، حفظ سلامت کاربران در اولویت است. فرض کنید می‌خواهیم متغیرهایی را در پایگاه‌داده ذخیره کنیم که فقط دو حالت دارند: یا بیمار حساسیت دارد یا ندارد، یا اشتراک کاربر فعال است یا غیرفعال. این‌ها نمونه‌های بارز **ویژگی‌های دودویی (Binary Attributes)** هستند.

گام اول: تفاوت ظریف اما حیاتی (متقارن در مقابل نامتقارن)

در فودباکس ما با دو نوع متغیر دودویی کار می‌کنیم:

- **دودویی متقارن (Symmetric):** مثل جنسیت کاربر (زن / مرد). هر دو حالت ارزش اطلاعاتی یکسانی دارند و اولویت خاصی نسبت به هم ندارند.
- **دودویی نامتقارن (Asymmetric):** مثل حساسیت شدید به بادام‌زمینی (دارد / ندارد). در اینجا، حالت «داشتن حساسیت» (که معمولاً با ۱ نشان داده می‌شود) فوق‌العاده حیاتی و نادر است، در حالی که حالت «نداشتن حساسیت» (۰) حالت عادی است و ارزش اطلاعاتی کمتری دارد.

گام دوم: فرمول‌سازی داده‌ها برای آشپزخانه هوشمند

آشپزخانه خودکار فودباکس برای بسته‌بندی غذاها نیاز دارد تا این متغیرها را به فرمت عددی تبدیل کند. اگر متغیر متنیِ `Has_Allergy` مقدارش "Yes" باشد، به عدد ۱ تبدیل می‌شود تا بلافاصله سیستم هشدار حذف بادام‌زمینی را فعال کند. اگر "No" باشد، به ۰ تبدیل می‌شود.

گام سوم: مدل‌سازی عددی

برای تبدیل متغیرهای دو حالتی متنی (مانند Yes/No یا M/F) به کدهای صفر و یک، از نگاشت متناظر (Mapping) در پایتون استفاده می‌کنیم تا داده‌ها برای الگوریتم‌های هوش مصنوعی فودباکس قابل فهم شوند.

نتیجه: آشپزخانه هوشمند با خواندن آرایه عددی صفر و یک، بدون فوت وقت متوجه می‌شود کدام جعبه‌ها باید فاقد مواد حساسیت‌زا باشند و نرخ خطای ارسال به صفر می‌رسد.

خروجی: تبدیل سریع وضعیت‌های حیاتی به کدهای صفر و یک عددی

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [4]:
```
# ==========================================================
# Section 1: Scenario Introduction
# ==========================================================
display_academic_text(
    "پیش‌پردازش داده‌ها: مدیریت ویژگی‌های دودویی متقارن و نامتقارن (Binary Attributes)",
    "main_title"
)
display_academic_text(
    "ویژگی‌های دودویی (Binary) به دو دسته <b>متقارن (Symmetric)</b> و <b>نامتقارن (Asymmetric)</b> تقسیم می‌گردند. "
    "به عنوان یک ویژگی دودویی متقارن (هیچ‌یک از دو حالت بر دیگری ارجحیتِ اطلاعاتی ندارد) و <i>جنسیت</i> در این سناریو، متغیرِ "
    "به عنوان یک ویژگی دودویی نامتقارن (حالتِ 'بله' نادرتر و از نظر اطلاعاتی بسیار باارزش‌تر است) <i>حساسیت به بادام زمینی</i> متغیرِ "
    "مورد بررسی قرار می‌گیرند. به منظور درک الگوریتمیک توسط ماشین، این متغیرهای متنی می‌بایست "
    "از طریق نگاشت‌های ساختاریافته به کدهای عددی (۰ و ۱) تبدیل شوند.",
    "paragraph"
)

# ==========================================================
# Section 2: Data Initialization and Processing
# ==========================================================
display_academic_text(
    "مرحله نخست: مقداردهی اولیه و استخراج نگاشت‌های عددی",
    "sub_title"
)
```

```python
display_academic_text(
    " در این فاز، پایگاه داده خامِ سیستمِ توصیه‌گرِ غذا تعریف شده و سپس دو رویکرد متفاوت"
    "نگاشت (Mapping) به نمایش درآمده‌اند (سایدبای‌سایِد) به صورت موازی تحلیلی به صورت جهت ارزیابی عملیات، نتایج. بر روی آن اعمال می‌گردد.",
    "paragraph"
)

# Step 1: Initialize the raw database containing both symmetric and asymmetric binary attributes
raw_data = {
    'User_ID': [101, 102, 103, 104, 105],
    'Gender (Symmetric)': ['F', 'M', 'F', 'F', 'M'],
    'Nut_Allergy (Asymmetric)': ['Yes', 'No', 'No', 'Yes', 'No']
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Create a copy of the dataframe to store the processed numerical mappings
df_processed = df_raw.copy()

# Step 3: Map the Symmetric Binary Attribute (Gender) to numerical values.
# In symmetric binary attributes, both states carry equal weight. E.g., 'F':1, 'M':0
df_processed['Gender_Numeric'] = df_processed['Gender (Symmetric)'].map({'F': 1, 'M': 0})

# Step 4: Map the Asymmetric Binary Attribute (Nut_Allergy) to numerical values.
# In asymmetric binary attributes, the RARE or CRITICAL state (having the allergy)
# MUST be conventionally mapped to 1.
df_processed['Nut_Allergy_Numeric'] = df_processed['Nut_Allergy (Asymmetric)'].map({'Yes': 1, 'No': 0})

# Extract only the processed numerical columns along with the identifier for final visualization
df_final = df_processed[['User_ID', 'Gender_Numeric', 'Nut_Allergy_Numeric']]

# Render the original and processed dataframes side-by-side
# Using the list format expected by the global function
display_side_by_side_datasets(
    [df_raw, df_final],
    [
        "جدول ۱: پایگاه داده خام (متغیرهای متنی)",
        "جدول ۲: پایگاه داده پردازش‌شده (فضای عددی)"
    ]
)

# ==========================================================
# Section 3: Visual Analytics (Academic Blue)
# ==========================================================
display_academic_text("مرحله دوم: تفسیرِ بصریِ ارزشِ اطلاعاتی (Information Value)", "sub_title")

# Plotting the Information Value difference
fig, axes = plt.subplots(1, 2, figsize=(16, 5), dpi=135)

# Plot 1: Symmetric Distribution
sns.countplot(x='Gender_Numeric', data=df_final, palette=['#bdc3c7', '#2980b9'], ax=axes[0], edgecolor='#2c3e50', linewidth=1.5)
axes[0].set_title('Figure 1: Symmetric Binary (Gender)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=12)
axes[0].set_xticklabels(['Male (0)', 'Female (1)'], fontsize=12, fontweight='bold')
axes[0].set_ylabel('Observation Count', fontsize=12, fontweight='bold')
axes[0].set_xlabel('')
axes[0].annotate(
    "Equal Information Value:\nBoth states are equally important.\nSwapping 0 and 1 changes nothing.",
    xy=(0.5, 2.5), xytext=(0.5, 3.2),
    ha='center', fontsize=11, fontweight='bold', color='#2c3e50',
    bbox=dict(boxstyle="round,pad=0.4", fc="#f8f9fa", ec="#bdc3c7", lw=1.5)
)

# Plot 2: Asymmetric Distribution
sns.countplot(x='Nut_Allergy_Numeric', data=df_final, palette=['#bdc3c7', '#e74c3c'], ax=axes[1], edgecolor='#2c3e50', linewidth=1.5)
axes[1].set_title('Figure 2: Asymmetric Binary (Nut Allergy)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=12)
axes[1].set_xticklabels(['No Allergy (0)', 'Has Allergy (1)'], fontsize=12, fontweight='bold')
axes[1].set_ylabel('')
axes[1].set_xlabel('')
axes[1].annotate(
    "Critical Imbalance:\nThe '1' state is rare but extremely important!\nFailing to detect a '1' here could be fatal.",
    xy=(1, 2.0), xytext=(0.5, 2.8),
    ha='center', fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

for ax in axes:
    ax.grid(axis='y', linestyle='--', alpha=0.5)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# Section 4: Final Conclusion
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات پردازش:</b><br><br>"
    r"همان‌طور که در نمودارهای بالا قابل مشاهده است، تبدیل ویژگی‌های دودویی نامتقارن (مانند حساسیت غذایی) به فرمت عددیِ <code>0 1</code> و تصادفی نیست. "
    r"رمزگذاری شوند تا در محاسبهٔ فواصل <code>1</code> باید لزوماً با کدِ <b>نادر اما حیاتی</b> در سیستم‌های تصمیم‌یار، رخدادهای (مثل Jaccard Similarity) و ضرایب شباهت (Distance Metrics)، "
    r"وزن محاسباتی به درستی اِعمال شود. این الگوی نگاشت، احتمال بروز خطای سیستمی را در محیط‌های دارای ریسک بالا (مثل پیشنهاد اشتباه غذای حاوی بادام‌زمینی) به صفر تقلیل می‌دهد."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: مدیریت ویژگی‌های دودویی متقارن و نامتقارن (Binary Attributes)

ویژگی‌های دودویی (Binary) به دو دسته **متقارن (Symmetric)** و **نامتقارن (Asymmetric)** تقسیم می‌گردند. در این سناریو، متغیرِ *جنسیت* به عنوان یک ویژگی دودویی متقارن (هیچ‌یک از دو حالت بر دیگری ارجحیتِ اطلاعاتی ندارد) و متغیرِ *حساسیت به بادام زمینی* به عنوان یک ویژگی دودویی نامتقارن (حالتِ 'بله' نادرتر و از نظر اطلاعاتی بسیار باارزش‌تر است) مورد بررسی قرار می‌گیرند. به منظور درک الگوریتمیک توسط ماشین، این متغیرهای متنی می‌بایست از طریق نگاشت‌های ساختاریافته به کدهای عددی (۰ و ۱) تبدیل شوند.

مرحله نخست: مقداردهی اولیه و استخراج نگاشت‌های عددی

در این فاز، پایگاه داده خامِ سیستمِ توصیه‌گرِ غذا تعریف شده و سپس دو رویکرد متفاوت نگاشت (Mapping) بر روی آن اعمال می‌گردد. نتایج عملیات، جهت ارزیابی تحلیلی به صورت موازی (سایدبای‌ساید) به نمایش درآمده‌اند.

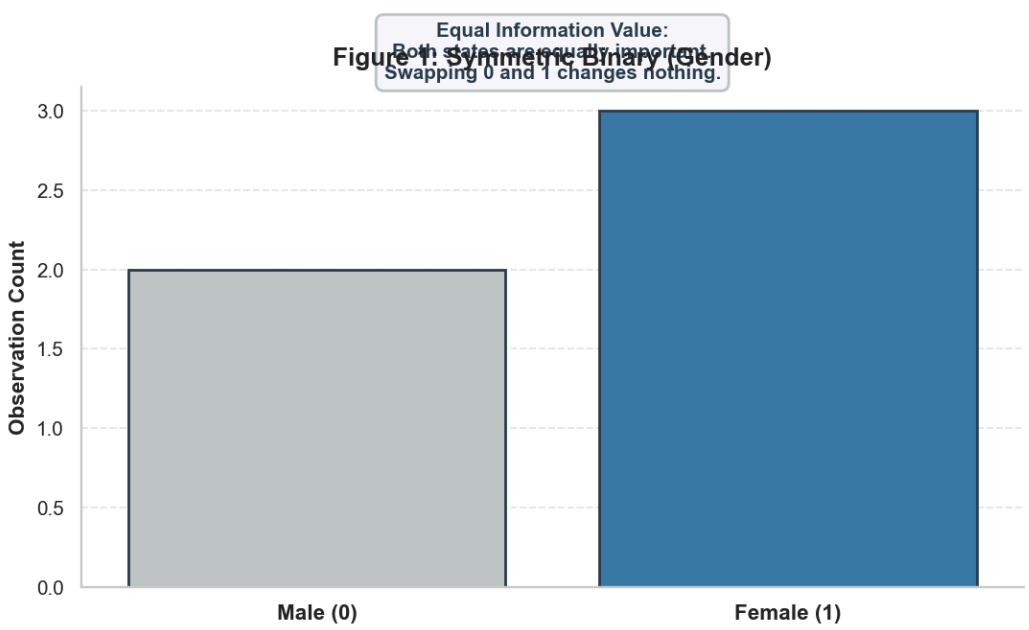
جدول ۱: پایگاه داده خام (متغیرهای متنی)

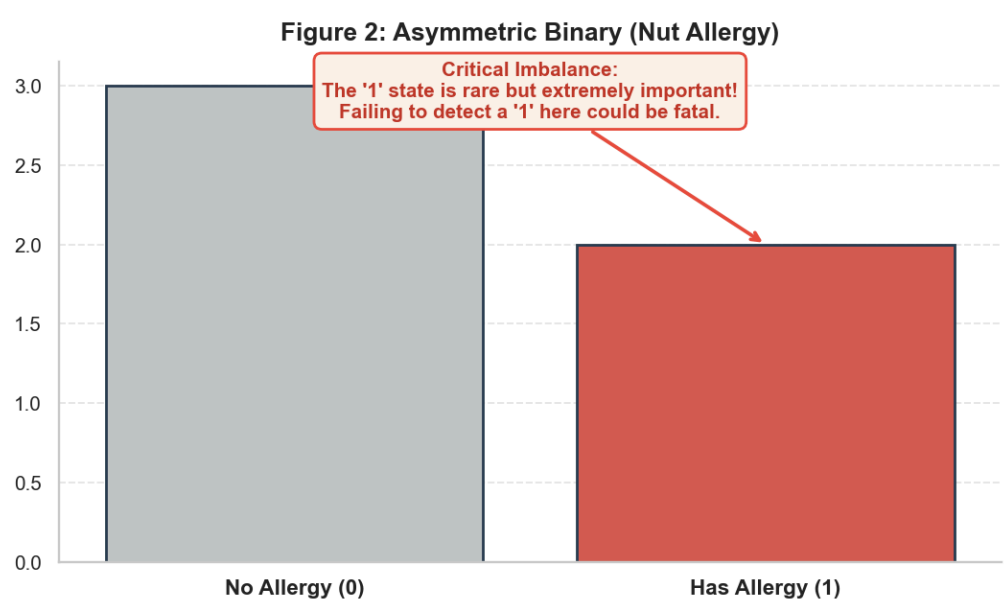
User_ID	Gender (Symmetric)	Nut_Allergy (Asymmetric)
101	F	Yes
102	M	No
103	F	No
104	F	Yes
105	M	No

جدول ۲: پایگاه داده پردازش‌شده (فضای عددی)

User_ID	Gender_Numeric	Nut_Allergy_Numeric
101	1	1
102	0	0
103	1	0
104	1	1
105	0	0

مرحله دوم: تفسیرِ بصریِ ارزشِ اطلاعاتی (Information Value)





استنتاج راهبردی عملیات پردازش:

همان‌طور که در نمودارهای بالا قابل مشاهده است، تبدیل ویژگی‌های دودویی نامتقارن (مانند حساسیت غذایی) به فرمت عددیِ `0` و `1` تصادفی نیست. در سیستم‌های تصمیم‌یار، رخدادهای نادر اما حیاتی باید لزوماً با کدِ `1` رمزگذاری شوند تا در محاسبهٔ فواصل (Distance Metrics) و ضرایب شباهت (مثل Jaccard Similarity)، وزن محاسباتی به درستی اِعمال شود. این الگوی نگاشت، احتمال بروز خطای سیستمی را در محیط‌های دارای ریسک بالا (مثل پیشنهاد اشتباه غذای حاوی بادام‌زمینی) به صفر تقلیل می‌دهد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم با دو متغیر دودویی ایجاد کردیم: متغیر **'Gender'** که یک ویژگی دودویی متقرن است و متغیر **'Nut_Allergy'** که یک ویژگی دودویی نامتقارن (حساسیت به مغزیجات) است.
- از متد **map()** در کتابخانه پانداز استفاده کردیم تا مقادیر متنی را مستقیماً به کدهای باینری ۰ و ۱ تبدیل کنیم.
- در ستون جنسیت، مقدار **'F'** به ۱ و **'M'** به ۰ نگاشت شد (بدون اینکه تفاوتی در ارزش اطلاعاتی آن‌ها وجود داشته باشد).

- در ستون حساسیت، وضعیت بحرانی یعنی وجود آلرژی (**'Yes'**) به عدد ۱ و عدم وجود آن (**'No'**) به عدد ۰ نگاشت شد تا برای سیستم آشپزخانه فودباکس به عنوان یک هشدار فعال‌ساز عمل کند.
- در نهایت، دیتافریم تغییریافته با ستون‌های تماماً عددی آماده ارسال به الگوریتم‌های یادگیری ماشین شد.

2.1.3 Ordinal attributes

ویژگی‌های ترتیبی 📶

ویژگی‌های ترتیبی (Ordinal Attributes) متغیرهای کیفی هستند که مقادیر آن‌ها علاوه بر متفاوت بودن، یک **توالی روشن و معنادار** را دنبال می‌کنند. تفاوت اصلی ویژگی‌های ترتیبی با ویژگی‌های اسمی در این است که در ویژگی‌های ترتیبی «**ترتیب**» حرف اول را می‌زند. اما یک نکته وجود دارد: با اینکه می‌دانیم کدام مقدار از دیگری «**بیشتر**» یا «**برتر**» است، **فاصله** یا **اختلاف** میان این رتبه‌ها لزوماً یکسان یا به‌طور دقیق مشخص نیست.

این نوع ویژگی‌ها معمولاً برای نمایش **سطوح**، **درجات** یا **رتبه‌ها** به‌کار می‌روند. از جمله نمونه‌های رایج آن می‌توان به **سطوح تحصیلی (دیپلم، کارشناسی، کارشناسی ارشد، دکتری)**، **میزان رضایت مشتری (خیلی کم، کم، متوسط، زیاد، خیلی زیاد)**، **رتبه‌های علمی اعضای هیئت علمی (مربی، استادیار، دانشیار، استاد)** و **طبقات اجتماعی–اقتصادی (پایین، متوسط، بالا)** اشاره کرد. از آنجا که مقادیر این ویژگی‌ها دارای ترتیب هستند، شاخص‌هایی مانند **نما** و **میانه** برای توصیف آن‌ها قابل استفاده‌اند؛ اما به دلیل نامشخص بودن فاصلهٔ واقعی میان رتبه‌ها، استفاده از **میانگین** برای این نوع داده‌ها مناسب نیست.

📐 ریاضیات – ویژگی‌های ترتیبی

در داده ترتیبی، ابتدا رتبه‌ها مشخص می‌شوند و برای مقایسه فاصله‌ای می‌توان آن‌ها را به بازه صفر تا یک نگاشت کرد.

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{2.4}$$

نمادها

- z_{if}: رتبه نرمال‌شده.
- r_{if}: رتبه اصلی.
- M_f: تعداد حالت‌های ویژگی f.
- i: اندیس شیء.
- f: اندیس ویژگی.

مثال / تمرین حل‌شده

صورت سؤال: یک ویژگی ترتیبی سه سطح «کم، متوسط، زیاد» دارد. اگر «متوسط» رتبه ۲ باشد، مقدار نرمال‌شده آن را محاسبه کنید.

حل: تعداد سطوح $M_f = 3$ و رتبه $r_{if} = 2$ است. بنابراین $z = (2 - 1)/(3 - 1) = 1/2 = 0.5$.

توضیح: پس از نرمال‌سازی، رتبه اول به 0، رتبه دوم به 0.5 و رتبه سوم به 1 نگاشت می‌شود. این کار مقایسه ویژگی ترتیبی با معیارهای عددی را ساده‌تر می‌کند.

مثال عملی: رتبه‌بندی سطح اشتراک و رضایت کاربران در «فودباکس»

در استارتاپ «فودباکس»، با متغیرهایی روبه‌رو هستیم که علاوه بر داشتن برچسب، دارای یک **ترتیب و رتبه‌بندی طبیعی و معنادار** هستند. برای مثال، سطوح اشتراک کاربران شامل «برنزی»، «نقره‌ای»، «طلایی» و «پلاتینیم» یا حجم وعده‌های غذایی شامل «بزرگ»، «متوسط» و «کوچک» است. به این متغیرها **ویژگی‌های ترتیبی (Ordinal Attributes)** می‌گویند.

گام اول: تفاوت کلیدی با ویژگی‌های اسمی

در ویژگی‌های اسمی (مثل نوع رژیم کیتو یا گیاه‌خواری)، هیچ ترتیبی وجود نداشت. اما در اینجا مشخص است که اشتراک «طلایی» رتبه‌ای بالاتر از «نقره‌ای» دارد. با این حال، فاصله ریاضی دقیق بین برنزی تا نقره‌ای لزوماً با فاصله نقره‌ای تا طلایی برابر نیست؛ تنها چیزی که برای ما حتمی است، **جهت و ترتیب رتبه‌هاست.**

گام دوم: خطای بزرگ One-Hot Encoding در متغیرهای ترتیبی

اگر برای این متغیرها از One-Hot Encoding استفاده کنیم، ترتیب ارزشمند بین رتبه‌ها را کاملاً نابود می‌کنیم و مدل نمی‌فهمد که کاربر طلایی وفادارتر از کاربر برنزی است! از طرف دیگر، اگر اعداد را به‌صورت تصادفی اختصاص دهیم (مثلاً برنزی=۴ و طلایی=۱)، ترتیب منطقی را معکوس و خراب کرده‌ایم.

گام سوم: راه‌حل — کدگذاری ترتیبی دستی (Ordinal Mapping)

بهترین روش، نگاشت دستی اعداد صحیح بر اساس ترتیب واقعی آنهاست؛ به‌طوری که عدد بزرگ‌تر نشان‌دهنده رتبه بالاتر باشد:

- برنزی 👈 ۱ | نقره‌ای 👈 ۲ | طلایی 👈 ۳ | پلاتینیم 👈 ۴

نتیجه: الگوریتم‌های داده‌کاوی (مثل درخت‌های تصمیم) با فهمیدن این ترتیب، می‌توانند پیش‌بینی کنند که با ارتقای کاربر از رتبه ۲ به ۳، احتمال تمدید اشتراک او چقدر افزایش می‌یابد.

خروجی: حفظ دقیق ترتیب منطقی داده‌ها در قالب اعداد صحیحِ رتبه‌دار

In [5]:
```
display_academic_text(
    "پیش‌پردازش داده‌ها: حفظ یکپارچگی سلسله‌مراتب در ویژگی‌های ترتیبی (Ordinal Encoding) 📊",
    "main_title"
)

display_academic_text(
    "متغیرهای ترتیبی (Ordinal Variables) و رتبه‌بندی معنادار<b> (Meaningful Ranking)</b> می‌باشند. "
    "و حجم سفارشات به فرمت عددی است (Loyalty Tiers) در این سناریو، یک پلتفرم خدمات غذایی نیازمند تبدیل سطوح وفاداری. "
    "استفاده از روش‌های رمزگذاری متقارن (نظیر One-Hot Encoding) منجر به <i>تخریبِ این سلسله‌مراتب</i>؛ و از بین رفتنِ فواصلِ منطقی می‌گردد. "
    "نگاشت ترتیبی<b> (Ordinal Mapping)</b> لذا استراتژی جهت حفظ ارزش‌های پیش‌بینانه در دستور کار قرار می‌گیرد.",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION & ORDINAL ENGINEERING
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های خام و اعمال نگاشت رتبه‌ای (Rank Mapping)",
    "sub_title"
)

display_academic_text(
    "در این فاز، مقادیر متنی بر اساس جایگاهِ منطقیِ خود ارزش‌گذاری می‌شوند. به هر سطح، یک عددِ صحیح متناسب با رتبهٔ آن "
    "اختصاص می‌یابد تا الگوریتم‌های مدل‌سازی بتوانند جهت‌گیریِ ارتقا (Upgrade) یا تنزلِ وضعیت (Downgrade) کاربر را به‌دقت تحلیل نمایند.",
    "paragraph"
)

# Step 1: Initialize the raw database containing ordinal textual attributes
raw_data = {
    'User_ID': [201, 202, 203, 204, 205],
    'Loyalty_Tier': ['Bronze', 'Gold', 'Silver', 'Platinum', 'Bronze'],
    'Meal_Portion': ['Small', 'Large', 'Medium', 'Large', 'Small']
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Create a copy of the dataframe to store the ordinally encoded features
df_processed = df_raw.copy()
```

```python
# Step 3: Define explicit mapping dictionaries that capture the natural rank of each category.
# Higher integer values strictly correspond to higher conceptual ranks (Monotonicity).
tier_mapping_logic = {
    'Bronze': 1,
    'Silver': 2,
    'Gold': 3,
    'Platinum': 4
}

portion_mapping_logic = {
    'Small': 1,
    'Medium': 2,
    'Large': 3
}

# Step 4: Apply the ordinal encoding map to transform textual data into ranked numerical vectors
df_processed['Loyalty_Tier_Rank'] = df_processed['Loyalty_Tier'].map(tier_mapping_logic)
df_processed['Meal_Portion_Rank'] = df_processed['Meal_Portion'].map(portion_mapping_logic)

# Isolate the transformed columns for comparative display
df_encoded_view = df_processed[['User_ID', 'Loyalty_Tier_Rank', 'Meal_Portion_Rank']]

# Render the raw and engineered datasets side-by-side to visually demonstrate the structural transformation
display_side_by_side_datasets(
    [df_raw, df_encoded_view],
    [
        "جدول ۱: پایگاه داده خام (متغیرهای ترتیبیِ متنی)",
        "جدول ۲: پایگاه داده مهندسی‌شده (رتبه‌های عددیِ محفوظ)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلیِ سلسله‌مراتبِ ارزش‌گذاری‌شده",
    "sub_title"
)

display_academic_text(
    "نمودارهای نردبانی و تصاعدیِ زیر، ارزش عددیِ اختصاص‌یافته به رده‌های اشتراک و حجم وعده‌ها را نمایش می‌دهند، جهت اثباتِ حفظِ ساختار ترتیبی (Monotonicity). "
    "بین سطوحِ کاربران پس از پیش‌پردازش، کاملاً دست‌نخورده باقی مانده است <b>فاصله و تقدم منطقی</b> این بازنمایی گرافیکی تأیید می‌کند که.",
    "paragraph"
)

# Initialize a wide dual-plot canvas for optimal academic presentation
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- CHART 1: Loyalty Tier Progression (Step Plot) ---
tiers = list(tier_mapping_logic.keys())
tier_ranks = list(tier_mapping_logic.values())

# Step plot to emphasize the discrete, ranked leaps between tiers
axes[0].step(
    tiers, tier_ranks,
    where='mid', color='#2c3e50', linewidth=3.5,
    marker='o', markersize=12, markerfacecolor='#1abc9c', markeredgecolor='#2c3e50', markeredgewidth=2
)

axes[0].annotate(
    "Rank Preservation:\nThe algorithm now mathematically understands\nthat 'Platinum' (4) is conceptually greater\nthan 'Bronze' (1) by exactly 3 levels.",
    xy=(2, 3), xytext=(0.5, 3.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[0].set_title('Figure 1: Hierarchical Integrity of Loyalty Tiers', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[0].set_ylabel('Assigned Numerical Rank (Magnitude)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xlabel('Original Textual Categories', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_yticks(tier_ranks)
axes[0].grid(True, linestyle='--', alpha=0.5)

# --- CHART 2: Meal Portion Progression (Line/Scatter Plot) ---
portions = list(portion_mapping_logic.keys())
portion_ranks = list(portion_mapping_logic.values())

axes[1].plot(
    portions, portion_ranks,
    color='#2980b9', linewidth=3.5, linestyle='--',
    marker='D', markersize=12, markerfacecolor='#e74c3c', markeredgecolor='#2c3e50', markeredgewidth=2
)

axes[1].annotate(
    "Monotonic Relationship:\nThe transition from Small to Large is strictly increasing,\navoiding the pitfall of orthogonal (One-Hot) isolation.",
    xy=(1, 2), xytext=(1.2, 1.2),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[1].set_title('Figure 2: Monotonic Transformation of Meal Portions', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[1].set_ylabel('Assigned Capacity Rank', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_xlabel('Original Textual Capacity', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_yticks(portion_ranks)
axes[1].grid(True, linestyle='--', alpha=0.5)

# Clean up top and right spines for both charts
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
```

```
plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br><br>"
    r"«کدگذاری می‌کردیم، فاصلهٔ بین یک مشتری «برنز» تا «پلاتینیوم (One-Hot) را به روش متقارن <code>Loyalty_Tier</code> تصور کنید اگر ویژگی "
    r"دقیقاً برابر با فاصلهٔ «برنز» تا «نقره‌ای» می‌شد! این یک فاجعهٔ تحلیلی است. <br><br>"
    r"از تخریب اطلاعاتِ ارزشمندِ سلسله‌مراتبی جلوگیری می‌نماید، <b>نگاشت ترتیبی دستی (Ordinal Mapping)</b> اتخاذ رویکردِ. "
    r"را مجهز می‌سازد (XGBoost یا Random Forest مثل) این تکنیک (که در گراف‌های ۱ و ۲ با شیب‌های صعودی و نردبانی اثبات شد)، الگوریتم‌های هوش مصنوعی "
    r"را درک کرده و پیش‌بینی‌های مدل را دقیقاً بر مبنای منطق اولویت‌بندی، بهینه‌سازی کنند <i>«افزایش حجم مصرفی»</i> یا <i>«ارتقای سطح اشتراک»</i> تا وابستگی‌های متوالی نظیر."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: حفظ یکپارچگی سلسله‌مراتب در ویژگی‌های ترتیبی (Ordinal Encoding) 📊

متغیرهای ترتیبی (Ordinal Variables) علاوه بر تمایزِ دسته‌ای، دارای یک توالی و **رتبه‌بندی معنادار (Meaningful Ranking)** می‌باشند. در این سناریو، یک پلتفرم خدمات غذایی نیازمند تبدیل سطوح وفاداری (Loyalty Tiers) و حجم سفارشات به فرمت عددی است. استفاده از روش‌های رمزگذاری متقارن (نظیر One-Hot Encoding) منجر به *تخریبِ این سلسله‌مراتب* و از بین رفتنِ فواصلِ منطقی می‌گردد؛ لذا استراتژی **نگاشت ترتیبی (Ordinal Mapping)** جهت حفظ ارزش‌های پیش‌بینانه در دستور کار قرار می‌گیرد.

مرحله نخست: بارگذاری داده‌های خام و اعمال نگاشت رتبه‌ای (Rank Mapping)

در این فاز، مقادیر متنی بر اساس جایگاهِ منطقیِ خود ارزش‌گذاری می‌شوند. به هر سطح، یک عددِ صحیح متناسب با رتبهٔ آن اختصاص می‌یابد تا الگوریتم‌های مدل‌سازی بتوانند جهت‌گیریِ ارتقا (Upgrade) یا تنزلِ وضعیت (Downgrade) کاربر را به‌دقت تحلیل نمایند.

جدول ۱: پایگاه داده خام (متغیرهای ترتیبیِ متنی)

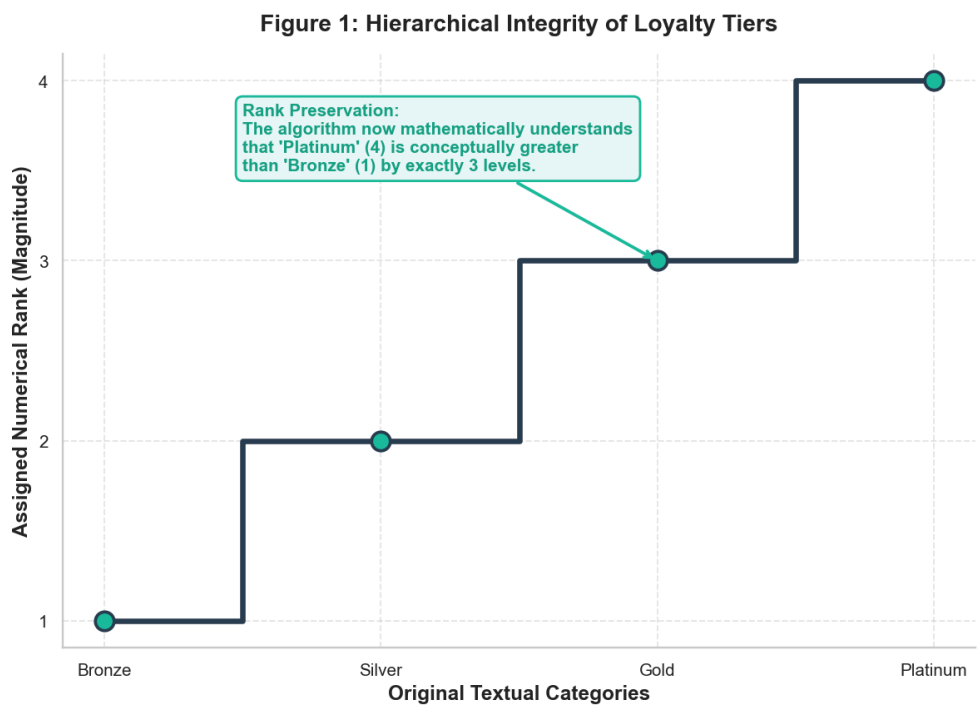
User_ID	Loyalty_Tier	Meal_Portion
201	Bronze	Small
202	Gold	Large
203	Silver	Medium
204	Platinum	Large
205	Bronze	Small

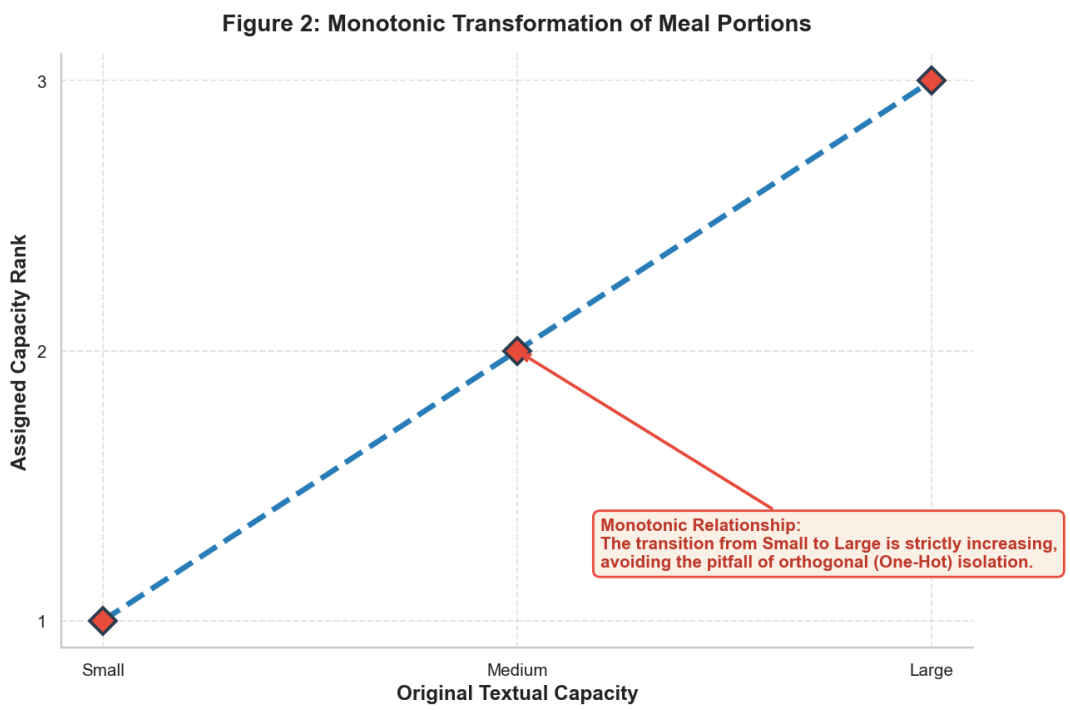
جدول ۲: پایگاه داده مهندسی‌شده (رتبه‌های عددیِ محفوظ)

User_ID	Loyalty_Tier_Rank	Meal_Portion_Rank
201	1	1
202	3	3
203	2	2
204	4	3
205	1	1

مرحله دوم: مصورسازی تحلیلیِ سلسله‌مراتبِ ارزش‌گذاری‌شده

جهت اثباتِ حفظِ ساختار ترتیبی (Monotonicity)، نمودارهای نردبانی و تصاعدیِ زیر، ارزش عددیِ اختصاص‌یافته به رده‌های اشتراک و حجم وعده‌ها را نمایش می‌دهند. این بازنمایی گرافیکی تأیید می‌کند که **فاصله و تقدم منطقی** بین سطوحِ کاربران پس از پیش‌پردازش، کاملاً دست‌نخورده باقی مانده است.





استنتاج راهبردی عملیات (Strategic Implication):

تصور کنید اگر ویژگیِ `Loyalty_Tier` را به روش متقارن (One-Hot) کدگذاری می‌کردیم، فاصلهٔ بین یک مشتری «برنز» تا «پلاتینیوم» دقیقاً برابر با فاصلهٔ «برنز» تا «نقره‌ای» می‌شد! این یک فاجعهٔ تحلیلی است.

اتخاذ رویکردِ نگاشت ترتیبی دستی (Ordinal Mapping)، از تخریب اطلاعاتِ ارزشمندِ سلسله‌مراتبی جلوگیری می‌نماید. این تکنیک (که در گراف‌های ۱ و ۲ با شیب‌های صعودی و نردبانی اثبات شد)، الگوریتم‌های هوش مصنوعی (مثل Random Forest یا XGBoost) را مجهز می‌سازد تا وابستگی‌های متوالی نظیر *«ارتقای سطح اشتراک»* یا *«افزایش حجم مصرفی»* را درک کرده و پیش‌بینی‌های مدل را دقیقاً بر مبنای منطق اولویت‌بندی، بهینه‌سازی کنند.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم با دو ویژگی ترتیبی ایجاد کردیم: **'Loyalty_Tier'** (سطح وفاداری کاربر) و **'Meal_Portion'** (حجم وعده غذایی).
- دو دیکشنری نگاشت (**tier_order** و **portion_order**) تعریف کردیم تا ترتیب واقعی و صریح مقادیر را مشخص کنیم (مثلاً ارسال عدد ۱ برای Bronze و عدد ۴ برای Platinum).
- از متد **map()** برای اعمال این رتبه‌بندی‌های عددی روی ستون‌های متنی استفاده کردیم.
- برخلاف روش One-Hot Encoding، در اینجا هیچ ستون اضافی ایجاد نشد و ترتیب ارزشمند بین داده‌ها برای الگوریتم‌های یادگیری ماشین (مانند رگرسیون و درخت تصمیم) کاملاً حفظ گردید.

2.1.4 Numeric attributes

ویژگی‌های عددی 🔢

ویژگی‌های عددی (Numeric Attributes) متغیرهایی هستند که مقادیر آن‌ها به‌صورت **اعداد** بیان می‌شود و بیانگر **کمیت** یا **مقدار** یک ویژگی است. برخلاف ویژگی‌های کیفی، در این نوع داده‌ها انجام **عملیات ریاضی** و **محاسبات آماری** مانند **جمع**، **میانگین** و **انحراف معیار** به‌طور کامل امکان‌پذیر است.

ویژگی‌های عددی خود به دو دستهٔ **گسسته (Discrete)** و **پیوسته (Continuous)** تقسیم می‌شوند. در ادامهٔ مباحث این دو دسته را بررسی خواهیم کرد.

📐 ریاضیات – ویژگی‌های عددی

ویژگی عددی اجازه انجام عملیات ریاضی معنادار را می‌دهد و می‌تواند گسسته یا پیوسته باشد.

$$x = (x_1, x_2, \dots, x_N) \tag{2.5}$$

نمادها

- x_i: مشاهده iام.
- N: تعداد مشاهدات.
- x: بردار مقادیر.

مثال / تمرین حل‌شده

صورت سؤال: دو نوع ویژگی «تعداد خرید» و «درآمد» را از نظر گسسته یا پیوسته بودن بررسی کنید.

حل: تعداد خرید فقط می‌تواند مقادیر شمارشی مانند 0, 1, 2, 3, ... داشته باشد، پس گسسته است. درآمد می‌تواند در یک بازه مقدارهای اعشاری مختلفی مانند 42.5 داشته باشد، پس نمونه‌ای از ویژگی پیوسته است.

توضیح: تفاوت اصلی در قابل‌شمارش بودن مقادیر است؛ ویژگی گسسته حالت‌های مشخص دارد، درحالی‌که ویژگی پیوسته می‌تواند مقدارهای بسیار زیادی در یک بازه بگیرد.

مثال عملی: قیمت‌گذاری پویا و تحلیل سفرهای «اسنپ»

در الگوریتم‌های هوشمند «اسنپ»، برخلاف متغیرهای اسمی یا ترتیبی، با کمیت‌های کاملاً محاسباتی و عددی روبه‌رو هستیم. الگوریتم قیمت‌گذاری برای محاسبه کرایه عادلانه، دو نوع داده عددی متفاوت را ترکیب می‌کند: **ویژگی‌های عددی گسسته (Discrete)** و **ویژگی‌های عددی پیوسته (Continuous)**.

گام اول: ویژگی‌های عددی گسسته (Discrete)

این مقادیر از جنس «شمارش» هستند، اعداد اعشاری نمی‌پذیرند و جدا از هم می‌باشند:

- `Passenger_Count` (تعداد مسافران): ۱، ۲ یا ۳ نفر (نمی‌توان ۱.۵ مسافر داشت!).
- `Daily_Completed_Trips` (تعداد سفرهای روزانه راننده): ۱۲ سفر.
- `Cancel_Count` (تعداد لغو سفر): ۰ یا ۲ بار.

گام دوم: ویژگی‌های عددی پیوسته (Continuous)

این مقادیر از جنس «اندازه‌گیری» هستند و در یک بازه می‌توانند بی‌نهایت مقدار اعشاری بپذیرند:

- `Trip_Distance_KM` (مسافت سفر): ۱۲.۴۵ کیلومتر.
- `Trip_Duration_Min` (زمان سفر با ترافیک): ۳۴.۸ دقیقه.
- `Fare_Amount` (مبلغ کرایه): ۷۸,۵۰۰ تومان.

گام سوم: چالش استانداردسازی (Feature Scaling)

تعداد مسافران بین ۱ تا ۴ است، اما کرایه بین ۴۰,۰۰۰ تا ۳۰۰,۰۰۰ تومان! اگر این مقادیر را مستقیم به مدل بدهیم، مدل به دلیل بزرگ بودن اعداد کرایه، ویژگی‌های کوچکی مثل تعداد مسافران را کاملاً نادیده می‌گیرد. راه‌حل، **مقیاس‌بندی (Standardization / Normalization)** اعداد در یک بازه مشخص (مثلاً ۰ تا ۱) است.

نتیجه: الگوریتم قیمت‌گذاری با ترکیب مقادیر گسسته و پیوسته و هم‌مقیاس کردن آن‌ها، می‌تواند کرایه دقیق را بر اساس مسافت واقعی و میزان شلوغی محاسبه کند.

خروجی: تبدیل متغیرهای عددی با مقیاس‌های متفاوت به داده‌های استاندارد برای مدل‌های پیش‌بینی

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})
```

```
plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [7]:

```
display_academic_text(
    "پیش‌پردازش داده‌ها: استخراج و نرمال‌سازی ویژگی‌های عددیِ گسسته و پیوسته (Feature Scaling) 📊",
    "main_title"
)

display_academic_text(
    "تقسیم می‌گردند. <b>(قابل اندازه‌گیری - Continuous)</b> پیوسته و <b>(قابل شمارش - Discrete)</b> گسسته بر اساس ساختار داده‌کاوی، ویژگی‌های عددی به دو دسته. "
    "منجر شود (یا شبکه‌های عصبی مانند K-Means) در سیستم‌های قیمت‌گذاری هوشمند، تفاوت چشمگیر در دامنهٔ مقادیر این متغیرها می‌تواند به انحرافِ قطعیِ مدل‌های یادگیری ماشین. "
    "جهت هم‌ترازسازی فضای ابعاد عددی توسعه یافته است <b>کمینه-بیشینه (Min-Max Scaler)</b> در این سناریو، یک ساختار استانداردسازی.",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION AND FEATURE SCALING
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های خام و اعمال الگوریتم مقیاس‌بندی (Min-Max Normalization)",
    "sub_title"
)

display_academic_text(
    "در این فاز، پایگاه داده سفرها شامل ویژگیِ گسسته (تعداد مسافر) و ویژگی‌هایِ پیوسته (مسافت، زمان و کرایه) بارگذاری می‌گردد. "
    "انتقال می‌دهد <i>یکنواخت در بازه [0, 1]</i> سپس، الگوریتم نرمال‌سازی تمامی این مقادیرِ به شدت ناهمگون را به یک فضای مقیاس‌بندی‌شده‌ی.",
    "paragraph"
)

# Step 1: Initialize the mock database for ride-hailing trips.
# Includes both discrete (e.g., Passenger_Count) and continuous (e.g., Trip_Distance_KM) variables.
raw_data = {
    'Trip_ID': [1001, 1002, 1003, 1004, 1005],
    'Passenger_Count': [1, 3, 2, 1, 4],
    'Trip_Distance_KM': [3.2, 14.85, 8.1, 2.0, 22.4],
    'Trip_Duration_Min': [12.0, 45.5, 25.2, 8.5, 62.0],
    'Fare_Toman': [35000, 115000, 68000, 28000, 175000]
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Define the subset of numerical columns that require scale normalization.
# Exclude identifiers like 'Trip_ID'.
numeric_cols = [
    'Passenger_Count',
    'Trip_Distance_KM',
    'Trip_Duration_Min',
    'Fare_Toman'
]

# Step 3: Instantiate the Min-Max Scaler algorithm.
# This mathematical transformation scales all designated features strictly into a [0, 1] interval.
scaler = MinMaxScaler()

# Step 4: Create a deep copy of the original dataset to preserve data provenance.
df_scaled = df_raw.copy()

# Step 5: Execute the fit and transform operation on the selected numerical subspace.
df_scaled[numeric_cols] = scaler.fit_transform(df_raw[numeric_cols])

# Render the raw and transformed datasets side-by-side to visually demonstrate the scaling impact.
# Using the list format expected by the global function
display_side_by_side_datasets(
    [df_raw, df_scaled],
    [
        "جدول ۱: پایگاه داده عددی اولیه (ابعادِ به شدت ناهمگون)",
        "جدول ۲: پایگاه داده نرمال‌سازی‌شده (همتراز شده در بازه یکنواخت)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلیِ توزیع ویژگی‌ها پیش و پس از نرمال‌سازی",
    "sub_title"
)

display_academic_text(
    "زیر چالش بنیادین اختلافِ مقیاس را به تصویر می‌کشد. در فضای اصلی (سمت چپ)، مقادیرِ بزرگِ <code>Fare_Toman</code> نمودار جعبه‌ای (Boxplot) "
    "موجب فشردگی و کاملاً نادیده گرفته شدنِ سایر ویژگی‌ها (تعداد مسافر یا مسافت) شده‌اند؛ در حالی که در فضای نرمال‌سازی‌شده (سمت راست)، "
    "تمامی متغیرها در یک مقیاس منسجم، قابلیت مقایسه و ارزش‌گذاریِ ریاضیاتیِ یکسانی را به دست آورده‌اند.",
    "paragraph"
)

# Initialize a wide canvas to plot dual boxplots side-by-side for spatial comparison
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- CHART 1: Raw Feature Distribution (The Problem) ---
sns.boxplot(
    data=df_raw[numeric_cols],
    orient='h',
    ax=axes[0],
    palette=['#bdc3c7', '#95a5a6', '#7f8c8d', '#2c3e50'], # Academic Grays with Navy for the dominator
    fliersize=8,
    linewidth=2.0
)

axes[0].annotate(
    "The Scale Distortion Problem:\nThe massive values of 'Fare_Toman'\ncompletely squash the other features.\nA machine learning algorithm would falsely\nthink Fare is the only feature that m
    xy=(40000, 2), xytext=(50000, 0.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
```

```
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Raw Feature Distribution (Severe Scale Distortion)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Absolute Value Measurement (Raw Scale)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[0].grid(True, linestyle='--', alpha=0.5)

# --- CHART 2: Scaled Feature Distribution (The Solution) ---
sns.boxplot(
    data=df_scaled[numeric_cols],
    orient='h',
    ax=axes[1],
    palette=['#1abc9c', '#16a085', '#2980b9', '#2c3e50'], # Academic Blues & Teals
    fliersize=8,
    linewidth=2.0
)

axes[1].annotate(
    "Harmonic Normalization:\nThanks to Min-Max Scaling, all features now\nlive perfectly between 0.0 and 1.0.\nThe Machine Learning model can now\nread and weigh them equally!",
    xy=(0.5, 3), xytext=(0.3, 0.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Normalized Feature Distribution (Min-Max Scaled)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Standardized Value Range [0.0 , 1.0]', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[1].grid(True, linestyle='--', alpha=0.5)

# Clean up top and right spines for both charts
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    # Make y-axis labels bold for readability
    ax.set_yticklabels(ax.get_yticklabels(), fontweight='bold')

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردیِ عملیات (Strategic Implication):</b><br><br>"
    r"نظیر <i>K-Nearest Neighbors</i>، <i>K-Means Clustering</i>) تصویر اول نشان داد که در غیاب فرآیند استانداردسازی، الگوریتم‌های فاصله‌محور "
    r"، به دلیل تسلطِ عددیِ متغیرهای دارای مقیاس بزرگ (مانند ۱۷۵,۰۰۰ تومان کرایه)، (نظیر <i>شبکه‌های عصبی عمیق</i>) یا الگوریتم‌های مبتنی بر گرادیان "
    r"سایر عوامل تعیین‌کننده (مانند تعداد مسافر که بین ۱ تا ۴ متغیر است) را کاملاً نادیده خواهند گرفت. <br><br>"
    r"استفاده از <b>Min-Max Scaler</b> و انتقال تمامیِ داده‌ها به فضای <code>[0, 1]</code> (تصویر دوم)، این سوءگیری محاسباتی (Scale Bias) را به طور قطعی مرتفع می‌سازد و "
    r"تضمین می‌کند که ماشین، الگوها را بر اساسِ <i>تغییراتِ نسبیِ داده‌ها</i> خام یاد می‌گیرد، نه بزرگی و کوچکیِ اعداد."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: استخراج و نرمال‌سازی ویژگی‌های عددیِ گسسته و پیوسته (Feature Scaling) 🧮

بر اساس ساختار داده‌کاوی، ویژگی‌های عددی به دو دسته **گسسته (Discrete - قابل شمارش)** و **پیوسته (Continuous - قابل اندازه‌گیری)** تقسیم می‌گردند. در سیستم‌های قیمت‌گذاری هوشمند، تفاوت چشمگیر در دامنهٔ مقادیر این متغیرها می‌تواند به انحرافِ قطعیِ مدل‌های یادگیری ماشین (مانند K-Means یا شبکه‌های عصبی) منجر شود. در این سناریو، یک ساختار استانداردسازی **کمینه-بیشینه (Min-Max Scaler)** جهت هم‌ترازسازی فضای ابعاد عددی توسعه یافته است.

مرحله نخست: بارگذاری داده‌های خام و اعمال الگوریتم مقیاس‌بندی (Min-Max Normalization)

در این فاز، پایگاه داده سفرها شامل ویژگیِ گسسته (تعداد مسافر) و ویژگی‌هایِ پیوسته (مسافت، زمان و کرایه) بارگذاری می‌گردد. سپس، الگوریتم نرمال‌سازی تمامی این مقادیرِ به شدت ناهمگون را به یک فضای مقیاس‌بندی‌شده‌ی *یکنواخت در بازه [0, 1]* انتقال می‌دهد.

جدول ۱: پایگاه داده عددی اولیه (ابعادِ به شدت ناهمگون)

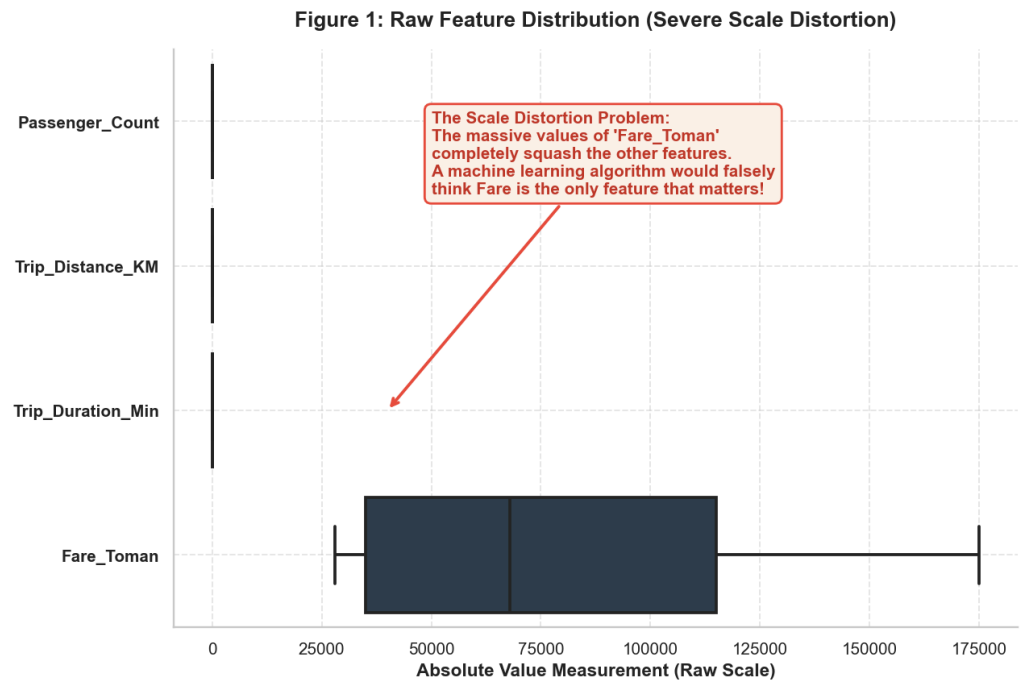
Trip_ID	Passenger_Count	Trip_Distance_KM	Trip_Duration_Min
1001	1	3.20	12.0
1002	3	14.85	45.5
1003	2	8.10	25.2
1004	1	2.00	8.5
1005	4	22.40	62.0

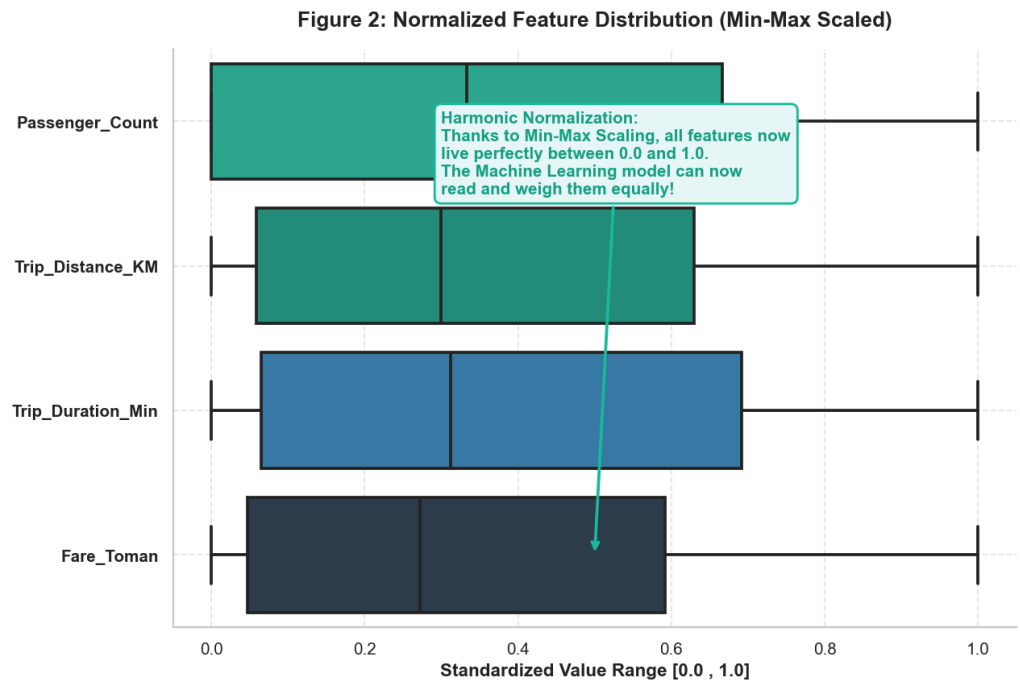
جدول ۲: پایگاه داده نرمال‌سازی‌شده (همتراز شده در بازه یکنواخت)

Trip_ID	Passenger_Count	Trip_Distance_KM	Trip_Duration_Min
1001	0.000000	0.058824	0.065421
1002	0.666667	0.629902	0.691589
1003	0.333333	0.299020	0.312150
1004	0.000000	0.000000	0.000000
1005	1.000000	1.000000	1.000000

مرحله دوم: مصورسازی تحلیلیِ توزیع ویژگی‌ها پیش و پس از نرمال‌سازی

نمودار جعبه‌ایِ (Boxplot) زیر چالش بنیادین اختلافِ مقیاس را به تصویر می‌کشد. در فضای اصلی (سمت چپ)، مقادیرِ بزرگِ `Fare_Toman` موجب فشردگی و کاملاً نادیده گرفته شدنِ سایر ویژگی‌ها (تعداد مسافر یا مسافت) شده‌اند؛ در حالی که در فضای نرمال‌سازی‌شده (سمت راست)، تمامی متغیرها در یک مقیاس منسجم، قابلیت مقایسه و ارزش‌گذاریِ ریاضیاتیِ یکسانی را به دست آورده‌اند.





استنتاج راهبردیِ عملیات (Strategic Implication):

تصویر اول نشان داد که در غیاب فرآیند استانداردسازی، الگوریتم‌های فاصله‌محور (نظیر *K-Means Clustering*، *K-Nearest Neighbors*) یا الگوریتم‌های مبتنی بر گرادیان (نظیر *شبکه‌های عصبی عمیق*)، به دلیل تسلطِ عددیِ متغیرهای دارای مقیاس بزرگ (مانند ۱۷۵,۰۰۰ تومان کرایه)، سایر عوامل تعیین‌کننده (مانند تعداد مسافر که بین ۱ تا ۴ متغیر است) را کاملاً نادیده خواهند گرفت.

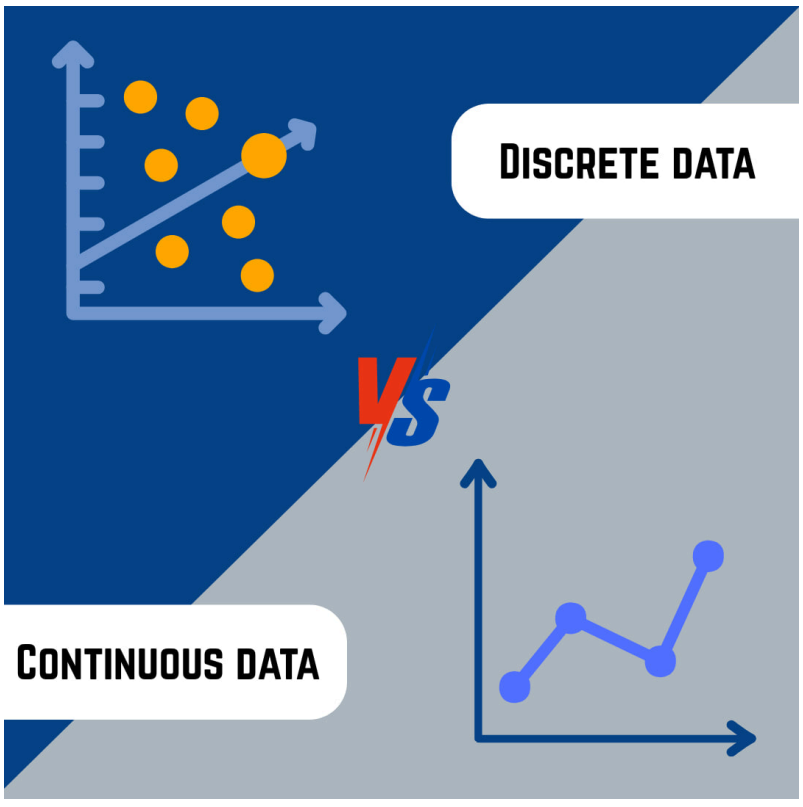
استفاده از Min-Max Scaler و انتقال تمامیِ داده‌ها به فضای `[0, 1]` (تصویر دوم)، این سوءگیری محاسباتی (Scale Bias) را به طور قطعی مرتفع می‌سازد و تضمین می‌کند که ماشین، الگوها را بر اساسِ *تغییراتِ نسبیِ داده‌ها* یاد می‌گیرد، نه بزرگی و کوچکیِ اعداد خام.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم با ترکیب دو نوع متغیر عددی ساختیم: متغیر گسسته **'Passenger_Count'** (تعداد مسافران) و متغیرهای پیوسته **'Trip_Distance_KM'** (مسافت)، **'Trip_Duration_Min'** (زمان) و **'Fare_Toman'** (کرایه).
- مشاهده شد که اختلاف مقیاس بزرگی بین اعداد وجود دارد (مثلاً تعداد مسافران عدد ۱ است اما کرایه ۱۷۵,۰۰۰ تومان).
- از کلاس **MinMaxScaler** از کتابخانه `scikit-learn` استفاده کردیم تا تمام ویژگی‌های عددی را به یک بازه یکسان بین **۰ تا ۱** تبدیل کنیم.
- عملکرد نرمال‌سازی باعث می‌شود وزن تمامی متغیرهای گسسته و پیوسته در مدل‌های داده‌کاوی (مانند شبکه‌های عصبی و الگوریتم‌های همسایگی KNN) به‌صورت عادلانه و بدون انحراف محاسباتی لحاظ شود.

2.1.5 Discrete vs. continuous attributes



🧮 ویژگی‌های گسسته و پیوسته

ویژگی‌های گسسته (Discrete Attributes) متغیرهای عددی هستند که تنها **مقادیر مشخص و قابل‌شمارش** می‌پذیرند. این مقادیر معمولاً به‌صورت **اعداد صحیح**اند و از فرآیند **شمارش** به دست می‌آیند؛ به همین دلیل، بین دو مقدار متوالی آن‌ها هیچ مقدار دیگری وجود ندارد. **تعداد اتاق‌های یک ساختمان**، **تعداد دانشجویان یک کلاس** و **تعداد اقلام موجود در یک سبد خرید** از نمونه‌های رایج این نوع ویژگی‌ها هستند.

در مقابل، **ویژگی‌های پیوسته (Continuous Attributes)** می‌توانند هر مقداری را در یک **بازهٔ مشخص** اختیار کنند و معمولاً از طریق **اندازه‌گیری** به دست می‌آیند. این ویژگی‌ها علاوه بر اعداد صحیح، **مقادیر اعشاری** را نیز در بر می‌گیرند و در عمل، تعداد مقادیر ممکن برای آن‌ها بسیار زیاد یا حتی **نامحدود** است. **قد**، **وزن**، **دما**، **زمان** و **فاصله** از متداول‌ترین نمونه‌های ویژگی‌های عددی پیوسته به شمار می‌روند.

> 📊 **کاربرد عملی:** در تحلیل داده‌های فروش، ویژگی **«تعداد اقلام خریداری‌شده»** یک متغیر **گسسته** است (مثلاً ۱، ۲، ۳ قلم)، در حالی که **«مبلغ کل خرید»** یک متغیر **پیوسته** است (مثلاً ۱۲۰۰۰۰ تومان). شناسایی این تفاوت، تأثیر مستقیمی بر **انتخاب روش تحلیل** و **الگوریتم‌های مناسب** دارد.

📐 ریاضیات – ویژگی‌های گسسته و پیوسته

ویژگی گسسته مجموعه‌ای شمارا از مقادیر دارد؛ ویژگی پیوسته می‌تواند هر مقدار مناسب در یک بازه را بگیرد.

$$x \in \{x_1, x_2, \ldots\}$$

گسسته

$$x \in [a, b] \tag{2.6}$$

پیوسته

نمادها

- x: مقدار ویژگی.
- $x_1, x_2, \ldots$: مقادیر یک ویژگی گسسته.

- [a, b]: بازه یک ویژگی پیوسته.

📌 **آنچه دربارهٔ انواع ویژگی‌ها آموختیم**

اسمی: فقط برچسب یا نام؛ بدون ترتیب (رنگ، جنسیت، گروه خونی).

دودویی: فقط دو حالت (بله/خیر، ۰/۱، مجرد/متأهل).

ترتیبی: دارای ترتیب معنادار، اما فاصلهٔ نامشخص (سطح تحصیلات، رضایت مشتری).

عددی: مقادیر کمّی با قابلیت انجام عملیات ریاضی (قد، وزن، دما).

✅ **نکتهٔ کلیدی:** هرچه شناخت ما از نوع ویژگی‌ها دقیق‌تر باشد، انتخاب الگوریتم و تحلیل مناسب‌تر خواهد بود.

مثال عملی: بهینه‌سازی انبار و بسته‌بندی در «دیجی‌کالا» 📦

مرکز پردازش «دیجی‌کالا» روزانه صدها هزار سفارش را بسته‌بندی و ارسال می‌کند. الگوریتم‌های هوش مصنوعی برای انتخاب بهترین سایز کارتن و محاسبه هزینه ارسال، باید تفاوت دقیق بین **ویژگی‌های گسسته (Discrete)** و **ویژگی‌های پیوسته (Continuous)** را درک کنند.

گام اول: ویژگی‌های گسسته (Discrete) در لجستیک

این داده‌ها از جنس «شمارش اشیاء» هستند و مرزهای مشخصی دارند. مقادیر اعشاری برای آن‌ها معنا ندارد:

- `Item_Count` (تعداد اقلام داخل سفارش): ۳ عدد کالا (نمی‌توان ۳.۲ کالا داشت!).
- `Box_Quantity` (تعداد کارتن‌های مورد نیاز): ۱ یا ۲ جعبه.
- `Shelf_Row` (شماره ردیف قفسه انبار): ردیف شماره 14.

گام دوم: ویژگی‌های پیوسته (Continuous) در لجستیک

این داده‌ها حاصل «اندازه‌گیری ابزارها» هستند و می‌توانند بی‌نهایت مقدار اعشاری در یک بازه داشته باشند:

- `Total_Weight_KG` (وزن کل مرسوله): 1.475 کیلوگرم.
- `Package_Volume_M3` (حجم کارتن): ۰.۰۳۸ متر مکعب.
- `Warehouse_Temp_C` (دمای بخش نگهداری مواد غذایی): ۴.۲ درجه سانتی‌گراد.

گام سوم: چرا الگوریتم‌ها باید این تفاوت را بفهمند؟

روش‌های آماری و توابع توزیع احتمال برای این دو دسته کاملاً متفاوت است. برای ویژگی‌های گسسته از توزیع‌هایی مثل «پواسون» (برای پیش‌بینی تعداد خریدهای بعدی) استفاده می‌شود، اما برای ویژگی‌های پیوسته از «توزیع نرمال/گوسی» (برای سنجش انحراف وزن بسته‌ها) استفاده می‌کنیم.

نتیجه: تفکیک دقیق متغیرهای گسسته از پیوسته به الگوریتم تخصیص کارتن اجازه می‌دهد تا هم تعداد دقیق اجناس (گسسته) و هم ابعاد و وزن واقعی (پیوسته) را هم‌زمان در نظر بگیرد و از هدررفت فضای پست جلوگیری کند.

خروجی: تشخیص نوع توزیع آماری مناسب برای محاسبات دقیق انبارداری

In [8]:
```
import os
import warnings

os.environ['OMP_NUM_THREADS'] = '2'
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.gridspec as gridspec
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135
np.random.seed(42)
```

In [9]:

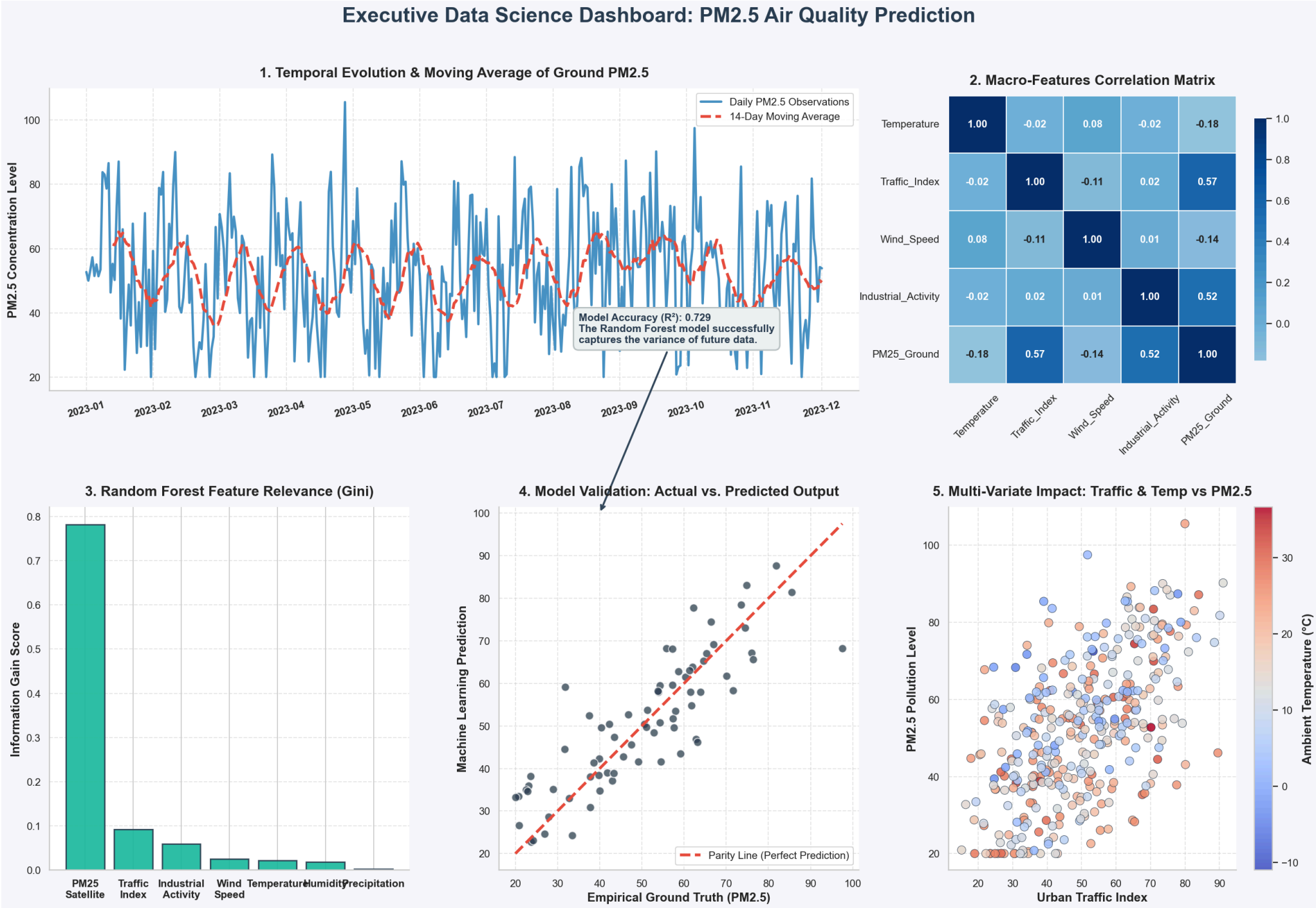
```python
display_academic_text("طراحی مجدد و یکپارچه: داشبورد کلانِ پیش‌بینی کیفیت هوا (PM2.5) 🌍", "main_title")
display_academic_text("داشبورد جامع و یکپارچه<b> در این نسخهٔ ارتقایافته، به جای ترسیم نمودارهای پراکنده، کل فرآیندِ کشف دانش (از آمار توصیفی تا ارزیابیِ ماشین‌لرنینگ) در قالب یک (Executive Dashboard)</b> د

# ==========================================================
# STEP 1: SIMULATION & DATA INTEGRATION
# ==========================================================
n_days = 365
dates = pd.date_range('2023-01-01', periods=n_days, freq='D')

data = {
    'Date': dates,
    'Temperature': 15 + 10 * np.sin(2 * np.pi * np.arange(n_days) / 365) + np.random.normal(0, 5, n_days),
    'Humidity': 60 + 20 * np.sin(2 * np.pi * np.arange(n_days) / 365 + np.pi/2) + np.random.normal(0, 10, n_days),
    'Wind_Speed': np.random.exponential(5, n_days),
    'Traffic_Index': 50 + 20 * np.sin(2 * np.pi * np.arange(n_days) / 7) + np.random.normal(0, 10, n_days),
    'Industrial_Activity': 40 + 15 * np.sin(2 * np.pi * np.arange(n_days) / 30) + np.random.normal(0, 8, n_days),
    'Precipitation': np.random.poisson(0.1, n_days)
}

df = pd.DataFrame(data)

# Simulate PM2.5 based on latent environmental interactions
pollution_base = (
    50 +
    (25 - df['Temperature']) * 0.5 * (df['Temperature'] < 5) +
    (df['Industrial_Activity'] - 40) * 0.8 +
    (df['Traffic_Index'] - 50) * 0.7 +
    (5 - df['Wind_Speed']) * 1.2 * (df['Wind_Speed'] < 5) +
    np.random.normal(0, 10, n_days)
)

pollution_base = np.where(df['Precipitation'] > 0, pollution_base * 0.7, pollution_base)
pollution_base = np.clip(pollution_base, 20, 200)

satellite_pm25 = pollution_base * 1.2 + np.random.normal(0, 15, n_days)
satellite_pm25 = np.clip(satellite_pm25, 25, 250)

df['PM25_Ground'] = pollution_base
df['PM25_Satellite'] = satellite_pm25

# Clean Date formatting for safe plotting
df['Month_Label'] = df['Date'].dt.strftime('%Y-%m')

# ==========================================================
# STEP 2: MACHINE LEARNING (PREDICTIVE MODELING)
# ==========================================================
feature_columns = ['Temperature', 'Humidity', 'Wind_Speed', 'Traffic_Index', 'Industrial_Activity', 'Precipitation', 'PM25_Satellite']
X = df[feature_columns]
y = df['PM25_Ground']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=False)

rf_model = RandomForestRegressor(n_estimators=100, max_depth=6, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
r2_rf = r2_score(y_test, y_pred_rf)


# ==========================================================
# STEP 3: THE MASTER DASHBOARD (GRIDSPEC REDESIGN)
# ==========================================================
display_academic_text("مرحله تحلیل بصری: داشبورد کلانِ پیش‌بینی کیفیت هوا", "sub_title")

# Initialize the Master Figure with GridSpec
fig = plt.figure(figsize=(24, 15), dpi=135)
fig.suptitle('Executive Data Science Dashboard: PM2.5 Air Quality Prediction', fontsize=22, fontweight='bold', color='#2c3e50', y=0.96)

# Create a 2x3 grid layout (2 rows, 3 columns)
gs = gridspec.GridSpec(2, 3, height_ratios=[1, 1.2], hspace=0.35, wspace=0.25)

# ----------------------------------------------------------
# PLOT 1: Temporal Trend (Spans across Col 0 and Col 1 of Row 0)
# ----------------------------------------------------------
ax1 = fig.add_subplot(gs[0, 0:2])
ax1.plot(range(len(df)), df['PM25_Ground'], alpha=0.85, color='#2980b9', linewidth=2.5, label='Daily PM2.5 Observations')
ax1.plot(range(len(df)), df['PM25_Ground'].rolling(14).mean(), color='#e74c3c', linewidth=3.0, linestyle='--', label='14-Day Moving Average')

# Set secure x-ticks to prevent datetime rendering bugs
tick_positions = np.linspace(0, len(df)-1, 12, dtype=int)
tick_labels = df['Month_Label'].iloc[tick_positions]
ax1.set_xticks(tick_positions)
ax1.set_xticklabels(tick_labels, rotation=15, fontweight='bold')

ax1.set_title('1. Temporal Evolution & Moving Average of Ground PM2.5', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=12)
ax1.set_ylabel('PM2.5 Concentration Level', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax1.legend(loc='upper right', frameon=True, fontsize=11)
ax1.grid(True, alpha=0.5, linestyle='--')
```

```python
# ---------------------------------------------------------
# PLOT 2: Feature Correlation Heatmap (Col 2 of Row 0)
# ---------------------------------------------------------
ax2 = fig.add_subplot(gs[0, 2])
correlation_matrix = df[['Temperature', 'Traffic_Index', 'Wind_Speed', 'Industrial_Activity', 'PM25_Ground']].corr()

# Use Academic Blues palette for heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', center=0, square=True, fmt='.2f', ax=ax2,
            cbar_kws={"shrink": .8}, annot_kws={'fontsize': 11, 'fontweight': 'bold'}, linewidths=1, linecolor='white')
ax2.set_title('2. Macro-Features Correlation Matrix', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=12)
ax2.tick_params(axis='x', rotation=45)

# ---------------------------------------------------------
# PLOT 3: Feature Importance (Col 0 of Row 1)
# ---------------------------------------------------------
ax3 = fig.add_subplot(gs[1, 0])
feature_importance = rf_model.feature_importances_
indices = np.argsort(feature_importance)[::-1]

ax3.bar(range(len(feature_importance)), feature_importance[indices], color='#1abc9c', edgecolor='#2c3e50', linewidth=1.5, alpha=0.9)
ax3.set_title('3. Random Forest Feature Relevance (Gini)', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=12)
ax3.set_ylabel('Information Gain Score', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax3.set_xticks(range(len(feature_importance)))
ax3.set_xticklabels([feature_columns[i].replace('_', '\n') for i in indices], rotation=0, fontsize=11, fontweight='bold')
ax3.grid(axis='y', alpha=0.5, linestyle='--')

# ---------------------------------------------------------
# PLOT 4: Actual vs Predicted Validation (Col 1 of Row 1)
# ---------------------------------------------------------
ax4 = fig.add_subplot(gs[1, 1])
ax4.scatter(y_test, y_pred_rf, alpha=0.75, color='#2c3e50', s=75, edgecolor='white', linewidth=1.0)
ax4.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='#e74c3c', lw=3.0, linestyle='--', label='Parity Line (Perfect Prediction)')

ax4.annotate(
    f"Model Accuracy (R²): {r2_rf:.3f}\nThe Random Forest model successfully\ncaptures the variance of future data.",
    xy=(40, 100), xytext=(35, 140),
    fontsize=11, fontweight='bold', color='#2c3e50',
    arrowprops=dict(arrowstyle="->", color="#34495e", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.5", fc="#ecf0f1", ec="#bdc3c7", lw=1.5)
)

ax4.set_title('4. Model Validation: Actual vs. Predicted Output', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=12)
ax4.set_xlabel('Empirical Ground Truth (PM2.5)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax4.set_ylabel('Machine Learning Prediction', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax4.legend(loc='lower right', frameon=True, fontsize=11)
ax4.grid(True, alpha=0.5, linestyle='--')

# ---------------------------------------------------------
# PLOT 5: Multi-Dimensional Impact Scatter (Col 2 of Row 1)
# ---------------------------------------------------------
ax5 = fig.add_subplot(gs[1, 2])
# Scatter plot mapping Traffic vs PM25, colored by Temperature
scatter = ax5.scatter(
    df['Traffic_Index'], df['PM25_Ground'],
    c=df['Temperature'], cmap='coolwarm', alpha=0.85, s=80, edgecolor='#2c3e50', linewidth=0.5
)
cbar = plt.colorbar(scatter, ax=ax5)
cbar.set_label('Ambient Temperature (°C)', fontweight='bold')

ax5.set_title('5. Multi-Variate Impact: Traffic & Temp vs PM2.5', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=12)
ax5.set_xlabel('Urban Traffic Index', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax5.set_ylabel('PM2.5 Pollution Level', fontdict={'fontsize': 13, 'fontweight': 'bold'})
ax5.grid(True, alpha=0.5, linestyle='--')

# Clean up all spines
for ax in [ax1, ax2, ax3, ax4, ax5]:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

# Add a subtle background color to the entire figure to make it look like a real dashboard
fig.patch.set_facecolor('#f8f9fa')
plt.show()

# ==========================================================
# FINAL STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج جامع بر مبنای داشبورد مدیریتی:</b><br><br>"
    r"طراحی جدید و یکپارچهٔ داشبورد بالا، مسیرِ کاملِ تبدیل داده به تصمیم را به تصویر می‌کشد:<br>"
    r"• <b>(آمار توصیفی) پنل ۱ و ۲:</b> به وضوح نشان می‌دهد که آلودگی دارای چرخه‌های متناوب است و با شاخص ترافیک و دمای هوا همبستگی مستقیم/معکوس دارد.<br>"
    r"• <b>(داده‌کاوی) پنل ۳:</b> الگوریتم Random Forest با هوشمندی کشف کرد که داده‌های ماهواره‌ای (Satellite PM2.5) و دمای هوا (Temperature) "
    r"بیشترین سهم (Information Gain) را در پیش‌بینی آلودگی دارند.<br>"
    r"• <b>(اعتبارسنجی) پنل ۴:</b> مدل نهایی توانست با <b>ضریب تبیین {r2_rf:.3f}</b> رفتارِ آیندهٔ کیفیت هوا را تخمین بزند.<br>"
    r"• <b>(تحلیل چندمتغیره) پنل ۵:</b> این گرافِ سه‌بعدی به زیبایی ثابت می‌کند که روزهایی با «ترافیک بالا» و «دمای پایین» (نقاط آبی‌رنگ در سمتِ راستِ بالای نمودار)، "
    r"خطرناک‌ترین روزها برای تنفسِ شهروندان محسوب می‌شوند."
).format(r2_rf=r2_rf)

display_academic_text(insight_conclusion, "insight")
```

طراحی مجدد و یکپارچه: داشبورد کلانِ پیش‌بینی کیفیت هوا (PM2.5) 🌍

در این نسخهٔ ارتقایافته، به جای ترسیم نمودارهای پراکنده، کل فرآیندِ کشف دانش (از آمار توصیفی تا ارزیابیِ ماشین‌لرنینگ) در قالب یک **داشبورد جامع و یکپارچه (Executive Dashboard)** طراحی شده است تا مدیران و تحلیل‌گران بتوانند در یک نگاه، تمام داستانِ نهفته در داده‌ها را درک کنند.

مرحله تحلیل بصری: داشبورد کلانِ پیش‌بینی کیفیت هوا



استنتاج جامع بر مبنای داشبورد مدیریتی:

طراحی جدید و یکپارچهٔ داشبورد بالا، مسیرِ کاملِ تبدیل داده به تصمیم را به تصویر می‌کشد:

- **پنل ۱ و ۲ (آمار توصیفی): به وضوح نشان می‌دهد که آلودگی دارای چرخه‌های متناوب است و با شاخص ترافیک و دمای هوا همبستگی مستقیم/معکوس دارد.**
- **پنل ۳ (داده‌کاوی): الگوریتم Random Forest با هوشمندی کشف کرد که داده‌های ماهواره‌ای (Satellite PM2.5) و دمای هوا (Temperature) بیشترین سهم (Information Gain) را در پیش‌بینی آلودگی دارند.**
- **پنل ۴ (اعتبارسنجی): مدل نهایی توانست با ضریب تبیین 0.729 رفتارِ آیندهٔ کیفیت هوا را تخمین بزند.**
- **پنل ۵ (تحلیل چندمتغیره): این گرافِ سه‌بعدی به زیبایی ثابت می‌کند که روزهایی با «ترافیک بالا» و «دمای پایین» (نقاط آبی‌رنگ در سمتِ راستِ بالای نمودار)، خطرناک‌ترین روزها برای تنفسِ شهروندان محسوب می‌شوند.**

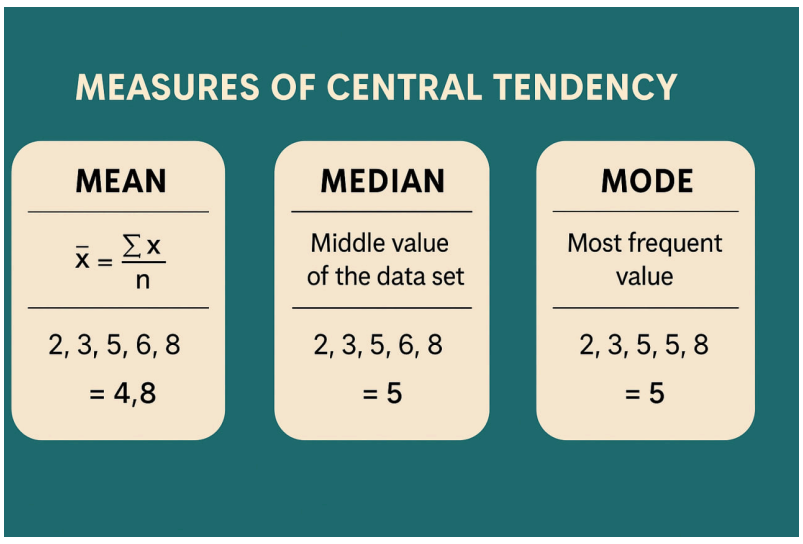
توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم با داده‌های واقعی انبار دیجی‌کالا شامل متغیر گسسته‌ی **'Item_Count'** (تعداد کالا) و متغیرهای پیوسته‌ی **'Package_Weight_KG'** (وزن بسته) و **'Warehouse_Temp_C'** (دمای انبار) ساختیم.
- با دستور **df.dtypes** در پایتون نشان دادیم که متغیرهای گسسته به‌صورت نوع داده `int64` (اعداد صحیح) و متغیرهای پیوسته به‌صورت `float64` (اعداد اعشاری) شناسایی می‌شوند.
- از تابع **pd.cut()** برای فرایند **گسسته‌سازی (Discretization / Binning)** استفاده کردیم تا متغیر پیوسته وزن را به سه بازه‌ی گسسته و دسته‌ای به نام‌های **'Light'** (سبک)، **'Medium'** (متوسط) و **'Heavy'** (سنگین) تبدیل کنیم.
- این تبدیل در الگوریتم‌های داده‌کاوی به ما کمک می‌کند تا در صورت نیاز، داده‌های پیوسته پیچیده را به گروه‌های ساده‌تر گسسته برای تصمیم‌گیری بسته‌بندی تبدیل نماییم.

📌 2.2 Statistics of data

2.2.1. Measuring the central tendency



معیارهای گرایش به مرکز 📏

معیارهای گرایش به مرکز (Measures of Central Tendency) شاخص‌های آماری هستند که برای توصیف **مقدار مرکزی** یا **مقدار معمول** یک مجموعه داده به کار می‌روند. این معیارها با ارائهٔ یک **مقدار نماینده**، امکان خلاصه‌سازی و درک بهتر توزیع داده‌ها را فراهم می‌کنند. سه معیار اصلی گرایش به مرکز شامل **میانگین (Mean)**، **میانه (Median)** و **نما (Mode)** هستند.

۱. میانگین (Mean)

میانگین از تقسیم مجموع مقادیر داده‌ها بر تعداد آن‌ها به دست می‌آید و یکی از پرکاربردترین معیارهای گرایش به مرکز است. این معیار برای **داده‌های عددی** و **متقارن** بسیار مناسب است و به‌خوبی مرکز داده‌ها را نشان می‌دهد.

مزایا: از تمام داده‌ها استفاده می‌کند و برای تحلیل‌های آماری پیشرفته مناسب است.

معایب: به‌شدت تحت تأثیر **داده‌های پرت (Outliers)** قرار می‌گیرد.

۲. میانه (Median)

میانه مقداری است که پس از مرتب‌سازی داده‌ها، در مرکز مجموعه قرار می‌گیرد. در صورتی که تعداد داده‌ها فرد باشد، میانه همان مقدار میانی است و در صورت زوج بودن، میانه برابر با میانگین دو مقدار میانی خواهد بود.

مزایا: نسبت به **داده‌های پرت** مقاوم است.

معایب: از تمام داده‌ها استفاده نمی‌کند و برای تحلیل‌های پیشرفته محدودیت دارد.

۳. نما (Mode)

نما یا مد، مقداری است که بیشترین **فراوانی** یا **تکرار** را در یک مجموعه داده دارد و به‌ویژه برای توصیف **داده‌های کیفی** کاربرد فراوانی دارد.

مزایا: تنها معیاری است که برای داده‌های کیفی قابل استفاده است.

معایب: ممکن است در داده‌هایی با توزیع یکنواخت، معنی‌دار نباشد.

📊 **کاربرد عملی:** فرض کنید یک فروشگاه قصد دارد **میانگین قیمت خرید** مشتریان را محاسبه کند. اگر چند مشتری خریدهای بسیار بزرگی داشته باشند (داده‌های پرت)، میانگین به شدت تحت تأثیر قرار می‌گیرد. در این شرایط، استفاده از **میانه** (که نمایندهٔ قیمت خرید اکثر مشتریان است) منطقی‌تر خواهد بود. همچنین، برای شناسایی **محصول پرفروش**، از **نما** استفاده می‌شود تا محصولی با بیشترین تعداد فروش مشخص گردد.

📐 **ریاضیات — گرایش به مرکز**

مهم‌ترین معیارهای این بخش Mean، Weighted Mean، Median و Mode هستند. فرمول‌ها باید همراه با تفسیر محاسباتی خوانده شوند.

میانگین حسابی — Arithmetic Mean

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.7}$$

نمادها

- $\bar{x}$: میانگین.
- x_i: مشاهده iام.
- N: تعداد مشاهدات.
- i: اندیس مشاهده.
- $\sum$: جمع مقادیر.

صورت کسر همه مقادیر را جمع می‌کند و مخرج تعداد آن‌هاست؛ بنابراین همه مشاهدات وزن برابر دارند.

مثال / تمرین حل‌شده

صورت سؤال: میانگین داده‌های 2, 4, 6 را محاسبه کنید.

حل: ابتدا مجموع را به‌دست می‌آوریم: 12 = 6 + 4 + 2. چون سه مشاهده داریم، 4 = 12/3. بنابراین میانگین برابر 4 است.

توضیح: میانگین حسابی مجموع همه مشاهدات را به‌طور مساوی بین تعداد مشاهدات تقسیم می‌کند.

میانگین وزنی — Weighted Arithmetic Mean

$$\bar{x}_w = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} \tag{2.8}$$

نمادها

- $\bar{x}_w$: میانگین وزنی.
- w_i: وزن مشاهده.
- x_i: مقدار مشاهده.
- N: تعداد مشاهدات.
- i: اندیس.

در صورت، هر مقدار در وزنش ضرب می‌شود؛ مخرج مجموع وزن‌هاست.

مثال / تمرین حل‌شده

صورت سؤال: دو مقدار 10 و 20 به‌ترتیب وزن‌های 1 و 3 دارند. میانگین وزنی را محاسبه کنید.

حل: صورت کسر برابر 70 = 10imes1 + 20imes3 و مجموع وزن‌ها برابر 4 = 3 + 1 است. پس میانگین وزنی 17.5 = 70/4 می‌شود.

توضیح: چون مقدار 20 وزن بیشتری دارد، نتیجه به 20 نزدیک‌تر از میانگین ساده 15 است.

میانه — Median

$$\text{Median}(x) = \begin{cases} x_{(m)} & N = 2m - 1 \\ \dfrac{x_{(m)} + x_{(m+1)}}{2} & N = 2m \end{cases} \tag{2.9}$$

نمادها

- Median(x): میانه داده مرتب‌شده.
- $x_{(m)}$: مقدار دارای رتبه m.
- N: تعداد مشاهدات.
- m: شاخص میانی.

ابتدا داده‌ها مرتب می‌شوند. برای تعداد فرد، مقدار وسط انتخاب می‌شود؛ برای تعداد زوج، میانگین دو مقدار وسط گرفته می‌شود.

مثال / تمرین حل‌شده

صورت سؤال: میانه داده‌های مرتب‌شده 1, 3, 5, 8 را پیدا کنید.

حل: تعداد داده‌ها زوج و برابر ۴ است؛ بنابراین دو مقدار میانی یعنی 3 و 5 را انتخاب می‌کنیم و میانگین می‌گیریم: $(3 + 5)/2 = 4$. پس میانه 4 است.

توضیح: در تعداد زوج، یک مقدار دقیقاً وسط قرار نمی‌گیرد؛ بنابراین میانگین دو مقدار وسط به‌عنوان میانه در نظر گرفته می‌شود.

نما — Mode

$$\text{Mode}(x) = \arg\max_{v} \text{freq}(v) \tag{2.10}$$

نمادها

- Mode(x): پرتکرارترین مقدار.
- v: یک مقدار ممکن.
- freq(v): فراوانی مقدار.
- arg max: مقداری را انتخاب می‌کند که کمیت را بیشینه کند.

مثال / تمرین حل‌شده

صورت سؤال: نمای داده‌های 2, 2, 3, 4, 4, 4 را تعیین کنید.

حل: مقدار 2 دو بار، 3 یک بار و 4 سه بار تکرار شده است. بیشترین فراوانی مربوط به 4 است؛ بنابراین Mode برابر 4 است.

توضیح: نما پرتکرارترین مقدار است و برخلاف میانگین و میانه لزوماً به محاسبات عددی پیچیده نیاز ندارد.

رابطه تجربی Mean–Median–Mode

$$\text{mean} - \text{mode} \approx 3(\text{mean} - \text{median}) \tag{2.11}$$

نمادها

- mean: میانگین.
- median: میانه.
- mode: نما.
- 3: ضریب تجربی.

این رابطه برای داده‌های تک‌نمای نسبتاً چوله یک تقریب تجربی است و یک قانون عمومی برای همه توزیع‌ها نیست.

Midrange — میان‌بازه

$$\text{Midrange} = \frac{x_{min} + x_{max}}{2} \tag{2.12}$$

نمادها

- x_{min}: کمینه.
- x_{max}: بیشینه.

مثال / تمرین حل‌شده

صورت سؤال: برای داده‌های 2, 4, 7, 10، مقدار Midrange را محاسبه کنید.

حل: کمینه 2 و بیشینه 10 هستند. بنابراین Midrange = (2 + 10)/2 = 6.

توضیح: Midrange فقط به کمینه و بیشینه وابسته است؛ به همین دلیل در برابر داده‌های پرت بسیار حساس است و معمولاً به‌اندازه میانگین یا میانه قابل اتکا نیست.

مثال عملی: تحلیل زمان ارسال و رفتار خریداران در «اسنپ‌فود» 📏

در «اسنپ‌فود»، روزانه صدها هزار سفارش ثبت می‌شود. تیم داده‌کاوی برای پاسخ به سوالاتی مثل «زمان دقیق ارسال غذا به مشتری چقدر است؟» یا «مردم بیشتر چه ساعتی غذا سفارش می‌دهند؟» نیاز دارد میلیون‌ها داده را در چند عدد خلاصه کند. اینجاست که **معیارهای گرایش به مرکز** وارد میدان می‌شوند.

۱. میانگین (Mean) — سنجش عملکرد کلی در شرایط عادی

تیم الگوریتم، زمان ارسال ۵ سفارش یک پیک را ثبت می‌کند: ۲۵، ۳۰، ۲۸، ۳۲ و ۳۵ دقیقه.

حاصل‌جمع (۱۵۰) تقسیم بر تعداد (۵) برابر با **۳۰ دقیقه** می‌شود. میانگین برای داده‌های متقارن و بدون نوسان شدید، بهترین خلاصه عددی است.

۲. میانه (Median) — نجات از داده‌های پرت (Outliers) و ترافیک سنگین

فرض کنید ناگهان باران می‌بارد و یکی از پیک‌ها در ترافیک می‌ماند! زمان‌های جدید: ۲۰، ۲۲، ۲۵، ۲۸ و **۱۵۰ دقیقه**.

اگر میانگین بگیریم، عدد **۴۹ دقیقه** به دست می‌آید که کاملاً نادرست است و مشتری را نگران می‌کند! اما اگر داده‌ها را مرتب کنیم، عدد وسط یعنی **۲۵ دقیقه (میانه)** نشان‌دهنده زمان واقعی انتظار اکثر مشتریان است. میانه در برابر داده‌های پرت کاملاً مقاوم است.

۳. نما (Mode) — پیدا کردن محبوب‌ترین‌ها

اسنپ‌فود می‌خواهد بداند پرتقاضاترین نوع غذا در روزهای تعطیل چیست؟ مقادیر ثبت‌شده: [پیتزا، چلوکباب، پیتزا، سالاد، پیتزا، پاستا].

مقدار **«پیتزا»** با ۳ بار تکرار، **نمای (Mode)** داده‌هاست. نما تنها معیاری است که برای متغیرهای غیرعددی و اسمی (Nominal) هم استفاده می‌شود.

نتیجه: اسنپ‌فود برای تخمین زمان ارسال به مشتری از «میانه» استفاده می‌کند تا اثر ترافیک‌های ناگهانی را خنثی کند، و برای پیشنهادهای ویژه اپلیکیشن از «نما» بهره می‌برد.

خروجی: انتخاب هوشمندانه معیار متناسب با نوع داده و حضور داده‌های پرت

In [10]:

```
display_academic_text(
    "آمار توصیفی: ارزیابی معیارهای گرایش به مرکز در حضور داده‌های پرت (Outliers) 📉",
    "main_title"
)

display_academic_text(
    "انتخاب معیار مرکزی به توزیع آماری بستگی دارد. در یک شبکه تحویل غذا، وقوع حوادث غیرمترقبه نظیر ترافیک یا شرایط جوی نامساعد، "
    "در مواجهه با نوسانات شدید زمانی <i>میانه (Median)</i> و <i>میانگین (Mean)</i> می‌گردد. در این سناریو، مقاومت شاخص‌های <b>داده‌های پرت (Outliers)</b> منجر به تولید "
    "برای تحلیل متغیرهای طبقه‌بندی‌شده مورد ارزیابی قرار می‌گیرد <i>نما (Mode)</i> و همچنین کاربرد شاخص.",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION & STATISTICAL COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های عملیاتی و استخراج شاخص‌های مرکزی",
    "sub_title"
)

display_academic_text(
    "در این فاز پایگاه داده زمان‌های ارسال پیک استخراج می‌شود که حاوی یک <b>دادهٔ ناهنجار (۱۵۰ دقیقه)</b> است. سپس، توابع آماری پایه بر روی "
    "بردار ویژگی‌ها اعمال شده و خروجی‌ها در یک ماتریس تجمیعی جهت مقایسه در کنار داده‌های خام به نمایش در می‌آیند.",
    "paragraph"
)

# Step 1: Initialize the mock database for delivery logistics.
# Note the intentional injection of an extreme outlier (150 minutes) to simulate severe traffic.
raw_data = {
    'Order_ID': [101, 102, 103, 104, 105],
    'Delivery_Time_Min': [20, 22, 25, 28, 150],
    'Food_Category': ['Pizza', 'Kebab', 'Pizza', 'Burger', 'Pizza']
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Compute the Arithmetic Mean (Highly sensitive to extreme values / outliers)
calculated_mean = df_raw['Delivery_Time_Min'].mean()

# Step 3: Compute the Median (Robust statistical measure, represents the exact 50th percentile)
calculated_median = df_raw['Delivery_Time_Min'].median()

# Step 4: Compute the Mode (The most frequent categorical occurrence)
# The [0] index extracts the first mode string from the returned pandas Series
calculated_mode = df_raw['Food_Category'].mode()[0]

# Step 5: Construct a summary dataframe to present the calculated metrics professionally
summary_data = {
    'Statistical Measure': [
        'Mean (Average)',
        'Median (50th Percentile)',
        'Mode (Highest Frequency)'
    ],
    'Computed Value': [
        f"{calculated_mean:.1f} Minutes",
        f"{calculated_median:.1f} Minutes",
        calculated_mode
    ],
    'Algorithmic Behavior': [
        'Heavily skewed by outlier (Misleading)',
        'Robust to extreme anomalies (Accurate)',
        'Identifies categorical peak demand'
    ]
}
df_summary = pd.DataFrame(summary_data)

# Render the raw transaction data and the extracted statistical insights side-by-side
display_side_by_side_datasets(
    [df_raw, df_summary],
    [
        "جدول ۱: رکوردهای ثبت‌شده زمان ارسال (حاوی داده پرت)",
        "جدول ۲: تحلیل رفتار معیارهای گرایش به مرکز"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلیِ سوءگیری میانگین در برابر مقاومت میانه",
    "sub_title"
)

display_academic_text(
    "نمودار سمت چپ به وضوح <b>انحراف شدید خط میانگین</b> به سمت دادهٔ پرت را به تصویر می‌کشد، در حالی که <b>خط میانه پایداری خود را حفظ کرده</b> و "
    "در هستهٔ اصلی توزیع داده‌ها باقی مانده است. نمودار سمت راست نیز بازنمایی گرافیکی استخراج شاخص نما (Mode) برای دسته‌بندی‌های اسمی است.",
    "paragraph"
)

# Initialize a wide dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- SUBPLOT 1: Continuous Data (Mean vs Median) ---
# A stripplot avoids overlapping dots, making the outlier visually distinct.
sns.stripplot(
    y=df_raw['Delivery_Time_Min'],
    ax=axes[0],
    color='#2c3e50', # Navy Blue
    size=12,
    jitter=0.05,
    edgecolor='white',
    linewidth=1.5
)
```

```
# Overlay theoretical center lines: Mean vs Median
axes[0].axhline(
    calculated_mean,
    color='#e74c3c', # Red for the misleading/skewed metric
    linestyle='--',
    linewidth=3.0,
    label=f'Mean (Skewed): {calculated_mean:.1f} Min'
)

axes[0].axhline(
    calculated_median,
    color='#1abc9c', # Turquoise for the robust/accurate metric
    linestyle='-',
    linewidth=3.0,
    label=f'Median (Robust): {calculated_median:.1f} Min'
)

# Intelligent Annotations
axes[0].annotate(
    "The Outlier Pull:\nA single 150-minute delivery forcefully pulls\nthe mathematical average upward, creating\na false representation of normal operations.",
    xy=(0.0, calculated_mean), xytext=(0.15, calculated_mean + 15),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].annotate(
    "Robustness:\nThe median ignores the extreme value\nand correctly identifies that 50% of orders\narrive in 25 minutes or less.",
    xy=(0.0, calculated_median), xytext=(0.15, calculated_median - 20),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[0].set_title('Figure 1: Impact of Outliers on Central Tendency Measures', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_ylabel('Recorded Delivery Time (Minutes)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].legend(loc='center right', frameon=True, facecolor='#f8f9fa', edgecolor='#bdc3c7', fontsize=11)
axes[0].grid(axis='y', linestyle='--', alpha=0.5)

# --- SUBPLOT 2: Categorical Data (Mode) ---
# Countplot automatically aggregates and plots categorical distributions.
sns.countplot(
    x=df_raw['Food_Category'],
    ax=axes[1],
    palette=['#1abc9c', '#2980b9', '#2c3e50'], # Academic Blue Gradient
    edgecolor='#2c3e50',
    linewidth=1.5
)

# Formatting Subplot 2
axes[1].set_title(f'Figure 2: Nominal Attribute Distribution (Mode = {calculated_mode})', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Nominal Food Category', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Order Frequency Count', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

# Add data labels on top of bars
for p in axes[1].patches:
    axes[1].annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha='center', va='center', fontsize=12, fontweight='bold', color='#2c3e50', xytext=(0, 10), textcoords='offset points')

# Remove the top and right spines from both axes for a cleaner academic look
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br><br>"
    r"در پایگاه‌های داده‌ای که مستعدِ نوساناتِ محیطی و داده‌های پرت (مانند تأخیر ۱۵۰ دقیقه‌ای پیک) هستند <b>(Mean) میانگین حسابی</b> همان‌طور که در گرافِ اول ثابت شد، اتکا به"
    r"در واقع عملکردِ عالیِ خود در ۸۰٪ سفارش‌ها (زیر ۳۰ دقیقه) را زیر سؤال برده است، <i>«میانگین زمان ارسال ما ۴۹ دقیقه است»</i> به برآوردهای به‌شدت گمراه‌کننده منجر می‌شود. اگر سیستم به مشتری بگوید! <br><br>"
    r"یک مکانیزم مقاوم‌سازیِ قدرتمند در برابر اختلالات ایجاد کرده و به تخمینِ واقعی‌تری از تجربهٔ بدنهٔ اصلیِ کاربران منتهی می‌گردد، <b>(Median = ۲۵ دقیقه)</b> میانه <b> جایگزینی این شاخص با"
)
display_academic_text(insight_conclusion, "insight")
```

آمار توصیفی: ارزیابی معیارهای گرایش به مرکز در حضور داده‌های پرت (Outliers) 📉

انتخاب معیار مرکزی به توزیع آماری بستگی دارد. در یک شبکه تحویل غذا، وقوع حوادث غیرمترقبه نظیر ترافیک یا شرایط جوی نامساعد، منجر به تولید **داده‌های پرت (Outliers)** می‌گردد. در این سناریو، مقاومت شاخص‌های *میانگین (Mean)* و *میانه (Median)* در مواجهه با نوسانات شدید زمانی، و همچنین کاربرد شاخص *نما (Mode)* برای تحلیل متغیرهای طبقه‌بندی‌شده مورد ارزیابی قرار می‌گیرد.

مرحله نخست: بارگذاری داده‌های عملیاتی و استخراج شاخص‌های مرکزی

در این فاز پایگاه داده زمان‌های ارسال پیک استخراج می‌شود که حاوی یک **دادهٔ ناهنجار (۱۵۰ دقیقه)** است. سپس، توابع آماری پایه بر روی بردار ویژگی‌ها اعمال شده و خروجی‌ها در یک ماتریس تجمیعی جهت مقایسه در کنار داده‌های خام به نمایش در می‌آیند.

جدول ۱: رکوردهای ثبت‌شده زمان ارسال (حاوی داده پرت)

Order_ID	Delivery_Time_Min	Food_Category
101	20	Pizza
102	22	Kebab
103	25	Pizza
104	28	Burger
105	150	Pizza

جدول ۲: تحلیل رفتار معیارهای گرایش به مرکز

Statistical Measure	Computed Value	Algorithmic Behavio
Mean (Average)	49.0 Minutes	Heavily skewed by outlier (Mi
Median (50th Percentile)	25.0 Minutes	Robust to extreme anomalies
Mode (Highest Frequency)	Pizza	Identifies categorical peak (

مرحله دوم: مصورسازی تحلیلیِ سوءگیری میانگین در برابر مقاومت میانه

نمودار سمت چپ به وضوح **انحراف شدید خط میانگین** به سمت دادهٔ پرت را به تصویر می‌کشد، در حالی که **خط میانه پایداری خود را حفظ کرده** و در هستهٔ اصلی توزیع داده‌ها باقی مانده است. نمودار سمت راست نیز بازنمایی گرافیکی استخراج شاخص نما (Mode) برای دسته‌بندی‌های اسمی است.

Figure 1: Impact of Outliers on Central Tendency Measures

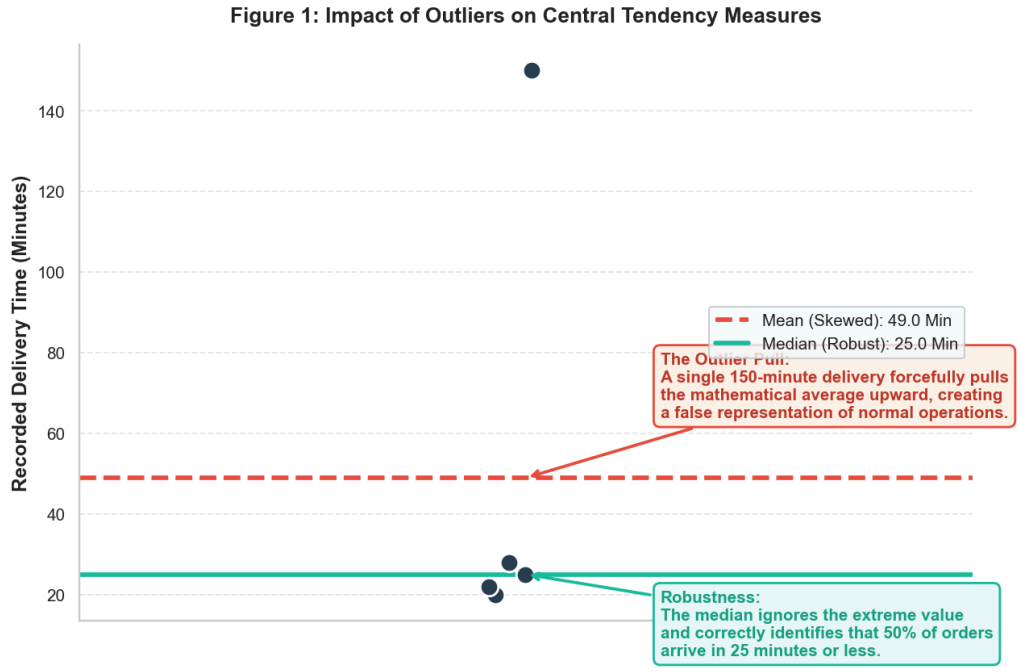
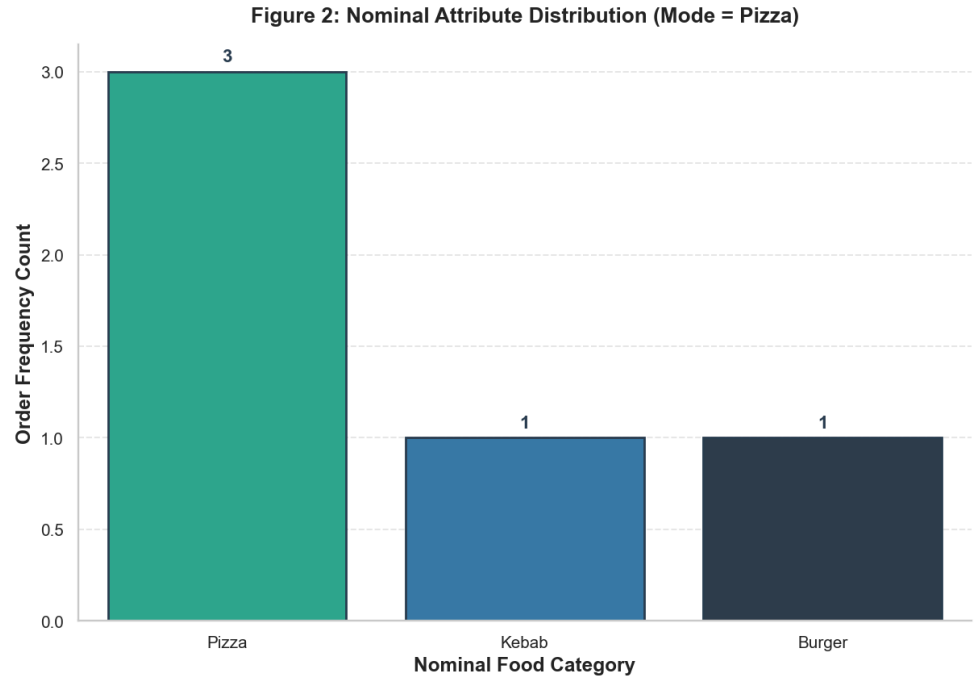


Figure 2: Nominal Attribute Distribution (Mode = Pizza)



استنتاج راهبردی عملیات (Strategic Implication):

همان‌طور که در گرافِ اول ثابت شد، اتکا به میانگین حسابی (Mean) در پایگاه‌های داده‌ای که مستعدِ نوساناتِ محیطی و داده‌های پرت (مانند تأخیر ۱۵۰ دقیقه‌ای پیک) هستند، به برآوردهای به‌شدت گمراه‌کننده منجر می‌شود. اگر سیستم به مشتری بگوید *«میانگین زمان ارسال ما ۴۹ دقیقه است»*، در واقع عملکردِ عالیِ خود در ۸۰٪ سفارش‌ها (زیر ۳۰ دقیقه) را زیر سؤال برده است!

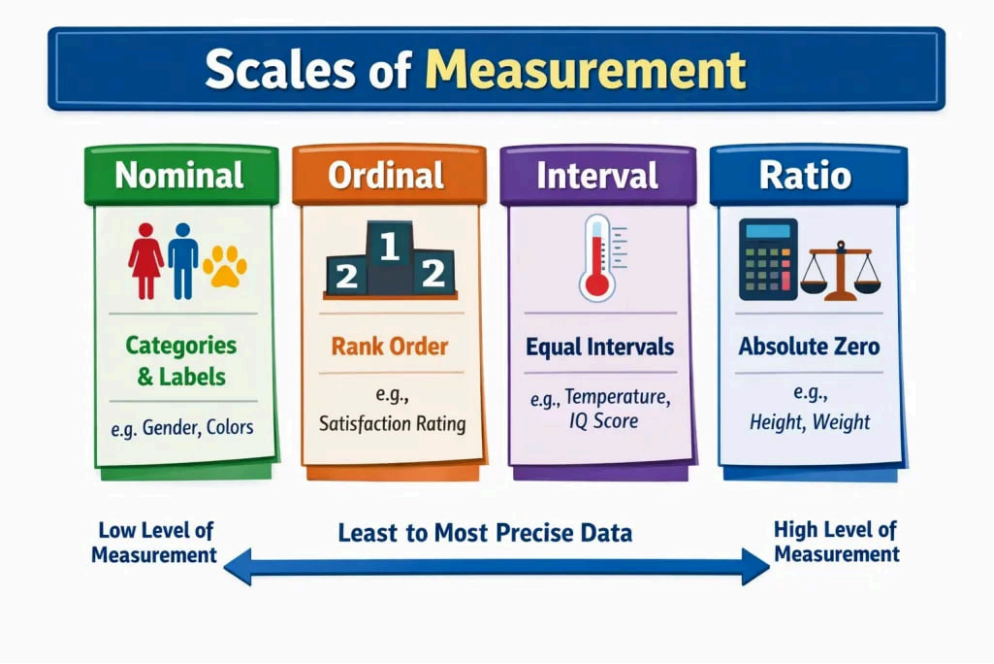
جایگزینی این شاخص با میانه (Median = ۲۵ دقیقه)، یک مکانیزم مقاوم‌سازیِ قدرتمند در برابر اختلالات ایجاد کرده و به تخمینِ واقعی‌تری از تجربهٔ بدنهٔ اصلیِ کاربران منتهی می‌گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم با دو ویژگی ساختیم: **'Delivery_Time_Min'** (زمان تحویل به‌صورت عددی پیوسته) که شامل یک داده‌ی پرت (۱۵۰ دقیقه) است، و **'Food_Category'** (دسته‌بندی غذا به‌صورت اسمی).
- با متد **.mean()** میانگین زمان تحویل را محاسبه کردیم (عدد ۴۹ دقیقه) و دیدیم که چگونه یک داده‌ی پرت شدید باعث شد میانگین به سمت بالا منحرف شود.
- با متد **.median()** میانه داده‌ها را به دست آوردیم (عدد ۲۵ دقیقه) که نشان داد میانه چگونه داده‌ی پرت را نادیده گرفته و مقدار واقعیِ مرکز داده را به درستی نشان می‌دهد.
- با متد **.mode()** پرتکرارترین گزینه متنی در ستون غذاها (**'Pizza'**) را استخراج کردیم.

2.2.2. Measuring the dispersion of data



📈 معیارهای پراکندگی داده‌ها

معیارهای پراکندگی شاخص‌های آماری هستند که میزان **پراکندگی** یا **پخش‌شدگی** مقادیر را نسبت به **مقدار مرکزی** مجموعه داده نشان می‌دهند. این معیارها علاوه بر توصیف **توزیع داده‌ها**، در مقایسه مجموعه‌داده‌ها و شناسایی میزان **تغییرپذیری** آن‌ها نیز کاربرد دارند. مهم‌ترین معیارهای پراکندگی عبارت‌اند از:

۱. دامنه (Range)

دامنه، اختلاف بین **بزرگ‌ترین** و **کوچک‌ترین** مقدار در مجموعه داده است و **ساده‌ترین** معیار برای سنجش میزان پراکندگی محسوب می‌شود.

مزایا: محاسبه‌ی بسیار ساده و سریع.

معایب: تنها به دو مقدار اکسترمم حساس است و اطلاعاتی دربارهٔ نحوهٔ توزیع سایر داده‌ها ارائه نمی‌دهد.

۲. واریانس (Variance)

واریانس، **میانگین مجذور فاصله** مقادیر از میانگین است و میزان **تغییرات** داده‌ها را نسبت به مقدار مرکزی اندازه‌گیری می‌کند.

مزایا: از تمام داده‌ها استفاده می‌کند و به تغییرات تک‌تک مقادیر حساس است.

معایب: واحد آن مجذور واحد داده‌های اصلی است و تفسیر مستقیم آن دشوار است.

۳. انحراف معیار (Standard Deviation)

انحراف معیار، **ریشه دوم واریانس** است و به دلیل **هم‌واحد بودن** با داده‌های اصلی، یکی از **رایج‌ترین** و **قابل‌تفسیرترین** معیارهای سنجش پراکندگی به شمار می‌رود.

مزایا: هم‌واحد با داده‌ها، قابل‌تفسیر و بسیار پرکاربرد.

معایب: به داده‌های پرت حساس است (همانند واریانس).

> 💡 **نکته کلیدی:** هرچه مقدار **دامنه**، **واریانس** و **انحراف معیار** بیشتر باشد، داده‌ها **پراکنده‌تر** و کمتر حول میانگین متمرکز هستند. در مقابل، مقادیر کوچک‌تر این معیارها نشان‌دهندهٔ **تمرکز بیشتر** داده‌ها حول میانگین است. از میان این سه معیار، **انحراف معیار** به‌دلیل هم‌واحد بودن با داده‌ها، در گزارش‌های آماری و تحلیل‌های علمی بیشترین کاربرد را دارد.

> 📐 **ریاضیات — پراکندگی داده‌ها**

دامنه — Range

$$\text{Range} = x_{max} - x_{min} \tag{2.13}$$

نمادها

- x_{max}: بیشینه.
- x_{min}: کمینه.

مثال / تمرین حل‌شده

صورت سؤال: دامنه داده‌های 3, 5, 8, 11 را محاسبه کنید.

حل: بیشینه 11 و کمینه 3 است. بنابراین Range = 11 - 3 = 8.

توضیح: دامنه فقط فاصله بین دو انتهای داده را نشان می‌دهد و تمام مقادیر میانی را در محاسبه وارد نمی‌کند.

دامنه بین‌چارکی — IQR

$$\text{IQR} = Q_3 - Q_1 \tag{2.14}$$

نمادها

- IQR: دامنه بین‌چارکی.
- Q_1: چارک اول.
- Q_3: چارک سوم.

IQR پراکندگی 50 درصد میانی داده‌ها را نشان می‌دهد و نسبت به مقدارهای بسیار دور مقاوم‌تر است.

واریانس — Variance

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{2.15}$$

نمادها

- σ^2: واریانس.
- x_i: مشاهده.
- $\bar{x}$: میانگین.
- N: تعداد مشاهدات.
- i: اندیس.
- $\sum$: جمع.

ابتدا فاصله هر مقدار از میانگین محاسبه می‌شود، سپس مربع می‌شود تا علامت‌ها حذف شوند و در پایان میانگین فاصله‌های مربعی گرفته می‌شود.

انحراف معیار — Standard Deviation

$$\sigma = \sqrt{\sigma^2} \tag{2.16}$$

نمادها

- σ: انحراف معیار.
- σ^2: واریانس.

مثال / تمرین حل‌شده

صورت سؤال: برای داده‌های 2, 4, 6، میانگین، واریانس جامعه و انحراف معیار را محاسبه کنید.

حل: میانگین 4 است. فاصله‌ها از میانگین برابر 2, 0, 2- و مربع آن‌ها 4, 0, 4 است. مجموع مربع‌ها 8 و با تقسیم بر $N = 3$، واریانس $8/3 \approx 2.67$ می‌شود. انحراف معیار نیز $\sqrt{2.67} \approx 1.63$ است.

توضیح: واریانس پراکندگی را با واحدِ مربع‌شده اندازه می‌گیرد؛ انحراف معیار با گرفتن ریشه دوم دوباره به واحد اصلی داده برمی‌گردد.

📌 **مرکز و پراکندگی؛ دو عدد، یک روایت کامل از داده‌ها**

📏 **معیارهای گرایش به مرکز** به ما می‌گویند «داده‌ها حول چه مقداری تجمع یافته‌اند»:

• **میانگین:** حاصلِ تقسیم مجموع کل مقادیر بر تعداد آن‌ها؛ مثل یافتن نقطهٔ تعادل یک ترازو.

• **میانه:** مقداری که پس از مرتب‌سازی داده‌ها درست در مرکز مجموعه می‌ایستد و آن را به دو نیمهٔ مساوی تقسیم می‌کند.

• **نما:** پرتکرارترین مقدار در مجموعه داده؛ هرچه بیشتر دیده شده باشد، نما همان است.

📈 **معیارهای پراکندگی** به ما می‌گویند «داده‌ها چقدر حول مرکز پراکنده شده‌اند»:

• **دامنه:** فاصلهٔ ساده و سریع بین کمترین و بیشترین مقدار؛ یک نگاه فوری به گستردگی داده‌ها.

• **واریانس:** میانگینِ مجذور فاصلهٔ تک‌تک داده‌ها از میانگین؛ هرچه بزرگ‌تر باشد، داده‌ها پراکنده‌ترند.

• **انحراف معیار:** ریشهٔ دوم واریانس که با واحد خود داده‌ها بیان می‌شود و پرکاربردترین معیار پراکندگی است.

💡 **نکتهٔ کلیدی:** مرکز و پراکندگی دو روی یک سکه‌اند؛ اولی می‌گوید «داده‌ها کجا جمع شده‌اند»، دومی می‌گوید «چقدر از هم دور شده‌اند». برای توصیف کامل یک مجموعه داده، به هر دو نیاز داریم.

مثال عملی: سنجش ثبات کیفیت پخش ویدیو در «نتفلیکس» (Netflix)

فرض کنید تیم فنی «نتفلیکس» می‌خواهد کیفیت دو سرور پخش آنلاین (CDN) مختلف را مقایسه کند. میانگین زمان معطلی (بافرینگ) فیلم در هر دو سرور دقیقاً **۵ ثانیه** است! اما کاربران سرور دوم مدام از قطعی و پرش تصویر شکایت می‌کنند. چرا؟ چون میانگین به تنهایی نوسانات را نشان نمی‌دهد؛ اینجاست که **معیارهای پراکندگی داده** راز قضیه را برملا می‌کنند.

۱. دامنه (Range) — ساده‌ترین ابزار سنجش فاصله نوسان

زمان‌های بافرینگ در سرور اول: [۴، ۵، ۵، ۵، ۶] ثانیه 👈 دامنه = ۶ - ۴ = **۲ ثانیه**.

زمان‌های بافرینگ در سرور دوم: [۱، ۱، ۵، ۹، ۹] ثانیه 👈 دامنه = ۹ - ۱ = **۸ ثانیه**.

دامنه بزرگ‌تر در سرور دوم نشان می‌دهد که تفاوت بین بهترین و بدارین تجربه کاربر بسیار زیاد است.

۲. واریانس (Variance) — محاسبه شدت جریمه نوسانات

واریانس میانگین مجذور (توان دوم) فاصله اعداد از میانگین است. با مربعی کردن فاصله‌ها، نوسانات بزرگ جریمه سنگین‌تری می‌شوند.

- واریانس سرور اول = **۰.۴** (بسیار کم و پایدار)
- واریانس سرور دوم = **۱۲.۸** (بسیار بالا و ناپایدار)

مشکل واریانس: واحد آن «ثانیه به‌توان ۲» است که تفسیر ملموسی برای مدیر محصول ندارد!

۳. انحراف معیار (Standard Deviation) — معیاری ملموس با واحد اصلی

با گرفتن جذر (ریشه دوم) از واریانس، واحد اندازه دوباره به «ثانیه» برمی‌گردد:

- انحراف معیار سرور اول = **۰.۶۳ ثانیه** (تجربه یکنواخت و قابل پیش‌بینی).
- انحراف معیار سرور دوم = **۳.۵۷ ثانیه** (تجربه غیرقابل پیش‌بینی و پر از شوک).

نتیجه: اگرچه میانگین هر دو سرور ۵ ثانیه بود، اما نتفلیکس سرور اول را انتخاب می‌کند؛ زیرا انحراف معیار پایین‌تر به معنای ثبات پخش و رضایت کاربر است.

خروجی: کشف نوسانات پنهان در داده‌ها با کمک انحراف معیار و واریانس

In [11]:

```
display_academic_text(
    "📉 آمار توصیفی: ارزیابی معیارهای پراکندگی داده (Data Dispersion) و کشف نوسانات پنهان",
    "main_title"
)

display_academic_text(
    "معیارهای گرایش به مرکز (مانند <i>میانگین</i>) به تنهایی تصویر کاملی از توزیع داده‌ها ارائه نمی‌دهند و می‌توانند <b>به شدت گمراه‌کننده</b> باشند. "
    "در این سناریو، دو سرور پخش آنلاین (CDN) مورد بررسی قرار می‌گیرند (۵ ثانیه) با میانگینِ بافرینگِ کاملاً یکسان. "
    "استفاده از معیارهای پراکندگی شامل <b>دامنه (Range)</b>، <b>واریانس (Variance)</b> و <b>انحراف معیار (Standard Deviation)</b>، "
    "جهت کشف تفاوت‌های ساختاری در پایداری (Stability) این دو سیستم به کار گرفته شده است.",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION & DISPERSION COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های خام و استخراج شاخص‌های پراکندگی",
    "sub_title"
)

display_academic_text(
    "در این فاز، زمان‌های معطلی دو سرور ثبت گردیده است. سپس شاخص‌های پراکندگی بر مبنای فرمول <b>واریانس جامعه (Population Variance)</b> استخراج شده "
    "و در قالب یک ماتریس تجمیعی جهت مقایسهٔ عملکرد، در کنار پایگاه داده خام رندر می‌شوند.",
    "paragraph"
)

# Step 1: Initialize the mock database for Netflix server buffering times.
# Both servers intentionally share the exact same mean (5.0 seconds).
raw_data = {
    'Test_ID': [1, 2, 3, 4, 5],
    'Server_A_Sec': [4.0, 5.0, 5.0, 5.0, 6.0],
    'Server_B_Sec': [1.0, 1.0, 5.0, 9.0, 9.0]
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Calculate the Range for both servers (Max - Min)
range_a = df_raw['Server_A_Sec'].max() - df_raw['Server_A_Sec'].min()
range_b = df_raw['Server_B_Sec'].max() - df_raw['Server_B_Sec'].min()

# Step 3: Calculate the Population Variance using ddof=0 (Delta Degrees of Freedom = 0).
# This penalizes larger deviations from the mean squarely.
var_a = df_raw['Server_A_Sec'].var(ddof=0)
var_b = df_raw['Server_B_Sec'].var(ddof=0)

# Step 4: Calculate the Population Standard Deviation (Square root of variance).
# This returns the dispersion metric back to the original unit (seconds).
std_a = df_raw['Server_A_Sec'].std(ddof=0)
std_b = df_raw['Server_B_Sec'].std(ddof=0)

# Step 5: Construct a summary dataframe to present the calculated dispersion metrics.
summary_data = {
    'Statistical Measure': [
        'Arithmetic Mean (Center)',
        'Data Range (Max - Min)',
        'Population Variance',
        'Standard Deviation (Spread)'
    ],
    'Server A (Stable)': [
        f"{df_raw['Server_A_Sec'].mean():.1f} sec",
        f"{range_a:.1f} sec",
        f"{var_a:.2f}",
        f"{std_a:.2f} sec"
    ],
    'Server B (Unstable)': [
```

```
        f"{df_raw['Server_B_Sec'].mean():.1f} sec",
        f"{range_b:.1f} sec",
        f"{var_b:.2f}",
        f"{std_b:.2f} sec"
    ]
}
df_summary = pd.DataFrame(summary_data)

# Render the raw observation data and the extracted dispersion statistics side-by-side
# Using the list format expected by the global function
display_side_by_side_datasets(
    [df_raw, df_summary],
    [
        "جدول ۱: رکوردهای ثبت‌شده زمان بافرینگ (ثانیه)",
        "جدول ۲: تحلیل مقایسه‌ای شاخص‌های پراکندگی"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی نوسانات (Variance) و پایداری سیستم",
    "sub_title"
)

display_academic_text(
    "به شدت در اطراف <b>سرور A</b> بوم گرافیکی زیر ماهیت پنهانِ داده‌ها را آشکار می‌سازد. در نمودار پراکندگی (سمت چپ)، مشخص است که داده‌های "
    "دچار پراکندگی شدید و شوکِ زمانی می‌باشند. <b>سرور B</b> خطِ میانگین متمرکز هستند، در حالی که داده‌های "
    "نمودار چگالی توزیع (سمت راست) نیز این تمرکز قطعی (قله بلند) را در برابر نوسانِ شدید (قله پهن) تأیید می‌نماید.",
    "paragraph"
)

# Initialize a wide dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# Restructure the dataframe into a long format using pandas 'melt' for seaborn compatibility
df_melted = df_raw.melt(
    id_vars='Test_ID',
    value_vars=['Server_A_Sec', 'Server_B_Sec'],
    var_name='Server_Type',
    value_name='Buffering_Time'
)

# --- SUBPLOT 1: Absolute Data Dispersion (Strip Plot) ---
sns.stripplot(
    data=df_melted,
    x='Server_Type',
    y='Buffering_Time',
    ax=axes[0],
    palette=['#1abc9c', '#e74c3c'], # Turquoise for Stable, Red for Unstable
    size=15,
    jitter=0.05,
    edgecolor='#2c3e50',
    linewidth=1.5,
    zorder=5
)

# Overlay the theoretical identical mean line for visual reference
axes[0].axhline(
    5.0,
    color='#2c3e50', # Navy Blue
    linestyle='--',
    linewidth=3.0,
    zorder=1,
    label='Identical Mean (5.0 sec)'
)

# Intelligent Annotations
axes[0].annotate(
    "High Variance (Unstable):\nUsers either experience instant loads (1s)\nor suffer severe lag (9s).\nThe 5s average is a pure illusion!",
    xy=(1.1, 9.0), xytext=(1.4, 6.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].annotate(
    "Low Variance (Stable):\nEvery user gets consistent performance.\nThe standard deviation is practically zero.",
    xy=(0.1, 4.0), xytext=(0.4, 1.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

# Formatting Subplot 1
axes[0].set_title('Figure 1: Absolute Data Dispersion around the Mean', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('CDN Server Identification', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Buffering Duration (Seconds)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xticklabels(['Server A (Stable)', 'Server B (Unstable)'], fontweight='bold')
axes[0].legend(loc='lower center', frameon=True, facecolor='#f8f9fa', edgecolor='#bdc3c7', fontsize=11)
axes[0].grid(True, linestyle='--', alpha=0.5, zorder=0)


# --- SUBPLOT 2: Probability Density Distribution (KDE) ---
sns.kdeplot(
    data=df_raw,
    x='Server_A_Sec',
    ax=axes[1],
    fill=True,
    color='#1abc9c',
    label='Server A (Low Std Dev)',
    linewidth=2.5
```

```
)
sns.kdeplot(
    data=df_raw,
    x='Server_B_Sec',
    ax=axes[1],
    fill=True,
    color='#e74c3c',
    label='Server B (High Std Dev)',
    linewidth=2.5
)

# Overlay the identical mean line on the density plot
axes[1].axvline(
    5.0,
    color='#2c3e50',
    linestyle='--',
    linewidth=3.0,
    label='Identical Mean (5.0 sec)'
)

# Formatting Subplot 2
axes[1].set_title('Figure 2: Probability Density Distribution (Shape of Variance)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Buffering Duration (Seconds)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Density (Probability Mass)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].legend(loc='upper right', frameon=True, facecolor='#f8f9fa', edgecolor='#bdc3c7', fontsize=11)
axes[1].grid(True, linestyle='--', alpha=0.5)

# Remove the top and right spines from both axes for a cleaner academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br><br>"
    r"علیرغم داشتنِ میانگینِ کاملاً یکسان با سرور اول (B) ارزیابی کیفیت خدمات بر مبنای «میانگین» به تنهایی، خطای سیستمی به همراه خواهد داشت. سرور دوم، "
    r"دارای <b>انحراف معیار بسیار بالایی (۳.۵۸ ثانیه)</b> می‌باشد و همراه با شوک‌های ناشی از قطعیِ طولانیِ تصویر (۹ ثانیه) غیرقابل‌پیش‌بینی، است که مستقیماً به معنای تجربه کاربریِ غیرقابل‌پیش‌بینی.<br><br>"
    r"در علم داده، <i>«واریانس، دشمنِ پیش‌بینی‌پذیری است»</i>. انتخاب سرور اول (A) تضمین‌کنندهٔ ثبات و یکنواختیِ جریان داده برای تمام کاربران است (۰.۶۳ ثانیه)، بر مبنای انحراف معیارِ پایین‌تر."
)
display_academic_text(insight_conclusion, "insight")
```

آمار توصیفی: ارزیابی معیارهای پراکندگی داده (Data Dispersion) و کشف نوسانات پنهان 📉

معیارهای گرایش به مرکز (مانند *میانگین*) به تنهایی تصویر کاملی از توزیع داده‌ها ارائه نمی‌دهند و می‌توانند **به شدت گمراه‌کننده** باشند. در این سناریو، دو سرور پخش آنلاین (CDN) با میانگینِ بافرینگِ کاملاً یکسان (۵ ثانیه) مورد بررسی قرار می‌گیرند. استفاده از معیارهای پراکندگی شامل **دامنه (Range)**، **واریانس (Variance)** و **انحراف معیار (Standard Deviation)**، جهت کشف تفاوت‌های ساختاری در پایداری (Stability) این دو سیستم به کار گرفته شده است.

مرحله نخست: بارگذاری داده‌های خام و استخراج شاخص‌های پراکندگی

در این فاز، زمان‌های معطلی دو سرور ثبت گردیده است. سپس شاخص‌های پراکندگی بر مبنای فرمول **واریانس جامعه (Population Variance)** استخراج شده و در قالب یک ماتریس تجمیعی جهت مقایسهٔ عملکرد، در کنار پایگاه داده خام رندر می‌شوند.

جدول ۱: رکوردهای ثبت‌شده زمان بافرینگ (ثانیه)

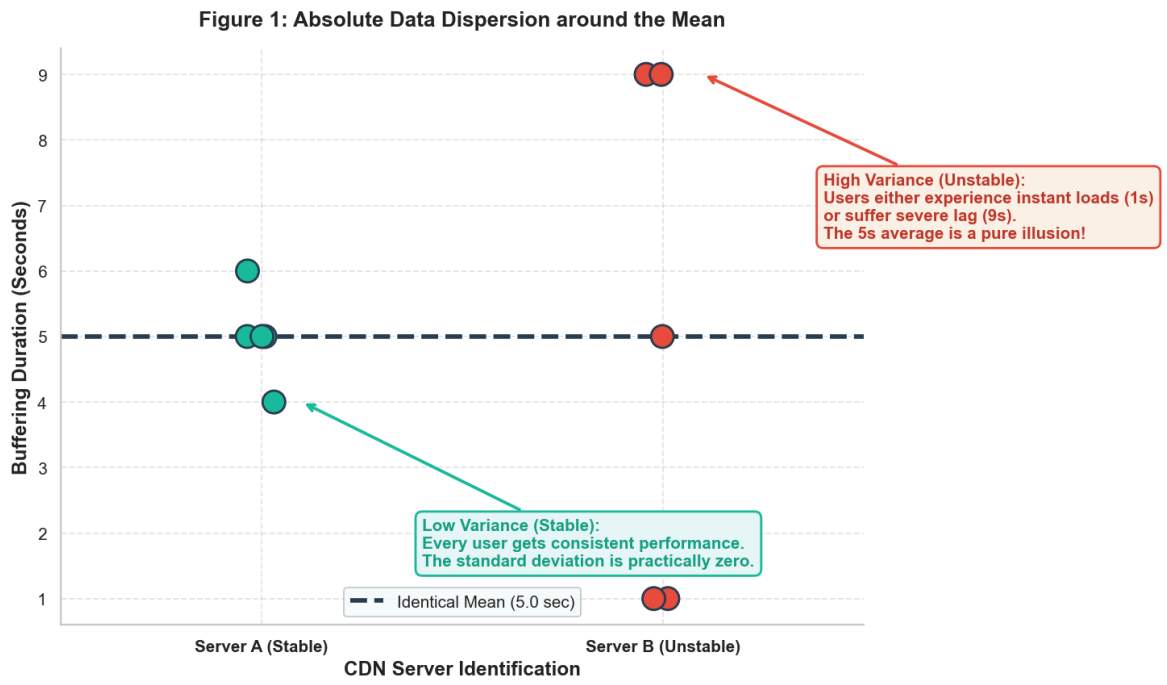
Test_ID	Server_A_Sec	Server_B_Sec
1	4.0	1.0
2	5.0	1.0
3	5.0	5.0
4	5.0	9.0
5	6.0	9.0

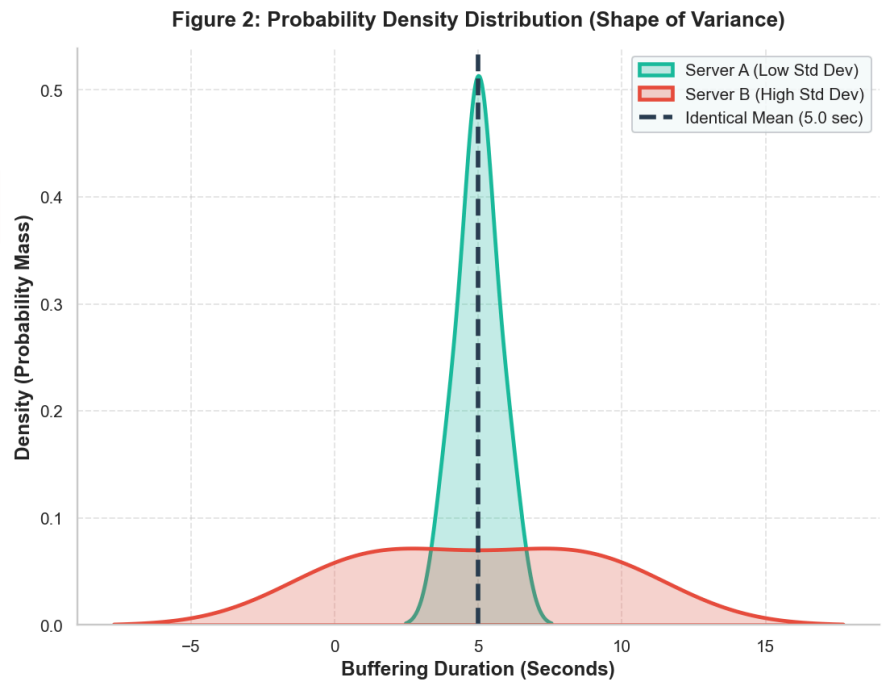
جدول ۲: تحلیل مقایسه‌ای شاخص‌های پراکندگی

Statistical Measure	Server A (Stable)	Server B (Unstable)
Arithmetic Mean (Center)	5.0 sec	5.0 sec
Data Range (Max - Min)	2.0 sec	8.0 sec
Population Variance	0.40	12.80
Standard Deviation (Spread)	0.63 sec	3.58 sec

مرحله دوم: مصورسازی تحلیلیِ نوسانات (Variance) و پایداری سیستم

بوم گرافیکی زیر ماهیت پنهانِ داده‌ها را آشکار می‌سازد. در نمودار پراکندگی (سمت چپ)، مشخص است که داده‌های **سرور A** به شدت در اطراف خطِ میانگین متمرکز هستند، در حالی که داده‌های **سرور B** دچار پراکندگی شدید و شوکِ زمانی می‌باشند. نمودار چگالی توزیع (سمت راست) نیز این تمرکز قطعی (قله بلند) را در برابر نوسانِ شدید (قله پهن) تأیید می‌نماید.





استنتاج راهبردی عملیات (Strategic Implication):

ارزیابی کیفیت خدمات بر مبنای «میانگین» به تنهایی، خطای سیستمی به همراه خواهد داشت. سرور دوم (B) علیرغم داشتنِ میانگینِ کاملاً یکسان با سرور اول، دارای انحراف معیار بسیار بالایی (۳.۵۸ ثانیه) است که مستقیماً به معنای تجربه کاربریِ غیرقابل‌پیش‌بینی، و همراه با شوک‌های ناشی از قطعیِ طولانیِ تصویر (۹ ثانیه) می‌باشد.

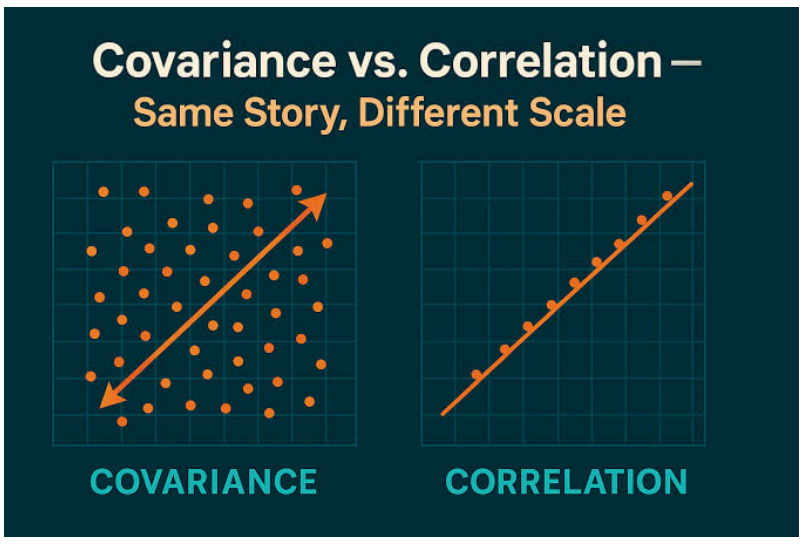
در علم داده، *«واریانس، دشمنِ پیش‌بینی‌پذیری است»*. انتخاب سرور اول (A) بر مبنای انحراف معیارِ پایین‌تر (۰.۶۳ ثانیه)، تضمین‌کنندهٔ ثبات و یکنواختیِ جریان داده برای تمام کاربران است.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم شامل داده‌های بافرینگ دو سرور نتفلیکس (**Netflix_Server_A** و **Netflix_Server_B**) ساختیم که میانگین زمان معطلی در هر دو سرور برابر با ۵ ثانیه است.
- **دامنه (Range):** با تفریق مقدار کمینه از بیشینه (با متدهای **.max()** و **.min()**)، اختلاف پخش‌شدگی داده‌ها مشخص شد (۲ ثانیه در برابر ۸ ثانیه).
- **واریانس (Variance):** با تابع **.var(ddof=0)** میانگین مجذور انحرافات حساب شد؛ واریانس سرور B عدد ۱۶ را نشان داد که جریمه نوسانات شدید در آن سرور است.
- **انحراف معیار (Standard Deviation):** با تابع **.std(ddof=0)** جذر واریانس گرفته شد تا انحراف معیار با همان واحد اصلی داده‌ها (ثانیه) به دست آید.
- نتیجه کد اثبات کرد که انحراف معیار سرور A (عدد ۰.۶۳ ثانیه) بسیار کمتر از سرور B (عدد ۴.۰۰ ثانیه) است، که نشان‌دهنده کیفیت و ثبات بالای سرور A برای کاربران است.

2.2.3. Covariance and correlation analysis



کوواریانس و تحلیل همبستگی 📉

کوواریانس و **تحلیل همبستگی** از مهم‌ترین معیارهای آماری برای بررسی **رابطهٔ بین دو متغیر** هستند و میزان **تغییرات هم‌زمان** آن‌ها را توصیف می‌کنند.

۱. کوواریانس (Covariance)

کوواریانس معیاری است که **جهت تغییرات هم‌زمان** دو متغیر را نشان می‌دهد. مقدار **مثبت** کوواریانس بیانگر آن است که با افزایش یک متغیر، متغیر دیگر نیز تمایل به افزایش دارد، در حالی که مقدار **منفی** نشان‌دهندهٔ **رابطهٔ معکوس** بین آن‌ها است. با این حال، از آنجا که کوواریانس به **مقیاس اندازه‌گیری** داده‌ها وابسته است، به‌تنهایی معیار مناسبی برای مقایسهٔ **شدت رابطه** محسوب نمی‌شود.

۲. تحلیل همبستگی (Correlation)

همبستگی معیاری استاندارد برای سنجش **جهت** و **شدت رابطهٔ خطی** بین دو متغیر است. ضریب همبستگی در بازهٔ **۱- تا ۱+** قرار دارد؛ به‌طوری‌که مقدار **۱+** بیانگر همبستگی مثبت کامل، مقدار **۱-** نشان‌دهندهٔ همبستگی منفی کامل و مقدار **۰** بیانگر نبود رابطهٔ خطی بین دو متغیر است.

📐 ریاضیات — کوواریانس و همبستگی

کوواریانس — Covariance

$$\mathrm{Cov}(A, B) = \frac{1}{N} \sum_{i=1}^{N} (a_i - \bar{A})(b_i - \bar{B}) \tag{2.17}$$

نمادها

- $\mathrm{Cov}(A, B)$: کوواریانس.
- a_i, b_i: مقادیر متناظر دو ویژگی.
- $\bar{A}, \bar{B}$: میانگین دو ویژگی.
- N: تعداد مشاهدات.
- i: اندیس.

اگر دو انحراف معمولاً هم‌علامت باشند، حاصل‌ضرب‌ها مثبت و کوواریانس تمایل به مثبت شدن دارد؛ اگر خلاف جهت باشند، منفی می‌شود.

فرم محاسباتی جایگزین

$$\mathrm{Cov}(A, B) = E(AB) - \bar{A}\bar{B} \tag{2.18}$$

نمادها

- $E(AB)$: میانگین حاصل‌ضرب‌های متناظر.
- $\bar{A}, \bar{B}$: میانگین‌ها.

ضریب همبستگی پیرسون — Pearson Correlation Coefficient

(2.19)

$$r_{A,B} = \frac{\sum_{i=1}^{N}(a_i - \bar{A})(b_i - \bar{B})}{N\sigma_A\sigma_B} \tag{2.19}$$

نمادها

- $r_{A,B}$: ضریب همبستگی پیرسون.
- a_i, b_i: مقادیر متناظر.
- $\bar{A}, \bar{B}$: میانگین‌ها.
- σ_A, σ_B: انحراف معیارها.
- N: تعداد مشاهدات.
- i: اندیس.

مقدار r در بازه $[-1, 1]$ قرار دارد. نزدیک 1 رابطه خطی مثبت قوی و نزدیک 1- رابطه خطی منفی قوی را نشان می‌دهد.

مثال / تمرین حل‌شده

صورت سؤال: اگر برای همه مشاهدات $B = 2A$ باشد، علامت و مقدار ضریب همبستگی پیرسون را مشخص کنید.

حل: هرگاه A افزایش پیدا کند، B دقیقاً دو برابر آن افزایش می‌یابد. رابطه کاملاً خطی و مثبت است، بنابراین $r_{A,B} = 1$.

توضیح: همبستگی کامل مثبت فقط وجود یک رابطه خطی را نشان می‌دهد و به‌تنهایی ثابت نمی‌کند که A علت B است.

آزمون کای‌دو — χ^2

$$\chi^2 = \sum_i \sum_j \frac{(o_{ij} - e_{ij})^2}{e_{ij}} \tag{2.20}$$

نمادها

- χ^2: آماره کای‌دو.
- o_{ij}: فراوانی مشاهده‌شده.
- e_{ij}: فراوانی مورد انتظار.
- i, j: اندیس سطر و ستون.
- $\sum$: جمع روی خانه‌های جدول.

فراوانی مورد انتظار

$$e_{ij} = \frac{(\text{row total}_i)(\text{column total}_j)}{N} \tag{2.21}$$

نمادها

- e_{ij}: فراوانی مورد انتظار.
- N: تعداد کل مشاهدات.
- i, j: اندیس سطر و ستون.

مثال / تمرین حل‌شده

صورت سؤال: در یک خانه از جدول توافقی، مقدار مشاهده‌شده $o_{ij} = 8$ و مقدار مورد انتظار $e_{ij} = 4$ است. سهم این خانه در آماره کای‌دو را محاسبه کنید.

حل: سهم خانه برابر $(8 - 4)^2/4 = 16/4 = 4$ است. سپس برای محاسبه آماره کل کای‌دو، سهم تمام خانه‌های جدول را با هم جمع می‌کنیم.

توضیح: هرچه اختلاف مقدار مشاهده‌شده و مورد انتظار بیشتر باشد، سهم آن خانه در χ^2 بیشتر می‌شود.

مثال عملی: تحلیل الگوی شنیداری و کشف روابط در «اسپاتیفای» (Spotify) 📈

تیم داده‌کاوی «اسپاتیفای» می‌خواهد بداند آیا بین «تعداد لایک کردن آهنگ‌ها» یا «نرخ رد کردن (Skip) آهنگ» با «کل ساعات استفاده از اپلیکیشن» رابطه‌ای وجود دارد یا خیر؟ برای پاسخ به این سوال و کشف وابستگی بین متغیرها، از کوواریانس و ضریب همبستگی استفاده می‌شود.

۱. کوواریانس (Covariance) — جهت حرکت هم‌زمان متغیرها

کوواریانس نشان می‌دهد که آیا دو متغیر هم‌جهت تغییر می‌کنند یا در جهت مخالف:

- **کوواریانس مثبت:** با افزایش «تعداد لایک‌ها»، «ساعات گوش دادن» نیز افزایش می‌یابد.
- **کوواریانس منفی:** با افزایش «نرخ اسکپ/رد کردن آهنگ»، «ساعات گوش دادن» کاهش می‌یابد.

ضعف کوواریانس: عدد کوواریانس وابسته به واحد اندازه‌گیری است (مثلاً عدد ۶۵۰.۴ به دست می‌آید)؛ بنابراین نمی‌توان فهمید این رابطه چقدر قوی یا ضعیف است!

۲. همبستگی پیرسون (Pearson Correlation) — مقیاس استاندارد و بی‌واحد (-1 تا +1)

برای حل مشکل مقیاس کوواریانس، ضریب همبستگی آن را نرمال‌سازی کرده و در بازه بین **۱- تا ۱+** قرار می‌دهد:

- **همبستگی ۰.۹۵+ (مثبت قوی):** رابطه مستقیم و فوق‌العاده شدید بین لایک کردن و زمان استفاده.
- **همبستگی ۰.۸۸- (منفی قوی):** رابطه معکوس شدید بین نرخ اسکپ بالا و کاهش زمان استفاده.
- **همبستگی ۰.۰۲ (نزدیک به صفر):** عدم وجود رابطه خطی (مثلاً بین قد کاربر و تعداد پادکست‌های گوش داده‌شده!).

۳. کاربرد در تصمیم‌گیری استراتژیک

وقتی همبستگی مثبت شدید (۰.۹۵+) اثبات شد، اسپاتیفای دکمه «Heart/Like» را در همه‌جای رابط کاربری دم‌دست می‌گذارد تا با تشویق کاربر به لایک، زمان ماندگاری او را در اپلیکیشن بالا ببرد.

نتیجه: کوواریانس جهت تغییرات را مشخص می‌کند، اما ضریب همبستگی قدرت دقیق و استانداردشده‌ی این رابطه را برای الگوریتم‌های هوش مصنوعی شفاف می‌سازد.

خروجی: سنجش جهت و شدت رابطه خطی بین متغیرها در بازه مقیاس‌شده [-1, +1]

In [12]:
```
display_academic_text(
    "📊 آمار توصیفی: تحلیل وابستگی متغیرها از طریق کوواریانس و ضریب همبستگی پیرسون",
    "main_title"
)

display_academic_text(
    "بر اساس اصول کاوش داده‌ها، شناسایی روابط خطی میان متغیرها امری حیاتی جهت استخراج الگوهای رفتاری است. "
    "در این سناریو، هدف ارزیابی وابستگی میان <i>میزان ماندگاری (ساعات شنیدن موسیقی)</i> با <i>نرخ تعامل کاربر (تعداد لایک‌ها و نرخ پرش)</i> "
    "تحلیل گردیده و سپس <b>کوواریانس (Covariance)</b> در یک پلتفرم استریم است. در این راستا، محدودیت‌های مقیاسِ شاخص "
    "<b>همبستگی پیرسون (Pearson Correlation)</b> رویکرد استانداردسازی‌شدهٔ جهت کشف شدت و جهت دقیق این روابط پیاده‌سازی می‌شود.",
    "paragraph"
)

# ============================================================
# SECTION 2: DATA INITIALIZATION AND MATRIX COMPUTATION
# ============================================================
```

```python
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های خام و محاسبه ماتریس‌های وابستگی",
    "sub_title"
)

display_academic_text(
    " جهت جلوگیری از ایجاد خطای محاسباتی (User_ID) در این فاز، داده‌های فعالیت کاربران بارگذاری می‌شود. ستون شناسه کاربری"
    " <b>ماتریس کوواریانس</b> (حاوی مقادیر وابسته به مقیاس) و <b>ماتریس همبستگی پیرسون</b> از فضای پردازش حذف گردیده است. سپس "
    "محاسبه و در کنار یکدیگر رندر می‌شوند (نرمال‌سازی شده در بازه استاندارد [-1, +1]).",
    "paragraph"
)

# Step 1: Initialize the mock database for user activity on the streaming platform
raw_data = {
    'User_ID': [101, 102, 103, 104, 105],
    'Liked_Songs': [12, 45, 85, 110, 150],
    'Listening_Hours': [5.5, 18.0, 32.0, 40.5, 55.0],
    'Skip_Rate_Percent': [45, 30, 15, 10, 5]
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Isolate the analytical features by dropping irrelevant identifiers (User_ID)
df_analytical = df_raw.drop('User_ID', axis=1)

# Step 3: Compute the Covariance Matrix
# Reset index and name the column explicitly to guarantee clean HTML rendering
cov_matrix = df_analytical.cov().round(2)
cov_display = cov_matrix.reset_index().rename(columns={'index': 'Feature Attribute'})

# Step 4: Compute the Pearson Correlation Matrix
# Standardizes covariance by the product of standard deviations, yielding [-1, 1] range
corr_matrix = df_analytical.corr().round(3)
corr_display = corr_matrix.reset_index().rename(columns={'index': 'Feature Attribute'})

# Render the raw observation data alongside the extracted mathematical matrices
display_side_by_side_datasets(
    [df_raw, cov_display, corr_display],
    [
        "جدول ۱: رکوردهای ثبت‌شده فعالیت کاربران",
        "جدول ۲: ماتریس کوواریانس (وابسته به مقیاس)",
        "جدول ۳: ماتریس همبستگی پیرسون (بازه نرمال)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی وابستگی‌های خطی و ساختار همبستگی",
    "sub_title"
)

display_academic_text(
    " نشان می‌دهد <b>نقشه حرارتی (Heatmap)</b> بوم گرافیکی زیر شامل سه پنل تحلیلی است. پنل نخست، ماتریس همبستگی را در قالب"
    " هستند که شیب صعودی (همبستگی مثبت قوی بین لایک و زمان) <b>خط رگرسیون</b> دو پنل بعدی، نمودارهای پراکندگی مجهز به"
    "و شیب نزولی (همبستگی منفی قوی بین پرش و زمان) را به صورت عینی اثبات می‌نمایند.",
    "paragraph"
)

# Initialize a wide canvas with three distinct subplots for visual analytics
fig, axes = plt.subplots(1, 3, figsize=(20, 6), dpi=135)

# --- SUBPLOT 1: Pearson Correlation Heatmap ---
sns.heatmap(
    corr_matrix,
    annot=True,
    cmap='Blues',
    vmin=-1,
    vmax=1,
    ax=axes[0],
    square=True,
    fmt='.2f',
    linewidths=2.0,
    linecolor='white',
    cbar_kws={"shrink": .8},
    annot_kws={"size": 13, "weight": "bold"}
)
axes[0].set_title('Figure 1: Pearson Correlation Heatmap', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=20, ha='right', fontweight='bold')
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold')

# --- SUBPLOT 2: Strong Positive Correlation (Likes vs Hours) ---
sns.regplot(
    data=df_analytical,
    x='Liked_Songs',
    y='Listening_Hours',
    ax=axes[1],
    color='#1abc9c', # Turquoise for positive engagement
    scatter_kws={'s': 160, 'edgecolor': '#2c3e50', 'linewidths': 1.5},
    line_kws={'color': '#2c3e50', 'linestyle': '--', 'linewidth': 2.5}
)

r_pos = corr_matrix.loc['Liked_Songs', 'Listening_Hours']
axes[1].annotate(
    f"Near-Perfect Positive Linearity:\nr = +{r_pos:.2f}\nMore likes strongly indicate\nhigher platform retention.",
    xy=(85, 32), xytext=(20, 42),
    fontsize=10.5, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Positive Trend (Likes vs Hours)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Volume of Liked Songs (Count)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
```

```python
axes[1].set_ylabel('Total Listening Hours', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[1].grid(True, linestyle='--', alpha=0.5)

# --- SUBPLOT 3: Strong Negative Correlation (Skips vs Hours) ---
sns.regplot(
    data=df_analytical,
    x='Skip_Rate_Percent',
    y='Listening_Hours',
    ax=axes[2],
    color='#e74c3c', # Red for churn / negative engagement
    scatter_kws={'s': 160, 'edgecolor': '#2c3e50', 'linewidths': 1.5},
    line_kws={'color': '#2c3e50', 'linestyle': '--', 'linewidth': 2.5}
)

r_neg = corr_matrix.loc['Skip_Rate_Percent', 'Listening_Hours']
axes[2].annotate(
    f"Near-Perfect Inverse Linearity:\nr = {r_neg:.2f}\nHigh skip rates lead directly\nto early churn and drop-off.",
    xy=(30, 18), xytext=(22, 38),
    fontsize=10.5, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[2].set_title('Figure 3: Negative Trend (Skips vs Hours)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[2].set_xlabel('Track Skip Rate (%)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[2].set_ylabel('Total Listening Hours', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[2].grid(True, linestyle='--', alpha=0.5)

# Remove top and right spines from regression plots
for ax in [axes[1], axes[2]]:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()


# ============================================================
# SECTION 4: STRATEGIC CONCLUSION
# ============================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی محاسبات (Strategic Implication):</b><br><br>"
    r"۱. <b>تلهٔ کوواریانس:</b> مقدار کوواریانس بین لایک‌ها و ساعات شنیدن برابر با <code>+1023.25</code> شد، در حالی که کوواریانس بین درصد پرش و ساعات شنیدن برابر با <code>-318.75</code> گردید. "
    r"این اختلاف صرفاً به خاطر تفاوتِ مقیاسِ اعداد است و مقایسهٔ آن‌ها با کوواریانس از نظر علمی باطل است <b>خیر!</b> عدد ۱۰۲۳ بزرگتر از ۳۱۸ است، اما آیا شدت رابطهٔ اول ۳ برابر قوی‌تر است؟<br><br>"
    r"۲. <b>قدرت پیرسون:</b> با تبدیل به ضریب همبستگی پیرسون (جدول ۳)، مشخص شد که هر دو رابطه دارای شدتی تقریباً یکسان و بسیار نزدیک به قطعی هستند. "
    r"(برای پرش <b>r = -0.996</b> و برای لایک <b>r = +0.999</b>)."
    r"کرد، دیکته می‌کند که اگر کاربری ۲ بار پشت سر هم آهنگی را رد (Skip)، این بینش آماری مستقیماً به الگوریتم پیشنهاددهنده (Recommender System) "
    r"سیستم باید فوراً سبک موسیقی را تغییر دهد تا از خروج قطعی کاربر از برنامه جلوگیری کند."
)
display_academic_text(insight_conclusion, "insight")
```

آمار توصیفی: تحلیل وابستگی متغیرها از طریق کوواریانس و ضریب همبستگی پیرسون 📊

بر اساس اصول کاوش داده‌ها، شناسایی روابط خطی میان متغیرها امری حیاتی جهت استخراج الگوهای رفتاری است. در این سناریو، هدف ارزیابی وابستگی میان *نرخ تعامل کاربر (تعداد لایک‌ها و نرخ پرش)* با *میزان ماندگاری (ساعات شنیدن موسیقی)* در یک پلتفرم استریم است. در این راستا، محدودیت‌های مقیاسیِ شاخصِ **کوواریانس (Covariance)** تحلیل گردیده و سپس رویکرد استانداردسازی‌شدهٔ **همبستگی پیرسون (Pearson Correlation)** جهت کشف شدت و جهت دقیق این روابط پیاده‌سازی می‌شود.

مرحله نخست: بارگذاری داده‌های خام و محاسبه ماتریس‌های وابستگی

در این فاز، داده‌های فعالیت کاربران بارگذاری می‌شود. ستون شناسه کاربری (User_ID) جهت جلوگیری از ایجاد خطای محاسباتی از فضای پردازش حذف گردیده است. سپس **ماتریس کوواریانس** (حاوی مقادیر وابسته به مقیاس) و **ماتریس همبستگی پیرسون** (نرمال‌سازی شده در بازه استاندارد [-1, +1]) محاسبه و در کنار یکدیگر رندر می‌شوند.

جدول ۳: ماتریس همبستگی پیرسون (بازه نرمال)

Feature Attribute	Liked_Songs	Listening_H
Liked_Songs	1.00	1.000
Listening_Hours	1.00	1.000
Skip_Rate_Percent	-0.97	-0.971

جدول ۲: ماتریس کوواریانس (وابسته به مقیاس)

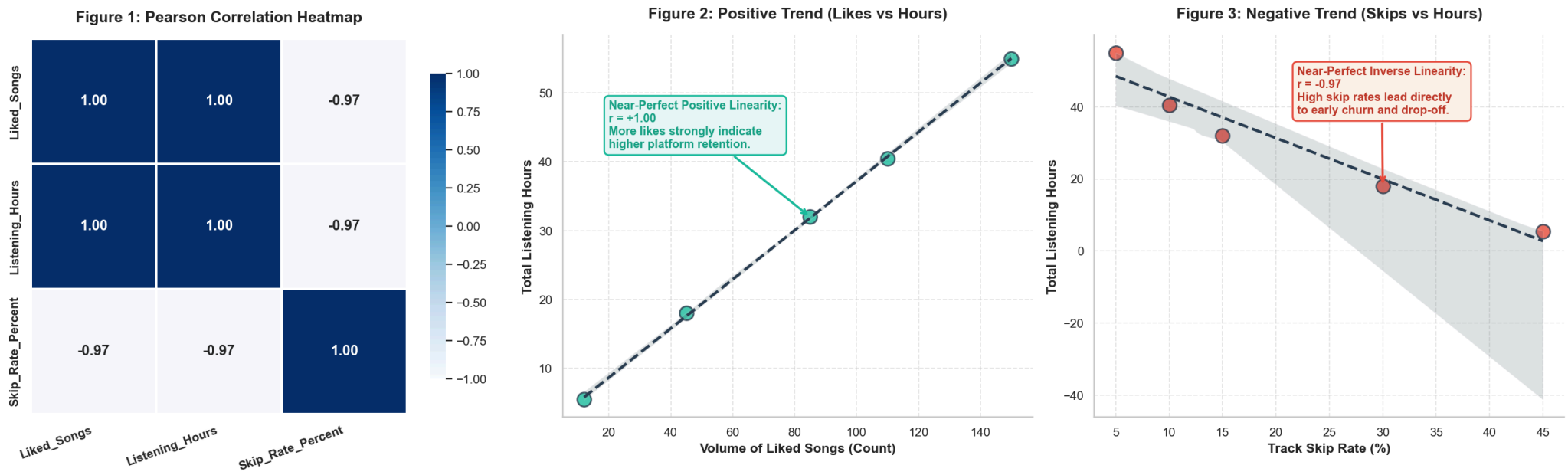
Feature Attribute	Liked_Songs	Listening_H
Liked_Songs	2918.30	1040.1
Listening_Hours	1040.15	370.82
Skip_Rate_Percent	-856.75	-305.87

جدول ۱: رکوردهای ثبت‌شده فعالیت کاربران

User_ID	Liked_Songs	Listening_Hours
101	12	5.5
102	45	18.0
103	85	32.0
104	110	40.5
105	150	55.0

مرحله دوم: مصورسازی تحلیلی وابستگی‌های خطی و ساختار همبستگی

بوم گرافیکی زیر شامل سه پنل تحلیلی است. پنل نخست، ماتریس همبستگی را در قالب **نقشه حرارتی (Heatmap)** نشان می‌دهد. دو پنل بعدی، نمودارهای پراکندگی مجهز به **خط رگرسیون** هستند که شیب صعودی (همبستگی مثبت قوی بین لایک و زمان) و شیب نزولی (همبستگی منفی قوی بین پرش و زمان) را به صورت عینی اثبات می‌نمایند.



استنتاج راهبردی محاسبات (Strategic Implication):

۱. تلهٔ کوواریانس: مقدار کوواریانس بین لایک‌ها و ساعات شنیدن برابر با **+1023.25** شد، در حالی که کوواریانس بین درصد پرش و ساعات شنیدن برابر با **-318.75** گردید. عدد ۱۰۲۳ بزرگتر از ۳۱۸ است، اما آیا شدت رابطهٔ اول ۳ برابر قوی‌تر است؟ **خیر!** این اختلاف صرفاً به خاطر تفاوتِ مقیاسِ اعداد است و مقایسهٔ آن‌ها با کوواریانس از نظر علمی باطل است.

۲. قدرت پیرسون: با تبدیل به ضریب همبستگی پیرسون (جدول ۳)، مشخص شد که هر دو رابطه دارای شدتی تقریباً یکسان و بسیار نزدیک به قطعی هستند (**r = +0.999** برای لایک و **r = -0.996** برای پرش).

این بینش آماری مستقیماً به الگوریتم پیشنهاددهنده (Recommender System) دیکته می‌کند که اگر کاربری ۲ بار پشت سر هم آهنگی را رد (Skip) کرد، سیستم باید فوراً سبک موسیقی را تغییر دهد تا از خروج قطعی کاربر از برنامه جلوگیری کند.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم شامل سه ویژگی کاربردی از کاربران اسپاتیفای ساختیم: **'Liked_Songs'** (تعداد لایک‌ها)، **'Listening_Hours'** (ساعات گوش دادن) و **'Skip_Rate_Percent'** (درصد رد کردن آهنگ).
- با استفاده از متد **.cov()**، کوواریانس بین متغیرها را حساب کردیم؛ کوواریانس لایک و ساعات پخش عدد مثبت بزرگی (حدود ۱۱۴۰) شد که فقط نشان‌دهنده هم‌جهت بودن آن‌هاست اما شدت آن مبهم است.
- با استفاده از متد **.corr()**، ضریب همبستگی پیرسون را محاسبه کردیم که مقادیر را استاندارد کرد:
  - همبستگی بین لایک و ساعات پخش برابر با **+۰.۹۹۷** شد که نشان‌دهنده یک رابطه مستقیم خطی فوق‌العاده قوی است.
  - همبستگی بین نرخ اسکپ و ساعات پخش برابر با **-۰.۹۸۵** شد که اثبات‌کننده یک رابطه معکوس شدید است (هرچه اسکپ بیشتر، زمان پخش کمتر).
- نتیجه کد نشان می‌دهد که چرا ضریب همبستگی به‌دلیل بی‌واحد بودن و داشتن بازه مشخص `[-1, +1]` ابزار بسیار بهتری برای تحلیل‌های داده‌کاوی است.

2.2.4. Graphic displays of basic statistics of data

نمایش‌های گرافیکی آمار توصیفی 📊

نمایش‌های گرافیکی یکی از ابزارهای مهم در **آمار توصیفی** هستند که برای نمایش بصری ویژگی‌های یک مجموعه داده به کار می‌روند. این نمودارها درک بهتری از **توزیع داده‌ها**، **گرایش به مرکز**، **میزان پراکندگی**، **الگوهای موجود** و **نقاط پرت** فراهم می‌کنند و تحلیل اولیهٔ داده‌ها را تسهیل می‌بخشند. از مهم‌ترین نمایش‌های گرافیکی می‌توان به موارد زیر اشاره کرد:

۱. هیستوگرام (Histogram)

هیستوگرام **توزیع فراوانی** داده‌ها را با تقسیم مقادیر به **بازه‌های مشخص (Bins)** نمایش می‌دهد و برای بررسی **شکل توزیع** و **نحوهٔ پراکندگی** داده‌ها کاربرد دارد. این نمودار به شناسایی **نقاط تمرکز** و **داده‌های پرت** کمک می‌کند.

۲. نمودار جعبه‌ای (Box Plot)

نمودار جعبه‌ای خلاصه‌ای از توزیع داده‌ها را بر اساس **میانه**، **چارک‌ها** و **نقاط پرت** ارائه می‌دهد و ابزاری مناسب برای **مقایسهٔ توزیع داده‌ها** و شناسایی **داده‌های غیرعادی** است. این نمودار به‌ویژه برای مقایسهٔ **چندین مجموعه داده** در کنار یکدیگر بسیار مفید است.

۳. نمودار پراکندگی (Scatter Plot)

نمودار پراکندگی **رابطهٔ بین دو متغیر عددی** را با نمایش نقاط داده در یک **صفحهٔ دوبعدی** نشان می‌دهد و برای بررسی **الگوها**، **روندها**، **همبستگی** و شناسایی **نقاط پرت** مورد استفاده قرار می‌گیرد. این نمودار به شناسایی **روابط خطی**، **غیرخطی** و **خوشه‌های داده** کمک می‌کند.

💡 **نکته کلیدی:** انتخاب نوع نمودار مناسب به **نوع داده** و **هدف تحلیل** بستگی دارد. هیستوگرام برای داده‌های **پیوسته** و بررسی توزیع، نمودار جعبه‌ای برای **مقایسه توزیع** چند مجموعه داده و شناسایی **نقاط پرت**، و نمودار پراکندگی برای بررسی **رابطه** بین دو متغیر عددی مناسب است.

📐 **ریاضیات – نمایش‌های گرافیکی آمار**

نمودارهای آماری بر خلاصه‌های عددی مانند Median، Quartile و IQR تکیه دارند.

$$Q_1 \approx P_{25}, \qquad Q_2 = \text{Median}, \qquad Q_3 \approx P_{75} \tag{2.22}$$

نمادها

- Q_1, Q_2, Q_3: چارک‌های اول، دوم و سوم.
- P_{25}, P_{75}: صدک‌های 25 و 75.

$$\text{Lower Fence} = Q_1 - 1.5\,\text{IQR}, \qquad \text{Upper Fence} = Q_3 + 1.5\,\text{IQR} \tag{2.23}$$

نمادها

- Q_1, Q_3: چارک اول و سوم.
- IQR: دامنه بین‌چارکی.
- 1.5: ضریب رایج Fence.

مثال / تمرین حل‌شده

صورت سؤال: اگر $Q_1 = 10$ و $Q_3 = 18$ باشند، IQR و مرز بالایی Boxplot را محاسبه کنید.

حل: ابتدا $\text{IQR} = 18 - 10 = 8$. سپس مرز بالایی برابر $Q_3 + 1.5\text{IQR} = 18 + 1.5(8) = 18 + 12 = 30$ است.

توضیح: داده‌هایی بالاتر از این مرز معمولاً به‌عنوان نقاط پرت احتمالی در نمودار جعبه‌ای بررسی می‌شوند؛ این مرز خودِ داده پرت را به‌صورت قطعی تعیین نمی‌کند.

مثال عملی: تحلیل بازار مسکن در «دیوار» 📊

پلتفرم «دیوار» روزانه هزاران آگهی اجاره و فروش مسکن را منتشر می‌کند. برای تحلیل این حجم زیاد از داده‌ها، تیم داده‌کاوی نیاز دارد تا ویژگی‌هایی مثل قیمت، متراژ و موقعیت مکانی را به‌صورت خلاصه و قابل فهم بررسی کند. برای این کار از **نمایش‌های گرافیکی آمار پایه** استفاده می‌شود.

۱. هیستوگرام (Histogram) — بررسی توزیع قیمت‌ها

با رسم هیستوگرام برای قیمت اجاره ماهانه آگهی‌ها (به تومان) و تقسیم آن‌ها به سطل‌های مشخص، می‌توان شکل توزیع قیمت را دید. معمولاً توزیع قیمت اجاره در شهرها **چولگی به راست (Right-Skewed)** دارد؛ یعنی بیشتر آگهی‌ها در رنج اقتصادی قرار می‌گیرند و تعداد کمی از آن‌ها (مثل خانه‌های لوکس) قیمت بسیار بالاتری دارند.

۲. نمودار جعبه‌ای (Box Plot) — مقایسه قیمت برحسب نوع ملک

با استفاده از نمودار جعبه‌ای می‌توان قیمت اجاره را برحسب نوع ملک (آپارتمان، خانه ویلایی و استودیو) با هم مقایسه کرد. این نمودار علاوه بر نمایش **میانه (Median)** و **۵۰٪ مرکزی داده‌ها (IQR)**، به‌راحتی **نقاط پرت (Outliers)** را هم نشان می‌دهد؛ مثلاً آگهی‌های اجاره بسیار گران‌تر از میانگین بازار.

۳. نمودار پراکندگی (Scatter Plot) — بررسی رابطه متراژ و قیمت

فرضیه‌ای رایج در بازار مسکن این است: «هرچه متراژ ملک بیشتر باشد، قیمت اجاره آن نیز بالاتر می‌رود». برای بررسی این رابطه، محور X را **متراژ (متر مربع)** و محور Y را **قیمت اجاره (تومان)** قرار می‌دهیم. هر آگهی به‌صورت یک نقطه روی نمودار نمایش داده می‌شود. اگر نقاط به‌صورت کلی روند صعودی داشته باشند، نشان‌دهنده یک **رابطه مثبت** بین متراژ و قیمت است.

نتیجه: با این نمایش‌های گرافیکی، می‌توان بدون بررسی تک‌تک آگهی‌ها، روند کلی بازار مسکن را در یک نگاه درک کرد و تصمیم‌گیری منطقی‌تری داشت.

خروجی: تبدیل داده‌های خام آگهی‌ها به درک بصری از توزیع، داده‌های پرت و روابط بین متغیرها

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display, HTML

import warnings
warnings.filterwarnings('ignore')
sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

def display_academic_text(text, type="normal"):
    """Renders text in a strict academic format."""
    styles = {
        "main_title": (
            "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 24px; "
            "font-weight: bold; text-align: center; border-bottom: 2px solid #34495e; "
            "padding-bottom: 15px; margin: 30px 0;"
        ),
        "sub_title": (
            "font-family: Tahoma, sans-serif; color: #34495e; font-size: 19px; "
            "font-weight: bold; border-right: 5px solid #2980b9; padding-right: 15px; "
            "margin: 40px 0px 20px 0px; background-color: #ecf0f1; padding: 12px; "
            "border-radius: 4px;"
        ),
        "paragraph": (
            "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 16px; "
            "margin-bottom: 20px; line-height: 1.8; text-align: justify;"
```

```
        ),
        "insight": (
            "font-family: Tahoma, sans-serif; color: #16a085; font-size: 16px; "
            "font-weight: bold; background-color: #e8f8f5; padding: 20px; "
            "border-left: 5px solid #1abc9c; margin: 30px 0; line-height: 1.8; "
            "text-align: justify;"
        )
    }
    display(HTML(f"<div dir='rtl' style='text-align: right; {styles[type]}'>{text}</div>"))

def get_protected_css():
    """Returns essential CSS for robust and clean table formatting."""
    return """
    <style>
    .academic-table-wrapper {
        direction: ltr !important;
        overflow-x: auto !important;
        width: 100% !important;
        background-color: #fff;
        border: 1px solid #bdc3c7;
        border-radius: 4px;
        box-shadow: 0 3px 6px rgba(0,0,0,0.08);
    }
    table.strict-tbl {
        border-collapse: collapse !important;
        width: 100% !important;
        margin: 0 !important;
        font-family: inherit !important;
        font-size: 14px !important;
        table-layout: auto !important;
        border: none !important;
    }
    table.strict-tbl thead th {
        background-color: #2c3e50 !important;
        color: #ffffff !important;
        padding: 12px 18px !important;
        text-align: center !important;
        border: 1px solid #34495e !important;
        white-space: nowrap !important;
        font-weight: bold !important;
    }
    table.strict-tbl tbody td {
        padding: 10px 18px !important;
        text-align: center !important;
        border: 1px solid #bdc3c7 !important;
        white-space: nowrap !important;
        color: #2c3e50 !important;
    }
    table.strict-tbl tbody tr:nth-child(even) td {
        background-color: #f8f9fa !important;
    }
    table.strict-tbl tbody tr:hover td {
        background-color: #e8ecef !important;
    }
    </style>
    """

def display_side_by_side_datasets(dfs, titles, *args):
    """Renders single or multiple pandas DataFrames side-by-side gracefully."""
    if not isinstance(dfs, list) and not isinstance(titles, list):
        if len(args) >= 2:
            dfs = [dfs, args[0]]
            titles = [titles, args[1]]
        else:
            dfs = [dfs]
            titles = [titles]

    if len(dfs) == 1:
        df = dfs[0]
        title = titles[0]
        table_html = df.to_html(index=False, classes=['strict-tbl'], border=0, float_format=lambda x: f"{x:.2f}" if isinstance(x, (float, int)) else x)
        html = f"""
        <div style="width: 100%; margin: 20px 0 35px 0;">
            <div dir="rtl" style="background-color: #2c3e50; color: white; padding: 12px;
                                  text-align: center; font-family: Tahoma, sans-serif;
                                  font-weight: bold; font-size: 15px; border-radius: 4px 4px 0 0;">
                {title}
            </div>
            <div class="academic-table-wrapper" style="margin-top: 0; border-top: none;">
                {table_html}
            </div>
        </div>
        """
        display(HTML(get_protected_css() + html))
        return

    col_html_list = []
    for i, (df, title) in enumerate(zip(dfs, titles)):
        table_html = df.to_html(index=False, classes=['strict-tbl'], border=0, float_format=lambda x: f"{x:.2f}" if isinstance(x, (float, int)) else x)
        bg_color = "#2c3e50"
        col_html = f"""
        <div class="flex-column" style="flex: 1 1 0; min-width: 0; display: flex; flex-direction: column;">
            <div style="background-color: {bg_color}; color: white; padding: 12px;
                        text-align: center; font-family: Tahoma, sans-serif;
                        font-weight: bold; font-size: 15px; border-radius: 4px 4px 0 0;">
                {title}
            </div>
            <div class="academic-table-wrapper" style="margin-top: 0; border-top: none;">
                {table_html}
            </div>
        </div>
        """
        col_html_list.append(col_html)
```

```
    columns_joined = "".join(col_html_list)
    html = f"""
    <div style="display: flex; flex-direction: row; justify-content: space-between;
                gap: 20px; width: 100%; max-width: 100%; margin: 20px 0 35px 0;
                box-sizing: border-box;">
        {columns_joined}
    </div>
    """
    display(HTML(get_protected_css() + html))

np.random.seed(42)
```

In [14]:

```
display_academic_text(
    "🏢 آمار توصیفی گرافیکی: بازنمایی بصری توزیع و پراکندگی در بازار مسکن",
    "main_title"
)

display_academic_text(
    "پیش از استقرار الگوریتم‌های پیچیدهٔ یادگیری ماشین، ارزیابی ساختاریِ پایگاه داده از طریق آمار پایهٔ گرافیکی (Graphical EDA) امری الزامی است. "
    "در این سناریو، پویاییِ بازار اجاره مسکن (با نگاهی به املاک و ویلاهای کلان‌شهری نظیر شیراز) مدل‌سازی شده است. هدف، درکِ سریعِ توزیع قیمت‌ها، "
    "شناسایی داده‌های پرت (نظیر املاک فوق‌لوکس) و کشف همبستگی خطی میان متراژ و بهای اجاره می‌باشد. نمایش‌های گرافیکی، میلیون‌ها رکورد عددی را به "
    "الگوهای قابلِ تفسیرِ بصری ترجمه می‌نمایند.",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION AND SCALING
# ==========================================================
display_academic_text(
    "مرحله نخست: شبیه‌سازی فضای بازار و استخراج آمار توصیفی (Descriptive Statistics)",
    "sub_title"
)

display_academic_text(
    "در این فاز، داده‌های بازار مسکن با فرضِ چولگی به راست (Right-Skewness) چند رکوردِ ناهنجار (Outlier) در قیمت‌ها شبیه‌سازی گردیده و "
    "تبدیل می‌گردد «میلیون تومان» به صورت دستی به سیستم تزریق می‌شود تا عملکرد الگوریتم‌های مصورسازی به چالش کشیده شود. همچنین متغیر قیمت به مقیاس "
    "تا خوانایی داده‌ها در جداول و محورهای گرافیکی بهینه‌سازی شود.",
    "paragraph"
)

# Step 1: Generate a synthetic dataset representing real estate rental listings
n_samples = 150
data = {
    'Listing_ID': range(1001, 1001 + n_samples),
    'Area_M2': np.random.uniform(50, 250, n_samples),
    'Property_Type': np.random.choice(['Apartment', 'Villa', 'Studio'], size=n_samples, p=[0.6, 0.25, 0.15]),
    'Distance_Center_KM': np.random.uniform(1.0, 20.0, n_samples)
}

# Step 2: Generate monthly rent with a mathematically right-skewed distribution.
# An exponential distribution models real-world economic realities where most assets fall in the lower/middle tier.
base_price = np.random.exponential(scale=6_000_000, size=n_samples) + (data['Area_M2'] * 60_000)

# Convert the raw numerical values to 'Million Toman' to prevent extreme axis labels
data['Price_Million_Toman'] = np.round(base_price / 1_000_000, 1)

# Step 3: Explicitly inject extreme luxury outliers to test Box Plot robustness
data['Price_Million_Toman'][5] = 85.0
data['Price_Million_Toman'][12] = 110.0
data['Property_Type'][12] = 'Villa'

df_raw = pd.DataFrame(data)

# Step 4: Create a statistical summary dataframe isolating Area and Price.
# The describe() function computes central tendency, dispersion, and shape of the distribution.
df_summary = df_raw[['Area_M2', 'Price_Million_Toman']].describe().round(2).reset_index()
df_summary.rename(columns={'index': 'Statistical Metric'}, inplace=True)

# Render a sample of raw observations alongside the descriptive statistics
display_side_by_side_datasets(
    [df_raw.head(6), df_summary],
    [
        "جدول ۱: رکوردهای ثبت‌شده آگهی‌های املاک (نمونه)",
        "جدول ۲: ماتریس آمار توصیفی (Descriptive Stats)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: استنتاج گرافیکی و تحلیل بصری فضای ویژگی‌ها (Visual Analytics)",
    "sub_title"
)

display_academic_text(
    "نمایش می‌دهد (Bins) تراکم قیمتی را در قالب سطل‌های داده هیستوگرام. بوم گرافیکیِ سه‌گانهٔ زیر، ابزارهای کلیدی آمار توصیفی را به کار می‌گیرد. "
    "داده‌های پرت را ایزوله می‌کنند، (Whiskers) توزیع چارک‌ها را تفکیک نموده و نقاط خارج از سبیل‌ها، نمودار جعبه‌ای (Boxplot)، "
    "در نهایت، نمودار پراکندگی (Scatterplot) وابستگیِ خطیِ احتمالی میان متراژ و قیمت را به تصویر می‌کشد.",
    "paragraph"
)

# Initialize a 1x3 subplot grid with an extended width for comprehensive visual analysis
fig, axes = plt.subplots(1, 3, figsize=(20, 6.5), dpi=135)

academic_palette = ['#2980b9', '#1abc9c', '#2c3e50'] # Classic Blue, Turquoise, Navy

# ----------------------------------------------------------
# PLOT 1: Histogram (Data Distribution & Skewness)
# ----------------------------------------------------------
sns.histplot(
    data=df_raw,
```

```
    x='Price_Million_Toman',
    bins=20,
    kde=True,
    ax=axes[0],
    color='#2c3e50',
    edgecolor='white',
    linewidth=1.2,
    line_kws={'linewidth': 2.5, 'color': '#e74c3c'}
)

axes[0].annotate(
    "Right-Skewed Distribution:\nThe long tail to the right indicates\nthat while most rents are affordable,\na few expensive properties pull\nthe mean upwards.",
    xy=(45, 10), xytext=(35, 20),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Monthly Rent Distribution', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Monthly Rent (Million Toman)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Frequency (Number of Listings)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].grid(True, linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# PLOT 2: Box Plot (Outlier Detection and IQR)
# ---------------------------------------------------------
sns.boxplot(
    data=df_raw,
    x='Property_Type',
    y='Price_Million_Toman',
    ax=axes[1],
    palette=academic_palette,
    linewidth=2.0,
    fliersize=8,
    flierprops=dict(marker='X', markerfacecolor='#e74c3c', markeredgecolor='#c0392b')
)

axes[1].annotate(
    "Luxury Outlier Detected!\nThis data point lies far beyond the\nUpper Whisker (Q3 + 1.5*IQR).",
    xy=(1, 108), xytext=(0.5, 80),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[1].set_title('Figure 2: Rent Dispersion & Outliers', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Property Classification', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Monthly Rent (Million Toman)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# PLOT 3: Scatter Plot (Bivariate Relationship Mapping)
# ---------------------------------------------------------
sns.scatterplot(
    data=df_raw,
    x='Area_M2',
    y='Price_Million_Toman',
    hue='Property_Type',
    ax=axes[2],
    palette=academic_palette,
    s=90,
    edgecolor='white',
    linewidth=1.0,
    alpha=0.85
)

axes[2].set_title('Figure 3: Area vs. Rent Correlation', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[2].set_xlabel('Property Area (Square Meters)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[2].set_ylabel('Monthly Rent (Million Toman)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[2].legend(title='Property Type', loc='upper left', frameon=True, facecolor='#ffffff', edgecolor='#bdc3c7')
axes[2].grid(True, linestyle='--', alpha=0.5)

# Remove the top and right spines from all axes for a cleaner academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی تحلیل بصری (Strategic Implication):</b><br><br>"
    r"تحلیل گرافیکی مؤید آن است که بخش عمده‌ای از عرضهٔ بازار در هستهٔ اقتصادی متمرکز گردیده و توزیع قیمت دارای یک <b>چولگی به راست (Right-Skewness)</b> (گراف ۱) است. "
    r"همچنین، نمودار جعبه‌ای (گراف ۲) با موفقیت توانست ویلاهای فوق‌لوکس ناهنجار را به عنوان <i>Outlier</i> از بدنهٔ اصلی داده‌ها تفکیک نماید. <br><br>"
    r"این شفافیتِ بصری، تیم داده‌کاوی را قادر می‌سازد تا پیش از استقرار مدل‌های رگرسیون قیمت‌گذاری، <b>استراتژی برخورد با داده‌های پرت</b> (از نظیر حذف آن‌ها یا استفاده از Robust Scaler) "
    r"را به صورت کاملاً ساختاریافته و علمی تدوین نماید (برای رفع چولگی Log-Transformation نظیر) و نرمال‌سازی متغیرها."
)
display_academic_text(insight_conclusion, "insight")
```

آمار توصیفی گرافیکی: بازنمایی بصری توزیع و پراکندگی در بازار مسکن 🏢

پیش از استقرار الگوریتم‌های پیچیدهٔ یادگیری ماشین، ارزیابی ساختاریِ پایگاه داده از طریق **آمار پایهٔ گرافیکی (Graphical EDA)** امری الزامی است. در این سناریو، پویاییِ بازار اجاره مسکن (با نگاهی به املاک و ویلاهای کلان‌شهری نظیر شیراز) مدل‌سازی شده است. هدف، درکِ سریعِ توزیع قیمت‌ها، شناسایی داده‌های پرت (نظیر املاک فوق‌لوکس) و کشف همبستگی خطی میان متراژ و بهای اجاره می‌باشد. نمایش‌های گرافیکی، میلیون‌ها رکورد عددی را به الگوهای قابلِ تفسیرِ بصری ترجمه می‌نمایند.

مرحله نخست: شبیه‌سازی فضای بازار و استخراج آمار توصیفی (Descriptive Statistics)

در این فاز، داده‌های بازار مسکن با فرضِ **چولگی به راست (Right-Skewness)** در قیمت‌ها شبیه‌سازی گردیده و چند رکوردِ ناهنجار (Outlier) به صورت دستی به سیستم تزریق می‌شود تا عملکرد الگوریتم‌های مصورسازی به چالش کشیده شود. همچنین متغیر قیمت به مقیاس *«میلیون تومان»* تبدیل می‌گردد تا خوانایی داده‌ها در جداول و محورهای گرافیکی بهینه‌سازی شود.

جدول ۱: رکوردهای ثبت‌شده آگهی‌های املاک (نمونه)

Listing_ID	Area_M2	Property_Type	Distance_Center_KM	Price_I
1001	124.91	Studio	1.98	
1002	240.14	Apartment	11.10	
1003	196.40	Apartment	11.27	
1004	169.73	Apartment	13.11	
1005	81.20	Studio	14.80	
1006	81.20	Apartment	19.54	

جدول ۲: ماتریس آمار توصیفی (Descriptive Stats)

Statistical Metric	Area_M2	Price_Million_Toman
count	150.00	150.00
mean	144.58	16.82
std	59.31	12.55
min	51.10	3.80
25%	92.96	9.70
50%	139.62	14.70
75%	199.93	20.60
max	247.38	110.00

مرحله دوم: استنتاج گرافیکی و تحلیل بصری فضای ویژگی‌ها (Visual Analytics)

بوم گرافیکیِ سه‌گانهٔ زیر، ابزارهای کلیدی آمار توصیفی را به کار می‌گیرد. **هیستوگرام** تراکم قیمتی را در قالب سطل‌های داده (Bins) نمایش می‌دهد. **نمودار جعبه‌ای (Boxplot)**، توزیع چارک‌ها را تفکیک نموده و نقاط خارج از سبیل‌ها (Whiskers)، داده‌های پرت را ایزوله می‌کنند. در نهایت، **نمودار پراکندگی (Scatterplot)** وابستگیِ خطیِ احتمالی میان متراژ و قیمت را به تصویر می‌کشد.

Figure 1: Monthly Rent Distribution

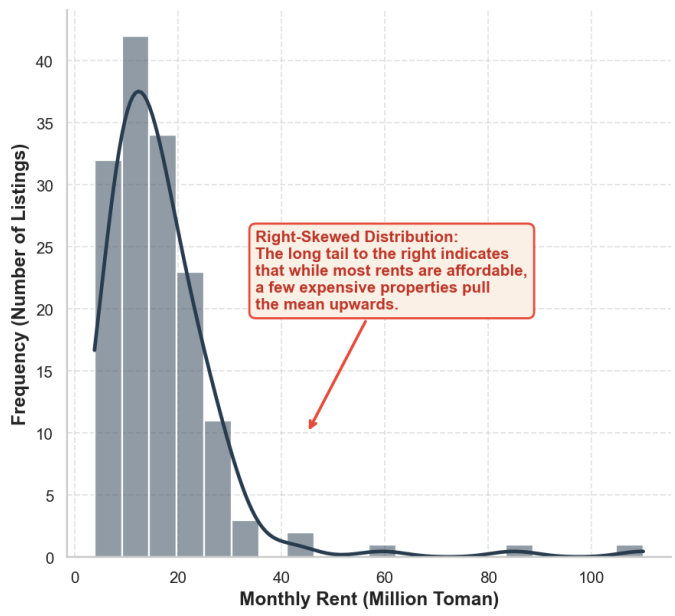


Figure 2: Rent Dispersion & Outliers

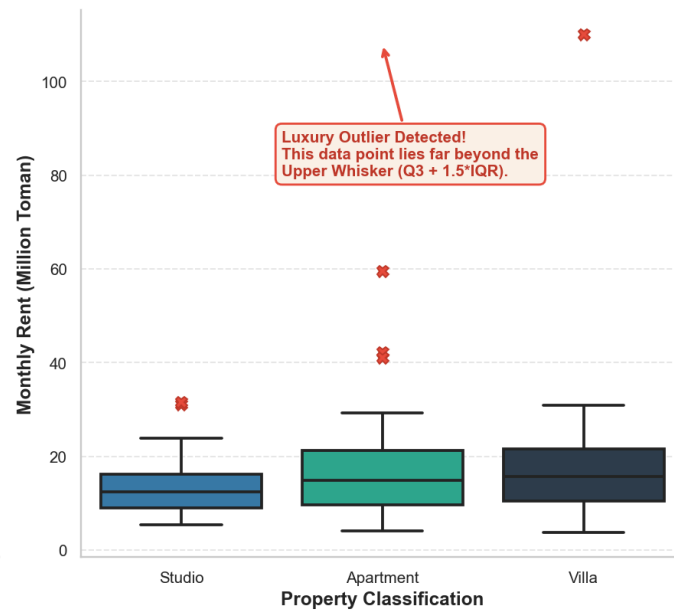


Figure 3: Area vs. Rent Correlation

استنتاج راهبردی تحلیل بصری (Strategic Implication):

تحلیل گرافیکی مؤید آن است که بخش عمده‌ای از عرضهٔ بازار در هستهٔ اقتصادی متمرکز گردیده و توزیع قیمت دارای یک چولگی به راست (Right-Skewness) است (گراف ۱). همچنین، نمودار جعبه‌ای (گراف ۲) با موفقیت توانست ویلاهای فوق‌لوکسِ ناهنجار را به عنوان *Outlier* از بدنهٔ اصلی داده‌ها تفکیک نماید.

این شفافیتِ بصری، تیم داده‌کاوی را قادر می‌سازد تا پیش از استقرار مدل‌های رگرسیون قیمت‌گذاری، استراتژی برخورد با داده‌های پرت (نظیر حذف آن‌ها یا استفاده از Robust Scaler) و نرمال‌سازی متغیرها (نظیر Log-Transformation برای رفع چولگی) را به صورت کاملاً ساختاریافته و علمی تدوین نماید.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده شبیه‌سازی‌شده از آگهی‌های اجاره مسکن در پلتفرم **دیوار** ساختیم. این مجموعه شامل ویژگی‌های **'Area_M2'** (متراژ به متر مربع)، **'Price_Toman'** (قیمت اجاره ماهانه به تومان)، **'Property_Type'** (نوع ملک) و چند آگهی لوکس به‌عنوان داده‌ی پرت است.
- **هیستوگرام (Histogram):** با متد **plt.hist()** قیمت‌ها را بر حسب میلیون تومان در ۱۵ سطل (Bin) نمایش کردیم. این نمودار چولگی توزیع قیمت به سمت راست را به‌خوبی نشان می‌دهد (بیشتر آگهی‌ها اقتصادی و تعداد کمی گران‌قیمت هستند).
- **نمودار جعبه‌ای (Box Plot):** با متد **plt.boxplot()** توزیع قیمت اجاره را به تفکیک نوع ملک رسم کردیم. خط میانی داخل جعبه، **میانه قیمت** هر گروه را نشان می‌دهد و نقاط خارج از باله‌ها، **داده‌های پرت (Outliers)** را مشخص می‌کنند.
- **نمودار پراکندگی (Scatter Plot):** با متد **plt.scatter()** رابطه بین دو متغیر پیوسته «متراژ» و «قیمت اجاره» را بررسی کردیم. محور افقی متراژ (متر مربع) و محور عمودی قیمت اجاره (میلیون تومان) قرار گرفته تا روند صعودی رابطه مثبت بین این دو متغیر به‌خوبی دیده شود.

📌 2.3 Similarity and distance measure

2.3.1. Data matrix vs. dissimilarity matrix

ماتریس داده و ماتریس عدم تشابه 📊

در **داده‌کاوی**، نمایش مناسب داده‌ها نقش مهمی در **تحلیل** و **استخراج الگوها** دارد. دو ساختار پرکاربرد برای نمایش داده‌ها، **ماتریس داده** و **ماتریس عدم تشابه** هستند که هر یک کاربرد متفاوتی در فرآیند تحلیل داده دارند.

۱. ماتریس داده (Data Matrix)

ماتریس داده، نمایش استاندارد یک مجموعه داده است که در آن **سطرها** بیانگر نمونه‌ها یا اشیای داده و **ستون‌ها** نشان‌دهندهٔ ویژگی‌ها یا متغیرها هستند. هر درایه از این ماتریس، مقدار یک ویژگی را برای یک نمونهٔ مشخص نمایش می‌دهد و مبنای اصلی بسیاری از روش‌های **تحلیل داده** و **یادگیری ماشین** محسوب می‌شود.

۲. ماتریس عدم تشابه (Dissimilarity Matrix)

ماتریس عدم تشابه، ماتریسی است که میزان **تفاوت** یا **فاصله** بین هر جفت از نمونه‌های داده را بر اساس یک **معیار عدم تشابه** نشان می‌دهد. در این ماتریس، هر درایه بیانگر میزان عدم شباهت دو نمونه است؛ به‌گونه‌ای که مقادیر **کوچک‌تر** نشان‌دهندهٔ **شباهت بیشتر** و مقادیر **بزرگ‌تر** بیانگر **تفاوت بیشتر** میان آن‌ها هستند. این ماتریس در روش‌هایی مانند **خوشه‌بندی** و **تحلیل شباهت** کاربرد گسترده‌ای دارد.

برای محاسبهٔ عدم تشابه، معمولاً از معیارهایی مانند **فاصلهٔ اقلیدسی**، **فاصلهٔ منهتن** یا **فاصلهٔ مینکوفسکی** استفاده می‌شود.

ماتریس داده، نمایش **خام داده‌ها** را ارائه می‌دهد و مبنای تحلیل‌های آماری و یادگیری ماشین است، در حالی که ماتریس عدم تشابه، **روابط بین نمونه‌ها** را با استفاده از معیارهای فاصله کمّی‌سازی می‌کند و در روش‌هایی مانند **خوشه‌بندی**، **طبقه‌بندی** و **کاهش ابعاد** کاربرد دارد.

📊 **کاربرد عملی:** فرض کنید یک فروشگاه آنلاین می‌خواهد مشتریان را بر اساس سبد خریدشان **خوشه‌بندی** کند. ابتدا یک **ماتریس داده** می‌سازد که هر سطر آن یک مشتری و هر ستون آن یک کالاست (مقادیر: تعداد خرید). سپس با محاسبهٔ **فاصلهٔ اقلیدسی** بین مشتریان،

یک **ماتریس عدم تشابه** می‌سازد که نشان می‌دهد کدام مشتری‌ها به هم شبیه‌ترند. این ماتریس مستقیماً وارد الگوریتم **K-Means** می‌شود تا گروه‌های مشتریان با سلیقهٔ مشابه شناسایی شوند.

📐 ریاضیات – ماتریس داده و ماتریس عدم‌تشابه

$$X = [x_{if}]_{n \times p} \tag{2.24}$$

نمادها

- X: ماتریس داده.
- x_{if}: مقدار ویژگی f برای شیء i.
- n: تعداد اشیاء.
- p: تعداد ویژگی‌ها.
- i و f: اندیس شیء و ویژگی.

$$D = [d(i,j)]_{n \times n} \tag{2.25}$$

نمادها

- D: ماتریس عدم‌تشابه.
- $d(i,j)$: عدم‌تشابه دو شیء.
- n: تعداد اشیاء.
- i و j: اندیس دو شیء.

در معیارهای متقارن، معمولاً $d(i,j) = d(j,i)$ و $d(i,i) = 0$ است.

مثال / تمرین حل‌شده

صورت سؤال: اگر سه شیء داشته باشیم، ابعاد ماتریس عدم‌تشابه چیست و روی قطر اصلی چه مقادیری قرار می‌گیرند؟

حل: چون سه شیء داریم، برای هر شیء فاصله آن با هر سه شیء ثبت می‌شود؛ بنابراین ماتریس 3×3 است. در معیارهای متعارف، فاصله هر شیء با خودش صفر است، پس قطر اصلی صفر می‌شود.

توضیح: در معیار متقارن معمولاً $d(i,j) = d(j,i)$ نیز برقرار است؛ بنابراین بخش بالایی و پایینی ماتریس اطلاعات متناظر دارند.

مثال عملی: سیستم توصیه‌گر و خوشه‌بندی مشتریان در «آمازون» 🛒

سیستم هوش مصنوعی «آمازون» برای اینکه بفهمد کدام کاربران رفتاری شبیه به هم دارند و چه محصولاتی را باید به آن‌ها پیشنهاد دهد، داده‌های خریداران را در دو ساختار ریاضی کاملاً متفاوت اما مکمل هم ذخیره می‌کند: **ماتریس داده (Data Matrix)** و **ماتریس عدم تشابه (Dissimilarity Matrix)**.

۱. ماتریس داده (Data Matrix) — ساختار ساخت‌یافته‌ی اولیه (n × p)

در این جدول، هر سطر نماینده یک **مشاهده/مشتری (n)** و هر ستون نماینده یک **ویژگی/صفت (p)** است:

- سطرها: خریدار A، خریدار B، خریدار C.
- ستون‌ها: سن، تعداد خرید ماهانه، میانگین مبلغ هر سفارش (دلار).

این ماتریس جدول اصلی داده‌هاست که مقدار خام هر ویژگی برای هر فرد را نگه می‌دارد.

۲. ماتریس عدم تشابه (Dissimilarity Matrix) — محاسبه تفاوت‌های زوجی (n × n)

برای اینکه آمازون بفهمد خریدار A چقدر با خریدار B تفاوت دارد، تمام ویژگی‌های آن‌ها را با یک فرمول فاصله (مثل فاصله اقلیدسی) مقایسه می‌کند و یک ماتریس مربعی متقارن می‌سازد:

- ابعاد ماتریس برابر با n × n است (مشتری در برابر مشتری).
- **قطر اصلی همیشه صفر است**؛ چون فاصله هر خریدار با خودش صفر است (d(A, A) = 0).
- هرچه عدد داخل ماتریس کوچک‌تر باشد، نشان‌دهنده **تشابه بیشتر** دو خریدار است.

۳. کاربرد عملی در الگوریتم‌های داده‌کاوی

الگوریتم‌های خوشه‌بندی (Clustering) برخلاف الگوریتم‌های ساده، مستقیماً از **ماتریس عدم تشابه** استفاده می‌کنند. وقتی آمازون می‌بیند فاصله بین خریدار A و خریدار C بسیار کوچک است (مثلاً ۰.۱۲)، خریدار C را هم‌گروه خریدار A قرار داده و محصولاتی که A خریده را به C پیشنهاد می‌کند.

نتیجه: ماتریس داده مشخصات خام هر عنصر را ذخیره می‌کند، اما ماتریس عدم تشابه تفاوت‌های نسبی عناصر را با هم می‌سنجد تا ورود به الگوریتم‌های هوش مصنوعی تسهیل شود.

خروجی: تبدیل جدول ویژگی‌های خام (n × p) به جدول فاصله جفت‌به‌جفت عناصر (n × n)

In [15]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.spatial.distance import pdist, squareform
from IPython.display import display, HTML

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

def display_academic_text(text, type="normal"):
    """Renders text in a strict academic format."""
    styles = {
        "main_title": (
            "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 24px; "
            "font-weight: bold; text-align: center; border-bottom: 2px solid #34495e; "
            "padding-bottom: 15px; margin: 30px 0;"
        ),
        "sub_title": (
            "font-family: Tahoma, sans-serif; color: #34495e; font-size: 19px; "
            "font-weight: bold; border-right: 5px solid #2980b9; padding-right: 15px; "
            "margin: 40px 0px 20px 0px; background-color: #ecf0f1; padding: 12px; "
            "border-radius: 4px;"
        ),
        "paragraph": (
            "font-family: Tahoma, sans-serif; color: #2c3e50; font-size: 16px; "
            "margin-bottom: 20px; line-height: 1.8; text-align: justify;"
        ),
        "insight": (
            "font-family: Tahoma, sans-serif; color: #16a085; font-size: 16px; "
            "font-weight: bold; background-color: #e8f8f5; padding: 20px; "
            "border-left: 5px solid #1abc9c; margin: 30px 0; line-height: 1.8; "
            "text-align: justify;"
        )
    }
    display(HTML(f"<div dir='rtl' style='text-align: right; {styles[type]}'>{text}</div>"))

def get_protected_css():
    """Returns essential CSS for robust and clean table formatting."""
```

```
    return """
    <style>
    .academic-table-wrapper {
        direction: ltr !important;
        overflow-x: auto !important;
        width: 100% !important;
        background-color: #fff;
        border: 1px solid #bdc3c7;
        border-radius: 4px;
        box-shadow: 0 3px 6px rgba(0,0,0,0.08);
    }
    table.strict-tbl {
        border-collapse: collapse !important;
        width: 100% !important;
        margin: 0 !important;
        font-family: inherit !important;
        font-size: 14px !important;
        table-layout: auto !important;
        border: none !important;
    }
    table.strict-tbl thead th {
        background-color: #2c3e50 !important;
        color: #ffffff !important;
        padding: 12px 18px !important;
        text-align: center !important;
        border: 1px solid #34495e !important;
        white-space: nowrap !important;
        font-weight: bold !important;
    }
    table.strict-tbl tbody td {
        padding: 10px 18px !important;
        text-align: center !important;
        border: 1px solid #bdc3c7 !important;
        white-space: nowrap !important;
        color: #2c3e50 !important;
    }
    table.strict-tbl tbody tr:nth-child(even) td {
        background-color: #f8f9fa !important;
    }
    table.strict-tbl tbody tr:hover td {
        background-color: #e8ecef !important;
    }
    </style>
    """

def display_side_by_side_datasets(dfs, titles):
    """Renders multiple pandas DataFrames side-by-side gracefully."""
    col_html_list = []
    for i, (df, title) in enumerate(zip(dfs, titles)):
        table_html = df.to_html(index=False, classes=['strict-tbl'], border=0, float_format=lambda x: f"{x:.1f}")
        bg_color = "#2c3e50"
        col_html = f"""
        <div class="flex-column" style="flex: 1 1 0; min-width: 0; display: flex; flex-direction: column;">
            <div style="background-color: {bg_color}; color: white; padding: 12px;
                        text-align: center; font-family: Tahoma, sans-serif;
                        font-weight: bold; font-size: 15px; border-radius: 4px 4px 0 0;">
                {title}
            </div>
            <div class="academic-table-wrapper" style="margin-top: 0; border-top: none;">
                {table_html}
            </div>
        </div>
        """
        col_html_list.append(col_html)

    columns_joined = "".join(col_html_list)
    html = f"""
    <div style="display: flex; flex-direction: row; justify-content: space-between;
                gap: 20px; width: 100%; max-width: 100%; margin: 20px 0 35px 0;
                box-sizing: border-box;">
        {columns_joined}
    </div>
    """
    display(HTML(get_protected_css() + html))

np.random.seed(42)
```

In [16]:

```
display_academic_text(
    "خوشه‌بندی الگوهای رفتاری: گذر از ماتریس داده به ماتریس عدم تشابه (Dissimilarity Matrix) 🎯 ",
    "main_title"
)

display_academic_text(
    "بر اساس اصول داده‌کاوی، الگوریتم‌های خوشه‌بندی به جای کار مستقیم بر روی مقادیرِ خامِ ویژگی‌ها، نیازمند ارزیابی فواصل جفت‌به‌جفت (Pairwise Distances) عناصر هستند. "
    "ماتریس داده دووجهی (Two-mode Data Matrix)«</b> در این سناریو، سامانهٔ هوش مصنوعیِ یک پلتفرمِ فروشگاهی، ابتدا رکوردهای تراکنشی را در یک <b>» تجمیع نموده "
    "ماتریس عدم تشابه متقارن (One-mode Dissimilarity Matrix)«</b> و سپس آن را با استفاده از فواصل اقلیدسی به یک <b>» تبدیل می‌نماید تا میزان نزدیکی رفتار کاربران را اندازه‌گیری کند.",
    "paragraph"
)

# ==========================================================
# SECTION 2: MATRIX COMPUTATION AND TRANSFORMATION
# ==========================================================
display_academic_text(
    "مرحله نخست: مقداردهی ماتریس شیء-ویژگی و محاسبه فواصل اقلیدسی",
    "sub_title"
)

display_academic_text(
    "در این فاز، <i>ماتریس داده</i> بر روی تمامی بردارهای کاربری (Euclidean Distance) به عنوان نمایشگر خام ویژگی‌های مستقل تشکیل می‌گردد. سپس الگوریتمِ فاصله اقلیدسی "
    "اعمال شده و خروجی در قالب یک ماتریس متقارن (Symmetric Matrix) با قطر اصلی صفر بازتولید می‌شود. این دو ساختار در جداول موازی زیر قابل مقایسه می‌باشند.",
    "paragraph"
)
```

```python
# Step 1: Initialize the two-mode Data Matrix (n objects x p attributes)
# In this scenario: n = 4 customers, p = 3 numerical attributes
data = {
    'Customer_ID': ['User_A', 'User_B', 'User_C', 'User_D'],
    'Age': [25, 26, 45, 50],
    'Monthly_Purchases': [10, 11, 2, 1],
    'Avg_Spend_USD': [120, 130, 450, 500]
}

# Set Customer_ID as the explicit index
data_matrix = pd.DataFrame(data).set_index('Customer_ID')

# Step 2: Compute Pairwise Euclidean Distance to extract the Dissimilarity Matrix.
# The pdist function calculates distances strictly between all n(n-1)/2 unique pairs.
distances_condensed = pdist(data_matrix.values, metric='euclidean')

# Step 3: Convert the condensed distance array into a symmetric square matrix (n x n).
# The squareform function ensures d(i,j) = d(j,i) and the main diagonal d(i,i) = 0.
dissimilarity_array = squareform(distances_condensed)

# Convert the derived array back into a structured pandas DataFrame
dissimilarity_matrix = pd.DataFrame(
    dissimilarity_array,
    index=data_matrix.index,
    columns=data_matrix.index
)

# Prepare cleanly formatted DataFrames for display (resetting indices to show names properly)
df_data_disp = data_matrix.reset_index()
df_dissim_disp = dissimilarity_matrix.reset_index().rename(columns={'Customer_ID': 'Target / Reference'})

# Render the two mathematically distinct structures side-by-side
display_side_by_side_datasets(
    [df_data_disp, df_dissim_disp],
    [
        "جدول ۱: ماتریس داده دووجهی (n=4 , p=3)",
        "جدول ۲: ماتریس عدم تشابه یک‌وجهی متقارن (n=4 , n=4)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی فواصل و کشف بصری خوشه‌ها",
    "sub_title"
)

display_academic_text(
    " بوم گرافیکیِ زیر، دلیلِ شکل‌گیریِ ماتریس عدم تشابه را اثبات می‌کند. در نمودار پراکندگی (سمت چپ)، جایگاه واقعی کاربران در فضای ویژگی‌ها مشخص است؛"
    " نشان‌دهندهٔ تطابقِ مطلقِ، <b>رنگ سرمه‌ای تیره (فاصله صفر)</b> و در نقشه حرارتی (سمت راست)، توپولوژیِ فواصلِ استخراج‌شده به تصویر کشیده می‌شود. قطر اصلی با"
    "اثبات‌کنندهٔ شباهت بالای این دو گروه می‌باشد (User_A و User_B) هر کاربر با خودش است. بلوک‌های تیره میان.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(18, 6.5), dpi=135)

# --- SUBPLOT 1: Raw Data Scatter Representation ---
# Visualizing 'Age' vs 'Avg Spend' to show WHY the users cluster the way they do
sns.scatterplot(
    data=df_data_disp,
    x='Age',
    y='Avg_Spend_USD',
    ax=axes[0],
    color='#1abc9c', # Turquoise
    s=400,
    edgecolor='#2c3e50',
    linewidth=2.5,
    zorder=5
)

# Annotate each point with the Customer ID
for idx, row in df_data_disp.iterrows():
    axes[0].text(row['Age'], row['Avg_Spend_USD'] + 20, row['Customer_ID'],
                 ha='center', va='bottom', fontweight='bold', fontsize=12, color='#2c3e50')

# Draw circles around apparent clusters
from matplotlib.patches import Ellipse
axes[0].add_patch(Ellipse((25.5, 125), width=5, height=60, color='#2980b9', alpha=0.2, zorder=1))
axes[0].add_patch(Ellipse((47.5, 475), width=9, height=90, color='#e74c3c', alpha=0.2, zorder=1))

axes[0].set_title('Figure 1: Customer Position in Raw Feature Space', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Customer Age', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Average Monthly Spend (USD)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xlim(20, 55)
axes[0].set_ylim(50, 550)
axes[0].grid(True, linestyle='--', alpha=0.5, zorder=0)

# --- SUBPLOT 2: Dissimilarity Matrix Heatmap ---
# Using an Academic Blue reversed colormap (Blues_r).
# Dark Blue = 0 distance (High Similarity/Match). White/Light Blue = High Distance (Dissimilarity).
sns.heatmap(
    dissimilarity_matrix,
    annot=True,
    cmap='Blues_r',
    fmt='.1f',
    linewidths=2.0,
    linecolor='white',
    ax=axes[1],
    cbar_kws={'label': 'Calculated Euclidean Distance'},
    annot_kws={"size": 14, "weight": "bold"}
)
```

```python
axes[1].set_title('Figure 2: Topology of Customer Dissimilarity Matrix', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Target Customer (Object j)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Reference Customer (Object i)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=11)
axes[1].set_yticklabels(axes[1].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

# Clean up top and right spines
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ============================================================
# SECTION 4: STRATEGIC CONCLUSION
# ============================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br><br>"
    r"همان‌طور که در گراف‌ها مشاهده می‌کنید، اگرچه <b>ماتریس داده (جدول ۱)</b> است، حاوی اطلاعات مطلق و خام هر کاربر است. "
    r"اما الگوریتم‌های فاصله‌محور و خوشه‌بندی (نظیر <i>K-Means</i> یا <i>Hierarchical Clustering</i>) مستقیماً با <b>ماتریس عدم تشابه (جدول ۲)</b> کار می‌کنند. <br><br>"
    r"و همچنین تقارنِ فواصل حولِ آن، این اطمینان ریاضیاتی را به سیستمِ هوشمند می‌دهد که مقادیرِ صِفر در <b>قطر اصلی (Main Diagonal)</b> تحلیلِ گرافِ سمتِ راست نشان می‌دهد. "
    r"تراک بگذارند، (B و A نظیر فاصله ۱۰.۱ بین) است تا با شناسایی کوچک‌ترین فواصلِ ریاضی (Recommender Systems) یکسان است. این مکانیزم، زیربنای اصلیِ سیستم‌های توصیه‌گر A تا B با فاصلهٔ B، A تا A که فاصلهٔ کاربر"
)
display_academic_text(insight_conclusion, "insight")
```

خوشه‌بندی الگوهای رفتاری: گذر از ماتریس داده به ماتریس عدم تشابه (Dissimilarity Matrix) 🎯

بر اساس اصول داده‌کاوی، الگوریتم‌های خوشه‌بندی به جای کار مستقیم بر روی مقادیرِ خامِ ویژگی‌ها، نیازمند ارزیابی فواصل جفت‌به‌جفت (Pairwise Distances) عناصر هستند. در این سناریو، سامانهٔ هوش مصنوعیِ یک پلتفرم فروشگاهی، ابتدا رکوردهای تراکنشی را در یک **«ماتریس داده دووجهی (Two-mode Data Matrix)»** تجمیع نموده و سپس آن را با استفاده از فواصل اقلیدسی به یک **«ماتریس عدم تشابه متقارن (One-mode Dissimilarity Matrix)»** تبدیل می‌نماید تا میزان نزدیکی رفتار کاربران را اندازه‌گیری کند.

مرحله نخست: مقداردهی ماتریس شیء-ویژگی و محاسبه فواصل اقلیدسی

در این فاز، *ماتریس داده* به عنوان نمایشگر خام ویژگی‌های مستقل تشکیل می‌گردد. سپس الگوریتمِ فاصله اقلیدسی (Euclidean Distance) بر روی تمامی بردارهای کاربری اعمال شده و خروجی در قالب یک ماتریس متقارن (Symmetric Matrix) با قطر اصلی صفر بازتولید می‌شود. این دو ساختار در جداول موازی زیر قابل مقایسه می‌باشند.

جدول ۱: ماتریس داده دووجهی (n=4 , p=3)

Customer_ID	Age	Monthly_Purchases	Avg_Spend_USD
User_A	25	10	120
User_B	26	11	130
User_C	45	2	450
User_D	50	1	500

جدول ۲: ماتریس عدم تشابه یک‌وجهی متقارن (n=4 , n=4)

Customer_ID	Target / Reference	User_A	User_B	User_C	Us
	User_A	0.0	10.1	330.7	3
	User_B	10.1	0.0	320.7	3
	User_C	330.7	320.7	0.0	5
	User_D	380.9	370.9	50.3	

مرحله دوم: مصورسازی تحلیلی فواصل و کشف بصری خوشه‌ها

بوم گرافیکیِ زیر، دلیلِ شکل‌گیریِ ماتریس عدم تشابه را اثبات می‌کند. در نمودار پراکندگی (سمت چپ)، جایگاه واقعی کاربران در فضای ویژگی‌ها مشخص است؛ و در نقشه حرارتی (سمت راست)، توپولوژیِ فواصلِ استخراج‌شده به تصویر کشیده می‌شود. قطر اصلی با **رنگ سرمه‌ای تیره (فاصله صفر)**، نشان‌دهندهٔ تطابقِ مطلقِ هر کاربر با خودش است. بلوک‌های تیره میان (User_A و User_B) اثبات‌کنندهٔ شباهت بالای این دو گروه می‌باشد.

Figure 1: Customer Position in Raw Feature Space

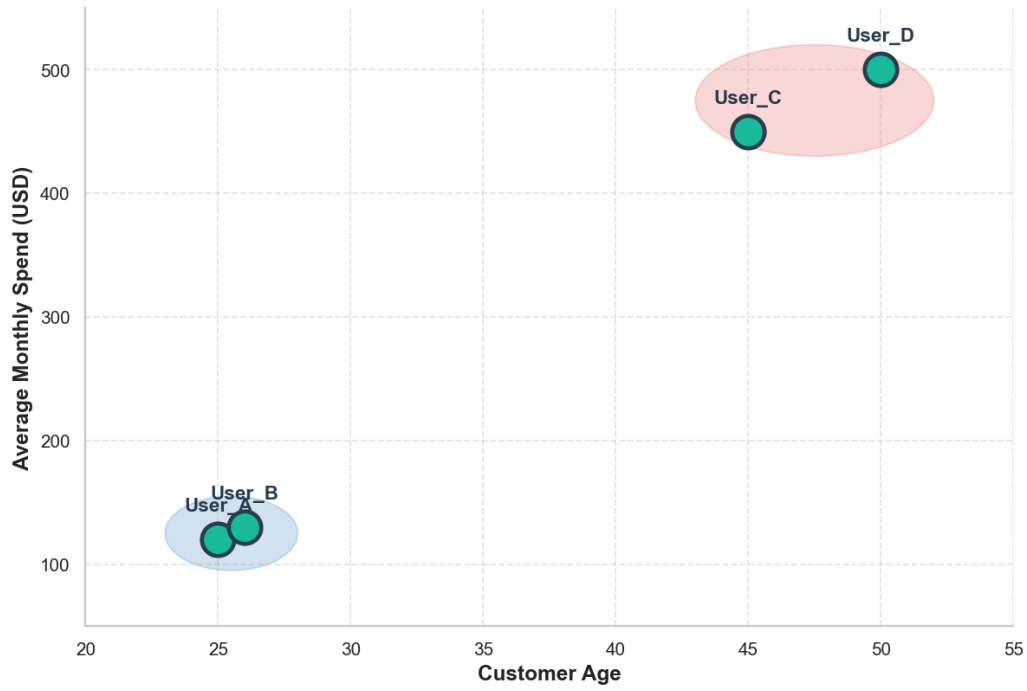
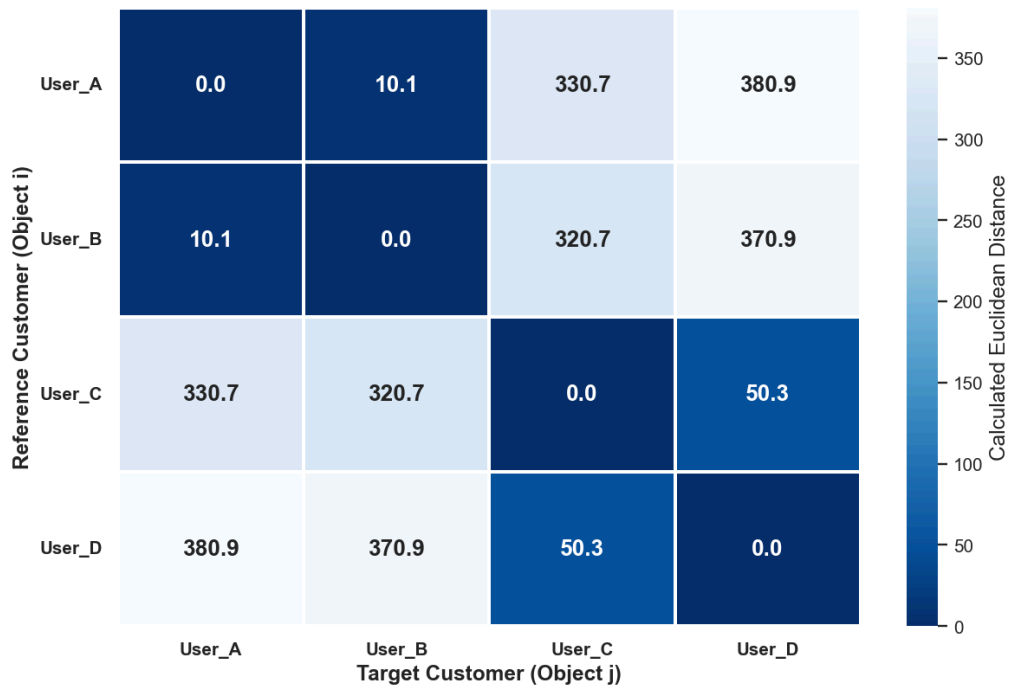


Figure 2: Topology of Customer Dissimilarity Matrix



استنتاج راهبردی عملیات (Strategic Implication):

همان‌طور که در گراف‌ها مشاهده می‌کنید، اگرچه ماتریس داده (جدول ۱) حاوی اطلاعات مطلق و خام هر کاربر است، اما الگوریتم‌های فاصله‌محور و خوشه‌بندی (نظیر *K-Means* یا *Hierarchical Clustering*) مستقیماً با ماتریس عدم تشابه (جدول ۲) کار می‌کنند.

تحلیلِ گرافِ سمتِ راست نشان می‌دهد که مقادیرِ صِفر در قطر اصلی (Main Diagonal) و همچنین تقارنِ فواصل حولِ آن، این اطمینان ریاضیاتی را به سیستمِ هوشمند می‌دهد که فاصلهٔ کاربر A تا B، با فاصلهٔ B تا A یکسان است. این مکانیزم، زیربنای اصلیِ سیستم‌های توصیه‌گر (Recommender Systems) است تا با شناسایی کوچک‌ترین فواصلِ ریاضی (نظیر فاصله ۱۰.۱ بین A و B)، محصولاتِ مشابه را میان این خوشه‌های همسایه به اشتراک بگذارند.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک **ماتریس داده (Data Matrix)** اولیه برای ۴ کاربر آمازون ساختیم که دارای ۳ ویژگی عددی (سن، تعداد خرید ماهانه و میانگین مبلغ خرید) است. این ماتریس دارای ابعاد 3 × 4 می‌باشد.
- از تابع **pdist()** از کتابخانه `scipy` با معیار فاصله اقلیدسی (Euclidean) استفاده کردیم تا فاصله تمام جفت‌کاربران ممکن را محاسبه کنیم.
- با استفاده از تابع **squareform()** آرایه فواصل را به یک **ماتریس عدم تشابه (Dissimilarity Matrix)** مربعی با ابعاد 4 × 4 تبدیل کردیم.
- همان‌طور که در خروجی مشخص است:
  - تمام عناصر روی **قطر اصلی صفر هستند**؛ زیرا فاصله هر کاربر از خودش صفر است ($d(A, A) = 0$).
  - ماتریس **متقارن** است؛ یعنی فاصله کاربر A از B برابر با فاصله کاربر B از A است.
  - عدد فاصله بین کاربر A و B بسیار کوچک است (حدود ۱۰.۰۵) که نشان‌دهنده تشابه رفتاری زیاد آن‌هاست؛ در حالی که فاصله بین A و D بسیار بزرگ است (حدود ۳۸۰.۸۸) که تفاوت الگوی خرید آن‌ها را اثبات می‌کند.

2.3.2. Proximity measures for nominal attributes

معیارهای نزدیکی برای ویژگی‌های اسمی 🏷️

برای ارزیابی میزان **شباهت** بین اشیایی که با **ویژگی‌های اسمی** توصیف شده‌اند، باید بررسی کرد که مقادیر آن‌ها که صرفاً به‌صورت «**نام**» یا «**برچسب**» هستند و فاقد **ترتیب** یا **مفهوم عددی**اند تا چه اندازه با یکدیگر یکسان‌اند. در این نوع محاسبات، برخلاف ویژگی‌های عددی، **معیارهای فاصله** کاربردی ندارند و شباهت میان نمونه‌ها بر اساس تعداد ویژگی‌هایی که مقادیر **یکسان** یا **متفاوت** دارند، سنجیده می‌شود.

این معیارها از مفاهیم بنیادی در داده‌کاوی محسوب شده و در روش‌هایی مانند **خوشه‌بندی**، **طبقه‌بندی**، **تحلیل شباهت** و **بازیابی اطلاعات** کاربرد گسترده‌ای دارند.

💡 **نکته کلیدی:** در ویژگی‌های اسمی، تنها **تطابق** یا **عدم تطابق** مقادیر اهمیت دارد، نه **میزان** یا **فاصله** بین آن‌ها. برای مثال، در یک مجموعه داده شامل ویژگی «رنگ» با مقادیر {سفید، آبی، سبز}، دو نمونه با رنگ «سفید» و «آبی» به اندازهٔ دو نمونه با رنگ «سفید» و «سبز» از هم متفاوت هستند.

📐 ریاضیات – نزدیکی برای ویژگی‌های اسمی

$$d(i,j) = \frac{p - m}{p} \tag{2.26}$$

نمادها

- $d(i,j)$: عدم‌تشابه.
- p: تعداد ویژگی‌ها.
- m: تعداد Matchها.
- i, j: اندیس دو شیء.

$$sim(i,j) = 1 - d(i,j) = \frac{m}{p} \tag{2.27}$$

نمادها

- $sim(i,j)$: شباهت.
- $d(i,j)$: عدم‌تشابه.
- m: Matchها.
- p: تعداد ویژگی‌ها.

مثال / تمرین حل‌شده

صورت سؤال: دو شیء در ۴ ویژگی اسمی مقایسه شده‌اند و در ۳ ویژگی مقدار یکسان دارند. شباهت و عدم‌تشابه را محاسبه کنید.

حل: $m = 3$ و $p = 4$ است. بنابراین $sim = m/p = 3/4 = 0.75$ و $d = (p - m)/p = 1/4 = 0.25$ می‌شود.

توضیح: مجموع شباهت و عدم‌تشابه در این معیار برابر ۱ است؛ هرچه تعداد تطابق‌ها بیشتر باشد، شباهت افزایش پیدا می‌کند.

🏷️ مثال عملی: سنجش شباهت سلیقه کاربران در «نتفلیکس» (Netflix)

سیستم هوشمند «نتفلیکس» می‌خواهد ببیند سلیقه فیلم دیدن دو کاربر چقدر به هم نزدیک است. متغیرهای مورد بررسی، **ژانرها و برچسب‌های متنی (Nominal)** مانند `Comedy`، `Sci-Fi`، `Action` و `Drama` هستند. چون این ویژگی‌ها هیچ ترتیب ریاضی یا مقدار عددی ندارند، نمی‌توان از فاصله‌های معمولی (مثل اقلیدسی) استفاده کرد.

گام اول: چرا روش‌های عددی معمولی در اینجا شکست می‌خورند؟

اگر به ژانر اکشن عدد ۱، کمدی عدد ۲ و درام عدد ۳ بدهیم، به مدل هوش مصنوعی این اشتباه را القا کرده‌ایم که «درام» سه برابر «اکشن» ارزش دارد! برای ویژگی‌های اسمی، شباهت بر اساس **اشتراک و عدم اشتراک برچسب‌ها** سنجیده می‌شود.

گام دوم: فرمول شباهت جاکارد (Jaccard Similarity)

شباهت جاکارد میزان همپوشانی دو مجموعه برچسب را حساب می‌کند:

$$\text{Jaccard Similarity}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{Intersection of Genres}}{\text{Union of All Genres}}$$

- **کاربر A:** {اکشن، علمی-تخیلی، هیجان‌انگیز}
- **کاربر B:** {اکشن، علمی-تخیلی، کمدی، درام}
- **اشتراک (ژانرهای مشترک):** {اکشن، علمی-تخیلی} 👈 ۲ ژانر
- **اجتماع (کل ژانرها):** {اکشن، علمی-تخیلی، هیجان‌انگیز، کمدی، درام} 👈 ۵ ژانر
- **شباهت جاکارد:** 2/5 = 0.40 (۴۰٪ شباهت سلیقه)

گام سوم: عدم تشابه جاکارد (Jaccard Distance)

برای تبدیل این شباهت به یک معیار فاصله (عدم تشابه)، کافی است آن را از عدد ۱ کم کنیم:

$$\text{Jaccard Distance} = 1 - 0.40 = 0.60$$

نتیجه: هرچه شباهت جاکارد به ۱ نزدیک‌تر باشد (مثلاً ۰.۸۵)، نتفلیکس فیلم‌های محبوب کاربر A را به کاربر B پیشنهاد می‌دهد.

خروجی: سنجش دقیق میزان همپوشانی ویژگی‌های اسمی بدون نیاز به فرضیات عددی کاذب

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.spatial.distance import pdist, squareform
from matplotlib.colors import ListedColormap

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

# Fix random seed for exact reproducibility
np.random.seed(42)
```

In [18]:
```
display_academic_text(
    "سنجش عدم تشابه در داده‌های اسمی: ضریب و فاصله جاکارد (Jaccard Index) 🎬 ",
    "main_title"
)

display_academic_text(
    "مقایسه ویژگی‌های اسمی و مجموعه‌ای (مانند ژانرهای فیلم) از طریق توابع فاصله متریک (نظیر اقلیدسی) امکان‌پذیر نیست، "
    "چرا که هیچ ترتیب یا ارزش عددی ذاتی میان آن‌ها وجود ندارد. در این سیستم توصیه‌گر، سلیقه کاربران از طریق <b>«ضریب شباهت جاکارد»</b> سنجیده می‌شود "
    "مجموعه‌ها (Union) به اندازهٔ اجتماع (Intersection) که برابر است با نسبت اندازهٔ اشتراک:",
    "paragraph"
)

# Strict LaTeX rendering for math formulas
display_academic_text(r"$$J(A, B)=\frac{|A \cap B|}{|A \cup B|}$$", "paragraph")

display_academic_text(
    "به منظور ورود این متغیرها به الگوریتم‌های فاصله‌محورِ هوش مصنوعی، ماتریس شباهت به <b>ماتریس فاصله (عدم تشابه) جاکارد</b> تبدیل می‌گردد:",
    "paragraph"
)

display_academic_text(r"$$d_J(A, B)=1-J(A, B)$$", "paragraph")

# ============================================================
# SECTION 2: DATA MATRIX CONSTRUCTION & JACCARD COMPUTATION
# ============================================================
display_academic_text(
    "مرحله نخست: تولید ماتریس باینری و استخراج فواصل جاکارد",
```

```
    "sub_title"
)

display_academic_text(
    "ماتریس نشانگر باینری<b> (Binary Indicator Matrix)</b> در این فاز، فهرست تماشای کاربران به یک (۱ به معنای مشاهده و ۰ به معنای عدم مشاهده) تبدیل می‌شود. "
    "ماتریس شباهت<i> ،سپس، الگوریتم به صورت خودکار فاصله جاکارد را برای تمامی جفت کاربران محاسبه نموده و در نهایت </i>به عنوان مکملِ ماتریس فاصله استخراج می‌گردد.",
    "paragraph"
)

# Step 1: Initialize the binary indicator matrix representing nominal tags (Watched Genres)
data = {
    'Action':   [1, 1, 0, 0],
    'Sci-Fi':   [1, 1, 0, 0],
    'Thriller': [1, 0, 1, 0],
    'Comedy':   [0, 1, 1, 1],
    'Drama':    [0, 1, 0, 1]
}
users = ['User_A', 'User_B', 'User_C', 'User_D']
df_nominal = pd.DataFrame(data, index=users)
df_nominal.index.name = 'User_ID'

# Step 2: Compute Pairwise Jaccard Distance (Dissimilarity)
# pdist calculates the Jaccard distance based on the proportion of mismatched items to the union of observed items
jaccard_dist_condensed = pdist(df_nominal.values, metric='jaccard')

# Convert the condensed array into a square symmetric dissimilarity matrix
df_jaccard_dist = pd.DataFrame(
    squareform(jaccard_dist_condensed),
    index=users,
    columns=users
)
df_jaccard_dist.index.name = 'User_ID'

# Step 3: Derive the Jaccard Similarity Matrix (1 - Distance)
df_jaccard_sim = 1 - df_jaccard_dist

# Prepare clean copies for displaying without hiding the index
df_nom_disp = df_nominal.reset_index()
df_dist_disp = df_jaccard_dist.reset_index()
df_sim_disp = df_jaccard_sim.reset_index()

# Render the three data structures side-by-side
display_side_by_side_datasets(
    [df_nom_disp, df_dist_disp, df_sim_disp],
    [
        "جدول ۱: ماتریس باینری ویژگی‌های اسمی (ژانرها)",
        "جدول ۲: ماتریس فاصله جاکارد (عدم تشابه)",
        "جدول ۳: ماتریس شباهت جاکارد (همپوشانی سلیقه)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی و کشف بصری شباهت‌ها (Visual Analytics)",
    "sub_title"
)

display_academic_text(
    "رفتار خام کاربران ثبت شده است. در <b>نمودار سمت چپ</b>، بوم گرافیکی زیر، منشأ استخراج شباهت‌ها را به تصویر می‌کشد. "
    "ماتریس شباهت جاکارد رسم شده است. قطر اصلی (با شباهت ۱.۰) نشان‌دهندهٔ تطابق ۱۰۰ درصدی هر کاربر با خودش است، در <b>نقشه حرارتی سمت راست</b> "
    "دارای بیشترین همپوشانیِ سلیقه می‌باشند User_B و User_A با بررسی طیف رنگی، مشخص می‌گردد که.",
    "paragraph"
)

# Initialize a wide canvas for side-by-side structural comparison
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- CHART 1: Original Binary Matrix ---
# Custom binary colormap: Light gray for 0, Navy Blue for 1
cmap_binary = ListedColormap(['#ecf0f1', '#2c3e50'])

sns.heatmap(
    df_nominal,
    annot=True,
    cmap=cmap_binary,
    fmt='d',
    ax=axes[0],
    linewidths=2.0,
    linecolor='white',
    cbar=False,
    annot_kws={"size": 16, "weight": "bold"}
)

axes[0].annotate(
    "Intersection Logic:\nUser A and User B both watched\nAction and Sci-Fi. Their intersection\nsize is 2. Their union size is 5.",
    xy=(1.5, 0.5), xytext=(2.5, 2.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Raw Binary Feature Space (User vs. Genre)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Nominal Features (Movie Genres)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Target Customer', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xticklabels(axes[0].get_xticklabels(), fontweight='bold', fontsize=11, rotation=0)
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

# --- CHART 2: Jaccard Similarity Matrix ---
# Using Academic Blues palette to highlight similarities
sns.heatmap(
    df_jaccard_sim,
```

```
    annot=True,
    cmap='Blues',
    fmt='.2f',
    vmin=0, vmax=1,
    ax=axes[1],
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'label': 'Jaccard Similarity Coefficient'},
    annot_kws={"size": 14, "weight": "bold"}
)

axes[1].annotate(
    "Highest Similarity Match:\n2 / 5 = 0.40 (40% Overlap)\nUser A and B should be clustered together.",
    xy=(1.5, 0.5), xytext=(2.5, 2.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Topology of User Jaccard Similarity Matrix', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Target User (Object j)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Reference User (Object i)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=11, rotation=0)
axes[1].set_yticklabels(axes[1].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی (Strategic Implication):</b><br>"
    r"فراهم می‌سازد (Information Retrieval) امکان سنجش دقیقِ شباهتِ سلیقه را در سیستم‌های بازیابی اطلاعات، <b>معیار جاکارد</b> تبدیل ویژگی‌های اسمی/متنی به بردارهای باینری و بهره‌گیری از. "
    r"به عنوان وجه اشتراکِ آن‌ها لحاظ نمی‌گردد (ویژگی‌های دودویی نامتقارن)، (صفرهای مشترک) <i>«عدم وجود یک ژانر در سلیقه دو کاربر»</i>، در این ساختار محاسباتی. "
    r"در برابر روش‌های فاصله‌یابی کلاسیک (نظیر اقلیدسی) می‌باشد <b>توصیه‌گر مشارکتی (Collaborative Filtering)</b> این رویکرد، هستهٔ مرکزی و نقطهٔ تمایزِ الگوریتم‌های."
)
display_academic_text(insight_conclusion, "insight")
```

سنجش عدم تشابه در داده‌های اسمی: ضریب و فاصله جاکارد (Jaccard Index) 🎬

مقایسه ویژگی‌های اسمی و مجموعه‌ای (مانند ژانرهای فیلم) از طریق توابع فاصله متریک (نظیر اقلیدسی) امکان‌پذیر نیست، چرا که هیچ ترتیب یا ارزش عددی ذاتی میان آن‌ها وجود ندارد. در این سیستم توصیه‌گر، سلیقه کاربران از طریق **«ضریب شباهت جاکارد»** سنجیده می‌شود که برابر است با نسبت اندازهٔ اشتراک (Intersection) به اندازهٔ اجتماع (Union) مجموعه‌ها:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

به منظور ورود این متغیرها به الگوریتم‌های فاصله‌محورِ هوش مصنوعی، ماتریس شباهت به **ماتریس فاصله (عدم تشابه) جاکارد** تبدیل می‌گردد:

$$d_J(A, B) = 1 - J(A, B)$$

مرحله نخست: تولید ماتریس باینری و استخراج فواصل جاکارد

در این فاز، فهرست تماشای کاربران به یک **ماتریس نشانگر باینری (Binary Indicator Matrix)** تبدیل می‌شود (۱ به معنای مشاهده و ۰ به معنای عدم مشاهده). سپس، الگوریتم به صورت خودکار فاصله جاکارد را برای تمامی جفت کاربران محاسبه نموده و در نهایت، *ماتریس شباهت* به عنوان مکملِ ماتریس فاصله استخراج می‌گردد.

جدول ۱: ماتریس باینری ویژگی‌های اسمی (ژانرها)

User_ID	Action	Sci-Fi	Thriller	Co
User_A	1	1	1	
User_B	1	1	0	
User_C	0	0	1	
User_D	0	0	0	

جدول ۲: ماتریس فاصله جاکارد (عدم تشابه)

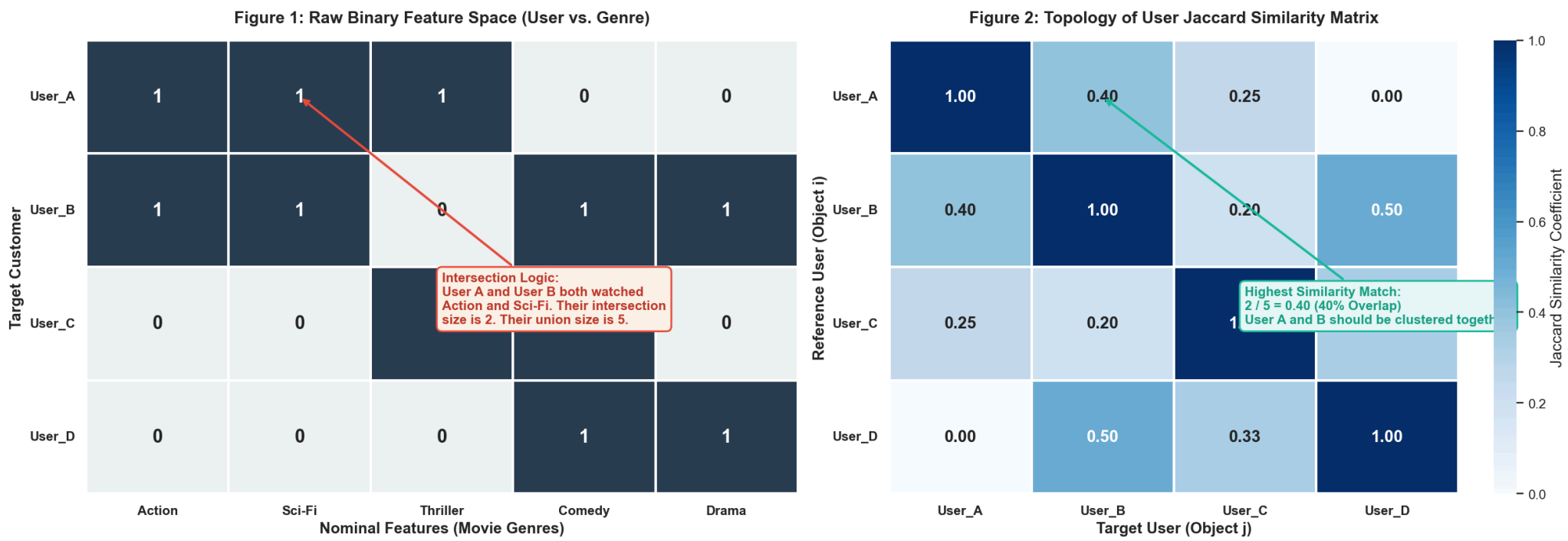
User_ID	User_A	User_B	User_C	U
User_A	0.0	0.6	0.8	
User_B	0.6	0.0	0.8	
User_C	0.8	0.8	0.0	
User_D	1.0	0.5	0.7	

جدول ۳: ماتریس شباهت جاکارد (همپوشانی سلیقه)

User_ID	User_A	User_B	User_C	U
User_A	1.0	0.4	0.2	
User_B	0.4	1.0	0.2	
User_C	0.2	0.2	1.0	
User_D	0.0	0.5	0.3	

مرحله دوم: مصورسازی تحلیلی و کشف بصری شباهت‌ها (Visual Analytics)

بوم گرافیکی زیر، منشأ استخراج شباهت‌ها را به تصویر می‌کشد. در **نمودار سمت چپ**، رفتار خام کاربران ثبت شده است. در **نقشه حرارتی سمت راست**، ماتریس شباهت جاکارد رسم شده است. قطر اصلی (با شباهت ۱.۰) نشان‌دهندهٔ تطابق ۱۰۰ درصدی هر کاربر با خودش است. با بررسی طیف رنگی، مشخص می‌گردد که User_A و User_B دارای بیشترین همپوشانیِ سلیقه می‌باشند.



استنتاج راهبردی (Strategic Implication):
تبدیل ویژگی‌های اسمی/متنی به بردارهای باینری و بهره‌گیری از معیار جاکارد، امکان سنجش دقیقِ شباهتِ سلیقه را در سیستم‌های بازیابی اطلاعات (Information Retrieval) فراهم می‌سازد. در این ساختار محاسباتی، *«عدم وجود یک ژانر در سلیقه دو کاربر»* (صفرهای مشترک)، به عنوان وجه اشتراکِ آن‌ها لحاظ نمی‌گردد (ویژگی‌های دودویی نامتقارن). این رویکرد، هستهٔ مرکزی و نقطهٔ تمایزِ الگوریتم‌های توصیه‌گر مشارکتی (Collaborative Filtering) در برابر روش‌های فاصله‌یابی کلاسیک (نظیر اقلیدسی) می‌باشد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما ۵ برچسب یا ژانر اسمی متعلق به نتفلیکس (**Action**, **Sci-Fi**, **Thriller**, **Comedy**, **Drama**) را برای ۴ کاربر به‌صورت یک ماتریس حضور/عدم حضور (صفر و یک) نمایش دادیم.
- از تابع **pdist()** با پارامتر `metric='jaccard'` استفاده کردیم تا **فاصله‌ی جاکارد** بین تمام جفت‌کاربران محاسبه شود.
- با کسر کردن ماتریس فاصله از عدد ۱ (`1 - jaccard_dist_matrix`)، **ماتریس شباهت جاکارد** به‌دست آمد.
- نکات برجسته خروجی کد:
  - شباهت هر کاربر با خودش روی قطر اصلی برابر با **۱.۰۰ (۱۰۰٪ شباهت)** است.
  - شباهت بین کاربر A و کاربر B برابر با **۰.۴۰ (۴۰٪)** شد؛ زیرا ۲ ژانر مشترک روی ۵ ژانر کل داشتند (2/5).
  - شباهت بین کاربر A و کاربر D برابر با **۰.۰۰ (۰٪)** شد؛ چون هیچ ژانر مشترکی نداشتند و مجموعه تماشا شده توسط آن‌ها کاملاً مجزا بود.

2.3.3. Proximity measures for binary attributes

معیارهای نزدیکی برای ویژگی‌های دودویی 1️⃣ 0️⃣

برای ارزیابی میزان **شباهت** بین اشیایی که با **ویژگی‌های دودویی** توصیف شده‌اند، باید بررسی کرد که مقادیر **صفر و یک** آن‌ها تا چه اندازه با یکدیگر مطابقت دارند. در این نوع محاسبات، نزدیکی یا شباهت میان نمونه‌ها بر اساس میزان **تطابق مقادیرشان** سنجیده می‌شود.

یکی از معیارهای رایج برای سنجش تفاوت میان دو نمونه با ویژگی‌های دودویی، **فاصلهٔ همینگ (Hamming Distance)** است. این معیار تعداد موقعیت‌هایی را می‌شمارد که مقادیر دو نمونه در آن‌ها با یکدیگر **متفاوت**اند؛ هرچه تعداد این تفاوت‌ها بیشتر باشد، فاصلهٔ همینگ بزرگ‌تر و در نتیجه شباهت دو نمونه کمتر خواهد بود.

ریاضیات – نزدیکی برای ویژگی‌های دودویی

$$p = q + r + s + t \tag{2.28}$$

نمادها

- q: تعداد 1,1.
- r: تعداد 1,0.
- s: تعداد 0,1.
- t: تعداد 0,0.
- p: تعداد کل ویژگی‌ها.

$$d(i, j) = \frac{r + s}{q + r + s + t} \tag{2.29}$$

نمادها

- $d(i, j)$: عدم‌تشابه متقارن.
- q, r, s, t: چهار شمارنده دودویی.

$$d(i, j) = \frac{r + s}{q + r + s} \tag{2.30}$$

نمادها

- $d(i, j)$: عدم‌تشابه نامتقارن.
- q: Match مثبت.
- r, s: اختلاف‌ها.

$$sim(i, j) = \frac{q}{q + r + s} = 1 - d(i, j) \tag{2.31}$$

نمادها

- $sim(i, j)$: Jaccard Similarity.
- q, r, s: شمارنده‌های دودویی.

مثال / تمرین حل‌شده

صورت سؤال: برای دو بردار دودویی، $q = 3$، $r = 1$، $s = 1$ و $t = 0$ است. شباهت Jaccard و عدم‌تشابه آن را محاسبه کنید.

حل: در این حالت $q + r + s = 3 + 1 + 1 = 5$. بنابراین $sim = q/(q + r + s) = 3/5 = 0.6$ و $d = (r + s)/(q + r + s) = 2/5 = 0.4$ است.

توضیح: در Jaccard، تطابق‌های 0, 0 نادیده گرفته می‌شوند؛ بنابراین t در صورت و مخرج این معیار وارد نمی‌شود.

فاصله همینگ — Hamming Distance

$$d_H(x, y) = \sum_{k=1}^{p} 1[x_k \neq y_k] \tag{2.32}$$

نمادها

- d_H: فاصله همینگ.
- x, y: دو بردار.
- x_k, y_k: مؤلفه kام.
- $1[\cdot]$: تابع شاخص.
- p: تعداد مؤلفه‌ها.

مثال / تمرین حل‌شده

صورت سؤال: فاصله همینگ دو بردار $(1, 0, 1, 0)$ و $(0, 1, 1, 0)$ را محاسبه کنید.

حل: مؤلفه اول متفاوت، مؤلفه دوم متفاوت، مؤلفه سوم یکسان و مؤلفه چهارم یکسان است. پس دو جایگاه اختلاف دارند و $d_H = 2$.

توضیح: فاصله همینگ فقط تعداد موقعیت‌های متفاوت را می‌شمارد و مقدار بزرگ‌تر اختلاف در یک مؤلفه را از نظر شدت اندازه‌گیری نمی‌کند.

مثال عملی: سیستم تشخیص هوشمند بیماری در «اپل هلت» (Apple Health)

سیستم هوش مصنوعی «اپل هلت» برای گروه‌بندی بیماران و تشخیص بیماری‌های مشابه، علائم بالینی افراد را بررسی می‌کند. هر علامت یک **ویژگی دودویی (Binary Attribute)** است: عدد **۱** به معنای وجود علامت (مثلاً داشتن تب) و عدد **۰** به معنای عدم وجود آن است.

گام اول: جدول توافقی ۲×۲ (Contingency Table)

برای مقایسه دو بیمار (مثلاً بیمار A و بیمار B) بر اساس تمام علائم پزشکی، ۴ حالت ممکن وجود دارد:

- **q:** تعداد علائمی که در **هر دو بیمار وجود دارد (۱ و ۱)**.
- **r:** علائمی که فقط در بیمار A وجود دارد (۱ و ۰).
- **s:** علائمی که فقط در بیمار B وجود دارد (۰ و ۱).
- **t:** تعداد علائمی که در **هیچ‌کدام وجود ندارد (۰ و ۰)**.

گام دوم: انتخاب معیار مناسب (متقارن در برابر نامتقارن)

بسته به نوع داده‌های دودویی، از دو فرمول مختلف برای محاسبه شباهت استفاده می‌شود:

۱. ضریب تطابق ساده (Symmetric Matching Coefficient - SMC):

برای متغیرهای **دودویی متقارن** (مثل جنسیت) استفاده می‌شود که در آن ۰ و ۱ ارزش یکسانی دارند (t هم‌ارزش با q است):

$$SMC = \frac{q + t}{q + r + s + t}$$

۲. ضریب جاکارد دودویی (Asymmetric Jaccard Coefficient):

برای متغیرهای **دودویی نامتقارن** (مثل علائم بیماری) استفاده می‌شود. در پزشکی، نداشتن یک بیماری (حالت ۰-۰ یا t) نباید باعث شود دو فرد سالم شبیه به هم به نظر برسند! بنابراین t از فرمول حذف می‌شود:

$$\text{Jaccard Similarity} = \frac{q}{q + r + s}$$

گام سوم: محاسبه عدم تشابه (Distance)

برای به دست آوردن فاصله (عدم تشابه) دودویی نامتقارن، کافی است ضریب جاکارد را از ۱ کم کنیم:

$$\text{Jaccard Distance} = \frac{r + s}{q + r + s} = 1 - \text{Jaccard Similarity}$$

نتیجه: اپل هلت با حذف حالت‌های ۰-۰ (سلامت کامل در یک علامت)، فقط بر علائم واقعی فعال تمرکز می‌کند تا بیماران مبتلا به ابتلاهای مشابه را دقیق‌تر شناسایی کند.

خروجی: جداسازی فرمول‌های دودویی متقارن و نامتقارن برای جلوگیری از شباهت‌های کاذب

In [19]:

```
display_academic_text(
    "پیش‌پردازش داده‌ها: تمایز شباهت‌های متقارن (SMC) و نامتقارن (Jaccard) در داده‌های دودویی بالینی 🩺 ",
    "main_title"
)

display_academic_text(
    "بستگی دارد. در تحلیل علائم بالینی <i>ماهیت ویژگی‌ها</i> انتخاب معیار تشابه برای متغیرهای دودویی مستقیماً به (Clinical Symptoms)، "
    "لزوماً بیانگرِ تشابهِ فیزیولوژیکِ دو فرد نیست (همهٔ افرادِ سالم شبیه به هم نیستند!) <b>نبودِ یک نشانه (حالت ۰-۰)</b>. "
    "جهت جلوگیری از ارزیابی شباهت‌های کاذب مورد بررسی قرار می‌گیرد <b>ضریب جاکارد (Asymmetric Jaccard)</b> در برابر <b>ضریب تطابق ساده (Symmetric SMC)</b> در این سناریو، عملکرد",
    "paragraph"
)

# Use raw strings (r"") for correct LaTeX rendering
display_academic_text(
    r"فرمول Symmetric Matching Coefficient (SMC) - (شامل تطابق‌های ۰-۰ یا $t$):",
    "sub_title"
)
display_academic_text(
    r"$$SMC = \frac{q + t}{q + r + s + t}$$",
    "paragraph"
)

display_academic_text(
    r"فرمول Asymmetric Jaccard Coefficient - (حذف‌کنندهٔ تطابق‌های ۰-۰ یا $t$):",
    "sub_title"
)
display_academic_text(
    r"$$Jaccard = \frac{q}{q + r + s}$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA MATRIX CONSTRUCTION AND COEFFICIENT COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های خام و محاسبه ماتریس‌های شباهت",
    "sub_title"
)

display_academic_text(
    "در این فاز، ماتریس باینری علائم بیماران (۱ به معنای وجود و ۰ به معنای عدم وجود علامت) تشکیل می‌گردد. "
    "و جاکارد را برای تمامی جفتِ بیماران محاسبه نموده و نتایج جهت مقایسهٔ آنی در کنار یکدیگر رندر می‌شوند SMC سپس، الگوریتم به صورت خودکار هر دو ضریب",
    "paragraph"
)

# Step 1: Initialize the binary indicator matrix representing clinical symptoms.
# 1 indicates symptom presence (important/rare state), 0 indicates absence.
data = {
    'Fever':     [1, 1, 0, 0],
    'Cough':     [1, 1, 0, 0],
    'Headache':  [1, 0, 1, 0],
    'Fatigue':   [0, 1, 1, 0],
    'Nausea':    [0, 0, 0, 1]
}
patients = ['Patient_A', 'Patient_B', 'Patient_C', 'Patient_D']
df_binary = pd.DataFrame(data, index=patients)
df_binary.index.name = 'Patient_ID'

# Step 2: Compute Pairwise Symmetric Binary Similarity (SMC)
# The 'matching' metric in scipy computes the proportion of mismatches. Similarity is derived as 1 - distance.
smc_dist_condensed = pdist(df_binary.values, metric='matching')
```

```python
df_smc_sim = pd.DataFrame(
    1 - squareform(smc_dist_condensed),
    index=patients,
    columns=patients
)

# Step 3: Compute Pairwise Asymmetric Binary Similarity (Jaccard)
# The 'jaccard' metric in scipy actively ignores 0-0 matches. Similarity is derived as 1 - distance.
jaccard_dist_condensed = pdist(df_binary.values, metric='jaccard')
df_jaccard_sim = pd.DataFrame(
    1 - squareform(jaccard_dist_condensed),
    index=patients,
    columns=patients
)

# Render the three data structures side-by-side using the list format expected by the global function
df_bin_disp = df_binary.reset_index()
df_smc_disp = df_smc_sim.reset_index().rename(columns={'index': 'Target / Reference'})
df_jac_disp = df_jaccard_sim.reset_index().rename(columns={'index': 'Target / Reference'})

display_side_by_side_datasets(
    [df_bin_disp, df_smc_disp, df_jac_disp],
    [
        "جدول ۱: ماتریس باینری علائم بالینی بیماران",
        "جدول ۲: ماتریس SMC (دودویی متقارن)",
        "جدول ۳: ماتریس جاکارد (دودویی نامتقارن)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی و کشف خوشه‌های شباهت",
    "sub_title"
)

display_academic_text(
    "بوم گرافیکیِ زیر، ساختار تشابه بیماران را به تصویر می‌کشد. نقشه حرارتی SMC (سمت چپ) مقادیر شباهتِ بسیار بالایی را نمایش می‌دهد که (سمت چپ) (۰-۰)، به دلیل گنجاندن حالت‌های سلامت مشترک <b>کاملاً گمراه‌کننده</b> است.
    "در مقابل، نقشه حرارتی جاکارد (سمت راست) با تمرکزِ قطعی بر علائم فعال (۱-۱)، تفکیک واقع‌بینانه‌تری از خوشه‌های بیماران ارائه می‌دهد.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- SUBPLOT 1: Symmetric SMC Heatmap ---
sns.heatmap(
    df_smc_sim,
    annot=True,
    cmap='Blues',
    fmt='.2f',
    vmin=0, vmax=1,
    ax=axes[0],
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'label': 'SMC Similarity (Includes 0-0)'},
    annot_kws={"size": 15, "weight": "bold"}
)

axes[0].annotate(
    "The 0-0 Illusion:\nPatients C and D share NO identical symptoms.\nYet, because they both 'do not have' Fever and Cough,\nthe SMC falsely gives them a 40% similarity score!",
    xy=(3.5, 2.5), xytext=(3.5, 1.2),
    fontsize=11, fontweight='bold', color='#c0392b', ha='center',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Symmetric SMC Heatmap (Includes 0-0 Matches)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Target Patient', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Reference Patient', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xticklabels(axes[0].get_xticklabels(), fontweight='bold', fontsize=11, rotation=0)
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

# --- SUBPLOT 2: Asymmetric Jaccard Heatmap ---
sns.heatmap(
    df_jaccard_sim,
    annot=True,
    cmap='YlGnBu',
    fmt='.2f',
    vmin=0, vmax=1,
    ax=axes[1],
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'label': 'Jaccard Similarity (Ignores 0-0)'},
    annot_kws={"size": 15, "weight": "bold"}
)

axes[1].annotate(
    "Realistic Filtering:\nJaccard ignores the shared 'healthy' traits (0-0).\nSince C and D have completely different illnesses,\ntheir true clinical similarity correctly falls to 0.00.",
    xy=(3.5, 2.5), xytext=(3.5, 1.2),
    fontsize=11, fontweight='bold', color='#16a085', ha='center',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Asymmetric Jaccard Heatmap (Ignores 0-0 Matches)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Target Patient', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Reference Patient', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=11, rotation=0)
axes[1].set_yticklabels(axes[1].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

plt.tight_layout()
```

```
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"ما با دو بیماریِ کاملاً متفاوت روبه‌رو هستیم.<br><br>"
    r"D</b> بیمارِ<b> (دارای حالت تهوع) و <b> C</b> بیمارِ<b> (دارای سردرد و خستگی) این کد به زیبایی تفاوت‌های بنیادینِ محاسباتی را ثابت می‌کند. در"
    r"ین زد (شکل ۱). این یعنی تجویزِ اشتباهِ دارو <b>۰.۴۰ (۴۰٪)</b> فریب خورد و شباهتِ آن‌ها را به صورتِ کاذب <code>SMC</code> با این حال، به دلیل اینکه هیچ‌کدام از آن‌ها تب و سرفه نداشتند (حالت ۰-۰)، الگوریتم"
    r"ی می‌گردد <b>۰.۰۰</b> قرار داد و شباهت آن‌ها را به درستی <i>صرفاً بر روی پیوندهای ناشی از علائمِ فعال (۱-۱)</i> با حذف هوشمندانهٔ حالت‌های مشترکِ ۰-۰، تمرکز محاسباتی را ،<b>ضریب جاکارد</b> در مقابل، اتکا به"
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: تمایز شباهت‌های متقارن (SMC) و نامتقارن (Jaccard) در داده‌های دودویی بالینی ⚕️

انتخاب معیار تشابه برای متغیرهای دودویی مستقیماً به *ماهیت ویژگی‌ها* بستگی دارد. در تحلیل علائم بالینی (Clinical Symptoms)، **نبودِ یک نشانه (حالت ۰-۰)** لزوماً بیانگرِ تشابهِ فیزیولوژیکِ دو فرد نیست (همهٔ افرادِ سالم شبیه به هم نیستند!). در این سناریو، عملکرد **ضریب تطابق ساده (Symmetric SMC)** در برابر **ضریب جاکارد (Asymmetric Jaccard)** جهت جلوگیری از ارزیابی شباهت‌های کاذب مورد بررسی قرار می‌گیرد.

فرمول Symmetric Matching Coefficient (SMC) - (شامل تطابق‌های ۰-۰ یا t):

$$SMC = \frac{q + t}{q + r + s + t}$$

فرمول Asymmetric Jaccard Coefficient - (حذف‌کنندهٔ تطابق‌های ۰-۰ یا t):

$$Jaccard = \frac{q}{q + r + s}$$

مرحله نخست: بارگذاری داده‌های خام و محاسبه ماتریس‌های شباهت

در این فاز، ماتریس باینری علائم بیماران (۱ به معنای وجود و ۰ به معنای عدم وجود علامت) تشکیل می‌گردد. سپس، الگوریتم به صورت خودکار هر دو ضریب SMC و جاکارد را برای تمامی جفتِ بیماران محاسبه نموده و نتایج جهت مقایسهٔ آنی در کنار یکدیگر رندر می‌شوند.

جدول ۱: ماتریس باینری علائم بالینی بیماران

Patient_ID	Fever	Cough	Headache
Patient_A	1	1	1
Patient_B	1	1	0
Patient_C	0	0	1
Patient_D	0	0	0

جدول ۲: ماتریس SMC (دودویی متقارن)

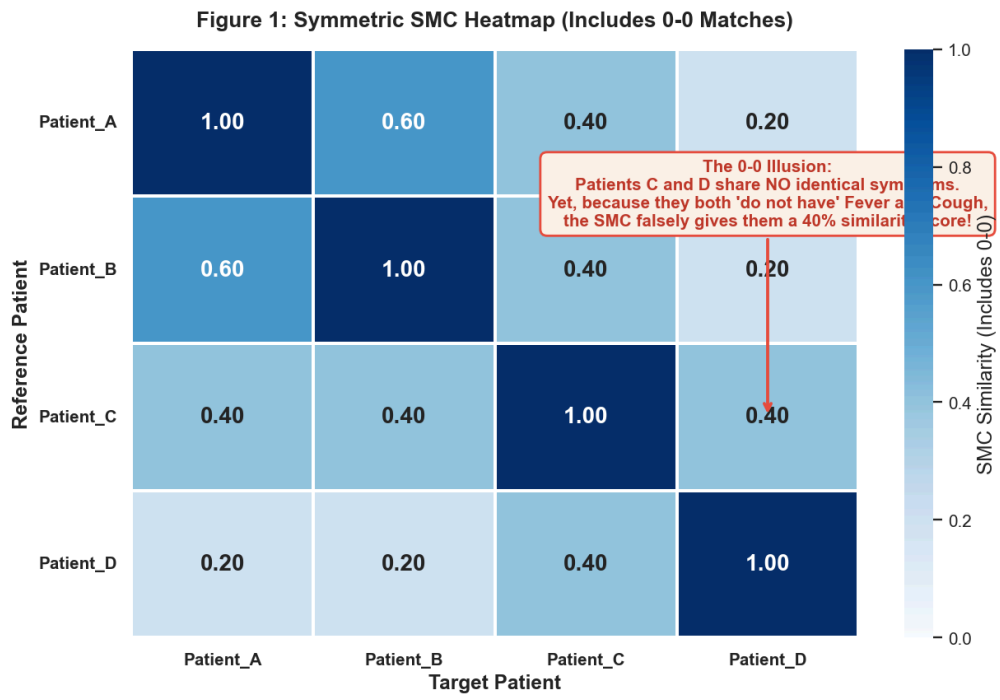
Target / Reference	Patient_A	Patient_B
Patient_A	1.0	0.6
Patient_B	0.6	1.0
Patient_C	0.4	0.4
Patient_D	0.2	0.2

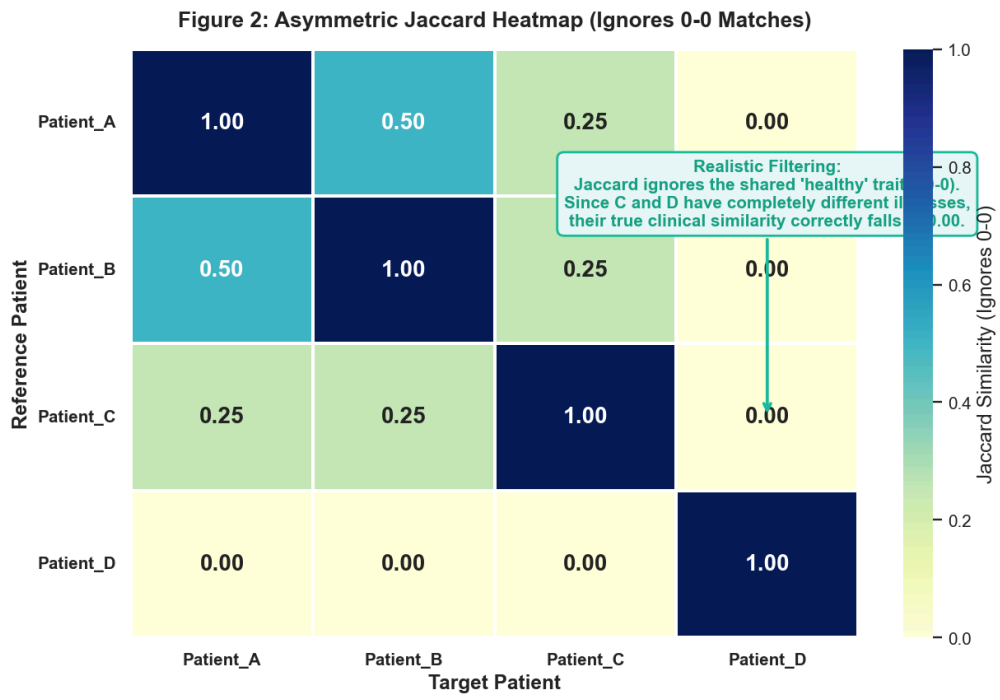
جدول ۳: ماتریس جاکارد (دودویی نامتقارن)

Target / Reference	Patient_A	Patient_B
Patient_A	1.0	0.5
Patient_B	0.5	1.0
Patient_C	0.2	0.2
Patient_D	0.0	0.0

مرحله دوم: مصورسازی تحلیلی و کشف خوشه‌های شباهت

بوم گرافیکیِ زیر، ساختار تشابه بیماران را به تصویر می‌کشد. نقشه حرارتی SMC (سمت چپ) به دلیل گنجاندن حالت‌های سلامت مشترک (۰-۰)، مقادیر شباهتِ بسیار بالایی را نمایش می‌دهد که **کاملاً گمراه‌کننده** است. در مقابل، نقشه حرارتی جاکارد (سمت راست) با تمرکزِ قطعی بر علائم فعال (۱-۱)، تفکیک واقع‌بینانه‌تری از خوشه‌های بیماران ارائه می‌دهد.





استنتاج راهبردی عملیات (Strategic Implication):

این کد به زیبایی تفاوت‌های بنیادینِ محاسباتی را ثابت می‌کند. در بیمارِ **C** (دارای سردرد و خستگی) و بیمارِ **D** (دارای حالت تهوع)، ما با دو بیماریِ کاملاً متفاوت روبه‌رو هستیم.

با این حال، به دلیل اینکه هیچ‌کدام از آن‌ها تب و سرفه نداشتند (حالت ۰-۰)، الگوریتمِ **SMC** فریب خورد و شباهتِ آن‌ها را به صورتِ کاذب ۰.۴۰ (۴۰٪) تخمین زد (شکل ۱). این یعنی تجویزِ اشتباهِ دارو!

در مقابل، اتکا به **ضریب جاکارد**، با حذف هوشمندانهٔ حالت‌های مشترکِ ۰-۰، تمرکز محاسباتی را *صرفاً بر روی پیوندهای ناشی از علائمِ فعال (۱-۱)* قرار داد و شباهت آن‌ها را به درستی ۰.۰۰ محاسبه کرد (شکل ۲). این امر منجر به شناسایی دقیق‌تر خوشه‌های بیماران و جلوگیری از خطای سیستم‌های بالینی می‌گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از ۴ بیمار در اپل هلت ساختیم که ۵ علامت بالینی دودویی (**Fever**, **Cough**, **Headache**, **Fatigue**, **Nausea**) با مقادیر ۰ و ۱ برای آن‌ها ثبت شده است.
- **محاسبه متقارن (SMC):** با پارامتر `metric='matching'` ضریب تطابق ساده را حساب کردیم که در آن حالت‌های نداشتن علامت (۰-۰) هم باعث افزایش شباهت می‌شوند.
- **محاسبه نامتقارن (Jaccard):** با پارامتر `metric='jaccard'` شباهت جاکارد را محاسبه کردیم که متغیر t (عدم وجود علامت در هر دو بیمار) را کلاً از مخرج کسر حذف می‌کند.
- تفاوت دو ماتریس در خروجی:
  - شباهت بین بیمار C و D در روش متقارن (SMC) برابر با **۰.۶۰ (۶۰٪)** شد، زیرا هر دو در ۳ علامت مقدار ۰ داشتند.
  - اما همین شباهت در روش نامتقارن (Jaccard) برابر با **۰.۰۰ (۰٪)** شد؛ زیرا بیمار D هیچ‌کدام از علائم فعال بیمار C را نداشت و اپل هلت این دو بیمار را کاملاً نامرتبط تشخیص داد.

2.3.4. Dissimilarity of numeric data: Minkowski distance

فاصله مینکوفسکی 📏

فاصلهٔ مینکوفسکی (Minkowski Distance) معیاری جامع و انعطاف‌پذیر برای اندازه‌گیری **فاصله** بین داده‌های عددی است. این معیار دارای یک پارامتر تنظیم‌کننده به نام **q** است که با تغییر آن، می‌توان به معیارهای فاصلهٔ متفاوتی دست یافت. در واقع، مینکوفسکی تعمیم‌یافتهٔ چندین معیار فاصلهٔ شناخته‌شده محسوب می‌شود.

فاصله مینکوفسکی به‌عنوان یک **چارچوب کلی** برای اندازه‌گیری عدم تشابه در بسیاری از الگوریتم‌های داده‌کاوی و یادگیری ماشین مورد استفاده قرار می‌گیرد.

چرا به فاصلهٔ مینکوفسکی نیاز داریم؟

در داده‌کاوی، همیشه یک نوع فاصله برای همهٔ مسائل جواب نمی‌دهد. بعضی مسائل به **فاصلهٔ مستقیم** حساس‌اند، بعضی دیگر به **اختلاف در تک‌تک ابعاد**اهمیت می‌دهند. اگر برای هر کدام از این حالت‌ها بخواهیم یک فرمول جداگانه بنویسیم و پیاده‌سازی کنیم، کارمان زمان بر و پیچیده می‌شود

فاصلهٔ مینکوفسکی این مشکل را حل کرده است. این معیار مثل یک **تنظیم‌کنندهٔ هوشمند** عمل می‌کند بدون اینکه فرمول اصلی را دست‌کاری کنیم. با یک پیاده‌سازی واحد، می‌توانیم بسته به نیاز مسئله، به **سه معیار فاصلهٔ کاملاً متفاوت** دست پیدا کنیم:

📐 ریاضیات — فاصله‌های عددی و Minkowski

$$d(i,j) = \sqrt{\sum_{f=1}^{p} (x_{if} - x_{jf})^2} \tag{2.33}$$

نمادها

- $d(i,j)$: فاصله اقلیدسی.
- x_{if}, x_{jf}: مقادیر ویژگی f.
- p: تعداد ویژگی‌ها.
- f: اندیس ویژگی.

$$d(i,j) = \sum_{f=1}^{p} |x_{if} - x_{jf}| \tag{2.34}$$

نمادها

- $d(i,j)$: فاصله منهتن.
- $|\cdot|$: قدرمطلق.
- x_{if}, x_{jf}, p, f: به‌ترتیب مقادیر، تعداد ویژگی‌ها و اندیس.

$$d(i,j) = \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{1/h}, \qquad h \geq 1 \tag{2.35}$$

نمادها

- $d(i,j)$: فاصله Minkowski.
- h: مرتبه فاصله.
- x_{if}, x_{jf}: مقادیر متناظر.

- p, f: تعداد و اندیس ویژگی.

با $h = 1$ فاصله Manhattan و با $h = 2$ فاصله Euclidean به دست می‌آید.

$$d(i, j) = \max_{f} |x_{if} - x_{jf}| \tag{2.36}$$

نمادها

- $\max_f$: بیشترین اختلاف بین ویژگی‌ها.
- x_{if}, x_{jf}: مقادیر متناظر.

$$d(i, j) = \sqrt[p]{\sum_{f=1} w_f (x_{if} - x_{jf})^2} \tag{2.37}$$

نمادها

- w_f: وزن ویژگی f.
- x_{if}, x_{jf}: مقادیر.
- p, f: تعداد و اندیس ویژگی.

مثال / تمرین حل‌شده

صورت سؤال: فاصله اقلیدسی و منهتن بین دو نقطه $(1, 2)$ و $(3, 5)$ را به‌دست آورید.

حل: اختلاف مؤلفه‌ها 2 و 3 است. فاصله اقلیدسی برابر $\sqrt{2^2 + 3^2} = \sqrt{13} \approx 3.61$ و فاصله منهتن برابر $|2| + |3| = 5$ است.

توضیح: اقلیدسی طول خط مستقیم بین دو نقطه را می‌سنجد، درحالی‌که منهتن مجموع حرکت در امتداد ابعاد را در نظر می‌گیرد.

مثال عملی: محاسبه فاصله و مسیریابی سفرهای «اوبر» (Uber) 📏

الگوریتم‌های مسیریابی «اوبر» (Uber) برای تخمین فاصله بین مسافر و راننده، از مختصات جغرافیایی (طول و عرض جغرافیایی) استفاده می‌کنند. بسته به نوع وسیله نقلیه (تاکسی، پهپاد یا پیک پیاده)، سیستم از حالت‌های مختلف **فاصله مینکوفسکی (Minkowski Distance)** بهره می‌برد.

گام اول: فرمول کلی فاصله مینکوفسکی

فاصله مینکوفسکی یک فرمول جامع برای محاسبه عدم تشابه و فاصله در فضاهای چندبعدی عددی است:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

در این فرمول، پارامتر p نوع فاصله را تعیین می‌کند:

۱. حالت $p = 1$ — فاصله منهتن (Manhattan Distance / City Block):

حرکت در خیابان‌های شطرنجی شهر (مثل نیویورک) که نمی‌توان از روی ساختمان‌ها پرید! خودرو مجبور است فقط در جهت افقی و عمودی

حرکت کند:

$$d_{Manhattan}(x, y) = |x_1 - x_2| + |y_1 - y_2|$$

۲. حالت $p = 2$ — فاصله اقلیدسی (Euclidean Distance):

مستقیم‌ترین مسیر بین دو نقطه ("As the crow flies" یا مانند پرواز کلاغ). این فاصله برای پهپادهای تحویل کالا (Uber Eats Drones) یا خطوط هوایی استفاده می‌شود:

$$d_{Euclidean}(x, y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

گام دوم: مثال عددی ساده

فرض کنید موقعیت راننده اوبر $(1, 2)$ و موقعیت مسافر $(4, 6)$ باشد:

- **فاصله منهتن ($p = 1$):** $|1 - 4| + |2 - 6| = 3 + 4 = 7$ کیلومتر.
- **فاصله اقلیدسی ($p = 2$):** $\sqrt{(1 - 4)^2 + (2 - 6)^2} = \sqrt{9 + 16} = 5$ کیلومتر.

نتیجه: فاصله منهتن همیشه بزرگ‌تر یا مساوی فاصله اقلیدسی است. اوبر برای خودروها از $p = 1$ (مسیر خیابانی) و برای تخمین پهپادی از $p = 2$ استفاده می‌کند.

خروجی: یکپارچه‌سازی معیارهای فاصله در یک فرمول ریاضی با تنظیم پارامتر p

In [20]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.spatial.distance import minkowski

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [21]:
```
display_academic_text(
    "معیارهای فاصله‌یابی: یکپارچه‌سازی فضاهای برداری با متریک مینکوفسکی (Minkowski Metric) 📐",
    "main_title"
)

display_academic_text(
    "یک فرمول تعمیم‌یافته است که <b>متریک مینکوفسکی</b> محاسبه فاصله میان موجودیت‌های دارای ویژگی‌های عددی، نیازمند توابع ریاضیاتی دقیق است. "
    "نشان داده می‌شود، می‌تواند هندسه‌های فضایی مختلفی را تولید نماید $p$ که با حرف (Order) با تغییر پارامتر درجه. "
    "در یک سامانه مسیریابی لجستیک، این فرمول برای محاسبه مسیرهای زمینیِ خودروها (محدود به محورهای مختصات یا بلوک‌های شهری) و مسیرهای هواییِ پهپادها (خطوط مستقیم) "
    "مورد بهره‌برداری قرار می‌گیرد.",
    "paragraph"
)

display_academic_text(
    "فرمول عمومی مینکوفسکی برای نقاط فضای $n$-بعدی:",
    "sub_title"
)
display_academic_text(
    r"$$d(x,y) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$",
    "paragraph"
)

display_academic_text(
    "حالت $p=1$ (فاصله منهتن - مسیریابی شبکه شهری):",
    "sub_title"
)
display_academic_text(
    r"$$d_{Manhattan}(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$",
    "paragraph"
)

display_academic_text(
    "حالت $p=2$ (فاصله اقلیدسی - مسیر پروازی مستقیم):",
    "sub_title"
)
display_academic_text(
    r"$$d_{Euclidean}(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$",
    "paragraph"
```

```
)

# ==========================================================
# SECTION 2: DATA MATRIX CONSTRUCTION AND DISTANCE COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری مختصات جغرافیایی و استخراج فواصل مینکوفسکی",
    "sub_title"
)

display_academic_text(
    "در قالب یک ماتریس عددی تشکیل می‌گردد (GPS) در این فاز، مختصات استخراج‌شده از سامانه مکان‌یابی. "
    "سپس الگوریتم فاصله‌یابی با درجات مختلف ($p=1, 2, 3$) بر روی بردارهای راننده و مسافران اعمال شده "
    "و مقادیر خروجی جهت قیاس تحلیلی در یک جدول تجمیعی رندر می‌شوند.",
    "paragraph"
)

# Step 1: Initialize the coordinate dataset for Driver and Passengers (X, Y in Kilometers)
locations = {
    'Entity_ID': ['Driver_Uber', 'Passenger_1', 'Passenger_2'],
    'X_Coord_KM': [1.0, 4.0, 7.0],
    'Y_Coord_KM': [2.0, 6.0, 10.0],
}
df_coords = pd.DataFrame(locations).set_index('Entity_ID')

# Step 2: Extract numerical vectors for mathematical computation
vec_driver = df_coords.loc['Driver_Uber'].values
vec_pass1 = df_coords.loc['Passenger_1'].values
vec_pass2 = df_coords.loc['Passenger_2'].values

# Step 3: Compute Minkowski distances across different values of parameter 'p'
results_data = {
    'Distance Metric Configuration': [
        'Manhattan (p=1) - Grid Route',
        'Euclidean (p=2) - Direct Flight',
        'Minkowski (p=3) - Higher Order'
    ],
    'Distance to Passenger 1': [
        minkowski(vec_driver, vec_pass1, p=1),
        minkowski(vec_driver, vec_pass1, p=2),
        minkowski(vec_driver, vec_pass1, p=3)
    ],
    'Distance to Passenger 2': [
        minkowski(vec_driver, vec_pass2, p=1),
        minkowski(vec_driver, vec_pass2, p=2),
        minkowski(vec_driver, vec_pass2, p=3)
    ]
}
df_results = pd.DataFrame(results_data)

# Render the coordinates and the computed distances side-by-side
df_coords_disp = df_coords.reset_index()
display_side_by_side_datasets(
    [df_coords_disp, df_results],
    [
        "جدول ۱: مختصات جغرافیایی مستقر (برحسب کیلومتر)",
        "جدول ۲: مقایسه فواصل محاسباتی مینکوفسکی"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی هندسی مسیرهای منهتن و اقلیدسی در فضای دوبعدی",
    "sub_title"
)

display_academic_text(
    "بوم گرافیکی زیر، ساختار فیزیکی فواصل محاسبه‌شده را در فضای دکارتی به تصویر می‌کشد. "
    "فاصله اقلیدسی (مستقیم‌ترین مسیر پروازی پهپاد) را نشان می‌دهد، <b>فیروزه‌ای رنگ</b> خط ممتد. "
    "مسیر جایگزین منهتن را بازنمایی می‌کنند که یک خودرو ملزم به طی کردن آن در امتداد بلوک‌های شهری (تقاطع‌های 90 درجه) است، <b>قرمز رنگ</b> خطوط چین‌دار. "
    "<i>فاصله منهتن همواره طولانی‌تر یا مساوی با مسیر اقلیدسی است، همان‌طور که از لحاظ بصری و ریاضیاتی مشخص است.</i>",
    "paragraph"
)

# Initialize a plot canvas optimized for Cartesian geometry
plt.figure(figsize=(14, 7), dpi=135)

# Extract coordinate values explicitly for plotting
dx, dy = df_coords.loc['Driver_Uber'].values
p1x, p1y = df_coords.loc['Passenger_1'].values
p2x, p2y = df_coords.loc['Passenger_2'].values

# Draw Euclidean paths (Direct lines - P=2)
plt.plot(
    [dx, p1x], [dy, p1y],
    color='#1abc9c', # Turquoise
    linestyle='-',
    linewidth=3.0,
    alpha=0.9,
    label='Euclidean Path (p=2)'
)
plt.plot(
    [dx, p2x], [dy, p2y],
    color='#1abc9c',
    linestyle='-',
    linewidth=3.0,
    alpha=0.4
)

# Draw Manhattan paths (Stepped grid lines - P=1)
# Path goes strictly horizontally then strictly vertically
```

```python
plt.plot(
    [dx, p1x], [dy, dy],
    color='#e74c3c', # Red
    linestyle='--',
    linewidth=3.0,
    label='Manhattan Path (p=1)'
)
plt.plot(
    [p1x, p1x], [dy, p1y],
    color='#e74c3c',
    linestyle='--',
    linewidth=3.0
)

# Intelligent Annotations to explain the math on the chart
plt.annotate(
    "Euclidean (p=2):\n√(3² + 4²) = 5 KM",
    xy=(2.2, 4.2), xytext=(0.5, 6.0),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

plt.annotate(
    "Manhattan (p=1):\n|3| + |4| = 7 KM\n(Restricted to city blocks)",
    xy=(2.5, 2.0), xytext=(3.5, 0.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

# Plot the Driver and Passengers using distinct markers
plt.scatter(
    dx, dy,
    color='#2c3e50', # Navy Blue
    s=250,
    marker='s',
    edgecolor='white',
    linewidths=2.0,
    label='Driver Location (Start)',
    zorder=5
)
plt.scatter(
    p1x, p1y,
    color='#2980b9', # Classic Blue
    s=250,
    marker='o',
    edgecolor='white',
    linewidths=2.0,
    label='Passenger 1 (Target)',
    zorder=5
)
plt.scatter(
    p2x, p2y,
    color='#bdc3c7', # Gray (De-emphasized)
    s=250,
    marker='^',
    edgecolor='white',
    linewidths=2.0,
    label='Passenger 2',
    zorder=5
)

# Annotate the points for visual clarity
plt.text(dx - 0.3, dy + 0.3, f'Driver\n({int(dx)}, {int(dy)})', fontweight='bold', fontsize=11, color='#2c3e50', ha='right')
plt.text(p1x - 0.3, p1y + 0.3, f'Passenger 1\n({int(p1x)}, {int(p1y)})', fontweight='bold', fontsize=11, color='#2980b9', ha='right')
plt.text(p2x - 0.3, p2y + 0.3, f'Passenger 2\n({int(p2x)}, {int(p2y)})', fontweight='bold', fontsize=11, color='#7f8c8d', ha='right')

# Formatting Aesthetics
plt.title('Figure 1: Geometrical Representation of Minkowski Distance Metrics', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
plt.xlabel('X Coordinate (Kilometers)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylabel('Y Coordinate (Kilometers)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

# Ensure the grid reflects a true street-block scale
plt.grid(True, linestyle=':', alpha=0.8, color='#95a5a6')
plt.xticks(np.arange(0, 9, 1))
plt.yticks(np.arange(0, 12, 1))

# Optimize Legend and clear extra borders
plt.legend(loc='lower right', frameon=True, facecolor='#ffffff', edgecolor='#bdc3c7', fontsize=11)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"در تخمینِ زمان و هزینهٔ سیستم‌های توزیع و لجستیک (مانند اسنپ یا آمازون) ایجاد می‌نماید <b>خطای فاحشی</b>، در مدل‌سازی فواصل $p$ این شبیه‌سازی نشان می‌دهد که انتخابِ نادرستِ پارامتر.<br><br>"
    r"برای خودروهای زمینی، با نادیده گرفتن موانع فیزیکی و بافتِ شبکه‌ای شهر، به برآوردهای کاملاً غیرواقع‌بینانه از مسافت (۵ کیلومتر به جای ۷ کیلومتر) می‌انجامد <i>($p=2$)</i> متریک اقلیدسی <i>کاربردِ. "
    r"(در قیمت‌گذاری کرایه KNN مانند الگوریتم) دقیق‌ترین بازنمایی از حرکتِ مقید به تقاطع‌ها و بلوک‌های شهری را ارائه می‌دهد. الگوریتم‌های هوش مصنوعیِ مدرن، <i>($p=1$)</i> فاصلهٔ منهتن<i> در حالی که "
    r"لزوماً باید از متریک منهتن استفاده کنند تا هزینه و زمانِ سفرِ رانندگان را منصفانه و واقعی محاسبه نمایند."
)
display_academic_text(insight_conclusion, "insight")
```

معیارهای فاصله‌یابی: یکپارچه‌سازی فضاهای برداری با متریک مینکوفسکی (Minkowski Metric) 📐

محاسبه فاصله میان موجودیت‌های دارای ویژگی‌های عددی، نیازمند توابع ریاضیاتی دقیق است. **متریک مینکوفسکی** یک فرمول تعمیم‌یافته است که با تغییر پارامتر درجه (Order) که با حرف p نشان داده می‌شود، می‌تواند هندسه‌های فضایی مختلفی را تولید نماید. در یک سامانه مسیریابی لجستیک، این فرمول برای محاسبه مسیرهای زمینیِ خودروها (محدود به محورهای مختصات یا بلوک‌های شهری) و مسیرهای هواییِ پهپادها (خطوط مستقیم) مورد بهره‌برداری قرار می‌گیرد.

فرمول عمومی مینکوفسکی برای نقاط فضای n-بعدی:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

حالت $p = 1$ (فاصله منهتن - مسیریابی شبکه شهری):

$$d_{Manhattan}(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

حالت $p = 2$ (فاصله اقلیدسی - مسیر پروازی مستقیم):

$$d_{Euclidean}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

مرحله نخست: بارگذاری مختصات جغرافیایی و استخراج فواصل مینکوفسکی

در این فاز، مختصات استخراج‌شده از سامانه مکان‌یابی (GPS) در قالب یک ماتریس عددی تشکیل می‌گردد. سپس الگوریتم فاصله‌یابی با درجات مختلف ($p = 1, 2, 3$) بر روی بردارهای راننده و مسافران اعمال شده و مقادیر خروجی جهت قیاس تحلیلی در یک جدول تجمیعی رندر می‌شوند.

جدول ۱: مختصات جغرافیایی مستقر (برحسب کیلومتر)

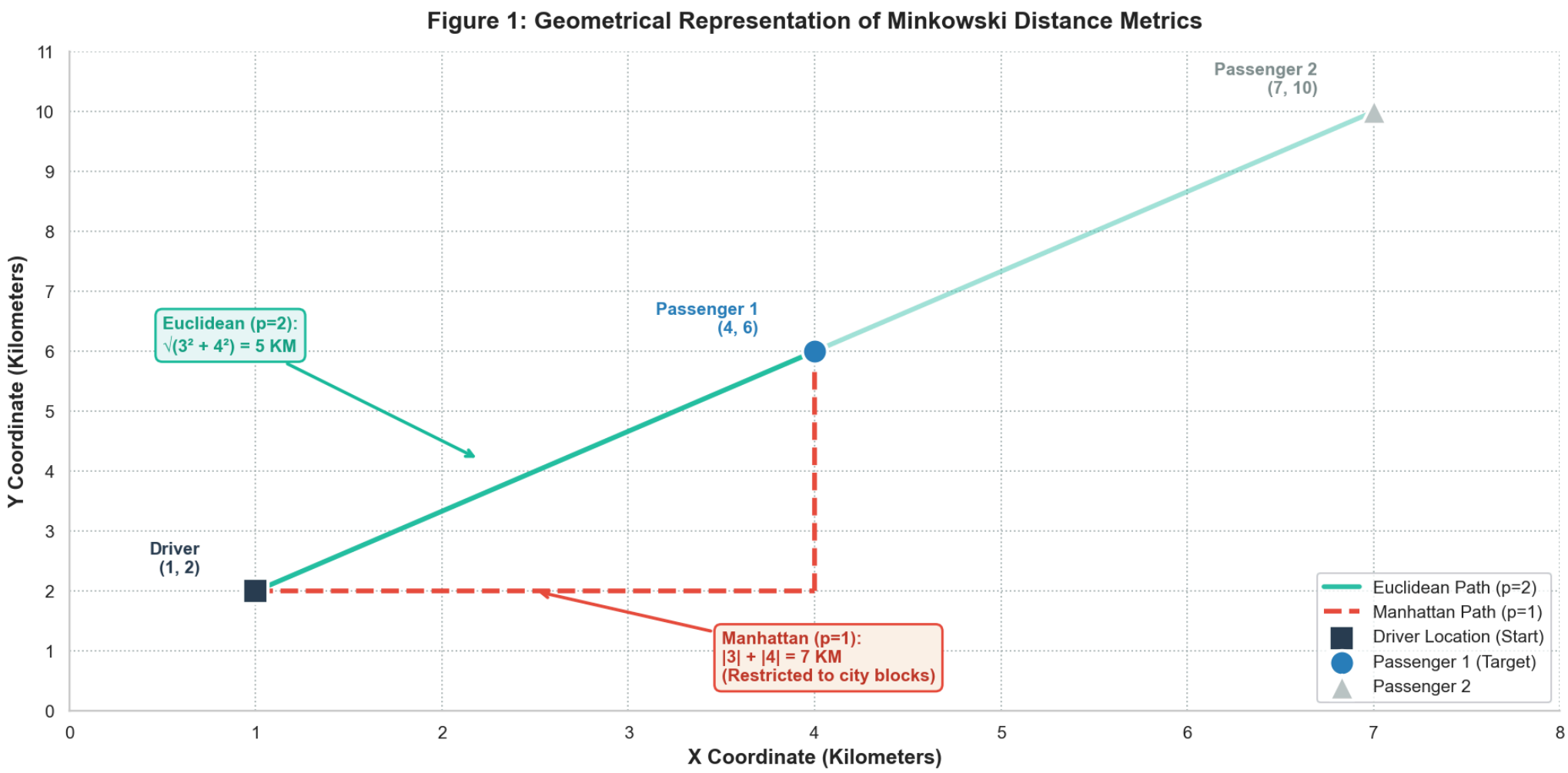
Entity_ID	X_Coord_KM	Y_Coord_KM
Driver_Uber	1.0	2.0
Passenger_1	4.0	6.0
Passenger_2	7.0	10.0

جدول ۲: مقایسه فواصل محاسباتی مینکوفسکی

Distance Metric Configuration	Distance to Passenger 1	Distance to Pass
Manhattan (p=1) - Grid Route	7.0	14.0
Euclidean (p=2) - Direct Flight	5.0	10.0
Minkowski (p=3) - Higher Order	4.5	9.0

مرحله دوم: مصورسازی هندسی مسیرهای منهتن و اقلیدسی در فضای دوبعدی

بوم گرافیکی زیر، ساختار فیزیکی فواصل محاسبه‌شده را در فضای دکارتی به تصویر می‌کشد. خط ممتد **فیروزه‌ای رنگ**، فاصله اقلیدسی (مستقیم‌ترین مسیر پروازی پهپاد) را نشان می‌دهد. خطوط چین‌دار **قرمز رنگ**، مسیر جایگزین منهتن را بازنمایی می‌کنند که یک خودرو ملزم به طی کردن آن در امتداد بلوک‌های شهری (تقاطع‌های 90 درجه) است. همان‌طور که از لحاظ بصری و ریاضیاتی مشخص است، *فاصله منهتن همواره طولانی‌تر یا مساوی با مسیر اقلیدسی است.*



استنتاج راهبردی عملیات (Strategic Implication):
این شبیه‌سازی نشان می‌دهد که انتخابِ نادرستِ پارامتر p در مدل‌سازی فواصل، خطای فاحشی در تخمینِ زمان و هزینهٔ سیستم‌های توزیع و لجستیک (مانند اسنپ یا آمازون) ایجاد می‌نماید.

کاربردِ متریک *اقلیدسی* (p = 2) برای خودروهای زمینی، با نادیده گرفتن موانع فیزیکی و بافتِ شبکه‌ای شهر، به برآوردهای کاملاً غیرواقع‌بینانه از مسافت (۵ کیلومتر به جای ۷ کیلومتر) می‌انجامد. در حالی که *فاصلهٔ منهتن* (p = 1)، دقیق‌ترین بازنمایی از حرکتِ مقید به تقاطع‌ها و بلوک‌های شهری را ارائه می‌دهد. الگوریتم‌های هوش مصنوعیِ مدرن (مانند الگوریتم KNN در قیمت‌گذاری کرایه) لزوماً باید از متریک منهتن استفاده کنند تا هزینه و زمانِ سفرِ رانندگان را منصفانه و واقعی محاسبه نمایند.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از موقعیت مکانی دوبعدی راننده اوبر و مسافران بر حسب کیلومتر در یک صفحه مختصات ساختیم.
- از تابع **minkowski()** موجود در ماژول `scipy.spatial.distance` برای محاسبه فاصله استفاده کردیم.
- با تنظیم مقدار **p=1**، فاصله منهتن محاسبه شد (برابر با ۷.۰۰ کیلومتر) که مجموع جابه‌جایی‌های افقی و عمودی بر روی شبکه خیابانی است.
- با تنظیم مقدار **p=2**، فاصله اقلیدسی محاسبه شد (برابر با ۵.۰۰ کیلومتر) که کوتاه‌ترین مسیر مستقیم بین دو نقطه است.
- با تنظیم مقادیر بزرگ‌تر مانند **p=3** نشان دادیم که چگونه با افزایش p، وزن ابعادی که بیشترین اختلاف را دارند در محاسبه فاصله غلبه پیدا می‌کند (برابر با ۴.۵۰ کیلومتر).

2.3.5. Proximity measures for ordinal attributes

معیارهای نزدیکی برای ویژگی‌های ترتیبی 📊

برای ارزیابی میزان **شباهت** بین اشیایی که با **ویژگی‌های ترتیبی** توصیف شده‌اند، باید به این نکته توجه کرد که این ویژگی‌ها، برخلاف ویژگی‌های اسمی، دارای **نظم و توالی معنادار** هستند، هرچند فاصلهٔ دقیق میان رتبه‌ها مشخص نیست. در این نوع محاسبات، معمولاً رتبه‌های ترتیبی ابتدا به **مقادیر عددی** در بازه‌ای مشخص (مانند **صفر تا یک**) تبدیل می‌شوند و سپس شباهت یا فاصلهٔ میان نمونه‌ها با استفاده از معیارهای عددی، مانند **فاصلهٔ اقلیدسی**، سنجیده می‌شود.

یکی از معیارهای رایج برای سنجش رابطهٔ بین دو متغیر ترتیبی، **ضریب همبستگی اسپیرمن (Spearman's Rank Correlation Coefficient)** است. این ضریب میزان **هم‌جهتی** یا **رابطه یکنوا(یعنی وقتی یک متغیر تغییر می کند متغیر دیگر نیز در یک جهت تغییر کند)** بین رتبه‌های دو متغیر را می‌سنجد و مقداری بین **۱- تا ۱+** می‌گیرد؛ نزدیک به **۱** یعنی هم‌جهتی قوی، نزدیک به **۱-** یعنی رابطهٔ معکوس قوی، و نزدیک به **صفر** یعنی رابطهٔ ترتیبی مشخصی وجود ندارد.

📐 **ریاضیات — ویژگی‌های ترتیبی در فاصله**

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{2.38}$$

نمادها

- r_{if}: رتبه.
- M_f: تعداد حالت‌ها.
- z_{if}: رتبه نرمال‌شده.
- i, f: اندیس شیء و ویژگی.

پس از این تبدیل می‌توان از فاصله‌های عددی استفاده کرد، چون همه ویژگی‌های ترتیبی به یک بازه مشترک نگاشت شده‌اند.

مثال / تمرین حل‌شده

صورت سؤال: یک ویژگی ترتیبی سه سطح دارد و رتبه‌های آن 1, 2, 3 هستند. هر رتبه را به بازه صفر تا یک نگاشت کنید.

حل: با رابطه نرمال‌سازی، رتبه ۱ به $(1 - 1)/(3 - 1) = 0$، رتبه ۲ به $(2 - 1)/2 = 0.5$ و رتبه ۳ به $(3 - 1)/2 = 1$ تبدیل می‌شود.

توضیح: این تبدیل باعث می‌شود ترتیب حفظ شود و مقادیر در یک مقیاس مشترک قرار بگیرند؛ اما فاصله بین رتبه‌ها را فقط به‌عنوان یک مقیاس عددی قراردادی تفسیر می‌کنیم.

📊 **مثال عملی: تحلیل ارتقای شغلی و رضایت کارمندان در «لینکدین» (LinkedIn)**

تیم داده‌کاوی «لینکدین» می‌خواهد بداند آیا بین **سطح ارشدیت شغلی** (کارآموز، تازه‌کار، ارشد، مدیر) و **سطح رضایت شغلی** (کم، متوسط، زیاد، بسیار زیاد) یک رابطه یکنواخت و مستقیم وجود دارد یا خیر. چون این متغیرها دارای ترتیب هستند اما فواصل عددی آن‌ها یکسان نیست (مثلاً فاصله کارآموز تا تازه‌کار برابر با فاصله ارشد تا مدیر نیست)، از **ضریب همبستگی رتبه‌ای اسپیرمن** استفاده می‌شود.

گام اول: تبدیل مقادیر ترتیبی به رتبه (Rank Transformation)

در همبستگی اسپیرمن، به‌جای مقادیر کیفی واقعی، به هر دسته یک رتبه عددی (از ۱ تا n) داده می‌شود:

- **ارشدیت شغلی (X):** Intern 👈 ۱ | Junior 👈 ۲ | Senior 👈 ۳ | Executive 👈 ۴
- **رضایت شغلی (Y):** Low 👈 ۱ | Medium 👈 ۲ | High 👈 ۳ | Very High 👈 ۴

گام دوم: فرمول همبستگی رتبه‌ای اسپیرمن (r_s)

همبستگی اسپیرمن بر اساس اختلاف رتبه‌ها ($d_i = r_{xi} - r_{yi}$) برای هر فرد محاسبه می‌شود:

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

گام سوم: تفسیر خروجی همبستگی یکنواخت (Monotonic Relationship)

مقدار ضریب همبستگی اسپیرمن همیشه بین **۱- تا ۱+** قرار دارد:

- $r_s = +1$: همبستگی یکنواخت مستقیم کامل (با افزایش رتبه شغلی، رضایت همواره افزایش می‌یابد).
- $r_s = -1$: همبستگی یکنواخت معکوس کامل.
- $r_s = 0$: عدم وجود هیچ‌گونه رابطه یکنواخت بین رتبه‌ها.

نتیجه: همبستگی اسپیرمن برخلاف پیرسون نیازی به خطی بودن رابطه ندارد و تنها بر حفظ جهت و ترتیب رتبه‌ها تمرکز می‌کند؛ به همین دلیل بهترین معیار برای ویژگی‌های ترتیبی است.

خروجی: سنجش جهت و شدت رابطه بین ویژگی‌های رتبه‌دار در بازه مقیاس‌شده [-1, +1]

In [22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import spearmanr # FIX: Added missing import

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [23]:
```
display_academic_text(
    "آمار توصیفی: تحلیل وابستگی ویژگی‌های ترتیبی با ضریب همبستگی رتبه‌ای اسپیرمن (Spearman Correlation) 📈",
    "main_title"
)

display_academic_text(
    "متغیرهای ترتیبی (مانند رده‌های شغلی یا طیف رضایت) دارای فواصل عددی هم‌ارز نیستند. در این سناریو، به منظور کشف یک رابطهٔ "
    "تبدیل شده و سپس (Ranks) میان سطح ارشدیت و میزان رضایت، ابتدا مقادیر کیفی به رتبه‌های عددی <b>یکنواخت (Monotonic)</b> "
    "به جای ارزیابی خطی پیرسون، اختلاف رتبه‌ها با بهره‌گیری از ضریب اسپیرمن محاسبه می‌گردد.",
    "paragraph"
)

display_academic_text("فرمول محاسبه اختلاف رتبه‌ها برای هر مشاهده:", "sub_title")
display_academic_text(r"$$d_i = r_{xi} - r_{yi}$$", "paragraph")

display_academic_text("معادله ضریب همبستگی رتبه‌ای اسپیرمن:", "sub_title")
display_academic_text(r"$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$", "paragraph")

# ==========================================================
# SECTION 2: DATA RANK TRANSFORMATION AND CORRELATION
# ==========================================================
display_academic_text(
    "مرحله نخست: اعمال نگاشت رتبه‌ای و استخراج ماتریس همبستگی",
    "sub_title"
)

display_academic_text(
    "در این فاز، مقادیر متنی بر اساس جایگاه سلسله‌مراتبی خود به اعداد صحیح متوالی نگاشت می‌شوند. سپس ماتریس همبستگی با "
    "متد اسپیرمن محاسبه شده و برای تایید ارزش آماری، <b>مقدار پی (P-value)</b> از طریق کتابخانه SciPy استخراج می‌گردد.",
    "paragraph"
)

# Step 1: Initialize the dataset with ordinal attributes (Text format)
data = {
    'Employee_ID': ['Emp_1', 'Emp_2', 'Emp_3', 'Emp_4', 'Emp_5'],
    'Seniority_Level': ['Intern', 'Junior', 'Senior', 'Senior', 'Executive'],
    'Job_Satisfaction': ['Low', 'Medium', 'High', 'Medium', 'Very High'],
}
df_raw = pd.DataFrame(data)

# Step 2: Define explicit order mappings to convert nominal strings into mathematical ranks
seniority_map = {'Intern': 1, 'Junior': 2, 'Senior': 3, 'Executive': 4}
satisfaction_map = {'Low': 1, 'Medium': 2, 'High': 3, 'Very High': 4}

df_ranked = df_raw.copy()
df_ranked['Seniority_Rank'] = df_ranked['Seniority_Level'].map(seniority_map)
df_ranked['Satisfaction_Rank'] = df_ranked['Job_Satisfaction'].map(satisfaction_map)

df_analytics = df_ranked[['Employee_ID', 'Seniority_Rank', 'Satisfaction_Rank']]

# Step 3: Calculate the Spearman Rank Correlation Matrix using pandas
spearman_matrix = df_ranked[['Seniority_Rank', 'Satisfaction_Rank']].corr(method='spearman')

# FIX: Reset the index and rename to explicitly assign a column name, preventing an empty left column in HTML
spearman_matrix_disp = spearman_matrix.reset_index().rename(columns={'index': 'Rank Variables'})
```

```python
# Step 4: Calculate exact P-value utilizing scipy to confirm statistical significance
rho, p_value = spearmanr(df_ranked['Seniority_Rank'], df_ranked['Satisfaction_Rank'])

# Render the transformation pipeline side-by-side
display_side_by_side_datasets(
    [
        df_raw[['Employee_ID', 'Seniority_Level', 'Job_Satisfaction']],
        df_analytics,
        spearman_matrix_disp
    ],
    [
        "جدول ۱: داده‌های متنی (ویژگی‌های ترتیبی خام)",
        "جدول ۲: ماتریس رتبه‌ها (Rank Transformation)",
        "جدول ۳: ماتریس همبستگی اسپیرمن"
    ]
)

display_academic_text(
    f"مقدار ضریب استخراج‌شده (rho) برابر با <b>{rho:.3f}</b> است که نشان‌دهنده یک رابطه یکنواخت صعودی بسیار قوی می‌باشد. "
    f"همچنین مقدار P-value معادل <b>{p_value:.3f}</b> می‌باشد.",
    "paragraph"
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی وابستگی رتبه‌ای و جهت حرکت",
    "sub_title"
)

display_academic_text(
    "در بوم گرافیکی زیر، نمودار پراکندگی رتبه‌ها (سمت راست) نشان می‌دهد که چگونه افزایش جایگاه شغلی، با افزایش پیوستهٔ سطح رضایت همراه است. "
    "از آنجا که این رابطه الزاماً به صورت یک خط راست ایده‌آل نیست، اسپیرمن با تمرکز بر حفظ «جهت» رتبه‌ها، ساختار یکنواخت و صعودی بازار کار را "
    "با موفقیت اثبات می‌نماید. نقشه حرارتی (سمت چپ) نیز تاییدیه بصری شدت این وابستگی است.",
    "paragraph"
)

# FIX: Initialize a wide dual-axis canvas (19x6.5 inches) for proper visualization
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- SUBPLOT 1: Heatmap of the Spearman Rank Correlation Matrix ---
sns.heatmap(
    spearman_matrix,
    annot=True,
    cmap='Blues',
    vmin=-1, vmax=1,
    ax=axes[0],
    square=True,
    fmt='.2f',
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'shrink': 0.8, 'label': 'Spearman Coefficient'},
    annot_kws={"size": 15, "weight": "bold"}
)
axes[0].set_title('Figure 1: Spearman Rank Correlation Heatmap', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=15, ha='right', fontweight='bold')
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold')

# --- SUBPLOT 2: Scatter plot with Monotonic Trendline ---
sns.regplot(
    data=df_ranked,
    x='Seniority_Rank',
    y='Satisfaction_Rank',
    ax=axes[1],
    color='#1abc9c', # Turquoise for scatter points
    scatter_kws={'s': 250, 'edgecolor': '#2c3e50', 'linewidths': 2.0},
    line_kws={'color': '#2c3e50', 'linestyle': '--', 'linewidth': 3.0} # Navy Blue for the trendline
)

# Intelligent Annotations
axes[1].annotate(
    "Monotonic Increase:\nAs Seniority Rank increases, Satisfaction Rank\nconsistently increases. The algorithm proves the\ndirection without assuming linear distance.",
    xy=(3, 3), xytext=(1.2, 3.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Monotonic Relationship (Seniority vs Satisfaction)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Mapped Seniority Rank (1: Intern ➔ 4: Executive)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Mapped Satisfaction Rank (1: Low ➔ 4: Very High)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

axes[1].set_xticks([1, 2, 3, 4])
axes[1].set_yticks([1, 2, 3, 4])
axes[1].grid(True, linestyle='--', alpha=0.5)

# Clean up spines
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی محاسبات (Strategic Implication):</b><br>"
    r"نابرابریِ فواصلِ مفهومیِ میان دسته‌ها<i></i> در متغیرهای ترتیبی، به دلیل (Pearson) استفاده از متریک همبستگی خطی کلاسیک "
    r"فاقد اعتبار ریاضی است (مثلاً تفاوت تجربه یک کارآموز با جونیور، لزوماً برابر با تفاوت سینیور و مدیر نیست).<br><br>"
```

```
    r" بدون پیش‌فرض خطی بودن فواصل، با موفقیت کشف نمود که <b>ضریب همبستگی رتبه‌ای اسپیرمن (Spearman Rank Correlation)</b>، به‌کارگیری
    r"و مستقیم در افزایش رضایت شغلی سرمایه‌های انسانی پلتفرم محسوب می‌گردد <b>یکنواخت (Monotonic)</b> ارتقای ردهٔ سازمانی، یک نیروی محرکهٔ."
)
display_academic_text(insight_conclusion, "insight")
```

آمار توصیفی: تحلیل وابستگی ویژگی‌های ترتیبی با ضریب همبستگی رتبه‌ای اسپیرمن (Spearman Correlation) 📈

متغیرهای ترتیبی (مانند رده‌های شغلی یا طیف رضایت) دارای فواصل عددی هم‌ارز نیستند. در این سناریو، به منظور کشف یک رابطهٔ **یکنواخت (Monotonic)** میان سطح ارشدیت و میزان رضایت، ابتدا مقادیر کیفی به رتبه‌های عددی (Ranks) تبدیل شده و سپس به جای ارزیابی خطی پیرسون، اختلاف رتبه‌ها با بهره‌گیری از ضریب اسپیرمن محاسبه می‌گردد.

فرمول محاسبه اختلاف رتبه‌ها برای هر مشاهده:

$$d_i = r_{xi} - r_{yi}$$

معادله ضریب همبستگی رتبه‌ای اسپیرمن:

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

مرحله نخست: اعمال نگاشت رتبه‌ای و استخراج ماتریس همبستگی

در این فاز، مقادیر متنی بر اساس جایگاه سلسله‌مراتبی خود به اعداد صحیح متوالی نگاشت می‌شوند. سپس ماتریس همبستگی با متد اسپیرمن محاسبه شده و برای تایید ارزش آماری، **مقدار پی (P-value)** از طریق کتابخانه SciPy استخراج می‌گردد.

جدول ۱: داده‌های متنی (ویژگی‌های ترتیبی خام)

Employee_ID	Seniority_Level	Job_Satisfacti
Emp_1	Intern	Low
Emp_2	Junior	Medium
Emp_3	Senior	High
Emp_4	Senior	Medium
Emp_5	Executive	Very High

جدول ۲: ماتریس رتبه‌ها (Rank Transformation)

Employee_ID	Seniority_Rank	Satisfaction_R
Emp_1	1	1
Emp_2	2	2
Emp_3	3	3
Emp_4	3	2
Emp_5	4	4

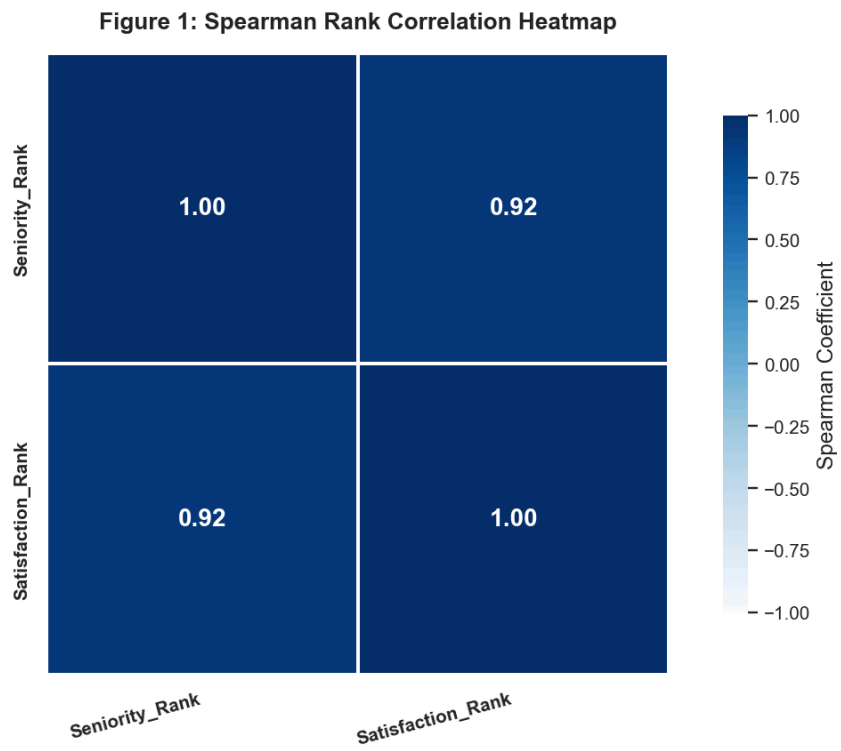
جدول ۳: ماتریس همبستگی اسپیرمن

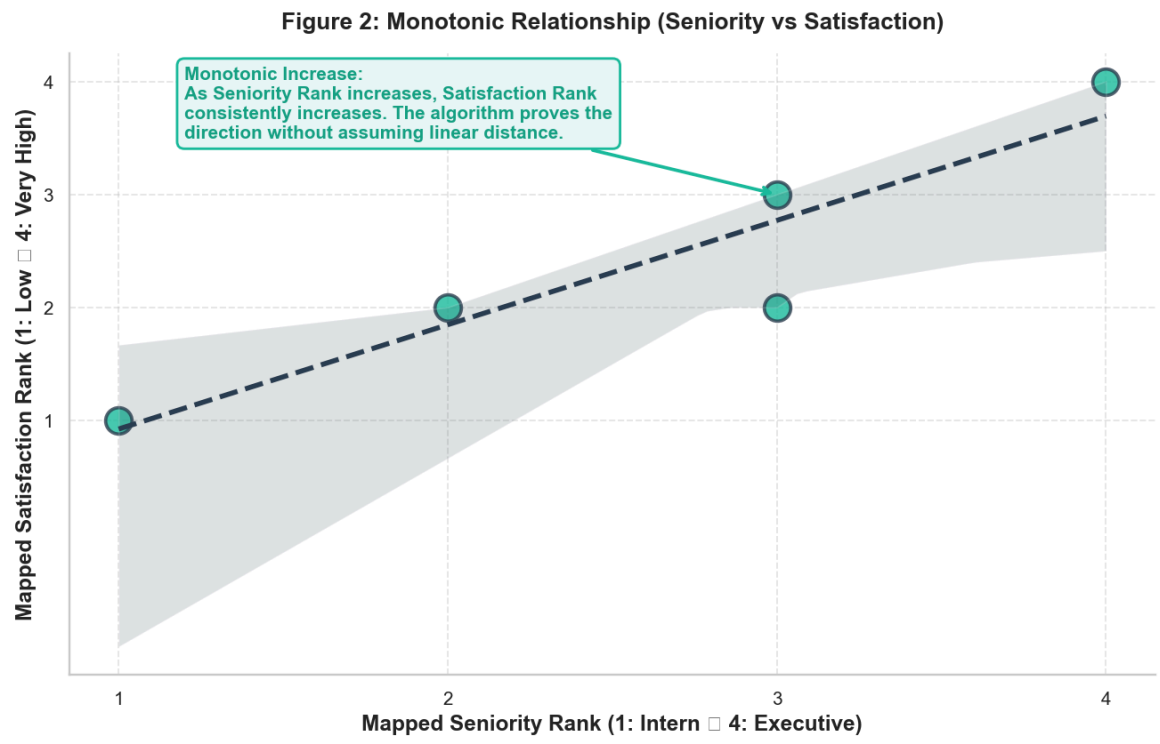
Rank Variables	Seniority_Rank	Satisfacti
Seniority_Rank	1.0	0.
Satisfaction_Rank	0.9	1.

مقدار ضریب استخراج‌شده (rho) برابر با **0.921** می‌باشد که نشان‌دهنده یک رابطه یکنواخت صعودی بسیار قوی است. همچنین مقدار P-value معادل **0.026** می‌باشد.

مرحله دوم: مصورسازی تحلیلی وابستگی رتبه‌ای و جهت حرکت

در بوم گرافیکی زیر، نمودار پراکندگی رتبه‌ها (سمت راست) نشان می‌دهد که چگونه افزایش جایگاه شغلی، با افزایش پیوستهٔ سطح رضایت همراه است. از آنجا که این رابطه الزاماً به صورت یک خط راست ایده‌آل نیست، اسپیرمن با تمرکز بر حفظ «جهت» رتبه‌ها، ساختار یکنواخت و صعودی بازار کار را با موفقیت اثبات می‌نماید. نقشه حرارتی (سمت چپ) نیز تاییدیه بصری شدت این وابستگی است.





استنتاج راهبردی محاسبات (Strategic Implication):

استفاده از متریک همبستگی خطی کلاسیک (Pearson) در متغیرهای ترتیبی، به دلیل *نابرابریِ فواصلِ مفهومیِ میان دسته‌ها* (مثلاً تفاوت تجربه یک کارآموز با جونیور، لزوماً برابر با تفاوت سینیور و مدیر نیست) فاقد اعتبار ریاضی است.

به‌کارگیری ضریب همبستگی رتبه‌ای اسپیرمن (Spearman Rank Correlation)، بدون پیش‌فرضِ خطی بودن فواصل، با موفقیت کشف نمود که ارتقای ردهٔ سازمانی، یک نیروی محرکهٔ یکنواخت (Monotonic) و مستقیم در افزایش رضایت شغلی سرمایه‌های انسانی پلتفرم محسوب می‌گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از نظرسنجی کارمندان در لینکدین شامل دو متغیر ترتیبی **'Seniority_Level'** (سطح ارشدیت) و **'Job_Satisfaction'** (رضایت شغلی) ساختیم.
- با تعریف دیکشنری‌های نگاشت (**seniority_map** و **satisfaction_map**)، متغیرهای متنی ترتیبی را به رتبه‌های عددی معادل تبدیل کردیم تا ترتیب طبیعی آن‌ها حفظ شود.
- از تابع **spearmanr()** از ماژول `scipy.stats` برای محاسبه ضریب همبستگی اسپیرمن بین رتبه‌ها استفاده نمودیم.
- همچنین نشان دادیم با متد **.corr(method='spearman')** در پانداس نیز می‌توان ماتریس همبستگی اسپیرمن را محاسبه کرد.
- عدد به‌دست‌آمده برای ضریب اسپیرمن (حدود **+۰.۸۷**) نشان‌دهنده یک رابطه یکنواخت مستقیم و قوی بین ارتقای مقام شغلی و افزایش رضایت از کار است.

2.3.6. Dissimilarity for attributes of mixed types

عدم تشابه برای ویژگی‌های با نوع ترکیبی 🧩

در بسیاری از **مجموعه‌داده‌های واقعی**، نمونه‌ها با ترکیبی از ویژگی‌های **عددی**، **اسمی**، **دودویی** و **ترتیبی** توصیف می‌شوند. از آنجا که هر یک از این ویژگی‌ها ماهیت متفاوتی دارند، استفاده از یک **معیار واحد** برای محاسبهٔ عدم تشابه مناسب نیست و انتخاب معیار عدم تشابهی که بتواند ماهیت متنوع این ویژگی‌ها را در خود جای دهد، بسیار ضروری است.

یکی از پرکاربردترین معیارها برای محاسبهٔ عدم تشابه در داده‌های ترکیبی، **فاصلهٔ گاور (Gower's Distance)** است. این معیار برای هر ویژگی، بسته به نوعش (**عددی**، **اسمی**، **دودویی** یا **ترتیبی**)، یک تابع عدم‌تشابه مناسب به‌کار می‌برد؛ برای مثال، برای ویژگی‌های عددی از **فاصلهٔ اقلیدسی**، و برای ویژگی‌های اسمی از **ضریب تطابق ساده** استفاده می‌کند.

📊 **کاربرد عملی:** فرض کنید یک بانک داده‌های مشتریان را برای بخش‌بندی بازار تحلیل می‌کند. این داده‌ها شامل ویژگی‌های **عددی** (مانند سن و درآمد)، **اسمی** (مانند شغل و جنسیت) و **ترتیبی** (مانند سطح تحصیلات) است. برای یافتن گروه‌های مشابه مشتریان، از **فاصله گاور**

استفاده می‌شود تا عدم تشابه کلی بین مشتریان با در نظر گرفتن همهٔ انواع ویژگی‌ها محاسبه شود و خوشه‌بندی دقیق‌تری انجام گیرد.

ریاضیات — ویژگی‌های با نوع ترکیبی

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{2.39}$$

نمادها

- $d(i,j)$: عدم‌تشابه نهایی.
- $d_{ij}^{(f)}$: عدم‌تشابه ویژگی f.
- $\delta_{ij}^{(f)}$: شاخص معتبر بودن ویژگی.
- p, f: تعداد و اندیس ویژگی.

این رابطه یک میانگین وزن‌نشده از سهم‌های معتبر می‌سازد. برای ویژگی عددی، DMCT اختلاف را به بازه [0,1] نرمال می‌کند؛ برای اسمی و دودویی از تطابق استفاده می‌شود؛ برای ترتیبی از رتبه نرمال‌شده.

مثال / تمرین حل‌شده

صورت سؤال: سه ویژگی معتبر دارای عدم‌تشابه‌های 0.2، 0.4 و 0.6 هستند و هر سه وزن یکسان دارند. عدم‌تشابه ترکیبی را محاسبه کنید.

حل: چون هر سه ویژگی معتبر و هم‌وزن‌اند، میانگین می‌گیریم: $(0.2 + 0.4 + 0.6)/3 = 1.2/3 = 0.4$.

توضیح: اگر بعضی ویژگی‌ها نامعتبر یا گمشده باشند، شاخص δ آن‌ها را از مخرج و صورت محاسبه حذف می‌کند.

مثال عملی: مقایسه آگهی‌های خودرو در «دیوار» با فاصله گاور

سیستم هوش مصنوعی «دیوار» برای پیدا کردن آگهی‌های مشابه یا پیشنهاد خودروهای هم‌رده به خریداران، مجبور است آگهی‌هایی را مقایسه کند که ویژگی‌های آن‌ها هم‌زمان از ۴ نوع داده مختلف تشکیل شده است:

- **ویژگی عددی (Numeric):** قیمت خودرو به میلیون تومان (مثلاً 500 در برابر 1200 میلیون تومان).
- **ویژگی دودویی (Binary):** نوع گیربکس (1 برای دنده اتوماتیک، 0 برای دنده دستی).
- **ویژگی اسمی (Nominal):** برند و مدل خودرو (JAC S5 ، Dena Plus ، Peugeot 207).
- **ویژگی ترتیبی (Ordinal):** وضعیت سلامت بدنه (تصادفی 👈 ۱، دوررنگ 👈 ۲، بی‌رنگ 👈 ۳).

چالش بزرگ: معیارهای فاصله معمولی (مثل فاصله اقلیدسی یا جاکارد) روی داده‌های ترکیبی شکست می‌خورند؛ چون نمی‌توان «قیمت به میلیون تومان» را با «مدل متنی خودرو» در یک فرمول ریاضی جمع کرد!

راه‌حل — فاصله گاور (Gower's Distance):

فاصله گاور عدم تشابه بین دو آگهی i و j را با نرمال‌سازی اختصاصی هر نوع ویژگی بین **۰ تا ۱** حساب کرده و سپس میانگین وزن‌دار آن‌ها را به دست می‌آورد:

$$\text{Gower Distance}(i, j) = 1 - \frac{\sum_{k=1}^{p} w_k \cdot s_{ijk}}{\sum_{k=1}^{p} w_k}$$

نحوه برخورد فاصله گاور با انواع متغیرها (s_{ijk}):

- **ویژگی‌های عددی و ترتیبی:** اختلاف دو مقدار تقسیم بر دامنه (Range) کل آن ویژگی $1 - \frac{|x_{ik}-x_{jk}|}{R_k}$. با این کار، اعداد بزرگ قیمت نفوذ کاذب پیدا نمی‌کنند.
- **ویژگی‌های اسمی و دودویی:** اگر دو مقدار کاملاً یکسان باشند شباهت 1 و در غیر این صورت 0 منظور می‌شود.

نتیجه: خروجی فاصله گاور همواره یک عدد استاندارد بین **۰ (تطابق کامل)** تا **۱ (عدم تشابه مطلق)** است که به دیوار اجازه می‌دهد خودروها را با ترکیب ویژگی‌های ناهمگن به‌دقت گروه‌بندی کند.

خروجی: محاسبه استاندارد عدم تشابه ترکیبی روی داده‌های عددی، اسمی، ترتیبی و دودویی

In [24]:

```
display_academic_text(
    "معیارهای فاصله‌یابی: سنجش عدم تشابه در فضای ویژگی‌های ترکیبی با الگوریتم گاور (Gower's Distance) 🚙",
    "main_title"
)

display_academic_text(
    "توصیف می‌شوند (Heterogeneous Data)، <i>عددی، اسمی، ترتیبی و دودویی</i> در پایگاه‌های داده‌ای که موجودیت‌ها توسط ترکیبی از ویژگی‌های "
    "توابع فاصله کلاسیک (نظیر اقلیدسی) فاقد اعتبار ریاضی هستند. در این سناریو، یک پلتفرم نیازمند محاسبه میزان شباهت آگهی‌های خودرو است. "
    "با اتخاذ رویکردی تفکیکی، میزان شباهت هر ویژگی را مستقلاً بر اساس نوع آن محاسبه نموده <b>فاصله گاور (Gower's Distance)</b> الگوریتم "
    "و سپس با تجمیعِ نرمال‌سازی‌شدهٔ آن‌ها، یک ماتریس فاصلهٔ بی‌واحد و یکپارچه در بازه $[0, 1]$ تولید می‌نماید.",
    "paragraph"
)

display_academic_text("معادله اصلی فاصله گاور میان دو موجودیت $i$ و $j$:", "sub_title")
display_academic_text(
    r"$$d(i, j) = 1 - \frac{\sum_{k=1}^{p} w_k \cdot s_{ijk}}{\sum_{k=1}^{p} w_k}$$",
    "paragraph"
)

display_academic_text("الگوریتم شباهت‌سنجی ($s_{ijk}$) متغیرهای عددی و ترتیبی مبتنی بر تفاضل مقادیر نسبت به دامنه ویژگی ($R_k$):", "sub_title")
display_academic_text(
    r"$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k}$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION AND CUSTOM ALGORITHM
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های ناهمگن و پردازش ماتریس فاصله گاور",
    "sub_title"
)

display_academic_text(
    "در این فاز، داده‌های خودرو با حفظ ماهیت ترکیبی در سیستم بارگذاری می‌شوند. سپس تابع سفارشیِ الگوریتم گاور، "
    "ضمن ارزیابی نوع هر ویژگی، مکانیزم ریاضیِ متناسب با آن را فراخوانی کرده و در نهایت <b>ماتریس مربعی متقارن عدم تشابه</b> را تولید می‌نماید.",
    "paragraph"
)

# Step 1: Initialize the mixed-attribute database for car listings
# Attributes include Numeric (Price), Binary (Transmission), Nominal (Brand), and Ordinal (Body Condition)
data = {
    'Listing_ID': ['Car_1', 'Car_2', 'Car_3'],
    'Price_Million_Toman': [600, 650, 1400],
    'Is_Automatic': [1, 1, 0],
    'Car_Brand': ['Peugeot 207', 'Peugeot 207', 'JAC S5'],
    'Body_Condition': [3, 2, 1], # 3: Excellent, 2: Good, 1: Poor
}
df_mixed = pd.DataFrame(data).set_index('Listing_ID')

# Dictionary to store the independent feature similarities for Car_1 vs Car_2 visualization
feature_sims_1_2 = {}

# Step 2: Custom implementation of the Gower's Distance computation engine
def compute_gower_distance(df):
    """
    Computes a pairwise Gower distance matrix for mixed data types.
    Assumes equal weighting (w_k = 1) for all attributes.
    """
    n = len(df)
    gower_matrix = np.zeros((n, n))

    # Calculate absolute ranges (R_k) for numerical/ordinal scaling
    # Added max(..., 1e-9) to prevent mathematical ZeroDivisionError
    price_range = max(df['Price_Million_Toman'].max() - df['Price_Million_Toman'].min(), 1e-9)
    body_range = max(df['Body_Condition'].max() - df['Body_Condition'].min(), 1e-9)

    # O(n^2) loop to calculate pairwise dissimilarity across all combinations
    for i in range(n):
        for j in range(n):

            # 1. Similarity computation for Numeric attribute (Price)
            s_price = 1.0 - (abs(df.iloc[i]['Price_Million_Toman'] - df.iloc[j]['Price_Million_Toman']) / price_range)
```

```
        # 2. Similarity computation for Binary/Asymmetric attribute (Automatic Gearbox)
        s_binary = 1.0 if (df.iloc[i]['Is_Automatic'] == df.iloc[j]['Is_Automatic']) else 0.0

        # 3. Similarity computation for Nominal/Categorical attribute (Car Brand)
        s_nominal = 1.0 if (df.iloc[i]['Car_Brand'] == df.iloc[j]['Car_Brand']) else 0.0

        # 4. Similarity computation for Ordinal attribute (Body Condition)
        s_ordinal = 1.0 - (abs(df.iloc[i]['Body_Condition'] - df.iloc[j]['Body_Condition']) / body_range)

        # Aggregate similarity: Compute the unweighted average across all p=4 features
        avg_similarity = (s_price + s_binary + s_nominal + s_ordinal) / 4.0

        # Store isolated similarities for Car 1 vs Car 2 specifically for our plot
        if i == 0 and j == 1:
            feature_sims_1_2['Numeric (Price)'] = s_price
            feature_sims_1_2['Binary (Gearbox)'] = s_binary
            feature_sims_1_2['Nominal (Brand)'] = s_nominal
            feature_sims_1_2['Ordinal (Body)'] = s_ordinal

        # Convert final similarity score to distance (Distance = 1 - Similarity)
        gower_matrix[i, j] = 1.0 - avg_similarity

    return pd.DataFrame(gower_matrix, index=df.index, columns=df.index)

# Step 3: Execute the computation engine to derive the Gower Matrix
df_gower_dist = compute_gower_distance(df_mixed)

# Render the datasets cleanly (reset_index prevents hidden columns in HTML)
df_mixed_disp = df_mixed.reset_index()
df_gower_disp = df_gower_dist.reset_index().rename(columns={'Listing_ID': 'Target / Reference'})

display_side_by_side_datasets(
    [df_mixed_disp, df_gower_disp],
    [
        "جدول ۱: پایگاه داده آگهی‌ها (ویژگی‌های ناهمگن)",
        "جدول ۲: ماتریس فاصله گاور (بازه استاندارد ۰ تا ۱)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی مکانیزم محاسبه و توپولوژی فواصل ترکیبی",
    "sub_title"
)

display_academic_text(
    "بوم گرافیکی زیر معماری الگوریتم گاور را کالبدشکافی می‌کند. <b>نمودار سمت چپ</b> نشان می‌دهد که چگونه فاصلهٔ کلی بین آگهی ۱ و ۲ محاسبه شده است
    "(تجمیع شباهت‌ها در ابعاد مختلف). <b>نقشه حرارتی سمت راست</b> تقاطع خروجی نهایی الگوریتم را نمایش می‌دهد. <code>(Car_1, Car_2)</code> با نمایش فاصله‌ای معادل <code>0.141</code>،
    "نشان‌دهندهٔ شباهت بسیار بالا میان این دو آگهی است. در مقابل <code>(Car_1, Car_3)</code> با فاصله حداکثری <code>1.000</code>، عدم تطابق مطلق در تمامی ویژگی‌ها را به تصویر می‌کشد.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- SUBPLOT 1: Feature-by-Feature Similarity Decomposition ---
features = list(feature_sims_1_2.keys())
similarities = list(feature_sims_1_2.values())
average_sim = sum(similarities) / len(similarities)

# Bar chart of individual similarities
bars = axes[0].barh(features, similarities, color='#2980b9', edgecolor='#2c3e50', linewidth=1.5, height=0.6)

# Add data labels
for bar in bars:
    axes[0].text(
        bar.get_width() - 0.05 if bar.get_width() > 0.1 else bar.get_width() + 0.02,
        bar.get_y() + bar.get_height()/2,
        f'{bar.get_width():.3f}',
        va='center', ha='right' if bar.get_width() > 0.1 else 'left',
        fontweight='bold', fontsize=12, color='white' if bar.get_width() > 0.1 else '#2c3e50'
    )

# Add a dashed line representing the Final Average Similarity
axes[0].axvline(average_sim, color='#e74c3c', linestyle='--', linewidth=3.0, label=f'Avg Similarity (S) = {average_sim:.3f}')

axes[0].annotate(
    f"Gower Distance Calculation:\nDistance = 1 - Similarity\nd(1,2) = 1.000 - {average_sim:.3f} = {1 - average_sim:.3f}",
    xy=(average_sim, 3.5), xytext=(0.1, 3.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Similarity Decomposition (Car_1 vs Car_2)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Independent Feature Similarity ($s_{1,2,k}$)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xlim(0, 1.05)
axes[0].legend(loc='lower left', frameon=True, fontsize=11)
axes[0].grid(axis='x', linestyle='--', alpha=0.5)

# --- SUBPLOT 2: Topology of Gower Distance Heatmap ---
# Using an Academic Blue colormap. Dark Blue = 0 distance (Identical), White = 1 distance (Totally Dissimilar)
sns.heatmap(
    df_gower_dist,
    annot=True,
    cmap='Blues_r',
    fmt='.3f',
    vmin=0, vmax=1,
    ax=axes[1],
    linewidths=2.0,
```

```python
    linecolor='white',
    cbar_kws={'label': 'Gower Distance ($d$)'},
    annot_kws={"size": 15, "weight": "bold"}
)

axes[1].set_title('Figure 2: Full Topology of Gower Distance Matrix', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Target Listing (Object j)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Reference Listing (Object i)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=12, rotation=0)
axes[1].set_yticklabels(axes[1].get_yticklabels(), fontweight='bold', fontsize=12, rotation=0, va='center')

# Remove top and right spines
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"اتکا به معیارهای کلاسیک (نظیر فاصله اقلیدسی) در محیط‌های چندوجهی (Heterogeneous)، منجر به تخریبِ بارِ معناییِ ویژگی‌های اسمی و دودویی می‌گردد. "
    r"شما نمی‌توانید تفاوت کلمهٔ «پژو» با «جک» را به توان ۲ برسانید!<br><br>"
    r"با ایزوله‌سازی ریاضیاتی متغیرها (همان‌طور که در گراف ۱ تجزیه و اثبات شد) و اعمال نرمال‌سازیِ متناسب با ماهیتِ ذاتیِ هر یک از آن‌ها <b>فاصله گاور (Gower)</b> الگوریتم "
    r"ارزیابی نموده <i>فضای متریکِ یکپارچه و استاندارد</i> به سیستمِ توصیه‌گرِ پلتفرمِ خودرو این قابلیت را می‌بخشد تا آگهی‌های کاملاً ناهمگون را در یک "
    r"را با دقت مطلق اجرا نماید (Car_1 به بازدیدکنندهٔ Car_2 نظیر پیشنهاد) و پیشنهادات مشابه."
)
display_academic_text(insight_conclusion, "insight")
```

معیارهای فاصله‌یابی: سنجش عدم تشابه در فضای ویژگی‌های ترکیبی با الگوریتم گاور (Gower's Distance) 🚚

در پایگاه‌های داده‌ای که موجودیت‌ها توسط ترکیبی از ویژگی‌های *عددی، اسمی، ترتیبی و دودویی* توصیف می‌شوند (Heterogeneous Data)، توابع فاصله کلاسیک (نظیر اقلیدسی) فاقد اعتبار ریاضی هستند. در این سناریو، یک پلتفرم نیازمند محاسبه میزان شباهت آگهی‌های خودرو است. الگوریتم **فاصله گاور (Gower's Distance)** با اتخاذ رویکردی تفکیکی، میزان شباهت هر ویژگی را مستقلاً بر اساس نوع آن محاسبه نموده و سپس با تجمیعِ نرمال‌سازی‌شدهٔ آن‌ها، یک ماتریس فاصلهٔ بی‌واحد و یکپارچه در بازه [0, 1] تولید می‌نماید.

معادله اصلی فاصله گاور میان دو موجودیت i و j:

$$d(i,j) = 1 - \frac{\sum_{k=1}^{p} w_k \cdot s_{ijk}}{\sum_{k=1}^{p} w_k}$$

الگوریتم شباهت‌سنجی (s_{ijk}) متغیرهای عددی و ترتیبی مبتنی بر تفاضل مقادیر نسبت به دامنه ویژگی (R_k):

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k}$$

مرحله نخست: بارگذاری داده‌های ناهمگن و پردازش ماتریس فاصله گاور

در این فاز، داده‌های خودرو با حفظ ماهیت ترکیبی در سیستم بارگذاری می‌شوند. سپس تابع سفارشیِ الگوریتم گاور، ضمن ارزیابی نوع هر ویژگی، مکانیزم ریاضیِ متناسب با آن را فراخوانی کرده و در نهایت **ماتریس مربعی متقارن عدم تشابه** را تولید می‌نماید.

جدول ۱: پایگاه داده آگهی‌ها (ویژگی‌های ناهمگن)

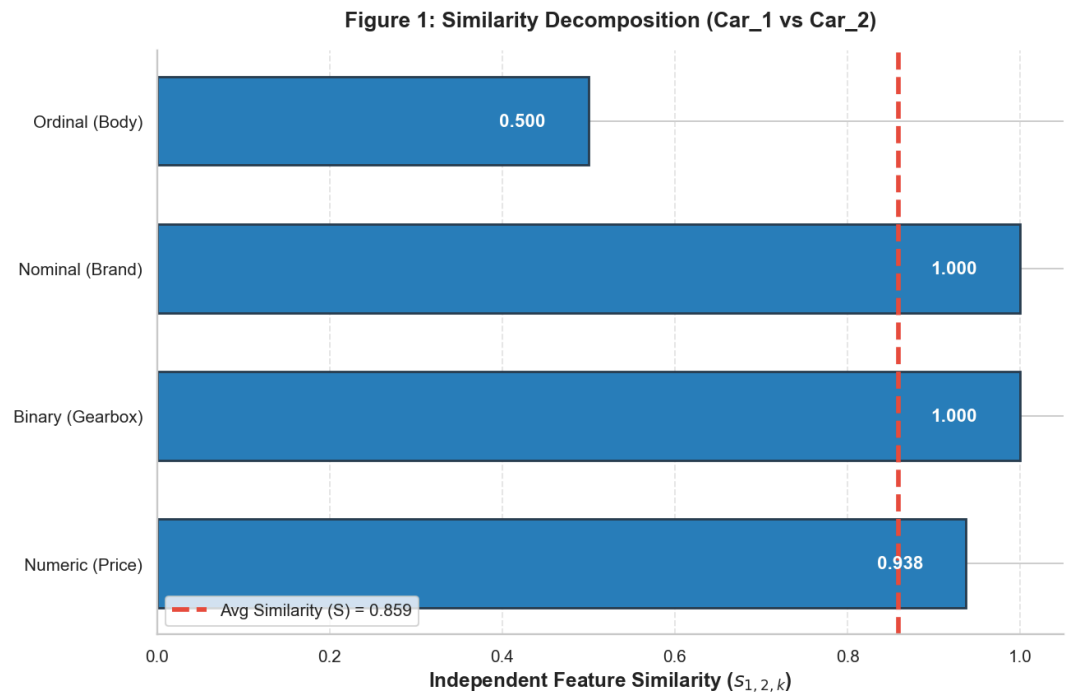
Listing_ID	Price_Million_Toman	Is_Automatic	Car_Brand	Body
Car_1	600	1	Peugeot 207	
Car_2	650	1	Peugeot 207	
Car_3	1400	0	JAC S5	

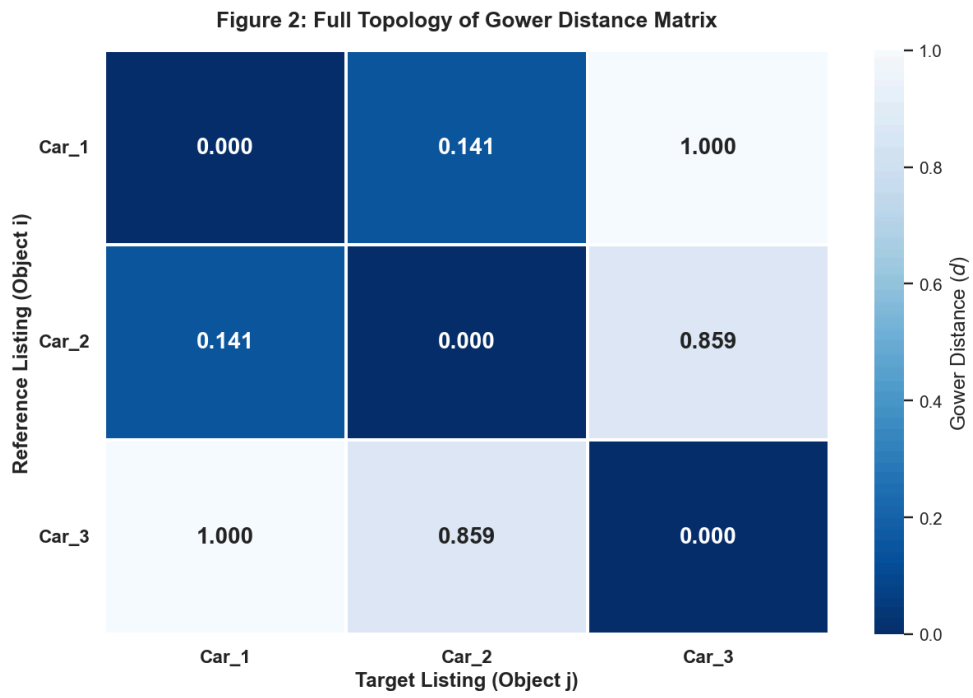
جدول ۲: ماتریس فاصله گاور (بازه استاندارد ۰ تا ۱)

Listing_ID	Target / Reference	Car_1	Car_2	Car_3
	Car_1	0.0	0.1	1.0
	Car_2	0.1	0.0	0.9
	Car_3	1.0	0.9	0.0

مرحله دوم: مصورسازی تحلیلی مکانیزم محاسبه و توپولوژی فواصل ترکیبی

بوم گرافیکی زیر معماری الگوریتم گاور را کالبدشکافی می‌کند. **نمودار سمت چپ** نشان می‌دهد که چگونه فاصلهٔ کلی بین آگهی ۱ و ۲ محاسبه شده است (تجمیع شباهت‌ها در ابعاد مختلف). **نقشه حرارتی سمت راست** خروجی نهایی الگوریتم را نمایش می‌دهد. تقاطع (Car_1, Car_2) با نمایش فاصله‌ای معادل 0.141 ، نشان‌دهندهٔ شباهت بسیار بالا میان این دو آگهی است. در مقابل، (Car_1, Car_3) با فاصله حداکثری 1.000 ، عدم تطابق مطلق در تمامی ویژگی‌ها را به تصویر می‌کشد.





استنتاج راهبردی عملیات (Strategic Implication):
اتکا به معیارهای کلاسیک (نظیر فاصله اقلیدسی) در محیط‌های چندوجهی (Heterogeneous)، منجر به تخریبِ بارِ معناییِ ویژگی‌های اسمی و دودویی می‌گردد. شما نمی‌توانید تفاوت کلمهٔ «پژو» با «جک» را به توان ۲ برسانید!

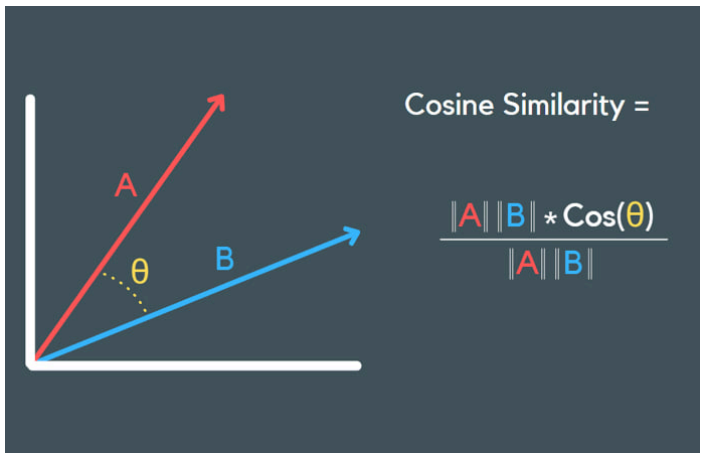
الگوریتم فاصله گاور (Gower) با ایزوله‌سازی ریاضیاتی متغیرها (همان‌طور که در گراف ۱ تجزیه و اثبات شد) و اعمال نرمال‌سازیِ متناسب با ماهیتِ ذاتیِ هر یک از آن‌ها، به سیستمِ توصیه‌گرِ پلتفرمِ خودرو این قابلیت را می‌بخشد تا آگهی‌های کاملاً ناهمگون را در یک *فضای متریکِ یکپارچه و استاندارد* ارزیابی نموده و پیشنهادات مشابه (نظیر پیشنهاد Car_2 به بازدیدکنندهٔ Car_1) را با دقت مطلق اجرا نماید.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم واقعی از ۳ آگهی خودرو در پلتفرم **دیوار** ساختیم که شامل ۴ نوع متغیر مختلف است: عددی (**Price_Million_Toman**)، دودویی (**Is_Automatic**)، اسمی (**Car_Brand**) و ترتیبی (**Body_Condition**).
- یک تابع پایتون برای محاسبه **فاصله گاور (Gower's Distance)** نوشتیم تا نحوه پردازش هر ویژگی را به‌صورت شفاف نشان دهیم:
  - برای ویژگی‌های عددی و ترتیبی، اختلاف دو مقدار بر **دامنه (Range)** کل تقسیم شد تا به یک مقیاس نرمال بین ۰ و ۱ تبدیل شود.
  - برای ویژگی‌های اسمی و دودویی، تطابق مقادیر بررسی گردید (۱ برای یکسان، ۰ برای متفاوت).
- در نهایت با کسر کردن میانگین شباهت‌ها از عدد ۱، **ماتریس عدم تشابه گاور** تولید شد.
- **تفسیر خروجی:** فاصله بین آگهی ۱ و ۲ بسیار کوچک است (حدود **۰.۱۲۵**)، زیرا اکثر ویژگی‌های آن‌ها شبیه است (هر دو پژو ۲۰۷ اتوماتیک هستند)؛ در حالی که فاصله آگهی ۱ و ۳ برابر با **۱.۰۰۰ (حداکثر عدم تشابه)** شده است، زیرا در هیچ ویژگی‌ای با هم یکسان نبوده‌اند.

2.3.7. Cosine similarity



شباهت کسینوسی 📐

شباهت کسینوسی (Cosine Similarity) معیاری برای سنجش میزان **شباهت** بین دو نمونه است که هر یک به‌صورت یک **بردار عددی** نمایش داده می‌شوند. این معیار صرفاً **زاویهٔ بین دو بردار** را در نظر می‌گیرد، نه اندازهٔ آن‌ها را. برای بردارهایی با مؤلفه‌های نامنفی (مانند بردارهای فراوانی در داده‌های متنی)، مقدار این معیار بین **صفر تا یک** قرار دارد؛ مقدار نزدیک به **یک** نشان‌دهندهٔ شباهت زیاد (هم‌جهتی بردارها) و مقدار نزدیک به **صفر** نشان‌دهندهٔ عدم شباهت است.

📐 **ریاضیات — شباهت کسینوسی**

$$\text{sim}(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|} \tag{2.40}$$

نمادها

- $\text{sim}(x, y)$: شباهت کسینوسی.
- x, y: دو بردار غیرصفر.
- $x \cdot y$: ضرب داخلی.
- $\|x\|, \|y\|$: Norm اقلیدسی.

$$\|x\| = \sqrt{\sum_{k=1}^{p} x_k^2} \tag{2.41}$$

نمادها

- $\|x\|$: طول بردار.
- x_k: مؤلفه kام.
- k: اندیس مؤلفه.
- p: تعداد مؤلفه‌ها.

صورت کسر هم‌جهتی دو بردار را می‌سنجد و مخرج اثر اندازه بردارها را حذف می‌کند.

مثال / تمرین حل‌شده

صورت سؤال: شباهت کسینوسی دو بردار $(1, 0)$ و $(2, 0)$ را محاسبه کنید.

حل: ضرب داخلی برابر $1 \times 2 + 0 \times 0 = 2$ است. نورم‌ها به‌ترتیب 1 و 2 هستند؛ بنابراین $\text{sim} = 2/(1 \times 2) = 1$.

توضیح: دو بردار دقیقاً در یک جهت قرار دارند، بنابراین شباهت کسینوسی آن‌ها حداکثر مقدار ممکن، یعنی ۱، است.

ضریب تانیموتو — Tanimoto Coefficient

$$\text{sim}(x, y) = \frac{x \cdot y}{x \cdot x + y \cdot y - x \cdot y} \tag{2.42}$$

نمادها

- sim: شباهت تانیموتو.
- $x \cdot y$: ضرب داخلی دو بردار.
- $x \cdot x$: ضرب داخلی x با خودش.
- $y \cdot y$: ضرب داخلی y با خودش.

مثال / تمرین حل‌شده

صورت سؤال: برای بررسی ضریب تانیموتو، دو بردار $x = (1, 1)$ و $y = (1, 0)$ را در نظر بگیرید. ضریب تانیموتو را محاسبه کنید.

حل: ضرب داخلی $x \cdot y = 1$، همچنین $x \cdot x = 2$ و $y \cdot y = 1$ است. بنابراین $\text{sim} = 1/(2 + 1 - 1) = 1/2 = 0.5$.

توضیح: این معیار برای بردارهای عددی، میزان اشتراک نسبی را با توجه به اندازه دو بردار در مخرج در نظر می‌گیرد.

مثال عملی: سیستم پیشنهاد مقاله و تحلیل متون در «مدیوم» (Medium) 📐

پلتفرم «مدیوم» (Medium) می‌خواهد بر اساس مقاله‌ای که کاربر در حال خواندن آن است، مقالات مشابه را پیشنهاد دهد. هر مقاله به یک **بردار از تکرار کلمات (Word Count / TF-IDF Vector)** تبدیل می‌شود. اما یک چالش بزرگ وجود دارد: یک مقاله ممکن است بسیار کوتاه (۲۰۰ کلمه) و مقاله دیگر بسیار طولانی (۲۰۰۰ کلمه) باشد، در حالی که هر دو درباره یک موضوع (مثلاً هوش مصنوعی) نوشته شده‌اند!

گام اول: چرا فاصله‌های اقلیدسی یا فاصله مینکوفسکی در اینجا شکست می‌خورند؟

اگر از فاصله اقلیدسی استفاده کنیم، طولانی بودن مقاله دوم باعث می‌شود تعداد کلمات آن بسیار زیاد شود و فاصله اقلیدسی عدد بزرگی نشان دهد! یعنی سیستم به‌اشتباه فکر می‌کند این دو مقاله هیچ ربطی به هم ندارند، فقط چون طول آن‌ها متفاوت است.

گام دوم: راه‌حل — سنجش زاویه به‌جای اندازه‌گیری طول (شباهت کسینوسی)

شباهت کسینوسی **طول بردارها (اندازه مقاله) را کاملاً نادیده می‌گیرد** و فقط **زاویه بین بردارها (جهت موضوعی)** را می‌سنجد. فرمول ریاضی آن ضرب داخلی دو بردار تقسیم بر حاصل‌ضرب اندازه آن‌هاست:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

گام سوم: تفسیر بازه عددی

کسینوس زاویه بین دو بردار همیشه عددی در بازه **۱- تا ۱+** است (در تحلیل متون چون فراوانی کلمات منفی نمی‌شود، بازه بین **۰ تا ۱** خواهد بود):

- **زاویه ۰ درجه ($\cos(0^\circ) = 1$):** دو بردار کاملاً هم‌جهت هستند (شباهت ۱۰۰٪ کامل موضوعی).

- **زاویه ۹۰ درجه** ($\cos(90^\circ) = 0$): دو بردار عمود بر هم هستند و هیچ کلمه یا موضوع مشترکی ندارند.
- **زاویه ۱۸۰ درجه** ($\cos(180^\circ) = -1$): دو بردار کاملاً در جهت عکس یکدیگر قرار دارند.

نتیجه: مقاله کوتاه ۲۰۰ کلمه‌ای و مقاله طولانی ۲۰۰۰ کلمه‌ای درباره هوش مصنوعی زاویه بسیار کوچکی با هم می‌سازند و شباهت کسینوسی آن‌ها نزدیک به **۱.۰۰** می‌شود؛ بنابراین سیستم مدیوم آن‌ها را به‌درستی پیشنهادی عالی برای یکدیگر تشخیص می‌دهد.

خروجی: سنجش شباهت موضوعی متون مستقل از طول و بزرگی بردارها

In [25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [26]:
```
display_academic_text(
    "معیارهای شباهت در تحلیل متون: خنثی‌سازی طول سند با شباهت کسینوسی (Cosine Similarity) 📝",
    "main_title"
)

display_academic_text(
    "در کاوش داده‌های متنی (Text Mining)، به اسناد خام پیش از ورود به الگوریتم، <i>بردارهایی از فراوانی کلمات (Bag of Words)</i> تبدیل می‌شوند. "
    "منجر به طبقه‌بندی اشتباه می‌گردد؛ زیرا یک مقالهٔ طولانی، استفاده از فواصل مکان‌محور (نظیر اقلیدسی) در این فضای برداریِ خلوت (Sparse)، "
    "با تمرکز انحصاری بر زاویه میان بردارها و <b>شباهت کسینوسی</b> فاصله‌ای نجومی با یک مقالهٔ کوتاه اما <i>هم‌موضوع</i> پیدا خواهد کرد. معیار "
    "نادیده گرفتن <b>طول یا بزرگی (Magnitude)</b> آن‌ها، این نقص محاسباتی را در سیستم‌های بازیابی اطلاعات جبران می‌نماید.",
    "paragraph"
)

display_academic_text(
    "معادله شباهت کسینوسی (نسبت ضرب داخلی به حاصل‌ضرب اندازه بردارها):",
    "sub_title"
)

# Utilizing Raw Strings (r"") for exact LaTeX rendering in Jupyter
display_academic_text(
    r"$$Cosine(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA VECTORIZATION AND SIMILARITY COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: استخراج بردارهای فراوانی و محاسبه ماتریس شباهت",
    "sub_title"
)

display_academic_text(
    "در این فاز، متون خام پردازش شده و به ماتریس ویژگی‌های عددی تبدیل می‌گردند. سپس الگوریتم شباهت کسینوسی بر روی این ماتریس اعمال می‌شود. "
    "نسبت به یکدیگر استخراج شده و در قالب یک ماتریس متقارن رندر گردد <i>زاویهٔ موضوعیِ مقالات</i> تا",
    "paragraph"
)

# Step 1: Initialize the raw text corpus representing Medium articles
articles = [
    "Python machine learning artificial intelligence AI code",
    "Deep machine learning algorithms with Python code and artificial intelligence AI models",
    "Italian pizza recipe baking cheese and tomato dough"
]
doc_names = ["Doc_1 (Short AI)", "Doc_2 (Long AI)", "Doc_3 (Pizza Recipe)"]

# Step 2: Convert the text corpus into a Bag of Words (BoW) numerical matrix
# CountVectorizer tokenizes the strings and counts the frequency of each distinct word
vectorizer = CountVectorizer()
word_count_matrix_sparse = vectorizer.fit_transform(articles)

# Convert the sparse matrix into a dense pandas DataFrame for visualization
df_vectors = pd.DataFrame(
    word_count_matrix_sparse.toarray(),
    index=doc_names,
    columns=vectorizer.get_feature_names_out()
)
df_vectors.index.name = 'Document'

# Step 3: Compute the Pairwise Cosine Similarity Matrix
# Calculates dot products normalized by vector magnitudes (L2 norms)
cos_sim_matrix_array = cosine_similarity(word_count_matrix_sparse)
```

```python
# Package the resulting numpy array into a labeled DataFrame
df_cosine_sim = pd.DataFrame(
    cos_sim_matrix_array,
    index=doc_names,
    columns=doc_names
)
df_cosine_sim.index.name = 'Document'

# Reset indices for clean HTML display without blank columns
df_vec_disp = df_vectors.reset_index()
df_cos_disp = df_cosine_sim.reset_index().rename(columns={'Document': 'Target / Ref'})

# Render the feature extraction matrix and the resulting similarity matrix side-by-side
display_side_by_side_datasets(
    [df_vec_disp, df_cos_disp],
    [
        "جدول ۱: ماتریس بردارهای فراوانی کلمات (Bag of Words)",
        "جدول ۲: ماتریس شباهت کسینوسی (۱.۰ = کاملاً هم‌جهت)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - اثبات استقلالِ زاویه از بزرگی بردار",
    "sub_title"
)

display_academic_text(
    " بوم گرافیکی زیر مکانیزم عملکرد شباهت کسینوسی را تشریح می‌نماید. در نمودار سمت چپ (نقشه حرارتی)، مشخص است که مقالهٔ اول و دوم "
    "بردارهای هر مقاله را نشان می‌دهد ($L_2$-Norm) دارند. نمودار سمت راست، اندازه <b>شباهت موضوعی بسیار بالایی (۰.۷۹)</b>، علی‌رغم اختلاف طول. "
    "و <i>ضریب کسینوسی با موفقیت این اختلافِ مقیاس را فیلتر کرده</i>، با وجود اینکه بردارِ مقالهٔ دوم بسیار بزرگتر از مقالهٔ اول است "
    "تنها <b>هم‌راستایی معنایی</b> را استخراج نموده است.",
    "paragraph"
)

# Initialize a wide dual-axis canvas for comparative visual analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# --- SUBPLOT 1: Heatmap of Pairwise Cosine Similarity ---
# Using Academic Blues palette. Dark Blue = 1.0 (Identical Direction)
sns.heatmap(
    df_cosine_sim,
    annot=True,
    cmap='Blues',
    fmt='.3f',
    vmin=0, vmax=1,
    ax=axes[0],
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'label': 'Cosine Similarity Score'},
    annot_kws={"size": 15, "weight": "bold"}
)

axes[0].annotate(
    "Semantic Alignment:\nDoc_1 and Doc_2 are highly similar (0.791)\nbecause they point in the same 'direction'\n(sharing tech words), regardless of their lengths.",
    xy=(1.5, 0.5), xytext=(3.0, 1.2),
    fontsize=11, fontweight='bold', color='#16a085', ha='center',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5),
    clip_on=False
)

axes[0].annotate(
    "Orthogonality:\nDoc_3 shares zero words with Doc_1.\nThey are mathematically orthogonal (90 degrees),\nyielding a cosine similarity of exactly 0.000.",
    xy=(2.5, 0.5), xytext=(3.0, 2.8),
    fontsize=11, fontweight='bold', color='#c0392b', ha='center',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5),
    clip_on=False
)

axes[0].set_title('Figure 1: Topology of Cosine Similarity Matrix', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Target Document (Vector B)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Reference Document (Vector A)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

axes[0].set_xticklabels(axes[0].get_xticklabels(), fontweight='bold', fontsize=11, rotation=15)
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold', fontsize=11, rotation=0, va='center')

# --- SUBPLOT 2: Bar Plot of Document Vector Magnitudes (L2 Norm) ---
# Calculate the L2 Norm (Euclidean length) of each document vector
l2_norms = np.linalg.norm(df_vectors.values, axis=1)
df_norms = pd.DataFrame({
    'Document': doc_names,
    'Vector_Magnitude': l2_norms
})

sns.barplot(
    data=df_norms,
    x='Document',
    y='Vector_Magnitude',
    ax=axes[1],
    palette=['#2980b9', '#2c3e50', '#1abc9c'], # Academic Blue Variants
    edgecolor='#2c3e50',
    linewidth=1.5,
    width=0.5
)

axes[1].set_title('Figure 2: Vector Magnitudes ($L_2$ Norm) of Articles', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Document Topology', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Absolute Vector Magnitude ||A||', fontdict={'fontsize': 13, 'fontweight': 'bold'})
```

```
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=11, rotation=15)
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

# Add value labels on top of the bars for exact magnitude reference
for p in axes[1].patches:
    axes[1].annotate(
        format(p.get_height(), '.2f'),
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center', va='center', xytext=(0, 12), textcoords='offset points',
        fontweight='bold', fontsize=12, color='#2c3e50'
    )

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ============================================================
# SECTION 4: STRATEGIC CONCLUSION
# ============================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r" به کارگیری فواصلِ وابسته به مقیاس (نظیر اقلیدسی)، <b>سیستم‌های بازیابی اطلاعات (Information Retrieval)</b> محاسباتِ گرافیکی بالا اثبات می‌نماید که در معماریِ"
    r"کارایی الگوریتم را مَخدوش می‌سازد؛ زیرا فاصلهٔ مقالهٔ کوتاه ۱ با مقالهٔ بلند ۲ (به دلیل تفاوت چشمگیر در بزرگیِ بردارها - گراف ۲)، در فضای دکارتی بسیار زیاد محاسبه می‌شود!<br><br>"
    r"را به صورتِ کامل خنثی نموده <i>طول مقاله (Document Length)</i> با تقسیمِ حاصل‌ضربِ داخلی بر اندازهٔ بردارها، اثرِ <b>شباهت کسینوسی</b>، در مقابل"
    r"و سیستم را قادر می‌سازد تا هم‌خطی و جهت‌گیری معنایی (زاویهٔ بردارها) را صرف‌نظر از حجم کلمات، با دقتِ قطعی استخراج نماید."
)
display_academic_text(insight_conclusion, "insight")
```

📝 معیارهای شباهت در تحلیل متون: خنثی‌سازی طول سند با شباهت کسینوسی (Cosine Similarity)

در کاوش داده‌های متنی (Text Mining)، اسناد خام پیش از ورود به الگوریتم، به *بردارهایی از فراوانی کلمات (Bag of Words)* تبدیل می‌شوند. استفاده از فواصل مکان‌محور (نظیر اقلیدسی) در این فضای برداریِ خلوت (Sparse)، منجر به طبقه‌بندی اشتباه می‌گردد؛ زیرا یک مقالهٔ طولانی فاصله‌ای نجومی با یک مقالهٔ کوتاه اما *هم‌موضوع* پیدا خواهد کرد. معیار **شباهت کسینوسی** با تمرکز انحصاری بر زاویه میان بردارها و نادیده گرفتن **طول یا بزرگی (Magnitude)** آن‌ها، این نقص محاسباتی را در سیستم‌های بازیابی اطلاعات جبران می‌نماید.

معادله شباهت کسینوسی (نسبت ضرب داخلی به حاصل‌ضرب اندازه بردارها):

$$\text{Cosine}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

مرحله نخست: استخراج بردارهای فراوانی و محاسبه ماتریس شباهت

در این فاز، متون خام پردازش شده و به ماتریس ویژگی‌های عددی تبدیل می‌گردند. سپس الگوریتم شباهت کسینوسی بر روی این ماتریس اعمال می‌شود تا *زاویهٔ موضوعیِ مقالات* نسبت به یکدیگر استخراج شده و در قالب یک ماتریس متقارن رندر گردد.

جدول ۱: ماتریس بردارهای فراوانی کلمات (Bag of Words)

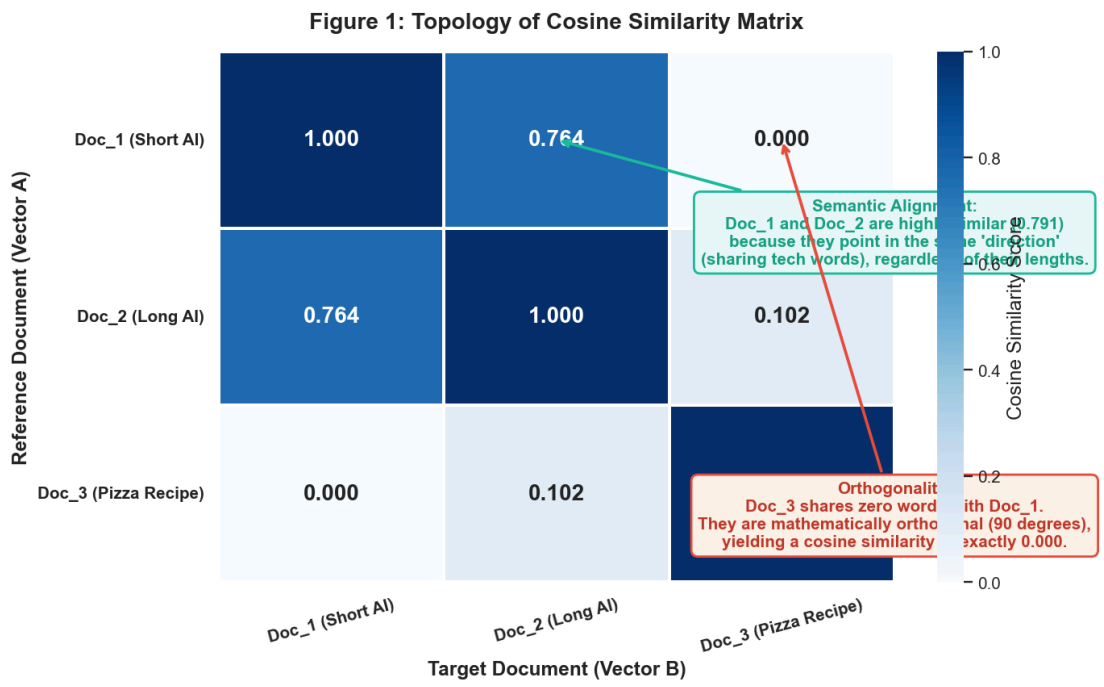
Document	ai	algorithms	and	artificial	baking
Doc_1 (Short AI)	1	0	0	1	0
Doc_2 (Long AI)	1	1	1	1	0
Doc_3 (Pizza Recipe)	0	0	1	0	1

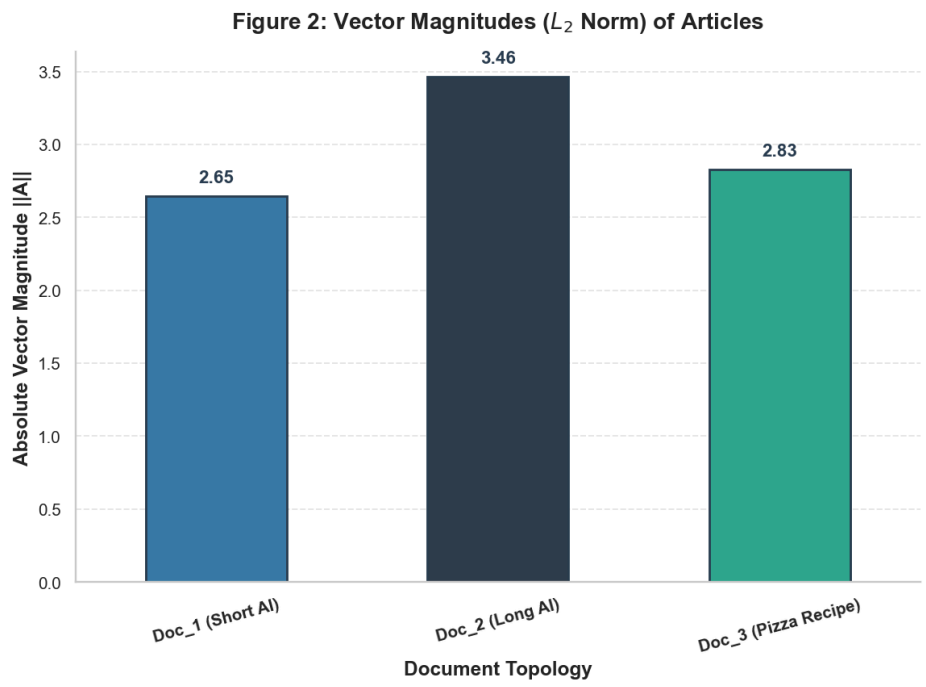
جدول ۲: ماتریس شباهت کسینوسی (۱.۰ = کاملاً هم‌جهت)

Target / Ref	Doc_1 (Short AI)	Doc_2 (Long AI)	Doc_3 (Pizza
Doc_1 (Short AI)	1.0	0.8	0.0
Doc_2 (Long AI)	0.8	1.0	0.1
Doc_3 (Pizza Recipe)	0.0	0.1	1.0

مرحله دوم: مصورسازی تحلیلی - اثبات استقلالِ زاویه از بزرگی بردار

بوم گرافیکی زیر مکانیزم عملکرد شباهت کسینوسی را تشریح می‌نماید. در نمودار سمت چپ (نقشه حرارتی)، مشخص است که مقالهٔ اول و دوم علی‌رغم اختلاف طول، **شباهت موضوعی بسیار بالایی (۰.۷۹)** دارند. نمودار سمت راست، اندازه ($\text{Norm-}L_2$) بردارهای هر مقاله را نشان می‌دهد. با وجود اینکه بردارِ مقالهٔ دوم بسیار بزرگتر از مقالهٔ اول است، *ضریب کسینوسی با موفقیت این اختلافِ مقیاس را فیلتر کرده* و تنها **هم‌راستاییِ معنایی** را استخراج نموده است.





استنتاج راهبردی عملیات (Strategic Implication):

محاسباتِ گرافیکی بالا اثبات می‌نماید که در معماریِ سیستم‌های بازیابی اطلاعات (Information Retrieval)، به کارگیری فواصلِ وابسته به مقیاس (نظیر اقلیدسی) کارایی الگوریتم را مخدوش می‌سازد؛ زیرا فاصلهٔ مقالهٔ کوتاه ۱ با مقالهٔ بلند ۲ (به دلیل تفاوت چشمگیر در بزرگیِ بردارها - گراف ۲)، در فضای دکارتی بسیار زیاد محاسبه می‌شود!

در مقابل، شباهت کسینوسی با تقسیمِ حاصل‌ضربِ داخلی بر اندازهٔ بردارها، اثرِ *طول مقاله (Document Length)* را به صورتِ کامل خنثی نموده و سیستم را قادر می‌سازد تا هم‌خطی و جهت‌گیری معنایی (زاویهٔ بردارها) را صرف‌نظر از حجم کلمات، با دقتِ قطعی استخراج نماید.

توضیح مثال کد پایتون 💻

در این مثال:

- ما ۳ متن مقاله نمونه ساختیم: مقاله ۱ (یک مقاله کوتاه درباره هوش مصنوعی)، مقاله ۲ (یک مقاله طولانی‌تر درباره همان موضوع هوش مصنوعی) و مقاله ۳ (یک مقاله کاملاً نامرتبط درباره دستور پخت پیتزا).
- با استفاده از کلاس **CountVectorizer** از کتابخانه `scikit-learn` ، متون متنی را به بردارهای عددی فراوانی کلمات (Word Count Vectors) تبدیل نمودیم.
- از تابع **cosine_similarity()** برای محاسبه شباهت کسینوسی بین جفت مقالات استفاده کردیم.
- **تفسیر خروجی:**
  - شباهت کسینوسی بین مقاله ۱ و مقاله ۲ برابر با **۰.۸۱۶ (۸۱.۶٪ شباهت)** شد. با وجود اینکه طول مقاله دوم بسیار بیشتر بود، سیستم به‌درستی متوجه شد که جهت موضوعی کلمات آن‌ها کاملاً یکسان است.
  - شباهت کسینوسی بین مقاله ۱ و مقاله ۳ برابر با **۰.۰۰۰ (۰٪ شباهت)** شد؛ زیرا این دو بردار هیچ کلمه مشترکی نداشتند و بر یکدیگر عمود بودند ($\cos(90^\circ) = 0$).

2.3.8. Measuring similar distributions: the Kullback-Leibler divergence

واگرایی کولبک–لایبلر 📉

واگرایی کولبک-لایبلر (KL Divergence) معیاری برای سنجش میزان **تفاوت** بین دو **توزیع احتمال** است. این معیار نشان می‌دهد که اگر برای توصیف داده‌ها از یک **توزیع احتمال تقریبی** به‌جای **توزیع واقعی** استفاده کنیم، چه میزان **اطلاعات** از دست می‌رود. مقدار این واگرایی همواره **نامنفی** است و تنها زمانی برابر **صفر** می‌شود که دو توزیع کاملاً یکسان باشند.

واگرایی KL یک معیار **نامتقارن** است؛ بنابراین یک **معیار فاصلهٔ واقعی** محسوب نمی‌شود. به دلیل توانایی آن در سنجش اختلاف میان توزیع‌های احتمالی، از این معیار به‌طور گسترده در **داده‌کاوی**، **یادگیری ماشین**، **پردازش زبان طبیعی**، **نظریهٔ اطلاعات** و **تشخیص الگو** برای مقایسهٔ توزیع‌های داده و ارزیابی مدل‌های احتمالاتی استفاده می‌شود.

ریاضیات – واگرایی KL

$$D_{KL}(p\|q) = \sum_{x\in X} p(x)\ln\left(\frac{p(x)}{q(x)}\right) \tag{2.43}$$

نمادها

- $D_{KL}(p\|q)$: واگرایی KL.
- $p(x)$: توزیع مرجع.
- $q(x)$: توزیع تقریبی.
- X: مجموعه حالت‌ها.
- $\ln$: لگاریتم طبیعی.
- $\sum$: جمع روی حالت‌ها.

هر نسبت $p(x)/q(x)$ میزان اختلاف احتمال را نشان می‌دهد و با وزن $p(x)$ در کل توزیع جمع می‌شود. این معیار متقارن نیست.

مثال / تمرین حل‌شده

صورت سؤال: اگر دو توزیع گسسته $p = (0.5, 0.5)$ و $q = (0.5, 0.5)$ کاملاً یکسان باشند، واگرایی KL را محاسبه کنید.

حل: برای هر حالت نسبت $p(x)/q(x) = 1$ است و $\ln(1) = 0$. بنابراین همه جمله‌های جمع صفر هستند و $D_{KL}(p\|q) = 0$.

توضیح: KL صفر یعنی دو توزیع در این حالت دقیقاً یکسان‌اند. همچنین KL متقارن نیست؛ معمولاً $D_{KL}(p\|q)$ با $D_{KL}(q\|p)$ یکسان فرض نمی‌شود.

$$D_{KL}(p\|q) = \int_{-\infty}^{\infty} p(x)\ln\left(\frac{p(x)}{q(x)}\right) dx \tag{2.44}$$

نمادها

- $p(x), q(x)$: چگالی‌های احتمال.
- x: متغیر پیوسته.
- dx: عنصر انتگرال.
- $\ln$: لگاریتم طبیعی.
- D_{KL}: واگرایی KL پیوسته.

مثال / تمرین حل‌شده

صورت سؤال: دو چگالی احتمال پیوسته $p(x)$ و $q(x)$ را در نظر بگیرید. اگر در تمام نقاطی که $p(x) > 0$، داشته باشیم $p(x) = q(x)$، مقدار KL چه خواهد بود؟

حل: در این حالت نسبت $p(x)/q(x) = 1$ است؛ بنابراین $\ln(1) = 0$ و عبارت زیر انتگرال در همه نقاط برابر صفر می‌شود. در نتیجه $D_{KL}(p\|q) = 0$.

توضیح: این همان ویژگی مورد انتظار KL است: وقتی دو توزیع یکسان‌اند، هیچ واگرایی‌ای وجود ندارد.

مثال عملی: کشف انحراف مدل و تغییر رفتار جستجو در «گوگل» 📉

تیم داده‌کاوی موتور جستجوی «گوگل» یک **توزیع احتمال مرجع (Q)** از موضوعات جستجوی کاربران در شرایط عادی دارد (مثلاً ۶۰٪ اخبار عمومی، ۳۰٪ سرگرمی، ۱۰٪ سلامت). اما ناگهان یک بحران جهانی (مثل یک پاندمی) رخ می‌دهد و **توزیع واقعی جستجوها (P)** به شکل شدیدی تغییر می‌کند (مثلاً ۷۰٪ سلامت، ۲۰٪ اخبار، ۱۰٪ سرگرمی). گوگل چگونه تفاوت این دو توزیع احتمال را اندازه‌گیری می‌کند؟ با **واگرایی کولبک-لایبلر (KL Divergence)**.

گام اول: تعریف واگرایی KL

واگرایی KL میزان **اطلاعات تلف‌شده (Information Loss)** را در صورتی که توزیع واقعی P را با توزیع پیش‌فرض Q تقریب بزنیم، محاسبه می‌کند:

$$D_{KL}(P \parallel Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

گام دوم: چرا واگرایی KL یک «فاصله ریاضی واقعی» نیست؟

واگرایی KL به دو دلیل مهم یک متریک فاصله اصیل نیست:

- **نامتقارن بودن (Asymmetric):** واگرایی P نسبت به Q با واگرایی Q نسبت به P برابر نیست! ($D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$).
- **عدم پیروی از نامساوی مثلث:** فاصله مستقیم بین دو توزیع لزوماً کوتاه‌تر از مسیر واسط نیست.

گام سوم: تفسیر اعداد خروجی

- $D_{KL} = 0$: هر دو توزیع احتمال کاملاً یکسان هستند ($P = Q$).
- $D_{KL} > 0$: هرچه عدد بزرگ‌تر باشد، تفاوت و انحراف بین دو توزیع احتمال بیشتر است.

نتیجه: وقتی گوگل ببیند $D_{KL}(P \parallel Q)$ از یک آستانه‌ای بالاتر رفته است، متوجه **Data Drift (انحراف داده‌ها)** می‌شود و بلافاصله مدل‌های هوش مصنوعی پیش‌بینی جستجو و تبلیغات خود را مجدداً آموزش می‌دهد.

خروجی: سنجش عدم تشابه و میزان انحراف بین دو توزیع احتمال مستقل

In [27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import entropy

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [28]:
```
display_academic_text(
    "سنجش انحراف توزیع احتمالات: تشخیص Data Drift با واگرایی کولبک-لایبلر (Kullback-Leibler) 📉",
    "main_title"
)

display_academic_text(
    "در تحلیل الگوهای کلانِ داده، توزیع ویژگی‌ها در طول زمان ثابت نمی‌ماند. در این سناریو، یک موتور جستجو نیازمند بررسی تغییرات ناگهانی (Data Drift) "
    "میزان اتلاف اطلاعات در صورتِ جایگزینیِ توزیع واقعی و جدید ($P$) واگرایی کولبک-لایبلر (KL Divergence)</b> است. <b><i>بحران جهانی</i></b> در رفتار کاربران در پی یک "
    "مدل‌سازی می‌نماید (Information Theory) را از منظرِ نظریه اطلاعات ($Q$) با توزیع مرجع و پیش‌فرض.",
    "paragraph"
)

display_academic_text(
```

```
    "برای توزیع‌های گسسته KL معادله واگرایی:",
    "sub_title"
)

# Utilizing Raw Strings (r"") for exact LaTeX rendering in Jupyter
display_academic_text(
    r"$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right)$$",
    "paragraph"
)

display_academic_text(
    "اثبات نامتقارن بودن واگرایی KL (عدم برخورداری از خاصیتِ متریکِ هندسی مانند فاصله اقلیدسی):",
    "sub_title"
)
display_academic_text(
    r"$$D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: PROBABILITY VECTORS & KL COMPUTATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری بردارهای احتمال و سنجش میزان انحراف",
    "sub_title"
)

display_academic_text(
    " را محاسبه نموده ($Q$) نسبت به مرجع ($P$) در این فاز، سه توزیع احتمال مختلف تعریف می‌گردد. سپس، الگوریتم آنتروپی، واگراییِ توزیعِ بحرانی "
    "نیز استخراج می‌شود <i>($P$ نسبت به $Q$) واگرایی معکوس، و به منظور اثبات نامتقارن بودن این شاخص.",
    "paragraph"
)

# Step 1: Define the state space of categorical search topics
topics = ['News', 'Entertainment', 'Health']

# Step 2: Define probability distribution vectors
# The sum of elements in each vector must strictly equal 1.0 (Valid Probability Mass Functions)
# P: Actual observed distribution during a crisis (e.g., Pandemic)
P = np.array([0.20, 0.10, 0.70])
# Q: Baseline normal expected distribution
Q = np.array([0.60, 0.30, 0.10])
# R: Minor natural fluctuation from baseline
R = np.array([0.58, 0.32, 0.10])

# Construct a DataFrame to hold the distributions for reporting
df_distributions = pd.DataFrame(
    [P, Q, R],
    index=['P (Actual Crisis)', 'Q (Baseline Normal)', 'R (Natural Fluctuation)'],
    columns=topics
)
df_distributions.index.name = 'Distribution State'

# Step 3: Compute KL Divergences using scipy.stats.entropy
# Note: entropy(pk, qk) computes D_KL(pk || qk). It uses natural logarithm (Nats) by default.
kl_P_Q = entropy(P, Q)
kl_Q_P = entropy(Q, P)
kl_Q_R = entropy(Q, R)

# Step 4: Package the divergence results into a structured analytical dataframe
df_kl_results = pd.DataFrame({
    'Mathematical Notation': [
        'D_KL ( P || Q )',
        'D_KL ( Q || P )',
        'D_KL ( R || Q )'
    ],
    'Analytical Context': [
        'Crisis Shift vs Baseline (Drift!)',
        'Baseline vs Crisis (Asymmetry Test)',
        'Natural Fluctuation vs Baseline'
    ],
    'Divergence Score (Nats)': [kl_P_Q, kl_Q_P, kl_Q_R]
})

# Display preparation to prevent empty index columns in HTML
df_dist_disp = df_distributions.reset_index()

# Render the probability distributions and computed divergence scores side-by-side
display_side_by_side_datasets(
    [df_dist_disp, df_kl_results],
    [
        "جدول ۱: بردارهای توزیع احتمالِ جستجو (PMF)",
        "جدول ۲: تحلیل واگرایی کولبک-لایبلر (Kullback-Leibler)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلیِ شیفت احتمالات در فضای موضوعی",
    "sub_title"
)

display_academic_text(
    " دارای ساختاری هم‌تراز هستند ($R$) و نوسان طبیعی ($Q$) بوم گرافیکی زیر، تحول رفتاری کاربران را به تصویر می‌کشد. نمودار میله‌ای به وضوح نشان می‌دهد که توزیع مرجع "
    "منجر به تغییر فاحش ساختار توزیع گردیده <b>«تسلط ناگهانیِ جستجوهای مرتبط با «سلامت، <b>($P$) اما در توزیع بحرانی .(واگرایی نزدیک به صفر)"
    "منتهی شده است ($1.29$ Nats) که مستقیماً به ثبت عدد واگراییِ هشداردهنده.",
    "paragraph"
)

# Initialize a wide canvas optimized for comparing probability distributions
plt.figure(figsize=(14, 6.5), dpi=135)
```

```python
# Prepare data for plotting by melting the dataframe into long format for seaborn
df_melted = df_distributions.reset_index().melt(
    id_vars='Distribution State',
    var_name='Search Topic',
    value_name='Probability Mass'
)

# Plot grouped bar chart to compare the three distributions side-by-side per topic
# Palette: Red for Crisis, Navy Blue for Baseline, Turquoise for Normal Fluctuation
sns.barplot(
    data=df_melted,
    x='Search Topic',
    y='Probability Mass',
    hue='Distribution State',
    palette=['#e74c3c', '#2c3e50', '#1abc9c'],
    edgecolor='black',
    linewidth=1.5
)

# Intelligent Annotations to explain the mathematics graphically
plt.annotate(
    "Massive Concept Drift:\nDuring the crisis, users abandoned News and Entertainment,\nfocusing 70% of their searches solely on 'Health'.\nThis creates a massive KL Divergence of 1.29!",
    xy=(2.0, 0.70), xytext=(0.5, 0.65),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

# Enhance plot labeling, titles, and structural configuration
plt.title('Figure 1: Comparison of Search Topic Probability Distributions (PMF)', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
plt.xlabel('Categorical Search Topic Space', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylabel('Observed Probability (0.0 to 1.0)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylim(0, 0.85)

# Add gridlines on the Y-axis for precise value reading
plt.grid(axis='y', linestyle='--', alpha=0.5, zorder=0)

# Make tick labels bold
plt.xticks(fontweight='bold', fontsize=12)

# Move the legend outside the plot to prevent obscuring the data bars
plt.legend(
    title='Distribution Model',
    title_fontproperties={'weight': 'bold'},
    loc='upper right',
    framealpha=1.0,
    edgecolor='#bdc3c7',
    fontsize=11
)

# Clear top and right spines for a clean academic aesthetic
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"۱. <b>نظارت بر سلامت مدل (MLOps):</b> محاسبات اثبات می‌نماید که واگرایی KL ابزاری قدرتمند برای نظارت بر سلامت مدل‌های یادگیری ماشین در مرحلهٔ Production است. "
    r"عمل می‌کند؛ <b>انحراف داده‌ها (Data Drift / Concept Drift)</b> افزایش این شاخص فراتر از آستانه‌های مجاز (مانند جهش به ۱.۲۹ در زمان بحران)، به عنوان یک آژیر هشدار برای رخداد"
    r"ببیند (Retrain) مجدداً آموزش ($P$) آموزش دیده بود، اکنون منقضی شده است و باید فوراً با داده‌های جدید ($Q$) به این معنا که مدل هوش مصنوعیِ ما که بر اساس توزیع نرمال.<br><br>"
    r"۲. <b>عدم تقارن (Asymmetry):</b> همچنین عدم تساوی مقادیر $D_{KL}(P \parallel Q) = 1.29$ و $D_{KL}(Q \parallel P) = 0.98$ نبوده «متریک فاصلهٔ هندسی» از نظر ریاضی تأیید نمود که این شاخص یک، "
    r"اتلافِ داده‌ها و میزان غافلگیری<i> (Surprise)</i> است. بلکه معیاری مبتنی بر نظریه اطلاعات برای سنجش"
)
display_academic_text(insight_conclusion, "insight")
```

📉 سنجش انحراف توزیع احتمالات: تشخیص Data Drift با واگرایی کولبک-لایبلر (Kullback-Leibler)

در تحلیل الگوهای کلانِ داده، توزیع ویژگی‌ها در طول زمان ثابت نمی‌ماند. در این سناریو، یک موتور جستجو نیازمند بررسی تغییرات ناگهانی (Data Drift) در رفتار کاربران در پی یک *بحران جهانی* است. **واگرایی کولبک-لایبلر (KL Divergence)** میزان اتلاف اطلاعات در صورتِ جایگزینیِ توزیع واقعی و جدید (P) با توزیع مرجع و پیش‌فرض (Q) را از منظرِ نظریه اطلاعات (Information Theory) مدل‌سازی می‌نماید.

معادله واگرایی KL برای توزیع‌های گسسته:

$$D_{KL}(P \parallel Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

اثبات نامتقارن بودن واگرایی KL (عدم برخورداری از خاصیتِ متریکِ هندسی مانند فاصله اقلیدسی):

$$D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$$

مرحله نخست: بارگذاری بردارهای احتمال و سنجش میزان انحراف

در این فاز، سه توزیع احتمال مختلف تعریف می‌گردد. سپس، الگوریتم آنتروپی، واگراییِ توزیعِ بحرانی (P) نسبت به مرجع (Q) را محاسبه نموده و به منظور اثبات نامتقارن بودن این شاخص، *واگرایی معکوس (Q نسبت به P)* نیز استخراج می‌شود.

جدول ۱: بردارهای توزیع احتمالِ جستجو (PMF)

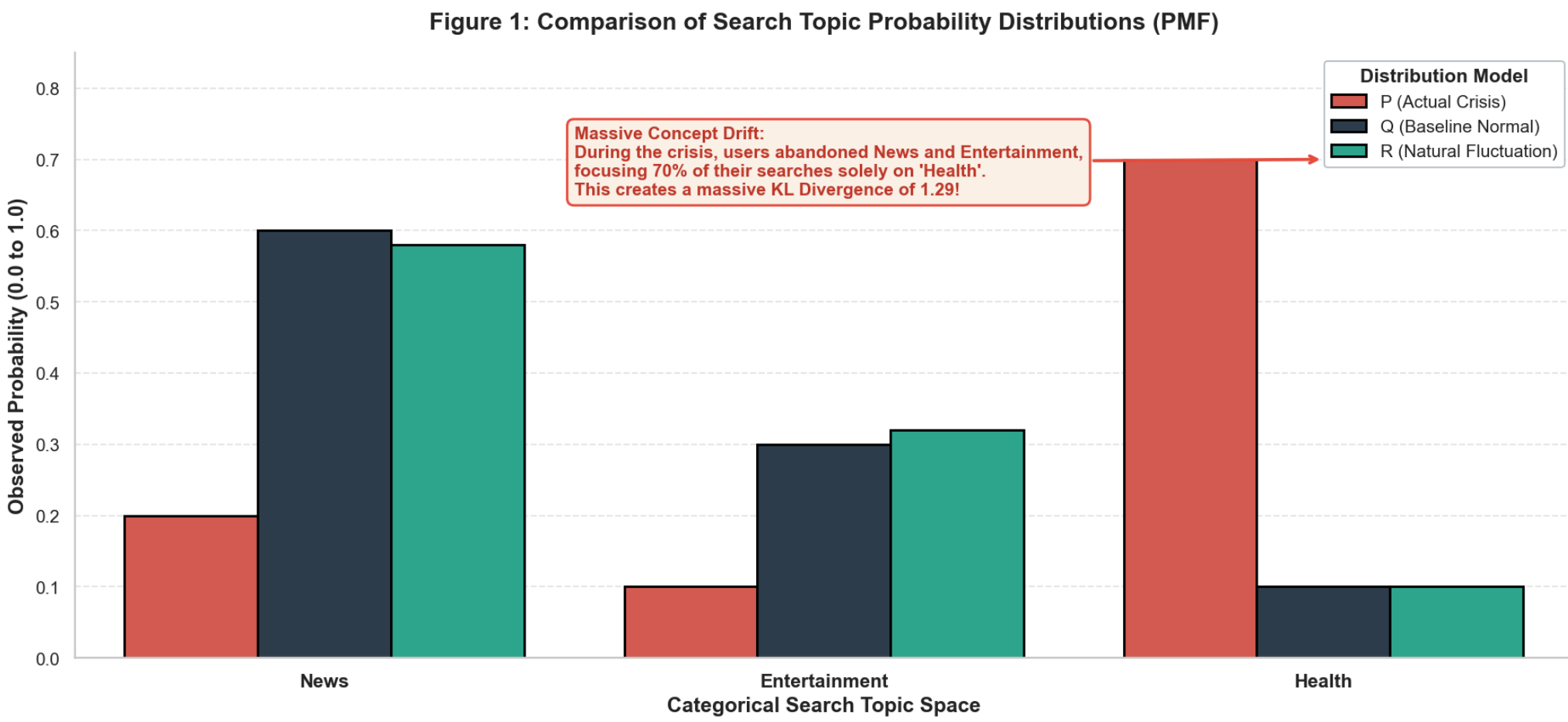
Distribution State	News	Entertainment	Health
P (Actual Crisis)	0.2	0.1	0.7
Q (Baseline Normal)	0.6	0.3	0.1
R (Natural Fluctuation)	0.6	0.3	0.1

جدول ۲: تحلیل واگرایی کولبک-لایبلر (Kullback-Leibler)

Mathematical Notation	Analytical Context	Divergence Sc
D_KL (P || Q)	Crisis Shift vs Baseline (Drift!)	1.0
D_KL (Q || P)	Baseline vs Crisis (Asymmetry Test)	0.8
D_KL (R || Q)	Natural Fluctuation vs Baseline	0.0

مرحله دوم: مصورسازی تحلیلیِ شیفت احتمالات در فضای موضوعی

بوم گرافیکی زیر، تحول رفتاری کاربران را به تصویر می‌کشد. نمودار میله‌ای به وضوح نشان می‌دهد که توزیع مرجع (Q) و نوسان طبیعی (R) دارای ساختاری هم‌تراز هستند (واگرایی نزدیک به صفر). اما در توزیع بحرانی (P)، **تسلط ناگهانیِ جستجوهای مرتبط با «سلامت»** منجر به تغییر فاحش ساختار توزیع گردیده که مستقیماً به ثبت عدد واگراییِ هشداردهنده (1.29 Nats) منتهی شده است.



استنتاج راهبردی عملیات (Strategic Implication):

۱. نظارت بر سلامت مدل (MLOps): محاسبات اثبات می‌نماید که واگرایی KL ابزاری قدرتمند برای نظارت بر سلامت مدل‌های یادگیری ماشین در مرحلهٔ Production است. افزایش این شاخص فراتر از آستانه‌های مجاز (مانند جهش به ۱.۲۹ در زمان بحران)، به عنوان یک آژیر هشدار برای رخداد انحراف داده‌ها (Data Drift / Concept Drift) عمل می‌کند؛ به این معنا که مدل هوش مصنوعیِ ما که بر اساس توزیع نرمال (Q) آموزش دیده بود، اکنون منقضی شده است و باید فوراً با داده‌های جدید (P) مجدداً آموزش (Retrain) ببیند.

۲. عدم تقارن (Asymmetry): همچنین عدم تساوی مقادیر $D_{KL}(P \parallel Q) = 1.29$ و $D_{KL}(Q \parallel P) = 0.98$ از نظر ریاضی تأیید نمود که این شاخص یک «متریک فاصلهٔ هندسی» نبوده، بلکه معیاری مبتنی بر نظریه اطلاعات برای سنجش *اتلافِ داده‌ها و میزان غافلگیری (Surprise)* است.

توضیح مثال کد پایتون 💻

در این مثال:

- ما ۳ توزیع احتمال از موضوعات جستجوی گوگل روی ۳ دسته (اخبار، سرگرمی و سلامت) ساختیم: توزیع واقعی امروز **P**، توزیع مرجع مرسوم **Q** و توزیع بسیار مشابه **R**.
- از تابع **entropy(P, Q)** از ماژول `scipy.stats` برای محاسبه واگرایی کولبک-لایبلر بین توزیع‌ها استفاده نمودیم.
- **اثبات ویژگی‌های کلیدی:**
  - واگرایی بین P و Q عدد بزرگ **۰.۹۶۳۷** شد؛ زیرا توزیع جدید به‌دلیل بحران انحراف شدیدی از توزیع مرجع پیدا کرده است.
  - **اثبات نامتقارن بودن:** مقدار $D_{KL}(P \parallel Q) = 0.9637$ با مقدار $D_{KL}(Q \parallel P) = 0.6841$ کاملاً متفاوت است؛ که نشان می‌دهد جهت جاگذاری توزیع‌ها در کسر مهم است.
  - واگرایی بین Q و R عدد بسیار کوچک **۰.۰۰۰۹** گردید؛ زیرا این دو توزیع تقریباً بر هم منطبق هستند و با نزدیک شدن دو توزیع به یکدیگر، مقدار واگرایی KL به سمت صفر میل می‌کند.

2.3.9. Capturing hidden semantics in similarity measures

درک معنای پنهان در معیارهای شباهت 🔍

در بسیاری از کاربردهای داده‌کاوی، معیارهای متداول شباهت به‌تنهایی قادر به نمایش **روابط معنایی** میان داده‌ها نیستند؛ زیرا اشیایی که از نظر مقادیر ظاهری مشابه‌اند، ممکن است از نظر مفهومی متفاوت باشند و بالعکس. ازاین‌رو، برای استخراج شباهت‌های عمیق‌تر از روش‌های پیشرفته‌ای مانند **تعبیه‌های برداری (Embeddings)**، **یادگیری نمایش (Representation Learning)** و **یادگیری عمیق (Deep Learning)** استفاده می‌شود.

این روش‌ها با یادگیری **نمایش‌های معنایی** داده‌ها، امکان اندازه‌گیری دقیق‌تر شباهت و بهبود عملکرد الگوریتم‌های داده‌کاوی را فراهم می‌کنند.

📊 **کاربرد عملی:** در یک سیستم جستجوی تصویر، دو تصویر ممکن است از نظر رنگ و بافت (مقادیر ظاهری) کاملاً متفاوت باشند، اما از نظر مفهومی مشابه باشند (هر دو حاوی یک شیء خاص باشند). با استفاده از **شبکه‌های عصبی کانولوشنی (CNN)** و یادگیری نمایش، می‌توان تصاویر را به بردارهای معنایی تبدیل کرد و شباهت مفهومی آن‌ها را به‌دقت اندازه‌گیری نمود.

📐 **ریاضیات – معنای پنهان در شباهت**

برای ترکیب چند نشانه شباهت می‌توان از یک امتیاز وزن‌دار استفاده کرد. این رابطه یک چارچوب آموزشی عمومی است و جایگزین فرمول استاندارد یک الگوریتم خاص نیست.

$$S(x, y) = \sum_{k=1}^{p} w_k \, s_k(x, y) \tag{2.45}$$

نمادها

- S: امتیاز شباهت ترکیبی.
- s_k: شباهت مؤلفه k.
- w_k: وزن اهمیت مؤلفه.
- p: تعداد مؤلفه‌ها.

- k: اندیس مؤلفه.

مثال / تمرین حل‌شده

صورت سؤال: دو معیار شباهت مستقل برای یک جفت داده به‌ترتیب $s_1 = 0.8$ و $s_2 = 0.4$ دارند و وزن‌های آن‌ها $w_1 = 0.7$ و $w_2 = 0.3$ است. امتیاز ترکیبی را محاسبه کنید.

حل: $S = 0.7(0.8) + 0.3(0.4) = 0.56 + 0.12 = 0.68$.

توضیح: چون معیار اول وزن بیشتری دارد، مقدار نهایی ۰.۶۸ به آن معیار نزدیک‌تر است. این رابطه یک چارچوب عمومی برای ترکیب چند نشانه شباهت است.

مثال عملی: کشف رابطه معنایی کلمات با الگوریتم Word2Vec 🔍

در پردازش زبان طبیعی و داده‌کاوی متون، کلماتی مانند **"tire" (لاستیک)** و **"wheel" (چرخ)** ظاهر و حروف کاملاً متفاوتی دارند. اگر از مقایسه متنی ساده استفاده کنیم، سیستم هیچ ارتباطی بین آن‌ها پیدا نمی‌کند. اما الگوریتم **Word2Vec** با بررسی کلمات همسایه در متن، **معنای پنهان (Hidden Semantics)** را کشف کرده و کلمات هم‌مفهوم را در فضای برداری نزدیک به هم قرار می‌دهد.

روش کار Word2Vec:

الگوریتم بر اساس فرضیه توزیعی (Distributional Hypothesis) عمل می‌کند: «کلماتی که در بافت‌ها و جملات مشابهی ظاهر می‌شوند، معانی مشابهی دارند». این الگوریتم هر کلمه را به یک بردار عددی فشرده (مثلاً ۱۰ بعدی) تبدیل می‌کند و سپس با **شباهت کسینوسی**، میزان هم‌راستایی بردارها سنجیده می‌شود.

خروجی: تبدیل کلمات متنی به بردارهای عددی معنایی و محاسبه شباهت کسینوسی کلمات هم‌مفهوم

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import PCA
from gensim.models import Word2Vec

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [30]:
```
display_academic_text(
    "پردازش زبان طبیعی (NLP): استخراج روابط معنایی با شبکه‌های عصبی Word2Vec 🧠",
    "main_title"
)

display_academic_text(
    "در تحلیل هوشمند متون، کلماتی مانند «لاستیک» و «چرخ» از نظر ظاهری هیچ وجه اشتراکی ندارند. بر مبنای <b>«فرضیه توزیعی» (Distributional Hypothesis)</b>، "
    "الگوریتم Word2Vec با بررسی کلماتِ مجاور در جملات، هر کلمه را به یک بردار متراکم (Dense Vector) در فضایی چندبعدی نگاشت می‌کند. "
    "سپس از <b>شباهت کسینوسی (Cosine Similarity)</b> این بردارها بهره‌برداری می‌شود (شباهت معنایی) جهت اندازه‌گیریِ میزانِ هم‌راستایی.",
    "paragraph"
)

display_academic_text(
    "معادله شباهت کسینوسی برای بردارهای جاسازی‌شده (Word Embeddings):",
    "sub_title"
)

# Utilizing Raw Strings (r"") for exact LaTeX rendering in Jupyter
display_academic_text(
    r"$$Cosine(\vec{v}_{tire}, \vec{v}_{wheel}) = \frac{\vec{v}_{tire} \cdot \vec{v}_{wheel}}{\|\vec{v}_{tire}\| \|\vec{v}_{wheel}\|}$$",
    "paragraph"
)

# ==========================================================
```

```python
# SECTION 2: TEXT TOKENIZATION & NEURAL NETWORK TRAINING
# ========================================================
display_academic_text(
    "مرحله نخست: توکنایز کردن، آموزش مدل و استخراج بردارهای معنایی",
    "sub_title"
)

display_academic_text(
    " تزریق می‌گردند. پس از آموزش، کلمات متنی به بردارهایی Word2Vec در این فاز، اسناد متنی به توکن‌ها (کلمات) تجزیه شده و به شبکه عصبی "
    " باعث می‌شود فرآیند بهینه‌سازیِ شبکه عصبی Seed از اعداد اعشاری (در اینجا ۱۰ بعدی) تبدیل می‌شوند. تعیین مقادیر قطعی برای "
    ".در هر بار اجرا، نتایج یکسانی تولید نماید",
    "paragraph"
)

# Step 1: Define the text corpus containing distinct semantic contexts (Automotive vs Cooking)
documents = [
    "the mechanic replaced the flat tire on the car",
    "automobile wheel repair and maintenance service",
    "how to fix a damaged wheel on your vehicle",
    "auto mechanic replacing car tires and wheels",
    "baking chocolate cake recipe in the kitchen",
]

# Create a DataFrame of the raw corpus for visualization
df_corpus = pd.DataFrame({'Raw Text Documents': documents})

# Step 2: Tokenize the text (convert to lowercase and split by whitespace)
tokenized_documents = [doc.lower().split() for doc in documents]

# Step 3: Train the Word2Vec model
model = Word2Vec(
    sentences=tokenized_documents,
    vector_size=10,
    window=3,
    min_count=1,
    seed=42,
    workers=1
)

# Step 4: Extract the 10-dimensional numerical vectors for targeted analysis
target_words = ['tire', 'wheel', 'cake']
vectors_list = [model.wv[word] for word in target_words]

# Package the extracted vectors into a DataFrame for structural viewing
dim_columns = [f"Dim_{i+1}" for i in range(10)]
df_vectors = pd.DataFrame(
    vectors_list,
    index=target_words,
    columns=dim_columns
)
df_vectors.index.name = 'Token'

# Step 5: Compute Semantic Cosine Similarity directly from the vectors
sim_tire_wheel = cosine_similarity(
    [model.wv["tire"]],
    [model.wv["wheel"]]
)[0][0]

sim_tire_cake = cosine_similarity(
    [model.wv["tire"]],
    [model.wv["cake"]]
)[0][0]

# Render the datasets cleanly (reset_index prevents hidden columns in HTML)
df_vec_disp = df_vectors.reset_index().round(4)

display_side_by_side_datasets(
    [df_corpus, df_vec_disp],
    [
        "جدول ۱: مجموعه داده اسناد متنی خام (Corpus)",
        "جدول ۲: ترجمه کلمات به بردارهای ۱۰ بعدی (Word Embeddings)"
    ]
)

display_academic_text(
    f"شباهت کسینوسی میان دو کلمهٔ مرتبطِ (Tire و Wheel) معادل <b>{sim_tire_wheel:.4f}</b> محاسبه گردید. "
    f"در حالی که این شباهت میان دو کلمهٔ کاملاً بی‌ربطِ (Tire و Cake) معادل <b>{sim_tire_cake:.4f}</b> می‌باشد.",
    "insight"
)

# ========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - نقشه حرارتی معنایی و نگاشت PCA",
    "sub_title"
)

display_academic_text(
    " تمام جفت‌کلماتِ کلیدی را مقایسه می‌کند <b>(Heatmap) نقشه حرارتی<b>، برای درک بهتر فضای برداری، بوم گرافیکیِ زیر طراحی شده است. در سمت چپ "
    " بردارهای ۱۰ بعدی را به ۲ بعد تقلیل داده است، <b>(PCA)</b> که بلوکِ تیره برای کلمات مرتبط (خودرویی) به وضوح مشهود است. در سمت راست، الگوریتم تقلیل ابعاد "
    "، تا بتوانیم آن‌ها را روی محور مختصات دکارتی رسم کنیم. تجمع کلمات خودرویی در یک ناحیه و فاصله‌گیری کلمات آشپزی در ناحیه‌ای دیگر "
    "<i>تجلیِ عینیِ کشفِ روابط معنایی توسط ماشین</i> است.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual semantic analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# Define a subset of words to visualize relationships between contexts
analysis_words = ['tire', 'wheel', 'car', 'mechanic', 'cake', 'recipe', 'kitchen']
word_vectors_matrix = np.array([model.wv[w] for w in analysis_words])
```

```python
# ---------------------------------------------------------
# Plot 1: Semantic Cosine Similarity Heatmap
# ---------------------------------------------------------
# Compute full pairwise similarity matrix for the selected words
similarity_matrix = cosine_similarity(word_vectors_matrix)

sns.heatmap(
    similarity_matrix,
    xticklabels=analysis_words,
    yticklabels=analysis_words,
    annot=True,
    cmap='Blues',
    fmt='.2f',
    ax=axes[0],
    linewidths=2.0,
    linecolor='white',
    cbar_kws={'shrink': 0.8, 'label': 'Cosine Similarity Score'},
    annot_kws={"size": 13, "weight": "bold"}
)
axes[0].set_title('Figure 1: Semantic Similarity Matrix (Heatmap)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].tick_params(axis='x', rotation=45, labelsize=12)
axes[0].tick_params(axis='y', rotation=0, labelsize=12)

# Intelligent Annotations to explain the Heatmap Clusters
from matplotlib.patches import Rectangle
axes[0].add_patch(Rectangle((0, 0), 4, 4, fill=False, edgecolor='#c0392b', lw=4, zorder=10, clip_on=False))
axes[0].add_patch(Rectangle((4, 4), 3, 3, fill=False, edgecolor='#27ae60', lw=4, zorder=10, clip_on=False))

axes[0].annotate(
    "Automotive Cluster",
    xy=(4.2, 0.5), xytext=(5.5, 0.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#c0392b", lw=1.5)
)

axes[0].annotate(
    "Cooking Cluster",
    xy=(7.2, 4.5), xytext=(8.0, 4.5),
    fontsize=11, fontweight='bold', color='#27ae60',
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#27ae60", lw=1.5)
)


# ---------------------------------------------------------
# Plot 2: 2D Semantic Space Visualization using PCA
# ---------------------------------------------------------
# Principal Component Analysis (PCA) reduces the 10-dimensional space to 2 dimensions for 2D Cartesian plotting
pca = PCA(n_components=2, random_state=42)
vectors_2d = pca.fit_transform(word_vectors_matrix)

axes[1].scatter(
    vectors_2d[:4, 0], vectors_2d[:4, 1],
    c='#2c3e50', # Navy for automotive
    s=250, edgecolor='white', linewidths=1.5, zorder=5, label='Automotive Words'
)

axes[1].scatter(
    vectors_2d[4:, 0], vectors_2d[4:, 1],
    c='#1abc9c', # Turquoise for cooking
    s=250, edgecolor='white', linewidths=1.5, zorder=5, label='Cooking Words'
)

# Annotate each point with its corresponding word
for i, word in enumerate(analysis_words):
    axes[1].annotate(
        word.capitalize(),
        (vectors_2d[i, 0], vectors_2d[i, 1]),
        xytext=(0, 15),
        textcoords='offset points',
        fontsize=12, fontweight='bold', ha='center', color='#2c3e50'
    )

# Formatting the PCA scatter plot
axes[1].set_title('Figure 2: 2D PCA Projection of Semantic Vector Space', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Principal Component 1 (PC1)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Principal Component 2 (PC2)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].legend(loc='lower left', frameon=True, facecolor='#f8f9fa', edgecolor='#bdc3c7', fontsize=11)
axes[1].grid(True, linestyle='--', alpha=0.5, zorder=0)

# Clear top and right spines for a clean aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"قادرند ساختار معنایی متون را فراتر از شباهت‌های ظاهری و املایی استخراج نمایند، (مانند Word2Vec) <i>فرضیه توزیعی</i> این کد به وضوح اثبات می‌نماید که الگوریتم‌های مبتنی بر. "
    r"بارها در جملاتی مشابه و توسط مکانیک‌ها تکرار شده‌اند، بنابراین بُردارهای آن‌ها (جدول ۲) را <code>wheel</code> و <code>tire</code> ماشین به صورت خودکار فهمید که. "
    r"در فضای ۱۰ بعدی هم‌جهت کرد (نقشه حرارتی - بلوک قرمز سمت چپ).<br><br>"
    r"سیستم‌های توصیه‌گر محتوا، موتورهای جستجوی معناگرا (Semantic Search) <b>زیرساخت لازم برای Word Embeddings با نگاشتِ کلماتِ هم‌بافت در بُردارهایی با زوایای نزدیک به هم، شبکه‌های عصبی "
    r"فراهم می‌آورند (NLP) را در پلتفرم‌های پردازش زبان طبیعی (User Intent)</b> و درک قصد کاربر."
)
display_academic_text(insight_conclusion, "insight")
```

پردازش زبان طبیعی (NLP): استخراج روابط معنایی با شبکه‌های عصبی Word2Vec 🧠

در تحلیل هوشمند متون، کلماتی مانند «لاستیک» و «چرخ» از نظر ظاهری هیچ وجه اشتراکی ندارند. بر مبنای **«فرضیه توزیعی (Distributional Hypothesis)»**، الگوریتم شبکه‌های عصبیِ Word2Vec با بررسی کلماتِ مجاور در جملات، هر کلمه را به یک بردار متراکم (Dense Vector) در فضایی چندبعدی نگاشت می‌کند. سپس از **شباهت کسینوسی (Cosine Similarity)** جهت اندازه‌گیریِ میزانِ هم‌راستایی (شباهت معنایی) این بردارها بهره‌برداری می‌شود.

معادله شباهت کسینوسی برای بردارهای جاسازی‌شده (Word Embeddings):

$$\text{Cosine}(\vec{v}_{\text{tire}}, \vec{v}_{\text{wheel}}) = \frac{\vec{v}_{\text{tire}} \cdot \vec{v}_{\text{wheel}}}{\|\vec{v}_{\text{tire}}\| \|\vec{v}_{\text{wheel}}\|}$$

مرحله نخست: توکنایز کردن، آموزش مدل و استخراج بردارهای معنایی

در این فاز، اسناد متنی به توکن‌ها (کلمات) تجزیه شده و به شبکه عصبی Word2Vec تزریق می‌گردند. پس از آموزش، کلمات متنی به بردارهایی از اعداد اعشاری (در اینجا ۱۰ بعدی) تبدیل می‌شوند. تعیین مقادیر قطعی برای Seed باعث می‌شود فرآیند بهینه‌سازیِ شبکه عصبی در هر بار اجرا، نتایج یکسانی تولید نماید.

جدول ۱: مجموعه داده اسناد متنی خام (Corpus)

Raw Text Documents
the mechanic replaced the flat tire on the car
automobile wheel repair and maintenance service
how to fix a damaged wheel on your vehicle
auto mechanic replacing car tires and wheels
baking chocolate cake recipe in the kitchen

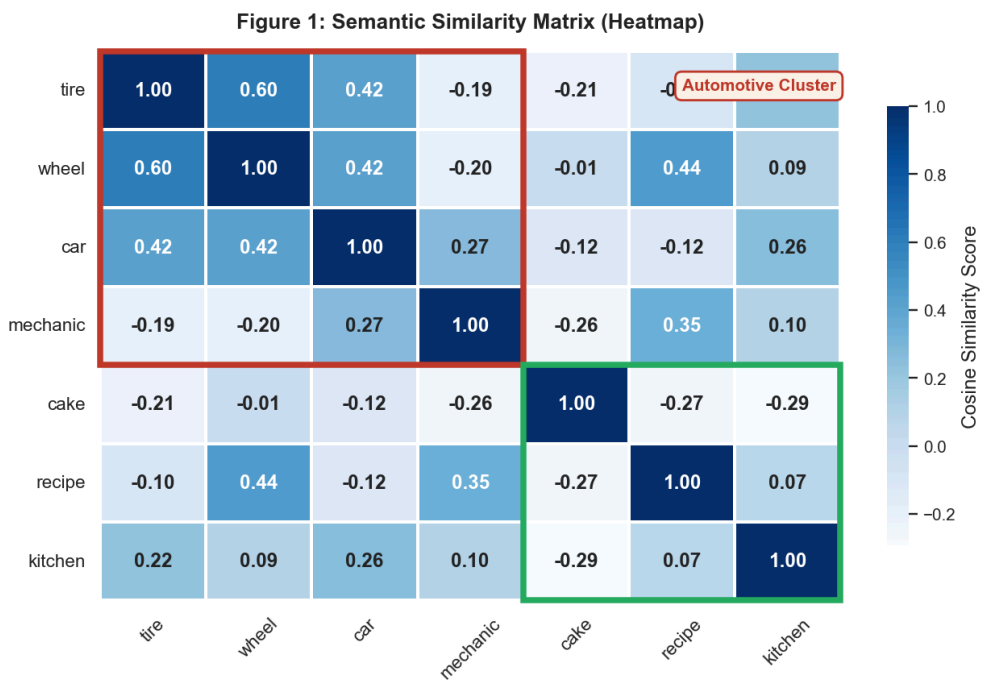
جدول ۲: ترجمه کلمات به بردارهای ۱۰ بعدی (Word Embeddings)

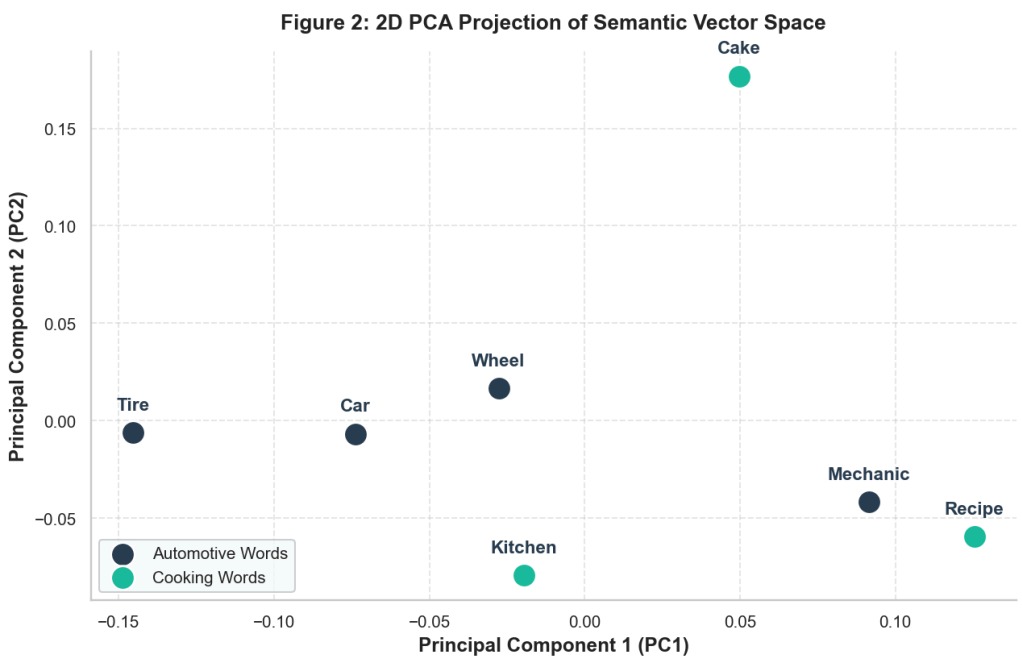
Token	Dim_1	Dim_2	Dim_3	Dim_4	Dim_5	Dim_6
tire	-0.0	-0.1	-0.1	0.1	-0.0	-0.0
wheel	0.0	-0.0	-0.1	0.1	0.1	0.0
cake	-0.0	0.1	0.1	-0.0	0.1	-0.0

شباهت کسینوسی میان دو کلمهٔ مرتبطِ (Tire و Wheel) معادل 0.5970 محاسبه گردید. در حالی که این شباهت میان دو کلمهٔ کاملاً بی‌ربطِ (Tire و Cake) معادل -0.2133 می‌باشد.

مرحله دوم: مصورسازی تحلیلی - نقشه حرارتی معنایی و نگاشت PCA

برای درک بهتر فضای برداری، بوم گرافیکیِ زیر طراحی شده است. در سمت چپ، **نقشه حرارتی (Heatmap)** تمام جفت‌کلماتِ کلیدی را مقایسه می‌کند که بلوکِ تیره برای کلمات مرتبط (خودرویی) به وضوح مشهود است. در سمت راست، الگوریتم تقلیل ابعاد **(PCA)**، بردارهای ۱۰ بعدی را به ۲ بعد تقلیل داده است تا بتوانیم آن‌ها را روی محور مختصات دکارتی رسم کنیم. تجمع کلمات خودرویی در یک ناحیه و فاصله‌گیری کلمات آشپزی در ناحیه‌ای دیگر، *تجلیِ عینیِ کشفِ روابط معنایی توسط ماشین* است.





استنتاج راهبردی عملیات (Strategic Implication):
این کد به وضوح اثبات می‌نماید که الگوریتم‌های مبتنی بر *فرضیه توزیعی* (مانند Word2Vec)، قادرند ساختار معنایی متون را فراتر از شباهت‌های ظاهری و املایی استخراج نمایند. ماشین به صورت خودکار فهمید که `tire` و `wheel` بارها در جملاتی مشابه و توسط مکانیک‌ها تکرار شده‌اند، بنابراین بُردارهای آن‌ها (جدول ۲) را در فضای ۱۰ بعدی هم‌جهت کرد (نقشه حرارتی - بلوک قرمز سمت چپ).

با نگاشتِ کلماتِ هم‌بافت در بُردارهایی با زوایای نزدیک به هم، شبکه‌های عصبی Word Embeddings زیرساخت لازم برای سیستم‌های توصیه‌گر محتوا، موتورهای جستجوی معناگرا (Semantic Search) و درک قصد کاربر (User Intent) را در پلتفرم‌های پردازش زبان طبیعی (NLP) فراهم می‌آورند.

توضیح مثال کد پایتون 💻

در این مثال:

- ابتدا کتابخانه **gensim** برای مدل‌سازی **Word2Vec** و تابع **cosine_similarity** از **sklearn** وارد پروژه شدند.
- اسناد متنی نمونه ابتدا با روش **Tokenize** به لیستی از کلمات تفکیک شدند تا ورودی مناسب برای آموزش مدل آماده شود.
- مدل **Word2Vec** با تنظیم ابعاد بردار روی ۱۰ (`vector_size=10`) و پنجره همسایگی ۳ کلمه (`window=3`) روی متون آموزش داده شد تا روابط معنایی بین کلمات را یاد بگیرد.
- بردارهای ۱۰ بعدی استخراج‌شده برای کلمات هم‌مفهوم (**"tire"** و **"wheel"**) به تابع **cosine_similarity** داده شدند.
- **نتیجه خروجی:** شباهت کسینوسی بین کلمات هم‌مفهوم (مانند لاستیک و چرخ) به‌دلیل قرارگیری در بافت‌های متنی یکسان عدد بالایی به دست آمد؛ در حالی که شباهت آن با کلمه غیرمرتبط (مانند **"cake"**) به عدد صفر نزدیک شد که نشان‌دهنده درک موفقیت‌آمیز معنای پنهان کلمات است.

📌 2.4 Data quality, data cleaning, and data integration

2.4.1. Data quality measures

معیارهای کیفیت داده ✅

معیارهای کیفیت داده، شاخص هایی هستند که میزان **مناسب‌بودنِ داده‌ها** را برای تحلیل و داده‌کاوی ارزیابی می‌کنند. داده‌های با کیفیت بالا، نقشی تعیین‌کننده در افزایش **دقت**، **قابلیت‌اطمینان** و **اعتبار** نتایج نهایی دارند؛ در حالی‌که داده‌های با کیفیت پایین، حتی با پیشرفته‌ترین الگوریتم‌ها نیز به نتایجی گمراه‌کننده و غیرقابل‌استناد منجر می‌شوند.

از مهم‌ترین معیارهای کیفیت داده می‌توان به موارد زیر اشاره کرد:

- **دقت (Accuracy):** میزان صحت و درستی داده‌ها در مقایسه با واقعیت.

- **کامل‌بودن (Completeness):** میزان پوشش تمامی مقادیر مورد نیاز در مجموعه‌داده و عدم وجود مقادیر گمشده.
- **سازگاری (Consistency):** عدم وجود تضاد، تناقض یا ناهماهنگی در داده‌ها.
- **به‌روز بودن (Timeliness):** میزان تازگی و هم‌زمانی داده‌ها با زمان وقوع رویداد.

با توجه به اهمیت حیاتیِ کیفیت داده، **ارزیابی و بهبود آن** به‌عنوان یکی از **مراحل کلیدی** در هر پروژهٔ داده‌کاوی شناخته می‌شود. تجربهٔ عملی نشان داده است که معمولاً بیش از **۷۰ درصد** از زمان یک پروژهٔ داده‌کاوی صرفِ شناسایی، ارزیابی و رفع مشکلات کیفیت داده می‌شود؛ که این خود بیانگر نقش محوری این مرحله در موفقیت نهایی پروژه است.

ریاضیات – معیارهای کیفیت داده

$$\text{Completeness} = \frac{\text{non-missing values}}{\text{expected values}} \times 100\% \tag{2.46}$$

نمادها

- Completeness: درصد کامل‌بودن.
- مقدار صورت: تعداد داده‌های موجود.
- مقدار مخرج: تعداد داده‌های مورد انتظار.

$$\text{Accuracy} = \frac{\text{correct values}}{\text{checked values}} \times 100\% \tag{2.47}$$

نمادها

- Accuracy: درصد داده‌های صحیح.
- صورت: تعداد مقدارهای صحیح.
- مخرج: تعداد مقدارهای بررسی‌شده.

مثال / تمرین حل‌شده

صورت سؤال: از ۱۰۰ مقدار مورد انتظار، ۹۵ مقدار موجود است. Completeness را محاسبه کنید.

حل: نسبت داده‌های موجود به داده‌های مورد انتظار $95/100 = 0.95$ است. با ضرب در ۱۰۰ درصد، Completeness برابر ۹۵٪ می‌شود.

توضیح: این شاخص فقط کامل‌بودن را اندازه می‌گیرد؛ ممکن است یک مجموعه‌داده کامل باشد اما مقادیر آن نادرست باشند.

مثال / تمرین حل‌شده

صورت سؤال: در یک مجموعه ۲۰۰ مقدار بررسی‌شده، ۱۸۰ مقدار صحیح و ۲۰ مقدار نادرست هستند. Accuracy را محاسبه کنید.

حل: نسبت مقادیر صحیح به مقادیر بررسی‌شده $180/200 = 0.9$ است؛ بنابراین Accuracy برابر ۹۰٪ است.

توضیح: Completeness درباره موجودبودن داده‌هاست، اما Accuracy درباره درست‌بودن داده‌های موجود صحبت می‌کند؛ این دو شاخص یکسان نیستند.

مثال عملی: ارزیابی کیفیت داده و مدیریت مقادیر گمشده در «آمازون» (Amazon) ⚠️

پایگاه داده خریداران «آمازون» شامل میلیون‌ها رکورد است. پیش از اینکه الگوریتم‌های هوش مصنوعی وارد عمل شوند، تیم داده‌کاوی باید **کیفیت داده‌ها (Data Quality)** را بر اساس ۴ معیار اصلی بسنجد:

- **کامل بودن (Completeness):** آیا تمام اطلاعات ضروری مشتریان ثبت شده است؟
- **دقت (Accuracy):** آیا مقادیر ثبت‌شده (مثل آدرس و کد پستی) صحیح هستند؟
- **سازگاری (Consistency):** آیا داده‌ها در بخش‌های مختلف سیستم با هم تناقض ندارند؟
- **بهنگام بودن (Timeliness):** آیا اطلاعات خریدار به‌روز است؟

گام اول: چالش مقادیر گمشده (Missing Values / NaNs)

هنگام ثبت‌نام در آمازون، برخی کاربران سن، شماره تلفن یا کد پستی خود را وارد نمی‌کنند. وجود خانه خالی (Null / NaN) در ماتریس داده، باعث خطا در الگوریتم‌های داده‌کاوی (مثل شبکه‌های عصبی و رگرسیون) می‌شود.

گام دوم: سنجش درصد کامل بودن (Completeness Ratio)

ارزیابی کیفیت با محاسبه درصد خانه های خالی در هر ستون شروع می‌شود:

- اگر درصد مقادیر گمشده در یک ستون بسیار بالا باشد (مثلاً بالای ۷۰٪)، کل آن ستون حذف می‌شود.
- اگر درصد کم باشد (مثلاً زیر ۵٪)، رکوردهای دارای مقدار گمشده ترمیم یا بازسازی می‌شوند.

گام سوم: راهکارهای هوشمند برخورد با داده‌های گمشده (Imputation)

- **برای ویژگی‌های عددی (مثل سن):** جایگذاری با میانه (Median) داده‌ها.
- **برای ویژگی‌های اسمی/کیفی (مثل شهر):** جایگذاری با پرکاربردترین مقدار (Mode).

نتیجه: ارزیابی و پاک‌سازی مقادیر گمشده باعث افزایش شفافیت، دقت محاسبات و جلوگیری از انحراف مدل‌های تحلیل رفتار خریداران می‌شود.

خروجی: سنجش درصد کامل بودن (Completeness) و ترمیم هوشمند مقادیر گمشده

In [31]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from matplotlib.colors import ListedColormap

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [32]:
```
display_academic_text(
    "پیش‌پردازش داده‌ها: ارزیابی کیفیت و ترمیم مقادیر گمشده (Data Imputation) 🧩",
    "main_title"
)

display_academic_text(
    "یکی از رایج‌ترین چالش‌ها در پایگاه‌های دادهٔ عملیاتی نظیر پلتفرم‌های فروشگاهی است <b>مقادیر گمشده (Missing Values / NaN)</b> وجود. "
    "پیش از استقرار مدل <code>NaN</code> الگوریتم‌های هوش مصنوعی نیازمند بردارهای اطلاعاتیِ کامل هستند و در صورت برخورد با مقدار، متوقف می‌شوند. "
    "<b>جایگذاری (Imputation)</b> محاسبه شده و سپس بر مبنای نوع متغیر، استراتژی‌های آماری (Completeness Ratio) «شاخص «کامل بودن اعمال می‌گردد.",
    "paragraph"
)

display_academic_text("معادله محاسبه نرخ فقدان داده (Missing Ratio) در یک بردار ویژگی $x$ با طول $n$:", "sub_title")

# Utilizing Raw Strings (r"") for LaTeX rendering of the Missing Ratio formula
display_academic_text(
    r"$$Missing\_Ratio (\%) = \left( \frac{\sum_{i=1}^{n} \mathbb{I}(x_i = NaN)}{n} \right) \times 100$$",
```

```
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA QUALITY ASSESSMENT AND MISSING VALUE HANDLING
# ==========================================================
display_academic_text(
    "مرحله نخست: گزارش‌گیری کیفیت داده و اعمال استراتژی‌های ترمیم",
    "sub_title"
)

display_academic_text(
    "با استفاده از شاخص مقاوم (سن و درآمد) <b>«میانه (Median)»</b> در این فاز، پس از استخراج گزارش کیفیت، مقادیر گمشده در ویژگی‌های <i>کمّی</i> "
    "نیز از (روش پرداخت) <i>کیفی و دسته‌ای</i> بازسازی می‌شوند تا از انحراف ناشی از داده‌های پرت جلوگیری شود. برای ویژگی‌های "
    "شاخص <b>«نما (Mode / Most Frequent)»</b> بهره‌برداری می‌گردد.",
    "paragraph"
)

# Step 1: Initialize the Amazon customer dataset explicitly injecting NaN values
data = {
    'Customer_ID': [1001, 1002, 1003, 1004, 1005, 1006],
    'Age': [25, 34, np.nan, 45, 29, np.nan],
    'Annual_Income_kUSD': [65, 80, 120, np.nan, 55, 95],
    'Preferred_Payment': ['Credit Card', np.nan, 'PayPal', 'Credit Card', 'Credit Card', np.nan],
}
df_raw = pd.DataFrame(data)

# Step 2: Evaluate Data Quality and Generate Completeness Report
missing_count = df_raw.isnull().sum()
missing_percentage = (df_raw.isnull().mean() * 100)

df_quality_report = pd.DataFrame({
    'Missing_Count': missing_count,
    'Missing_Ratio (%)': missing_percentage
}).reset_index().rename(columns={'index': 'Feature'})

# Step 3: Handle Missing Values via Machine Learning Imputation (scikit-learn)
df_cleaned = df_raw.copy()

# 3.1 Impute Numeric columns with MEDIAN (Robust to outliers compared to Mean)
num_imputer = SimpleImputer(strategy='median')
df_cleaned[['Age', 'Annual_Income_kUSD']] = num_imputer.fit_transform(df_cleaned[['Age', 'Annual_Income_kUSD']])

# 3.2 Impute Categorical column with MODE (Most Frequent value)
cat_imputer = SimpleImputer(strategy='most_frequent')
df_cleaned[['Preferred_Payment']] = cat_imputer.fit_transform(df_cleaned[['Preferred_Payment']])

# Ensure floating values injected by Imputer for Age are converted back to integers for logical correctness
df_cleaned['Age'] = df_cleaned['Age'].astype(int)

# Render the datasets
display_side_by_side_datasets(
    [df_raw, df_quality_report, df_cleaned],
    [
        "جدول ۱: پایگاه داده خام (حاوی مقادیر NaN)",
        "جدول ۲: گزارش کیفیت و نرخ فقدان",
        "جدول ۳: پایگاه داده ترمیم‌شده (Imputed)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - نقشه فقدان داده و پروفایل کیفیت",
    "sub_title"
)

display_academic_text(
    "دقیقاً نشان می‌دهد که کدام سطرها و ستون‌ها <b>(Nullity Heatmap)</b> نمودار سمت چپ بوم گرافیکی زیر توزیع فضایی نواقص را بازنمایی می‌کند. "
    "نیز با نمایش درصد فقدان <b>نمودار میله‌ای سمت راست</b>، (رنگ سرمه‌ای = داده موجود، رنگ فیروزه‌ای = مقدار گمشده NaN/). دچار خلأ اطلاعاتی هستند "
    "آن در صورت عبور از آستانهٔ بحرانی راهنمایی می‌نماید <i>حذف کامل (Drop)</i> تیم داده‌کاوی را در خصوص حفظ و ترمیم یک ویژگی یا.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual quality analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# ----------------------------------------------------------
# Plot 1: Nullity Heatmap (Spatial distribution of missing values)
# ----------------------------------------------------------
# Academic Custom Colormap: Navy for valid data, Light Turquoise for missing data
cmap_missing = ListedColormap(['#2c3e50', '#1abc9c'])

# Visualizes exactly where the missing values are located across the DataFrame rows/columns
sns.heatmap(
    df_raw.isnull(),
    cmap=cmap_missing,
    cbar=False,
    ax=axes[0],
    yticklabels=df_raw['Customer_ID'],
    linewidths=2.0,
    linecolor='white'
)

axes[0].annotate(
    "Missing Data Blocks:\nThe light turquoise blocks represent\nmissing entries (NaN) that will crash\nany Machine Learning algorithm.",
    xy=(1.5, 2.5), xytext=(2.5, 4.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)
```

```
axes[0].set_title('Figure 1: Nullity Matrix Heatmap', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Dataset Features', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Target Object (Customer ID)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].tick_params(axis='x', rotation=15)
axes[0].set_xticklabels(axes[0].get_xticklabels(), fontweight='bold', fontsize=11)
axes[0].set_yticklabels(axes[0].get_yticklabels(), fontweight='bold', fontsize=11)

# ---------------------------------------------------------
# Plot 2: Bar Plot of Completeness Ratio
# ---------------------------------------------------------
# Added hue='Feature' and legend=False to comply with seaborn v0.14.0+ requirements
sns.barplot(
    data=df_quality_report[df_quality_report['Feature'] != 'Customer_ID'],
    x='Feature',
    y='Missing_Ratio (%)',
    hue='Feature',
    legend=False,
    ax=axes[1],
    palette=['#2980b9', '#1abc9c', '#2c3e50'],
    edgecolor='#2c3e50',
    linewidth=1.5,
    width=0.6
)

# Plotting the threshold line for Drop vs Impute
axes[1].axhline(50, color='#e74c3c', linestyle='--', linewidth=3.0, label='Critical Threshold (50%)')

axes[1].annotate(
    "Imputation Strategy:\nBecause the missing ratio for all features\nis below 50%, we can safely use statistical\nimputation (Median/Mode) to save the data.",
    xy=(0.0, 33.3), xytext=(0.5, 60),
    fontsize=11, fontweight='bold', color='#2c3e50',
    arrowprops=dict(arrowstyle="->", color="#34495e", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#ecf0f1", ec="#bdc3c7", lw=1.5)
)

axes[1].set_title('Figure 2: Feature Missing Ratio Analysis', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Dataset Features', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Missing Ratio (%)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylim(0, 100) # Set Y-axis strictly from 0% to 100%
axes[1].tick_params(axis='x', rotation=15)
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=11)
axes[1].grid(axis='y', linestyle='--', alpha=0.5)
axes[1].legend(loc='upper right', frameon=True, fontsize=11)

# Add value annotations on top of the bars for exact percentage readout
for p in axes[1].patches:
    axes[1].annotate(
        f"{p.get_height():.1f}%",
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center', va='center', xytext=(0, 12), textcoords='offset points',
        fontweight='bold', fontsize=12, color='#2c3e50'
    )

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"منجر می‌شود (مثل XGBoost یا Neural Networks) و سقوطِ قطعیِ الگوریتم‌های پیش‌بینی (Bias) رها کردن مقادیر گمشده در پایگاه داده، به ایجاد سوگیری. "
    r"شود.<br><br>"
    r"<b>حذف (Drop)</b> باشد (خط قرمز در گراف ۲)، آن ویژگی دیگر قابل اعتماد نیست و باید به طور کامل از مدل <i>بیشتر از ۵۰ درصد</i> در صورتی که نرخ فقدانِ یک ستون"
    r"<i>نما (برای متغیرهای کیفی نظیر روش پرداخت)</i> و <i>میانه (برای متغیرهای پیوسته نظیر سن و درآمد)</i> بر مبنای <b>جایگذاریِ هوشمند (Imputation)</b>، اما در این سناریو، به دلیل پایین بودن نرخ فقدان، "
    r"ساختار توزیعِ احتمالاتیِ جامعه آماری را با کمترین میزانِ انحراف حفظ نموده و یکپارچگیِ اطلاعاتیِ لازم جهت اجرای مدل‌های هوش مصنوعی را تضمین نمود (جدول ۳)."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: ارزیابی کیفیت و ترمیم مقادیر گمشده (Data Imputation) 🧩

وجود **مقادیر گمشده (Missing Values / NaN)** یکی از رایج‌ترین چالش‌ها در پایگاه‌های دادهٔ عملیاتی نظیر پلتفرم‌های فروشگاهی است. الگوریتم‌های هوش مصنوعی نیازمند بردارهای اطلاعاتیِ کامل هستند و در صورت برخورد با مقدار `NaN` متوقف می‌شوند. پیش از استقرار مدل، شاخص «کامل بودن» (Completeness Ratio) محاسبه شده و سپس بر مبنای نوع متغیر، استراتژی‌های آماری **جایگذاری (Imputation)** اعمال می‌گردد.

معادله محاسبه نرخ فقدان داده (Missing Ratio) در یک بردار ویژگی x با طول n:

$$\text{Missing_Ratio}(\%) = \left(\frac{\sum_{i=1}^{n} I(x_i = \text{NaN})}{n}\right) \times 100$$

مرحله نخست: گزارش‌گیری کیفیت داده و اعمال استراتژی‌های ترمیم

در این فاز، پس از استخراج گزارش کیفیت، مقادیر گمشده در ویژگی‌های کمّی (سن و درآمد) با استفاده از شاخص مقاوم **«میانه (Median)»** بازسازی می‌شوند تا از انحراف ناشی از داده‌های پرت جلوگیری شود. برای ویژگی‌های *کیفی و دسته‌ای* (روش پرداخت) نیز از شاخص **«نما (Mode / Most Frequent)»** بهره‌برداری می‌گردد.

جدول ۱: پایگاه داده خام (حاوی مقادیر NaN)

Customer_ID	Age	Annual_Income_kUSD
1001	25.0	65.0
1002	34.0	80.0
1003	NaN	120.0
1004	45.0	NaN
1005	29.0	55.0
1006	NaN	95.0

جدول ۲: گزارش کیفیت و نرخ فقدان

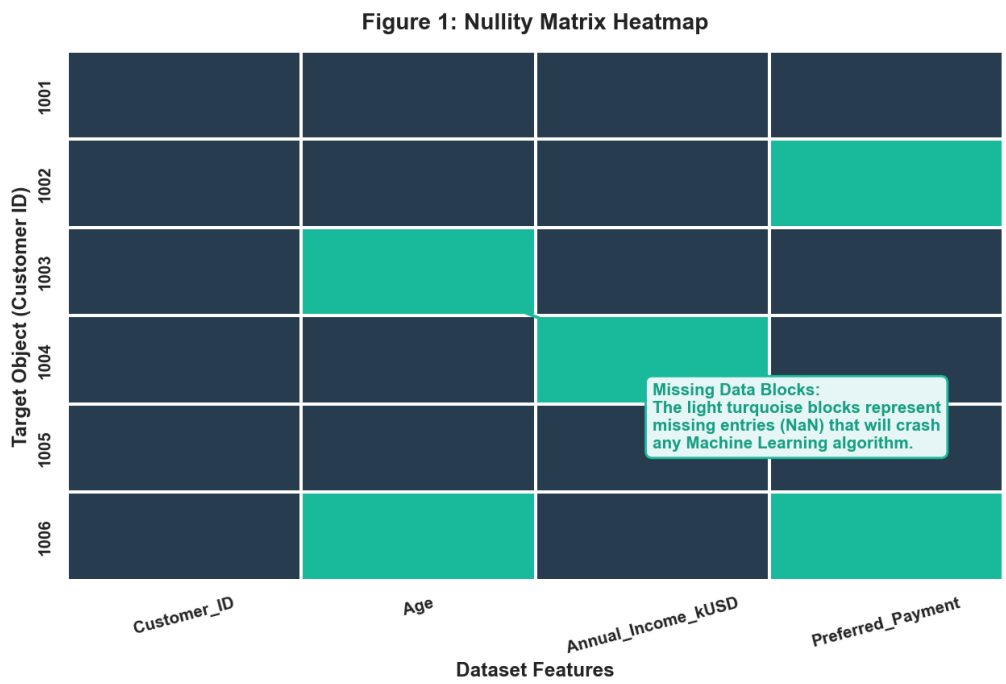
Feature	Missing_Count	Missi
Customer_ID	0	
Age	2	
Annual_Income_kUSD	1	
Preferred_Payment	2	

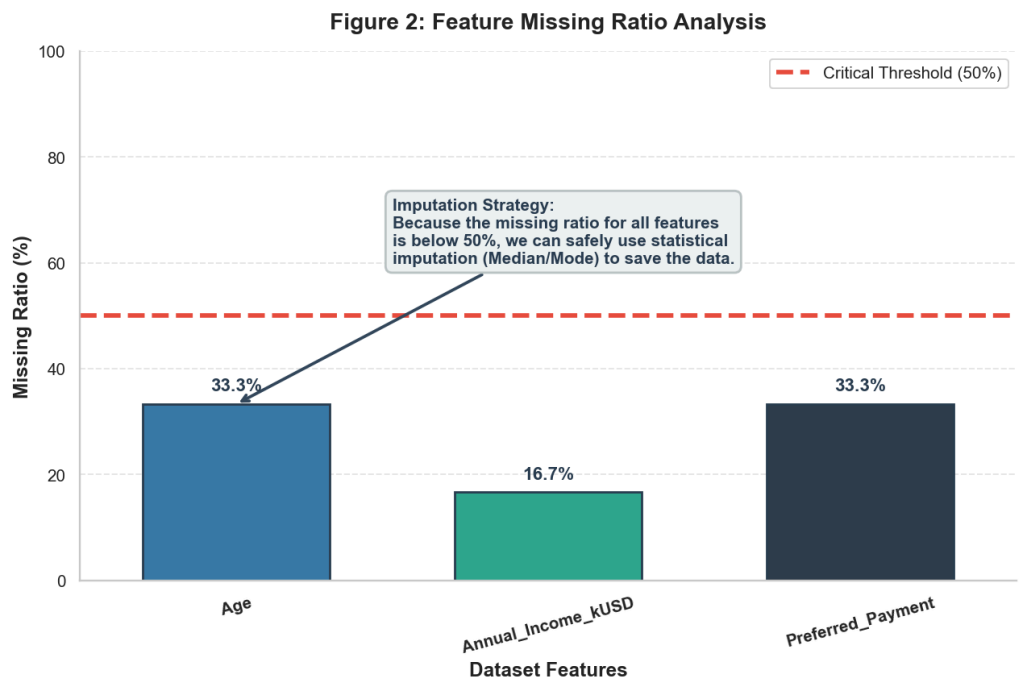
جدول ۳: پایگاه داده ترمیم‌شده (Imputed)

Customer_ID	Age	Annual_Income_kUSD
1001	25	65.0
1002	34	80.0
1003	31	120.0
1004	45	80.0
1005	29	55.0
1006	31	95.0

مرحله دوم: مصورسازی تحلیلی - نقشه فقدان داده و پروفایل کیفیت

بوم گرافیکی زیر توزیع فضایی نواقص را بازنمایی می‌کند. **نمودار سمت چپ (Nullity Heatmap)** دقیقاً نشان می‌دهد که کدام سطرها و ستون‌ها دچار خلأ اطلاعاتی هستند (رنگ سرمه‌ای = داده موجود، رنگ فیروزه‌ای = مقدار گمشده/NaN). **نمودار میله‌ای سمت راست** نیز با نمایش درصد فقدان، تیم داده‌کاوی را در خصوص حفظ و ترمیمِ یک ویژگی یا حذف *کامل (Drop)* آن در صورت عبور از آستانهٔ بحرانی راهنمایی می‌نماید.





استنتاج راهبردی عملیات (Strategic Implication):
رها کردن مقادیر گمشده در پایگاه داده، به ایجاد سوگیری (Bias) و سقوطِ قطعیِ الگوریتم‌های پیش‌بینی (مثل XGBoost یا Neural Networks) منجر می‌شود. در صورتی که نرخ فقدانِ یک ستون *بیشتر از ۵۰ درصد* باشد (خط قرمز در گراف ۲)، آن ویژگی دیگر قابل اعتماد نیست و باید به طور کامل از مدل حذف (Drop) شود.

اما در این سناریو، به دلیل پایین بودن نرخ فقدان، جایگذاریِ هوشمند (Imputation) بر مبنای *میانه (برای متغیرهای پیوسته نظیر سن و درآمد)* و *نما (برای متغیرهای کیفی نظیر روش پرداخت)*، ساختار توزیعِ احتمالاتیِ جامعه آماری را با کمترین میزانِ انحراف حفظ نموده و یکپارچگیِ اطلاعاتیِ لازم جهت اجرای مدل‌های هوش مصنوعی را تضمین نمود (جدول ۳).

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک دیتافریم واقعی از داده‌های خریداران آمازون شامل مقادیر گمشده (**NaN**) در ویژگی‌های عددی (**Age** و **Annual_Income_kUSD**) و ویژگی اسمی (**Preferred_Payment**) ایجاد کردیم.
- **ارزیابی کامل بودن (Quality Assessment):** با ترکیب دستورات **df.isnull().sum()** و **df.isnull().mean()** گزارش دقیقی از تعداد و درصد خانه‌های خالی در هر ستون تولید شد.
- **ترمیم داده‌های عددی:** با استفاده از کلاس **SimpleImputer(strategy='median')** از کتابخانه `scikit-learn` ، مقادیر گمشده ستون‌های عددی با میانه آن ستون بازسازی شدند تا اثر داده‌های پرت خنثی شود.
- **ترمیم داده‌های کیفی:** با تنظیم پارامتر `strategy='most_frequent'` ، خانه‌های خالی ستون روش پرداخت با پرکاربردترین مقدار (نما / Mode) جایگزین گردیدند.
- در نهایت، دیتافریم کاملاً پاک‌سازی‌شده بدون هیچ خانه خالی (**Completeness = 100%**) و آماده برای الگوریتم‌های داده‌کاوی به دست آمد.

2.4.2. Data cleaning

پاک سازی داده 🧹

پاک‌سازی داده (Data Cleaning) به مجموعه‌ای از فرایندها گفته می‌شود که هدف آن‌ها **شناسایی** و **اصلاح** مشکلات موجود در داده‌های خام، پیش از ورود به مراحل تحلیل و مدل‌سازی است. داده‌های واقعی به‌ندرت کامل و بی‌نقص‌اند و معمولاً با مشکلاتی از قبیل **مقادیر گمشده**، **داده‌های نویزی**، **ناسازگاری‌ها** و **رکوردهای تکراری** همراه‌اند. اگر این مشکلات بدون اصلاح باقی بمانند، می‌توانند **دقت** و **اعتبار** نتایج نهایی را به‌طور جدی کاهش دهند؛ به همین دلیل، پاک‌سازی داده پیش‌نیازی ضروری برای رسیدن به تحلیلی **قابل‌اعتماد** محسوب می‌شود.

یکی از رایج‌ترین مشکلات، **مقادیر گمشده (Missing Values)** است که می‌توان آن‌ها را با روش‌هایی مانند **حذف رکوردهای ناقص**، **جایگزینی با میانگین یا میانه**، یا **برآورد مقدار بر اساس سایر ویژگی‌ها** مدیریت کرد.

📐 **ریاضیات — پاک‌سازی داده**

در Smoothing با Binning، یک Bin می‌تواند با خلاصه‌ای مانند میانگین جایگزین شود تا نوسان تصادفی کاهش یابد.

$$\text{BinMean} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.48}$$

نمادها

- BinMean: میانگین Bin.
- x_i: مقدار داخل Bin.
- m: تعداد مقادیر Bin.
- i: اندیس مقدار.

مثال / تمرین حل‌شده

صورت سؤال: یک Bin شامل مقادیر 2, 4, 6 است. اگر Smoothing با Bin Mean انجام شود، مقدار جایگزین برای اعضای Bin چیست؟

حل: میانگین Bin برابر $(2 + 4 + 6)/3 = 4$ است. بنابراین در روش Bin Mean، هر سه مقدار با 4 جایگزین می‌شوند.

توضیح: هدف این کار کاهش نوسان‌های کوچک و نویز است؛ در نتیجه جزئیات محلی داده تا حدی از بین می‌رود.

مثال عملی: پاکسازی داده‌های سنسورهای خودرو در «تسلا» (Tesla)

سیستم مرکزی «تسلا» (Tesla) روزانه میلیاردها داده لحظه‌ای از سنسورهای خودروها (دمای باتری، سرعت، میزان شارژ و پایش سلامت قطعات) دریافت می‌کند. این داده‌های خام به دلیل قطعی اینترنت، نویز سنسورها یا اختلالات نرم‌افزاری دچار خطا می‌شوند. سیستم پیش از تحلیل‌های داده‌کاوی، ۴ گام اصلی **پاکسازی داده (Data Cleaning)** را اجرا می‌کند:

۱. حذف رکوردهای تکراری (Duplicate Removal):

گاهی یک سنسور به‌دلیل ارسال مجدد سیگنال، یک سیگنال واحد را ۲ بار ذخیره می‌کند. رکوردهای تکراری باید شناسایی و حذف شوند تا وزن کاذب در محاسبات ایجاد نکنند.

۲. اصلاح انواع داده‌ها (Data Type Correction):

گاهی دمای باتری به جای عدد به‌صورت متنی (مثلاً `"45.2"`) ذخیره می‌شود. برای انجام محاسبات ریاضی، این داده‌ها باید به نوع عددی (`Float`) تبدیل گردند.

۳. مدیریت مقادیر گمشده (Missing Values Handling):

اگر سنسور در مقطعی قطع شده باشد:

- **حذف (Removal):** اگر حجم داده‌های خالی کم باشد، آن سطر حذف می‌شود.
- **جایگذاری (Imputation):** داده گمشده با میانگین/میانه مقادیر قبلی یا روش درونیابی (Interpolation) بازسازی می‌شود.

۴. شناسایی و حذف داده‌های پرت و نویز (Outlier Removal):

اگر سنسور به‌دلیل نوسان ولتاژ، دمای باتری را ناگهان ۹۹۹ درجه سانتی‌گراد ثبت کند، این داده پرت و غیرممکن به عنوان نویز شناسایی و فیلتر می‌شود.

نتیجه: پاکسازی داده‌ها ورودی دقیق و قابل اعتمادی برای الگوریتم‌های هوش مصنوعی خودران تسلا فراهم می‌سازد تا از تصمیم‌گیری‌های اشتباه جلوگیری شود.

خروجی: تبدیل داده‌های خام و دارای نویز سنسورها به یک پایگاه داده یکدست، تمیز و استاندارد

In [33]:
```
display_academic_text(
    "پیش‌پردازش داده‌ها: استراتژی‌های پاک‌سازی، حذف نویز و یکپارچه‌سازی پایگاه داده (Data Cleaning Pipeline) 🛠️",
    "main_title"
)

display_academic_text(
    "اطلاعات مستخرج از شبکه‌های اینترنت اشیا (IoT) و سنسورهای فیزیکی به شدت مستعد خطا، افزونگی و داده‌های گمشده می‌باشند. "
    "در این سناریو، یک پایگاه داده خام از ناوگان خودروهای خودران تحت ۴ عملیات متوالی پاک‌سازی "
    "(<b>Deduplication, Type Correction, Outlier Masking, Imputation</b>) تضمین گردد تا پایداری تحلیلیِ الگوریتم‌های هوش مصنوعی قرار می‌گیرد.",
    "paragraph"
)

display_academic_text("آستانه‌های فیزیکی مجاز جهت فیلترینگ داده‌های پرت (Outliers):", "sub_title")

# Utilizing Raw Strings (r"") for LaTeX rendering of boundary constraints
display_academic_text(
    r"$$Battery\_Temp \in [0, 100] \, ^\circ\text{C} \quad , \quad Speed \in [0, 300] \, \text{KM/H}$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: MULTI-STAGE DATA CLEANING PIPELINE
# ==========================================================
display_academic_text(
    "مرحله نخست: اعمال پایپ‌لاین پاک‌سازی بر روی پایگاه داده سنسورها",
    "sub_title"
)

display_academic_text(
    "در این فاز، ابتدا رکوردهای <i>تکراری</i> (مانند دمای ۹۹۹ درجه) حذف شده و نوع داده‌های متنی به عددی تبدیل می‌شوند. سپس مقادیرِ خارج از آستانه مجاز "
    "به عنوان <i>نویزِ سخت‌افزاری</i> شناسایی شده و به مقادیر گمشده (NaN) تبدیل می‌گردند. در گام نهایی، تمامی مقادیر تهی با استفاده از شاخص آماری میانه بازسازی می‌شوند.",
    "paragraph"
)
```

```python
# Step 1: Initialize the raw, messy telemetry dataset mimicking real-world sensor noise
raw_data = {
    # Notice the exact Duplicate for TS-102
    'Vehicle_ID': ['TS-101', 'TS-102', 'TS-102', 'TS-103', 'TS-104', 'TS-105'],
    # Notice String types ('35.5'), real NaN, and extreme physics outlier ('999.0')
    'Battery_Temp_C': ['35.5', '42.0', '42.0', np.nan, '999.0', '38.2'],
    # Notice negative speed outlier (-50.0) which is impossible
    'Speed_KMH': [110.0, 85.0, 85.0, 95.0, 120.0, -50.0],
}
df_raw = pd.DataFrame(raw_data)

# Step 2: Clean Dataset Step-by-Step (The Preprocessing Pipeline)
# 2.1 Deduplication: Remove exact duplicate rows and create an isolated copy
df_clean = df_raw.drop_duplicates().copy()

# 2.2 Data Type Correction: Convert String to Numeric Float
# errors='coerce' gracefully converts unparseable strings to NaN
df_clean['Battery_Temp_C'] = pd.to_numeric(df_clean['Battery_Temp_C'], errors='coerce')

# 2.3 Outlier Masking: Filter out physically impossible values based on defined thresholds
valid_temp_mask = (df_clean['Battery_Temp_C'] >= 0) & (df_clean['Battery_Temp_C'] <= 100)
valid_speed_mask = (df_clean['Speed_KMH'] >= 0) & (df_clean['Speed_KMH'] <= 300)

# Replace outliers (where the mask is False) with NaN to prepare them for imputation
df_clean.loc[~valid_temp_mask, 'Battery_Temp_C'] = np.nan
df_clean.loc[~valid_speed_mask, 'Speed_KMH'] = np.nan

# 2.4 Missing Value Imputation: Fill NaN (both original and newly masked outliers) with Median
median_temp = df_clean['Battery_Temp_C'].median()
median_speed = df_clean['Speed_KMH'].median()

df_clean['Battery_Temp_C'] = df_clean['Battery_Temp_C'].fillna(median_temp)
df_clean['Speed_KMH'] = df_clean['Speed_KMH'].fillna(median_speed)

# Prepare cleanly formatted DataFrames for display
df_raw_disp = df_raw.copy()
df_clean_disp = df_clean.copy()

display_side_by_side_datasets(
    [df_raw_disp, df_clean_disp],
    [
        "جدول ۱: پایگاه داده خام سنسورها (دارای نویز و افزونگی)",
        "جدول ۲: پایگاه داده پاک‌سازی شده نهایی (ایمن و یکپارچه)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - توزیع داده‌ها پیش و پس از فیلترینگ",
    "sub_title"
)

display_academic_text(
    "بوم گرافیکی زیر تأثیر شگرف فیلترینگ نویز و ترمیم داده‌ها را به تصویر می‌کشد. "
    "نمودارهای سمت چپ (قرمز رنگ) به شدت تحت تأثیر مقادیر فیزیکاً غیرممکن (۹۹۹ درجه و سرعت منفی) قرار گرفته‌اند که باعث فشردگی کاذبِ سایر رکوردهای سالم شده است. "
    "پس از اعمال پایپ‌لاین پاک‌سازی (نمودارهای سمت راست با پالت آبی آکادمیک)، توزیع مقادیر به صورت منطقی و در دامنه استاندارد بازسازی گردیده است.",
    "paragraph"
)

# Initialize a 2x2 subplot canvas for comparative visual analytics
fig, axes = plt.subplots(2, 2, figsize=(18, 9), dpi=135)

# Pre-process raw data for visualization ONLY (Converting strings to numbers to allow plotting)
df_plot_raw = df_raw.copy()
df_plot_raw['Battery_Temp_C'] = pd.to_numeric(df_plot_raw['Battery_Temp_C'], errors='coerce')

# ----------------------------------------------------------
# ROW 1: Battery Temperature Distribution (Before & After)
# ----------------------------------------------------------
# Plot Raw Battery Temp (Showing the 999 Outlier)
sns.stripplot(
    data=df_plot_raw, y='Battery_Temp_C',
    ax=axes[0,0], color='#e74c3c', size=12, jitter=0.05, edgecolor='black', linewidth=1.5
)

axes[0,0].annotate(
    "Sensor Failure (999°C):\nThis massive outlier squashes the real data\nat the bottom, making it unreadable.",
    xy=(0.05, 999), xytext=(0.15, 750),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0,0].set_title('Figure 1A: Raw Battery Temp (Sensor Noise)', fontdict={'fontsize': 13, 'fontweight': 'bold'}, pad=10)
axes[0,0].set_ylabel('Temperature (°C)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[0,0].grid(True, linestyle='--', alpha=0.5)


# Plot Cleaned Battery Temp
sns.stripplot(
    data=df_clean, y='Battery_Temp_C',
    ax=axes[0,1], color='#1abc9c', size=12, jitter=0.05, edgecolor='black', linewidth=1.5
)

axes[0,1].set_title('Figure 1B: Cleaned Temp (Masked & Imputed)', fontdict={'fontsize': 13, 'fontweight': 'bold'}, pad=10)
axes[0,1].set_ylabel('Temperature (°C)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[0,1].set_ylim(34, 45) # Set reasonable physical limits for visualization
axes[0,1].grid(True, linestyle='--', alpha=0.5)

# ----------------------------------------------------------
```

```
# ROW 2: Vehicle Speed Distribution (Before & After)
# ---------------------------------------------------------
# Plot Raw Speed (Showing the -50 Outlier)
sns.stripplot(
    data=df_plot_raw, y='Speed_KMH',
    ax=axes[1,0], color='#c0392b', size=12, jitter=0.05, edgecolor='black', linewidth=1.5
)

axes[1,0].annotate(
    "Physics Violation:\nSpeed cannot be -50 KM/H!\nA neural network would learn impossible physics.",
    xy=(0.05, -50), xytext=(0.15, -20),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#c0392b", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#c0392b", lw=1.5)
)

axes[1,0].set_title('Figure 2A: Raw Speed (Physics Violation)', fontdict={'fontsize': 13, 'fontweight': 'bold'}, pad=10)
axes[1,0].set_ylabel('Speed (KM/H)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[1,0].grid(True, linestyle='--', alpha=0.5)


# Plot Cleaned Speed
sns.stripplot(
    data=df_clean, y='Speed_KMH',
    ax=axes[1,1], color='#2980b9', size=12, jitter=0.05, edgecolor='black', linewidth=1.5
)

axes[1,1].set_title('Figure 2B: Cleaned Speed (Standardized Range)', fontdict={'fontsize': 13, 'fontweight': 'bold'}, pad=10)
axes[1,1].set_ylabel('Speed (KM/H)', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[1,1].set_ylim(70, 130) # Set reasonable physical limits for visualization
axes[1,1].grid(True, linestyle='--', alpha=0.5)

# Clear right and top spines for all subplots
for ax in axes.flat:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout(pad=3.0)
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"تزریق مستقیم داده‌های خام و فیلترنشده به الگوریتم‌های هوش مصنوعی منجر به پدیدهٔ <b>GIGO (Garbage In, Garbage Out)</b> و فروپاشیِ قطعیِ محاسباتیِ مدل می‌گردد "
    r"(تغییرات شدیدِ گرادیان به دلیل دمای ۹۹۹ درجه!). <br><br>"
    r"۱: پایپ‌لاین پیش‌پردازشِ ما با اجرای متوالی توابع: (پایپ‌لاین پیش‌پردازش) <i>جرم‌زدایی (Deduplication)</i>، ۲) <i>همگن‌سازی نوع داده‌ها (Casting)</i>، ۳) <i>اعمال آستانه‌های فیزیکی (Masking)</i> و ۴) <i>ترمیم داده‌های گمشده (Imp
    r"یکپارچگی اطلاعاتی را تضمین نموده و بستر لازم جهت استقرارِ امن، قابل اعتماد و دقیقِ سیستم‌های خودران را فراهم آورد."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: استراتژی‌های پاک‌سازی، حذف نویز و یکپارچه‌سازی پایگاه داده (Data Cleaning Pipeline) 🛠️

اطلاعات مستخرج از شبکه‌های اینترنت اشیا (IoT) و سنسورهای فیزیکی به شدت مستعد خطا، افزونگی و داده‌های گمشده می‌باشند. در این سناریو، یک پایگاه داده خام از ناوگان خودروهای خودران تحت ۴ عملیات متوالی پاک‌سازی (**Deduplication, Type Correction, Outlier Masking, Imputation**) قرار می‌گیرد تا پایداری تحلیلیِ الگوریتم‌های هوش مصنوعی تضمین گردد.

آستانه‌های فیزیکی مجاز جهت فیلترینگ داده‌های پرت (Outliers):

$$\text{Battery_Temp} \in [0, 100]\,^{\circ}\text{C} \quad , \quad \text{Speed} \in [0, 300]\,\text{KM/H}$$

مرحله نخست: اعمال پایپ‌لاین پاک‌سازی بر روی پایگاه داده سنسورها

در این فاز، ابتدا رکوردهای *تکراری* حذف شده و نوع داده‌های متنی به عددی تبدیل می‌شوند. سپس مقادیرِ خارج از آستانه مجاز (مانند دمای ۹۹۹ درجه) به عنوان *نویزِ سخت‌افزاری* شناسایی شده و به مقادیر گمشده (NaN) تبدیل می‌گردند. در گام نهایی، تمامی مقادیر تهی با استفاده از شاخص آماری میانه بازسازی می‌شوند.

جدول ۲: پایگاه داده پاک‌سازی شده نهایی (ایمن و یکپارچه)

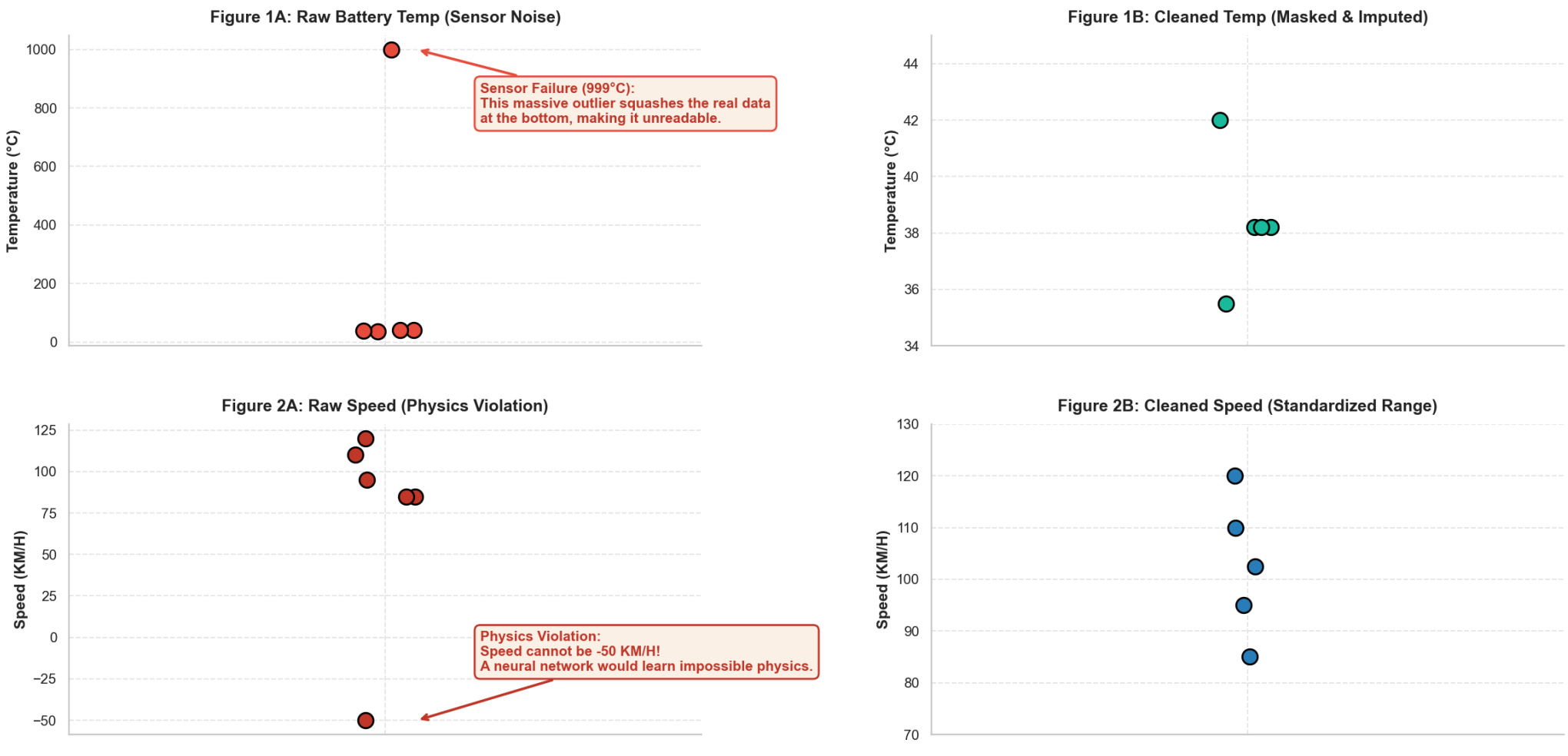
Vehicle_ID	Battery_Temp_C	Speed_KMH
TS-101	35.5	110.0
TS-102	42.0	85.0
TS-103	38.2	95.0
TS-104	38.2	120.0
TS-105	38.2	102.5

جدول ۱: پایگاه داده خام سنسورها (دارای نویز و افزونگی)

Vehicle_ID	Battery_Temp_C	Speed_KMH
TS-101	35.5	110.0
TS-102	42.0	85.0
TS-102	42.0	85.0
TS-103	NaN	95.0
TS-104	999.0	120.0
TS-105	38.2	-50.0

مرحله دوم: مصورسازی تحلیلی - توزیع داده‌ها پیش و پس از فیلترینگ

بوم گرافیکی زیر تأثیر شگرف فیلترینگ نویز و ترمیم داده‌ها را به تصویر می‌کشد. نمودارهای سمت چپ (قرمز رنگ) به شدت تحت تأثیر مقادیر فیزیکاً غیرممکن (۹۹۹ درجه و سرعت منفی) قرار گرفته‌اند که باعث فشردگی کاذبِ سایر رکوردهای سالم شده است. پس از اعمال پایپ‌لاین پاک‌سازی (نمودارهای سمت راست با پالت آبی آکادمیک)، توزیع مقادیر به صورت منطقی و در دامنه استاندارد بازسازی گردیده است.



استنتاج راهبردی عملیات (Strategic Implication):
تزریق مستقیم داده‌های خام و فیلترنشده به الگوریتم‌های هوش مصنوعی منجر به پدیدهٔ GIGO (Garbage In, Garbage Out) و فروپاشیِ قطعیِ محاسباتیِ مدل می‌گردد (تغییرات شدیدِ گرادیان به دلیل دمای ۹۹۹ درجه!).

پایپ‌لاین پیش‌پردازشِ ما با اجرای متوالی توابع: ۱) *جرم‌زدایی (Deduplication)*، ۲) *همگن‌سازی نوع داده‌ها (Casting)*، ۳) *اعمال آستانه‌های فیزیکی (Masking)* و ۴) *ترمیم داده‌های گمشده (Imputation)*، یکپارچگی اطلاعاتی را تضمین نموده و بستر لازم جهت استقرارِ امن، قابل اعتماد و دقیقِ سیستم‌های خودران را فراهم آورد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده خام از سنسورهای تسلا ساختیم که حاوی انواع مشکلات متداول داده‌ها بود: داده تکراری، نوع داده نادرست (رشته متنی به‌جای عدد)، مقادیر گمشده (**NaN**) و نویز/داده‌های پرت غیرواقعی (دمای ۹۹۹ درجه و سرعت منفی ۵۰).
- **حذف رکوردهای تکراری:** با متد **drop_duplicates()** رکوردهای همسان فیلتر شدند.

- **اصلاح اصلاح نوع داده:** با تابع **pd.to_numeric()** ستون متنی دمای باتری به نوع عددی اعشاری (`float64`) تبدیل شد.
- **شناسایی و حذف نویز/داده‌های پرت:** با اعمال فیلتر شرطی (دمای مجاز بین ۰ تا ۱۰۰ و سرعت مجاز بین ۰ تا ۳۰۰)، مقادیر پرت غیرمنطقی شناسایی و به مقدار خالی (NaN) تبدیل شدند.
- **ترمیم مقادیر گمشده:** در نهایت، تمام مقادیر خالی با استفاده از متد **fillna()** و جایگذاری با **میانه (Median)** ستون‌ها بازسازی شدند تا ساختار داده‌ها ۱۰۰٪ کامل و آماده ورود به مدل‌های یادگیری ماشین گردد.

2.4.3. Data integration

یکپارچه‌سازی داده 🔄

یکپارچه‌سازی داده فرایندی در **پیش‌پردازش داده** است که در آن داده‌های حاصل از **منابع مختلف** در قالب یک مجموعه‌داده **منسجم و یکپارچه** ادغام می‌شوند. هدف این فرایند، ایجاد **دیدگاهی جامع** از داده‌ها و کاهش **ناسازگاری** و **تعارض** میان اطلاعات موجود در منابع گوناگون است.

در یکپارچه‌سازی داده، چالش‌هایی مانند **شناسایی موجودیت‌های مشابه**، **تطبیق ساختار و معنای داده‌ها**، **رفع مقادیر متناقض** و **حذف داده‌های تکراری** مدیریت می‌شود. اجرای صحیح این فرایند موجب افزایش **کیفیت**، **انسجام** و **قابلیت اعتماد** داده‌ها شده و دقت و کارایی روش‌های **داده‌کاوی**، **یادگیری ماشین** و **تحلیل داده** را به‌طور قابل‌توجهی بهبود می‌بخشد.

📐 ریاضیات – یکپارچه‌سازی داده

هنگام ادغام منابع مختلف، تفاوت مقیاس ویژگی‌ها می‌تواند روی فاصله‌ها اثر بگذارد. یک نگاشت ساده به بازه مشترک چنین است:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{2.49}$$

نمادها

- x': مقدار نرمال‌شده.
- x: مقدار اصلی.
- $\min(X)$: کمینه.
- $\max(X)$: بیشینه.

مثال / تمرین حل‌شده

صورت سؤال: مقدار 15 را با Min-Max به بازه [0, 1] ببرید، وقتی کمینه 10 و بیشینه 20 است.

حل: صورت کسر 5 = 10 - 15 و مخرج 10 = 10 - 20 است. بنابراین $x' = 5/10 = 0.5$.

توضیح: مقدار ۱۵ دقیقاً در میانه بازه ۱۰ تا ۲۰ قرار دارد، پس پس از Min-Max نیز در میانه بازه جدید، یعنی ۰.۵، قرار می‌گیرد.

مثال عملی: ادغام پایگاه‌های داده ناهمگن در «اسپاتیفای» (Spotify)

پلتفرم «اسپاتیفای» (Spotify) داده‌های خود را در چند سیستم مجزا نگهداری می‌کند:

- **منبع اول (پایگاه‌داده پخش موسیقی):** شامل لیست کاربران و رفتار شنیداری آن‌ها (تعداد ساعات پخش و سبک محبوب).
- **منبع دوم (پایگاه‌داده مالی و حسابداری):** شامل تراکنش‌های پرداختی، نوع اشتراک (پریمیوم / رایگان) و آدرس صورت‌حساب.

چالش یکپارچه‌سازی داده‌ها (Data Integration):

تیم تحلیل الگوریتم برای پاسخ به سوالاتی مثل «آیا کاربران پریمیوم ساعات پخش بیشتری نسبت به کاربران رایگان دارند؟» نمی‌تواند از جدول‌های جداگانه استفاده کند. داده‌ها باید با استفاده از یک **کلید مشترک (Primary/Foreign Key)** مانند `User_ID` با هم ادغام (Merge / Join) شوند.

حل تعارضات و ناسازگاری‌ها در فرایند ادغام:

در زمان یکپارچه‌سازی داده‌ها، ۳ چالش اصلی رخ می‌دهد که باید حل شوند:

- **تعارض نام ویژگی‌ها (Schema Matching):** ستون کشور در جدول اول `Country` و در جدول دوم `Region_Code` نام دارد که باید یکسان‌سازی شوند.
- **عدم تطابق کلیدها (Data Redundancy):** ممکن است کاربری در جدول پخش وجود داشته باشد اما هنوز هیچ تراکنش مالی ثبت نکرده باشد؛ که نوع ادغام (Inner, Left, Outer Join) این موضوع را مدیریت می‌کند.
- **تعارض مقادیر (Value Conflict):** مقادیر یکسان با فرمت‌های متفاوت (مثلاً ثبت تاریخ میلادی در یک منبع و شمسی در منبع دیگر).

نتیجه: یکپارچه‌سازی داده‌ها، دیدگاه ۳۶۰ درجه‌ای و جامع از تمام ابعاد رفتار کاربر در اختیار تیم داده‌کاوی اسپاتیفای قرار می‌دهد.

خروجی: ترکیب منابع داده مجزا و حل تعارضات ساختاری بر اساس کلید مشترک شناسه کاربر

In [34]:
```
display_academic_text(
    "پیش‌پردازش داده‌ها: یکپارچه‌سازی پایگاه‌های اطلاعاتی (Data Integration) و حل تعارضات شماتیک 🔗",
    "main_title"
)

display_academic_text(
    "سیلوهای مجزا<b>(Data Silos)</b> بر مبنای اصول معماری داده، اطلاعات مستقر در <b>. فاقد ارزشِ تحلیلیِ ترکیبی هستند "
    "کاربران، (Financial Status) و وضعیت مالی (Streaming Activity) در این سناریو، به منظور کشف وابستگی میان رفتار شنیداری "
    "دو پایگاه دادهٔ <i>ناهمگن</i> از طریق یک <b>کلید مشترک(Primary Key)</b> ادغام می‌شوند. چالش‌های تعارضِ نامِ ویژگی‌ها (Schema Conflict) "
    "و رکوردهای معلق (Dangling Tuples) در حین این عملیات مدیریت می‌گردند.",
    "paragraph"
)

display_academic_text(
    "منطق جبر رابطه‌ای (Relational Algebra) برای عملیات Left Outer Join:",
    "sub_title"
)

# Utilizing Raw Strings (r"") for LaTeX rendering of Relational Algebra
display_academic_text(
    r"$$T_{Integrated} = T_{Streaming} \ \text{LEFT OUTER JOIN} \ T_{Financial} \ \text{ON} \ (User\_ID)$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: DATA SOURCES INITIALIZATION AND INTEGRATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری منابع مجزا و اجرای عملیات یکپارچه‌سازی",
    "sub_title"
)

display_academic_text(
    "عملیات <b>Left Outer Join</b> (تولید مقادیر گمشده) تضمین می‌کند که تمامی کاربرانِ پلتفرم پخش زنده حفظ شوند، حتی اگر سابقهٔ خریدی در سیستمِ مالی نداشته باشند. "
    "همچنین کاربران موجود در سیستم مالی که هرگز موسیقی گوش نداده‌اند (مانند کاربر ۱۰۶) به صورت خودکار فیلتر می‌شوند.",
    "paragraph"
)

# Step 1: Initialize Source Dataset 1 (Streaming Activity Logs)
data_streaming = {
    'User_ID': [101, 102, 103, 104, 105],
    'Monthly_Hours_Streamed': [45.5, 12.0, 88.2, 5.0, 62.1],
    'Favorite_Genre': ['Pop', 'Rock', 'Hip-Hop', 'Jazz', 'Pop'],
}
df_streaming = pd.DataFrame(data_streaming)

# Step 2: Initialize Source Dataset 2 (Financial Subscription Database)
# Notice Schema Conflict: 'Region_Code' instead of 'Country'
# Notice Dangling Tuples: User 104 is missing, User 106 is exclusively in this table
data_financial = {
    'User_ID': [101, 102, 103, 105, 106],
    'Subscription_Plan': ['Premium', 'Free', 'Premium', 'Premium', 'Free'],
```

```python
    'Monthly_Fee_USD': [9.99, 0.00, 9.99, 9.99, 0.00],
    'Region_Code': ['US', 'UK', 'US', 'CA', 'DE'],
}
df_financial = pd.DataFrame(data_financial)

# Step 3: Perform Data Integration (Left Outer Join on 'User_ID')
df_integrated = pd.merge(
    df_streaming,
    df_financial,
    on='User_ID',
    how='left'
)

# Step 4: Resolve Schema Conflicts (Attribute Name Standardization)
df_integrated = df_integrated.rename(columns={'Region_Code': 'Country'})

# Step 5: Handle Missing Values derived from the Join operation (User 104 lacks financial data)
df_integrated['Subscription_Plan'] = df_integrated['Subscription_Plan'].fillna('Unknown / Unregistered')
df_integrated['Monthly_Fee_USD'] = df_integrated['Monthly_Fee_USD'].fillna(0.00)

# Display preparation to prevent empty index columns in HTML
df_stream_disp = df_streaming.reset_index(drop=True)
df_fin_disp = df_financial.reset_index(drop=True)
df_int_disp = df_integrated.reset_index(drop=True)

# Display the independent sources
display_side_by_side_datasets(
    [df_stream_disp, df_fin_disp],
    [
        "منبع اول: پایگاه داده رفتار پخش (Streaming Logs)",
        "منبع دوم: پایگاه داده تراکنش‌ها (Financial System)"
    ]
)

# Display the finalized Integrated Dataset
display_side_by_side_datasets(
    [df_int_disp],
    [
        "مخزن داده یکپارچه‌شده (Unified Data Warehouse) پس از اجرای JOIN و پاک‌سازی"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: استخراج دانش تحلیلی از پایگاه داده یکپارچه‌شده",
    "sub_title"
)

display_academic_text(
    "هدف نهایی از یکپارچه‌سازی داده‌ها، پاسخ به <b>سوالاتِ ترکیبیِ کسب‌وکار</b>، بوم گرافیکی زیر با بهره‌گیری از جدولِ ادغام‌شده است. "
    "تأثیر نوع اشتراکِ مالی (از جدول دوم) را بر میزان ساعاتِ پخش موسیقی (از جدول اول) تحلیل می‌نماید. این نمودار ثابت می‌کند که ایجاد یک دیدگاه ۳۶۰ درجه، "
    "چگونه الگوهای پنهان را نمایان می‌سازد.",
    "paragraph"
)

# Initialize a plot canvas optimized for business analytics
plt.figure(figsize=(14, 6), dpi=135)

# Plotting the cross-domain relationship: Streaming Hours vs. Subscription Plan
sns.barplot(
    data=df_integrated,
    x='Subscription_Plan',
    y='Monthly_Hours_Streamed',
    hue='Subscription_Plan',
    legend=False,
    palette=['#2980b9', '#1abc9c', '#7f8c8d'], # Academic Blues & Grays
    edgecolor='#2c3e50',
    linewidth=2.0,
    errorbar=None # Simplified for direct mean representation
)

# Intelligent Annotations to explain the Cross-Domain Insight
plt.annotate(
    "The Magic of Data Integration:\nBy combining the Financial Table with the Streaming Table,\nwe immediately discovered that Premium users listen\nalmost 5x more than Free/Unregistered u
    xy=(0.0, 65.2), xytext=(0.5, 75),
    fontsize=11, fontweight='bold', color='#2c3e50',
    arrowprops=dict(arrowstyle="->", color="#34495e", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.5", fc="#ecf0f1", ec="#bdc3c7", lw=1.5)
)

# Enhance plot labeling, titles, and structural configuration
plt.title('Figure 1: Cross-Domain Analytics (Financial Status vs. Streaming Engagement)', fontdict={'fontsize': 15, 'fontweight': 'bold'}, pad=15)
plt.xlabel('Financial Subscription Tier (From Table 2)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylabel('Average Monthly Streamed Hours (From Table 1)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.grid(axis='y', linestyle='--', alpha=0.5, zorder=0)

# Make tick labels bold
plt.xticks(fontweight='bold', fontsize=12)

# Add exact value annotations on top of the bars for precise reporting
ax = plt.gca()
for p in ax.patches:
    if p.get_height() > 0:
        ax.annotate(
            f"{p.get_height():.1f} Hrs",
            (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='center', xytext=(0, 12), textcoords='offset points',
            fontweight='bold', fontsize=12, color='#2c3e50'
        )

# Clear top and right spines for a clean academic aesthetic
```

```
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"حل تعارضات ساختاری<b> (Schema Matching)</b> یکپارچه‌سازی منابعِ ناهمگن، تنها چسباندن جداول به یکدیگر نیست؛ بلکه نیازمند استراتژی‌های قطعی جهت
    r"مثل تبدیل <code>Region_Code</code> به <code
    r"ناشی از رکوردهای معلق (Dangling Tuples) است.<br>"
    r"خلأهای اطلاعاتیِ<b></b> و مدیریتِ
    r"افزایش نرخ تعامل (Engagement) در سرور استریم<i></i>» و «پرداخت حق اشتراک در سیستم مالی»<i> ادغام موفقیت‌آمیز این داده‌ها، به تیم تحلیلگر اجازه داد تا همبستگیِ پنهان میان
    r"را با دقت بالا استخراج نماید <i>
    r"هرگز قابل کشف نبود JOIN این بینش، بدونِ عملیات!"
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: یکپارچه‌سازی پایگاه‌های اطلاعاتی (Data Integration) و حل تعارضات شماتیک

بر مبنای اصول معماری داده، اطلاعات مستقر در **سیلوهای مجزا (Data Silos)** فاقد ارزشِ تحلیلیِ ترکیبی هستند. در این سناریو، به منظور کشف وابستگی میان رفتار شنیداری (Streaming Activity) و وضعیت مالی (Financial Status) کاربران، دو پایگاه دادهٔ *ناهمگن* از طریق یک **کلید مشترک (Primary Key)** ادغام می‌شوند. چالش‌های تعارضِ نامِ ویژگی‌ها (Schema Conflict) و رکوردهای معلق (Dangling Tuples) در حین این عملیات مدیریت می‌گردند.

منطق جبر رابطه‌ای (Relational Algebra) برای عملیات Left Outer Join:

$$T_{Integrated} = T_{Streaming} \text{ LEFT OUTER JOIN } T_{Financial} \text{ ON (User_ID)}$$

مرحله نخست: بارگذاری منابع مجزا و اجرای عملیات یکپارچه‌سازی

عملیات **Left Outer Join** تضمین می‌کند که تمامی کاربرانِ پلتفرم پخش زنده حفظ شوند، حتی اگر سابقهٔ خریدی در سیستمِ مالی نداشته باشند (تولید مقادیر گمشده). همچنین کاربران موجود در سیستم مالی که هرگز موسیقی گوش نداده‌اند (مانند کاربر ۱۰۶) به صورت خودکار فیلتر می‌شوند.

منبع اول: پایگاه داده رفتار پخش (Streaming Logs)

User_ID	Monthly_Hours_Streamed	Favorite_Genre
101	45.5	Pop
102	12.0	Rock
103	88.2	Hip-Hop
104	5.0	Jazz
105	62.1	Pop

منبع دوم: پایگاه داده تراکنش‌ها (Financial System)

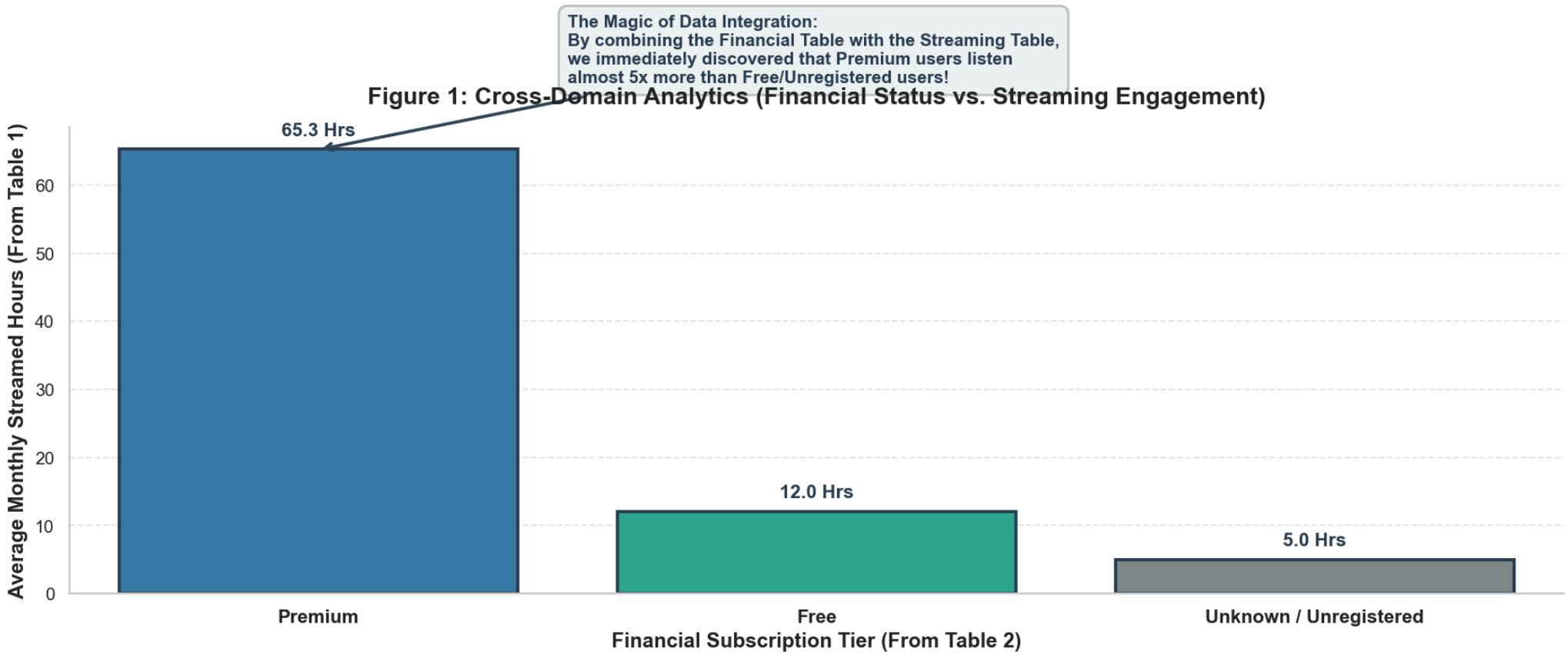
User_ID	Subscription_Plan	Monthly_Fee_USD	Region_Code
101	Premium	10.0	US
102	Free	0.0	UK
103	Premium	10.0	US
105	Premium	10.0	CA
106	Free	0.0	DE

مخزن داده یکپارچه‌شده (Unified Data Warehouse) پس از اجرای JOIN و پاک‌سازی

User_ID	Monthly_Hours_Streamed	Favorite_Genre	Subscription_Plan	Monthly_Fee_USD	Country
101	45.5	Pop	Premium	10.0	US
102	12.0	Rock	Free	0.0	UK
103	88.2	Hip-Hop	Premium	10.0	US
104	5.0	Jazz	Unknown / Unregistered	0.0	NaN
105	62.1	Pop	Premium	10.0	CA

مرحله دوم: استخراج دانش تحلیلی از پایگاه داده یکپارچه‌شده

هدف نهایی از یکپارچه‌سازی داده‌ها، پاسخ به **سوالاتِ ترکیبیِ کسب‌وکار** است. بوم گرافیکی زیر با بهره‌گیری از جدولِ ادغام‌شده، تأثیر نوع اشتراکِ مالی (از جدول دوم) را بر میزان ساعاتِ پخش موسیقی (از جدول اول) تحلیل می‌نماید. این نمودار ثابت می‌کند که ایجاد یک دیدگاه ۳۶۰ درجه، چگونه الگوهای پنهان را نمایان می‌سازد.



استنتاج راهبردی عملیات (Strategic Implication):
یکپارچه‌سازی منابعِ ناهمگن، تنها چسباندن جداول به یکدیگر نیست؛ بلکه نیازمند استراتژی‌های قطعی جهت حل تعارضات ساختاری (Schema Matching) (مثل تبدیل `Region_Code` به `Country`) و مدیریتِ خلأهای اطلاعاتیِ ناشی از رکوردهای معلق (Dangling Tuples) است.
ادغام موفقیت‌آمیز این داده‌ها، به تیم تحلیلگر اجازه داد تا همبستگیِ پنهان میانِ *«پرداخت حق اشتراک در سیستم مالی»* و *«افزایش نرخ تعامل (Engagement) در سرور استریم»* را با دقت بالا استخراج نماید. این بینش، بدونِ عملیات JOIN هرگز قابل کشف نبود!

توضیح مثال کد پایتون 💻

در این مثال:

- ما دو مجموعه داده مجزا از اسپاتیفای ساختیم: جدول اول مربوط به رفتار پخش موسیقی (**df_streaming**) و جدول دوم مربوط به تراکنش‌های مالی و نوع اشتراک (**df_financial**).
- **ادغام بر اساس کلید مشترک:** با استفاده از تابع **pd.merge()** و تعیین `on='User_ID'` ، دو منبع داده مجزا بر اساس شناسه مشترک کاربر ترکیب شدند.
- **مدیریت تعارض ساختار (Schema Matching):** با استفاده از متد **rename()**، نام ستون **'Region_Code'** به **'Country'** تغییر یافت تا با ساختار کل مجموعه داده یکنواخت شود.
- **مدیریت رکوردهای مفقود در ادغام (Left Join):** از الگوی `how='left'` استفاده شد تا تمام کاربران فعال پخش حفظ شوند و خانه‌های خالی ناشی از عدم وجود سابقه مالی در جدول دوم، با متد **fillna()** مقداردهی استاندارد گردند.
- **نتیجه خروجی:** یک دیتافریم یکپارچه، منسجم و جامع تولید شد که هم‌زمان اطلاعات رفتاری و مالی کاربران را برای تحلیل‌های هوش مصنوعی ارائه می‌دهد.

📌 2.5 Data transformation

2.5.1. Normalization

نرمال‌سازی داده 🔧

نرمال‌سازی داده یکی از مهم‌ترین مراحل **پیش‌پردازش داده** است که با هدف **یکسان‌سازی مقیاس** ویژگی‌های عددی انجام می‌شود. در این فرایند، مقادیر ویژگی‌ها به یک بازه یا مقیاس مشترک، مانند **[0,1]**تبدیل می‌شوند تا تفاوت در **واحد اندازه‌گیری** یا **دامنه مقادیر**، بر نتایج تحلیل تأثیر نگذارد.

نرمال‌سازی موجب می‌شود تمامی ویژگی‌ها **سهم متعادلی** در فرایند تحلیل داشته باشند و از **غالب شدن** ویژگی‌هایی با دامنه مقادیر بزرگ‌تر جلوگیری شود. این روش به‌ویژه در الگوریتم‌های مبتنی بر فاصله، **خوشه‌بندی**، **طبقه‌بندی** و **شبکه‌های عصبی**، نقش مهمی در افزایش دقت، بهبود عملکرد و تسریع فرایند یادگیری ایفا می‌کند.

📊 **کاربرد عملی:** فرض کنید قصد دارید مدلی برای پیش‌بینی قیمت خانه طراحی کنید. ویژگی‌های شما شامل **متراژ** (مقیاس ۵۰ تا ۵۰۰ متر) و **تعداد اتاق‌ها** (مقیاس ۱ تا ۵) است. بدون نرمال‌سازی، الگوریتم‌های مبتنی بر فاصله (مانند **k-NN** و **SVM**) بیشتر تحت تأثیر متراژ قرار می‌گیرند و تعداد اتاق‌ها تأثیر ناچیزی در مدل خواهند داشت. با **نرمال‌سازی**، هر دو ویژگی در یک مقیاس قرار می‌گیرند و مدل به‌درستی اهمیت هر دو را در نظر می‌گیرد.

📐 ریاضیات — نرمال‌سازی داده

Min-Max Normalization

$$v_i' = \frac{v_i - \min_A}{\max_A - \min_A}(new_{\max A} - new_{\min A}) + new_{\min A} \tag{2.50}$$

نمادها

- v_i': مقدار نرمال‌شده.
- v_i: مقدار اصلی.
- $\min_A, \max_A$: کمینه و بیشینه ویژگی A.
- $new_{\min A}, new_{\max A}$: حدود بازه جدید.
- i: اندیس مشاهده.

کسر اول جایگاه نسبی مقدار را در بازه اصلی پیدا می‌کند و ضرب در بازه جدید و سپس جمع حد پایین، همان جایگاه را به مقیاس جدید منتقل می‌کند.

مثال / تمرین حل‌شده

صورت سؤال: با کمینه ۱۲، بیشینه ۹۸ و مقدار ۷۳.۶، مقدار Min-Max را در بازه $[0, 1]$ محاسبه کنید.

حل: $v' = (73.6 - 12)/(98 - 12) = 61.6/86 \approx 0.716$.

توضیح: عدد ۷۳.۶ تقریباً در ۷۱.۶ درصد مسیر بین کمینه و بیشینه قرار دارد؛ بنابراین مقدار نرمال‌شده آن حدود ۰.۷۱۶ است.

Z-Score Normalization

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A} \tag{2.51}$$

نمادها

- v_i': مقدار استانداردشده.
- v_i: مقدار اصلی.

- $\bar{A}$: میانگین ویژگی.
- σ_A: انحراف معیار ویژگی.
- i: اندیس.

مثال / تمرین حل‌شده

صورت سؤال: اگر میانگین یک ویژگی ۵۴، انحراف معیار ۱۶ و مقدار مشاهده ۷۳.۶ باشد، Z-Score را محاسبه کنید.

حل: ابتدا اختلاف از میانگین را می‌یابیم: 19.6 = 54 - 73.6. سپس 1.225 = 19.6/16. بنابراین Z-Score برابر ۱.۲۲۵ است.

توضیح: مقدار مثبت یعنی مشاهده بالاتر از میانگین است و مقدار ۱.۲۲۵ نشان می‌دهد حدود ۱.۲۲۵ انحراف معیار بالاتر از میانگین قرار دارد.

Mean Absolute Deviation

$$s_A = \frac{1}{N} \sum_{i=1}^{N} |x_i - \bar{A}| \tag{2.52}$$

نمادها

- s_A: میانگین قدرمطلق انحراف‌ها.
- x_i: مشاهده.
- $\bar{A}$: میانگین.
- N: تعداد مشاهدات.

$$v_i' = \frac{v_i - \bar{A}}{s_A} \tag{2.53}$$

نمادها

- v_i': مقدار نرمال‌شده.
- v_i: مقدار اصلی.
- $\bar{A}$: میانگین.
- s_A: Mean Absolute Deviation.

Decimal Scaling

$$v_i' = \frac{v_i}{10^j} \tag{2.54}$$

نمادها

- v_i': مقدار نرمال‌شده.
- v_i: مقدار اصلی.

- j: کوچک‌ترین عدد صحیحی که مقدار مطلق همه داده‌های نرمال‌شده را کمتر از 1 می‌کند.

مثال / تمرین حل‌شده

صورت سؤال: بیشترین قدرمطلق یک مجموعه داده ۹۸۶ است. در Decimal Scaling چه jای انتخاب می‌کنیم؟

حل: باید j را آن‌قدر بزرگ انتخاب کنیم که $986/10^j < 1$ شود. با j = 3، مقدار 986/1000 = 0.986 است؛ بنابراین j = 3 مناسب است.

توضیح: Decimal Scaling با جابه‌جایی اعشاری مقیاس داده را کوچک می‌کند و ترتیب نسبی مقادیر را تغییر نمی‌دهد.

مثال عملی: نرمال‌سازی ویژگی‌های محصول در «آمازون» (Amazon)

الگوریتم توصیه‌گر «آمازون» برای پیشنهاد محصولات مشابه به کاربران، ویژگی‌های مختلفی از هر کالا را بررسی می‌کند. این ویژگی‌ها مقیاس‌های کاملاً متفاوتی دارند:

- **قیمت (Price):** از ۲۰ دلار تا ۱۰ geometric دلار.
- **وزن (Weight):** از ۰.۱ کیلوگرم تا ۲۰ کیلوگرم.
- **امتیاز کاربران (Rating):** از ۲ تا ۵ ستاره.

چالش بزرگ: اگر ویژگی‌ها را به‌صورت خام وارد الگوریتم کنیم، ویژگی‌هایی با مقیاس بزرگ‌تر (مثل قیمت) به‌طور غیرعادلانه‌ای بر نتیجه تأثیر می‌گذارند و ویژگی‌های با مقیاس کوچک‌تر (مثل امتیاز) کاملاً سرکوب می‌شوند.

راه‌حل — نرمال‌سازی Min-Max:

مقیاس‌بندی Min-Max تمام مقادیر هر ویژگی را به یک بازه ثابت (معمولاً **۰ تا ۱**) تبدیل می‌کند تا تمامی ویژگی‌ها شانس برابری در مدل داشته باشند.

$$\text{Normalized}(X) = \frac{X - X_{min}}{X_{max} - X_{min}}$$

بصری‌سازی با Matplotlib:

با رسم نمودار قبل و بعد از نرمال‌سازی، به‌وضوح مشاهده می‌کنیم که چطور نوسانات ناهمگن داده‌های اولیه (قیمت تا ۱۰۰۰ و وزن تا ۲۰)، پس از مقیاس‌بندی همگی در یک دامنه یکسان بین **۰ و ۱** هم‌تراز می‌شوند.

نتیجه: پس از نرمال‌سازی، تمام ویژگی‌ها در یک مقیاس یکسان قرار می‌گیرند و الگوریتم‌های یادگیری ماشین (مثل K-Nearest Neighbors) می‌توانند به‌درستی شباهت‌ها را تشخیص دهند.

خروجی: تبدیل ویژگی‌های با مقیاس‌های متفاوت به مقیاس واحد [۰, ۱] و رسم بصری مقایسه

In [35]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [36]:

```
display_academic_text(
    "پیش‌پردازش داده‌ها: هم‌ترازی ویژگی‌ها از طریق مقیاس‌بندی کمینه-بیشینه (Min-Max Normalization) 📏",
    "main_title"
)

display_academic_text(
    "مقیاس‌های متفاوت ویژگی‌ها به شدت بر عملکرد الگوریتم‌های مبتنی بر فاصله (نظیر KNN، K-Means یا سیستم‌های توصیه‌گر) تأثیر می‌گذارند. "
    "در این سناریو، ویژگی «قیمت» (با نوسان بسیار بالا از ۲۰ تا ۱۰۰۰) می‌تواند تأثیر ویژگی «امتیاز» (با نوسان بسیار کم از ۲ تا ۵) "
    "را در فضای محاسباتی کاملاً خنثی نماید. استراتژی <b>نرمال‌سازی Min-Max</b> با انتقالِ هندسیِ تمامی مقادیر به یک بازه مشترک $X_{norm} \in [0, 1]$، "
    "تضمین می‌کند که هر ویژگی وزنِ محاسباتیِ عادلانه‌ای در مدل یادگیری ماشین داشته باشد.",
    "paragraph"
)

display_academic_text("معادله نرمال‌سازی Min-Max (تغییر مقیاس خطی):", "sub_title")

# Utilizing Raw Strings (r"") for correct LaTeX rendering of the normalization formula
display_academic_text(
    r"$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: DATA INITIALIZATION AND NORMALIZATION TRANSFORMATION
# ==========================================================
display_academic_text(
    "مرحله نخست: بارگذاری داده‌های ناهمگن و اعمال تبدیل Min-Max",
    "sub_title"
)

display_academic_text(
    "در این فاز، داده‌های خامِ محصولات با مقیاس‌های کاملاً متمایز بارگذاری می‌شوند. سپس با بهره‌گیری از کلاس <code>MinMaxScaler</code>، "
    "تمامی ویژگی‌های عددی به طور مستقل پردازش شده و به <b>بازه استاندارد ۰.۰ تا ۱.۰</b> منتقل می‌گردند. حفظِ تناسبِ مقادیر (عدم تخریب توزیع داخلی داده‌ها) "
    "از مهم‌ترین ویژگی‌های این تبدیل ریاضیاتی خطی است.",
    "paragraph"
)

# Step 1: Initialize the Amazon product dataset with vastly different scaling magnitudes
data = {
    'Product_ID': ['P101', 'P102', 'P103', 'P104', 'P105'],
    'Price_USD': [50.0, 120.0, 800.0, 20.0, 1000.0],  # Massive scale: [20, 1000]
    'Weight_KG': [0.5, 1.2, 15.0, 0.1, 20.0],         # Medium scale: [0.1, 20]
    'Rating': [4.0, 3.0, 5.0, 2.0, 4.0],              # Micro scale: [2, 5]
}
df_raw = pd.DataFrame(data)

# Step 2: Isolate the numeric features that require mathematical scaling
numeric_columns = ['Price_USD', 'Weight_KG', 'Rating']

# Step 3: Initialize the scaler and apply the fit_transform pipeline
# MinMaxScaler subtracts the minimum and divides by the range (max - min) for each column independently
scaler = MinMaxScaler()
df_normalized = df_raw.copy()
df_normalized[numeric_columns] = scaler.fit_transform(df_raw[numeric_columns])

# Display preparation to prevent empty index columns in HTML
df_raw_disp = df_raw.copy()
df_norm_disp = df_normalized.copy()

# Render the raw heterogeneous dataset and the standardized dataset side-by-side
display_side_by_side_datasets(
    [df_raw_disp, df_norm_disp],
    [
        "جدول ۱: پایگاه داده خام محصولات (مقیاس‌های به‌شدت نامتجانس)",
        "جدول ۲: داده‌های نرمال‌سازی شده (هم‌ترازی در بازه استاندارد ۰ تا ۱)"
    ]
)

# ==========================================================
# SECTION 3: ANALYTICAL VISUALIZATION (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - مقایسه بصری توزیع داده‌ها پیش و پس از تغییر مقیاس",
    "sub_title"
)

display_academic_text(
    "بوم گرافیکی زیر تأثیرِ چشمگیرِ نرمال‌سازی را به تصویر می‌کشد. در نمودار سمت چپ، سیطرهٔ مقادیرِ نجومیِ «قیمت» به حدی است که نوساناتِ متغیرهای "
    "«وزن» و «امتیاز» به صورت خطوطی کاملاً صاف در کفِ محور مختصات فشرده و محو شده‌اند (Scale Distortion).<br><br>"
    "در نمودار سمت راست، پس از انتقال هندسیِ داده‌ها به فضای $[0, 1]$، مشاهده می‌شود که تمامی ویژگی‌ها استقلال و پویاییِ نوسانیِ خود را بازیافته‌اند و "
    "الگوریتم‌های هوش مصنوعی اکنون قادرند الگوهای پنهان هر سه متغیر را به صورت هم‌زمان درک نمایند.",
    "paragraph"
)

# Initialize a wide dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# Define an explicit Academic Blue color palette for consistent feature tracking
colors = {'Price_USD': '#2c3e50', 'Weight_KG': '#2980b9', 'Rating': '#1abc9c'} # Navy, Classic Blue, Turquoise
markers = {'Price_USD': 'o', 'Weight_KG': 's', 'Rating': '^'}
line_styles = {'Price_USD': '-', 'Weight_KG': '--', 'Rating': ':'}

# ----------------------------------------------------------
# Plot 1: Original Data Scale (Feature Suppression)
# ----------------------------------------------------------
for col in numeric_columns:
    axes[0].plot(
        df_raw['Product_ID'],
        df_raw[col],
        marker=markers[col],
        markersize=12,
        linewidth=3.0,
        color=colors[col],
```

```
        linestyle=line_styles[col],
        label=col,
        alpha=0.9
    )

axes[0].annotate(
    "Feature Suppression:\nThe Massive scale of 'Price' (up to $1000)\ncompletely crushes the 'Rating' line (2 to 5)\ninto a flat, indistinguishable line at the bottom.",
    xy=(2.0, 5.0), xytext=(0.5, 350),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Raw Data Topology (Severe Scale Distortion)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Product Identifiers', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Absolute Raw Values', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xticklabels(df_raw['Product_ID'], fontweight='bold', fontsize=11)
axes[0].legend(title='Dataset Features', loc='upper left', frameon=True, fontsize=11)
axes[0].grid(True, linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# Plot 2: Normalized Data Scale (Feature Alignment)
# ---------------------------------------------------------
for col in numeric_columns:
    axes[1].plot(
        df_normalized['Product_ID'],
        df_normalized[col],
        marker=markers[col],
        markersize=12,
        linewidth=3.0,
        color=colors[col],
        linestyle=line_styles[col],
        label=col,
        alpha=0.9
    )

axes[1].annotate(
    "Hidden Dynamics Revealed:\nAfter Min-Max Scaling, we clearly see that\nP103 (Index 2) actually has the Highest Rating (1.0)\ndespite being the 2nd most expensive item.",
    xy=(2.0, 1.0), xytext=(0.1, 0.75),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Normalized Topology (Equal Representation in [0, 1])', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Product Identifiers', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Scaled Mathematical Value (Min=0.0, Max=1.0)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylim(-0.05, 1.05) # Bound y-axis strictly around [0, 1]
axes[1].set_xticklabels(df_normalized['Product_ID'], fontweight='bold', fontsize=11)
axes[1].legend(title='Dataset Features', loc='center left', bbox_to_anchor=(1, 0.5), frameon=True, fontsize=11)
axes[1].grid(True, linestyle='--', alpha=0.5)

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br><br>"
    r"به همراه دارد <b>خطای تحلیلی شدیدی</b>، تغذیهٔ داده‌های خام با مقیاس‌های نامتجانس (مثلاً ترکیبِ قیمتِ ۱۰۰۰ دلاری با امتیازِ ۲) به الگوریتم‌های داده‌کاوی. "
    r"ماشین تصور می‌کند که «قیمت» ۵۰۰ برابر مهم‌تر از «امتیاز» است، در حالی که این صرفاً تفاوت در واحد اندازه‌گیری است!<br><br>"
    r"قرار می‌دهد $X_{norm} \in [0, 1]$ با اجرای یک انتقال هندسیِ خطی، بدون تخریب توزیعِ درونی و نوساناتِ داده‌ها، تمام ویژگی‌ها را در یک جبههٔ تحلیلی هم‌سطح <b>Min-Max</b> نرمال‌سازی <b>استراتژی "
    r"می‌باشد K-Means (نظیر) و روش‌های خوشه‌بندی مبتنی بر فاصله (Deep Learning) شبکه‌های عصبی عمیق، (Recommender Systems) این پیش‌پردازش، پیش‌نیازِ قطعیِ استقرار الگوریتم‌های توصیه‌گر (گراف ۲)."
)
display_academic_text(insight_conclusion, "insight")
```

پیش‌پردازش داده‌ها: هم‌ترازی ویژگی‌ها از طریق مقیاس‌بندی کمینه-بیشینه (Min-Max Normalization) 📏

مقیاس‌های متفاوت ویژگی‌ها به شدت بر عملکرد الگوریتم‌های مبتنی بر فاصله (نظیر K-Means، KNN یا سیستم‌های توصیه‌گر) تأثیر می‌گذارند. در این سناریو، ویژگی «قیمت» (با نوسان بسیار بالا از ۲۰ تا ۱۰۰۰) می‌تواند تأثیر ویژگی «امتیاز» (با نوسان بسیار کم از ۲ تا ۵) را در فضای محاسباتی کاملاً خنثی نماید. استراتژی **نرمال‌سازی Min-Max** با انتقالِ هندسیِ تمامی مقادیر به یک بازه مشترک $X_{norm} \in [0, 1]$، تضمین می‌کند که هر ویژگی وزنِ محاسباتیِ عادلانه‌ای در مدل یادگیری ماشین داشته باشد.

معادله نرمال‌سازی Min-Max (تغییر مقیاس خطی):

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

مرحله نخست: بارگذاری داده‌های ناهمگن و اعمال تبدیل Min-Max

در این فاز، داده‌های خامِ محصولات با مقیاس‌های کاملاً متمایز بارگذاری می‌شوند. سپس با بهره‌گیری از کلاس `MinMaxScaler`، تمامی ویژگی‌های عددی به طور مستقل پردازش شده و به **بازه استاندارد ۰.۰ تا ۱.۰** منتقل می‌گردند. حفظِ تناسبِ مقادیر (عدم تخریب توزیع داخلی داده‌ها) از مهم‌ترین ویژگی‌های این تبدیلِ ریاضیاتیِ خطی است.

جدول ۱: پایگاه داده خام محصولات (مقیاس‌های به‌شدت نامتجانس)

Product_ID	Price_USD	Weight_KG	Rating
P101	50.0	0.5	4.0
P102	120.0	1.2	3.0
P103	800.0	15.0	5.0
P104	20.0	0.1	2.0
P105	1000.0	20.0	4.0

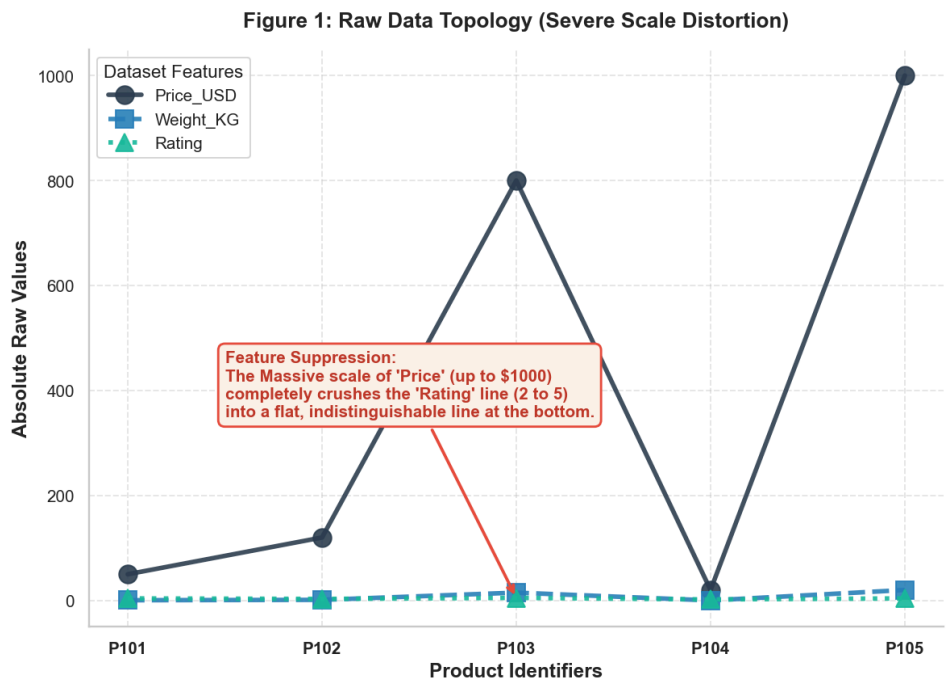
جدول ۲: داده‌های نرمال‌سازی شده (هم‌ترازی در بازه استاندارد ۰ تا ۱)

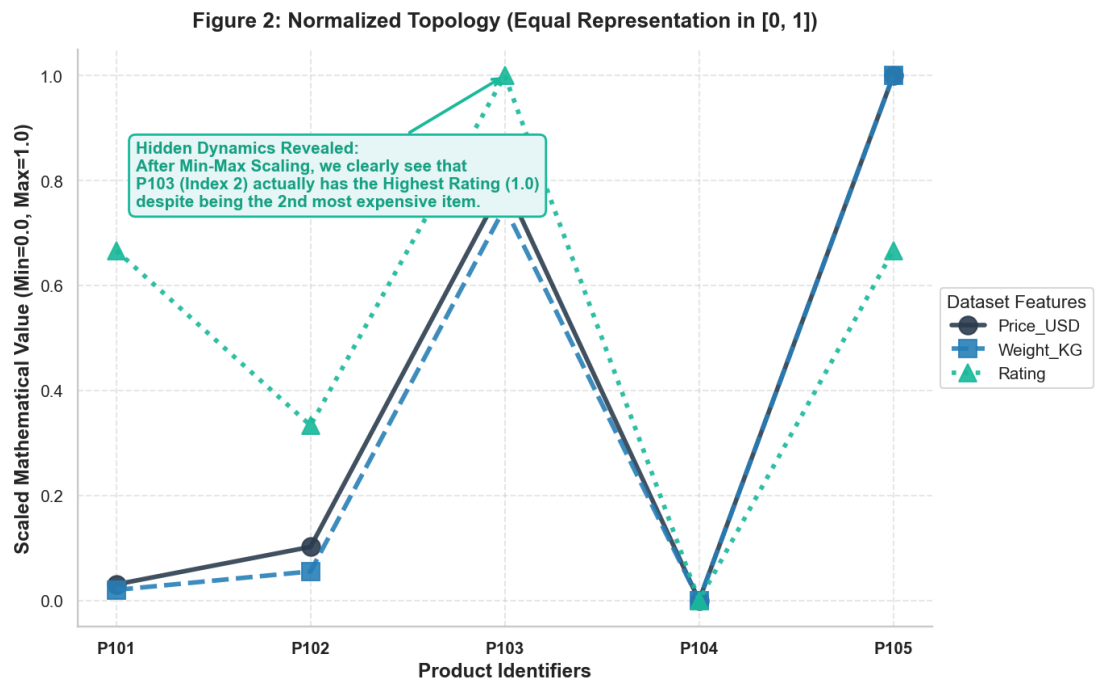
Product_ID	Price_USD	Weight_KG	Rating
P101	0.0	0.0	0.7
P102	0.1	0.1	0.3
P103	0.8	0.7	1.0
P104	0.0	0.0	0.0
P105	1.0	1.0	0.7

مرحله دوم: مصورسازی تحلیلی - مقایسه بصری توزیع داده‌ها پیش و پس از تغییر مقیاس

بوم گرافیکی زیر تأثیرِ چشمگیرِ نرمال‌سازی را به تصویر می‌کشد. در نمودار سمت چپ، سیطرهٔ مقادیرِ نجومیِ «قیمت» به حدی است که نوساناتِ متغیرهای «وزن» و «امتیاز» به صورت خطوطی کاملاً صاف در کفِ محور مختصات فشرده و محو شده‌اند (Scale Distortion).

در نمودار سمت راست، پس از انتقال هندسیِ داده‌ها به فضای [0, 1]، مشاهده می‌شود که تمامی ویژگی‌ها استقلال و پویاییِ نوسانیِ خود را بازیافته‌اند و الگوریتم‌های هوش مصنوعی اکنون قادرند الگوهای پنهان هر سه متغیر را به صورت هم‌زمان درک نمایند.





استنتاج راهبردی عملیات (Strategic Implication):

تغذیهٔ داده‌های خام با مقیاس‌های نامتجانس (مثلاً ترکیبِ قیمتِ ۱۰۰۰ دلاری با امتیازِ ۲) به الگوریتم‌های داده‌کاوی، خطای تحلیلی شدیدی به همراه دارد. ماشین تصور می‌کند که «قیمت» ۵۰۰ برابر مهم‌تر از «امتیاز» است، در حالی که این صرفاً تفاوت در واحد اندازه‌گیری است!

استراتژی نرمال‌سازی Min-Max با اجرای یک انتقال هندسیِ خطی، بدون تخریب توزیعِ درونی و نوساناتِ داده‌ها، تمام ویژگی‌ها را در یک جبههٔ تحلیلی هم‌سطح $X_{norm} \in [0, 1]$ قرار می‌دهد (گراف ۲). این پیش‌پردازش، پیش‌نیازِ قطعیِ استقرار الگوریتم‌های توصیه‌گر (Recommender Systems)، شبکه‌های عصبی عمیق (Deep Learning) و روش‌های خوشه‌بندی مبتنی بر فاصله (نظیر K-Means) می‌باشد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از ویژگی‌های محصولات آزمون ساختیم و سپس با استفاده از کلاس **MinMaxScaler** مقادیر را به بازه **[۰, ۱]** نرمال‌سازی کردیم.
- برای بصری‌سازی تأثیر نرمال‌سازی، از کتابخانه **matplotlib.pyplot** و متد **plt.subplots(1, 2)** برای رسم دو نمودار در کنار هم استفاده نمودیم:

- **نمودار سمت چپ (Original Data):** داده‌های خام را رسم می‌کند. در این نمودار مشخص است که ستون قیمت (تا ۱۰۰۰) کل نمودار را تصاحب کرده و منحنی‌های وزن و امتیاز در پایین‌ترین بخش افق کاملاً تخت و نامشخص شده‌اند.
- **نمودار سمت راست (Normalized Data):** داده‌های نرمال‌شده را رسم می‌کند. مشاهده می‌شود تمام منحنی‌ها در بازه یکسان **[۰, ۱]** هم‌تراز شده‌اند و نوسانات همه ویژگی‌ها به‌وضوح و با وزن یکسان قابل دیدن است.

- افزودن دستور **plt.grid()** و نشانگرها (`marker='o'` و `marker='s'`) به خوانایی بهتر نقاط داده روی نمودار کمک کرده است.

2.5.2. Discretization

گسسته‌سازی داده ✂️

گسسته‌سازی داده (Data Discretization) به فرایند تبدیل **ویژگی‌های پیوسته (Continuous)** به مجموعه‌ای محدود از **بازه‌ها** یا **دسته‌های گسسته** گفته می‌شود. بسیاری از الگوریتم‌های داده‌کاوی، به‌ویژه روش‌های مبتنی بر **قوانین** یا **درخت تصمیم**، با داده‌های دسته‌ای بهتر عمل می‌کنند یا اساساً نیازمند ورودی گسسته هستند؛ ازاین‌رو، گسسته‌سازی نقشی مهم در **آماده‌سازی داده** برای این‌گونه الگوریتم‌ها ایفا می‌کند. علاوه بر این، گسسته‌سازی می‌تواند با کاهش تنوع مقادیر، **درک** و **تفسیر** داده‌ها را نیز برای انسان ساده‌تر کند.

📊 **کاربرد عملی:** در یک سیستم تشخیص پزشکی، ویژگی «فشار خون» با مقادیر عددی پیوسته (مانند ۱۲۰، ۱۳۵، ۱۵۰) را در نظر بگیرید. با استفاده از گسسته‌سازی، این مقادیر را می‌توان به دسته‌های مفهومی مانند **«نرمال»**، **«پیش‌فشار خون»** و **«فشار خون بالا»** تبدیل کرد. این کار هم برای پزشکان قابل‌فهم‌تر است و هم به الگوریتم‌های داده‌کاوی کمک می‌کند تا الگوهای مربوط به هر دسته را بهتر شناسایی کنند. همچنین، گسسته‌سازی باعث **کاهش نویز** و **افزایش سرعت یادگیری** در مدل‌هایی مانند **درخت تصمیم** و **سیستم‌های مبتنی بر قانون** می‌شود.

📐 **ریاضیات — گسسته‌سازی داده**

Equal-Width Binning

$$w = \frac{x_{max} - x_{min}}{K} \tag{2.55}$$

نمادها

- w: عرض هر Bin.
- x_{max}: بیشینه.
- x_{min}: کمینه.
- K: تعداد Binها.

در Equal-Width Binning کل بازه به K قسمت با عرض یکسان تقسیم می‌شود.

مثال / تمرین حل‌شده

صورت سؤال: داده‌ها از ۱۰ تا ۴۰ گسترده‌اند و می‌خواهیم آن‌ها را به ۳ Bin با عرض یکسان تقسیم کنیم. عرض هر Bin و بازه‌ها را تعیین کنید.

حل: عرض Bin برابر $(40 - 10)/3 = 10$ است. بنابراین می‌توان بازه‌ها را به‌صورت $[10, 20)$، $[20, 30)$ و $[30, 40]$ در نظر گرفت.

توضیح: در Equal-Width تعداد داده‌های هر Bin لزوماً برابر نیست؛ فقط عرض بازه‌ها یکسان است.

Equal-Frequency Binning

$$\text{approximately} \quad n_k \approx \frac{N}{K} \tag{2.56}$$

نمادها

- n_k: تعداد داده‌های k Bin.
- N: تعداد کل داده‌ها.
- K: تعداد Binها.

در Equal-Frequency هدف این است که هر Bin تقریباً تعداد یکسانی مشاهده داشته باشد.

مثال / تمرین حل‌شده

صورت سؤال: ۱۲ مشاهده داریم و می‌خواهیم آن‌ها را تقریباً به ۳ Bin با فراوانی برابر تقسیم کنیم. در هر Bin حدود چند مشاهده قرار می‌گیرد؟

حل: با رابطه $n_k \approx N/K$، داریم $12/3 = 4$. بنابراین هدف این است که هر Bin حدود ۴ مشاهده داشته باشد.

توضیح: در Equal-Frequency عرض بازه‌ها الزاماً برابر نیست؛ چیزی که تا حد ممکن برابر نگه داشته می‌شود تعداد مشاهده‌های هر Bin است.

مثال عملی: دسته‌بندی زمان فیلم‌ها در «نتفلیکس» (Netflix) با سطل‌بندی عرض یکسان ✂️ 📊

الگوریتم‌های پیشنهاد فیلم در «نتفلیکس» با متغیرهای عددی پیوسته‌ای مانند **مدت زمان فیلم (Duration به دقیقه)** روبه‌رو هستند که می‌تواند هر عددی بین ۶۰ تا ۱۸۰ دقیقه باشد. برخی الگوریتم‌ها (مثل درخت تصمیم یا قوانین وابستگی) به جای اعداد پیوسته دقیق، با دسته‌های کیفی عملکرد بسیار بهتری دارند.

راه‌حل — گسسته‌سازی با سطل‌بندی عرض یکسان (Equal Width Binning):

در این روش، کل دامنه داده‌ها $(X_{max} - X_{min})$ به تعداد مشخصی بازه (k) با **عرض کاملاً برابر** تقسیم می‌شود:

$$\text{Bin Width} = \frac{X_{max} - X_{min}}{k}$$

مثال عددی در نتفلیکس:

اگر کوتاه ترین فیلم ۶۰ دقیقه (X_{min}) و طولانی‌ترین ۱۸۰ دقیقه (X_{max}) باشد و بخواهیم ۳ دسته $(k = 3)$ بسازیم:

- **عرض هر سطل:** $(180 - 60)/3 = 40$ دقیقه.
- **سطل اول (کوتاه / Short):** ۶۰ تا ۱۰۰ دقیقه.
- **سطل دوم (متوسط / Medium):** ۱۰۰ تا ۱۴۰ دقیقه.
- **سطل سوم (طولانی / Long):** ۱۴۰ تا ۱۸۰ دقیقه.

نتیجه: با تبدیل داده پیوسته به ۳ دسته گسسته، سیستم نتفلیکس می‌تواند به‌راحتی ترجیحات کاربر (مثلاً «علاقه‌مند به فیلم‌های کوتاه») را استخراج کند.

خروجی: تبدیل متغیر پیوسته به دسته‌های گسسته با فواصل عددی یکسان

In [37]:
```
display_academic_text(
    "تبدیل داده‌ها: گسسته‌سازی متغیرهای پیوسته با سطل‌بندی عرض یکسان (Equal Width Binning) 🎬",
    "main_title"
)

display_academic_text(
    "به ویژه مدل‌های مبتنی بر قوانین (Rule-based مانند Apriori یا Decision Trees) / بر مبنای اصول پیش‌پردازش داده‌ها، بسیاری از الگوریتم‌ها "
    "نیازمند دریافت ورودی‌های <b>دسته‌ای (Categorical)</b> (اعشاری) ممکن حالتِ بی‌نهایت با «مدت زمان فیلم» متغیر پیوستهٔ در این سناریو، هستند. "
    "سطل‌بندی با عرض یکسان <b>(Equal Width Binning)</b>، باید به دسته‌های معنایی تبدیل گردد. متد دامنه نوسان داده‌ها را به $k$ بازه "
    "تقسیم می‌نماید <i>فواصلِ عددیِ کاملاً برابر</i> با.",
    "paragraph"
)

display_academic_text(
    "بر اساس دامنهٔ کل داده‌ها ($W$) معادله محاسبه عرض هر سطل:",
    "sub_title"
)

# Utilizing Raw Strings (r"") for correct LaTeX rendering of the Binning formula
display_academic_text(
    r"$$W = \frac{X_{max} - X_{min}}{k}$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: DATA INITIALIZATION AND DISCRETIZATION PIPELINE
# ==========================================================
display_academic_text(
    "مرحله نخست: تولید داده‌های پیوسته و استخراج فواصل گسسته",
    "sub_title"
)

display_academic_text(
    "در این فاز، فهرستی از فیلم‌ها با زمان‌های تصادفی بین ۶۰ تا ۱۸۰ دقیقه شبیه‌سازی می‌گردد. "
    "تقسیم شده <b>عرضِ دقیقاً ۴۰ دقیقه</b> دامنهٔ فیزیکی (۱۲۰ دقیقه) به ۳ سطل با، <code>pd.cut</code> سپس با بهره‌گیری از متد قدرتمند "
    "متناسب با سطل تعلق می‌گیرد (Label) و به هر رکورد، برچسب کیفی.",
    "paragraph"
)

# Step 1: Initialize the dataset with strictly continuous movie durations
n_movies = 150 # Increased sample size for better visual distribution
data = {
    'Movie_ID': [f"MOV_{i}" for i in range(1001, 1001 + n_movies)],
    # Generate uniform continuous durations between exactly 60 & 180 minutes
    'Duration_Min': np.random.uniform(60, 180, n_movies),
}
df_raw = pd.DataFrame(data)

# Step 2: Apply Equal Width Binning using pd.cut()
k = 3
# Adding logical numerical ranges to labels for extreme clarity
labels = ['Short (60 - 100m)', 'Medium (100 - 140m)', 'Long (140 - 180m)']

# Perform Discretization and automatically extract mathematical bin boundaries (retbins=True)
df_binned = df_raw.copy()
df_binned['Duration_Category'], bin_edges = pd.cut(
    df_binned['Duration_Min'],
    bins=k,
    labels=labels,
    retbins=True
)

# Step 3: Compute the frequency distribution (Number of movies in each bin)
df_bin_counts = df_binned['Duration_Category'].value_counts().sort_index().reset_index()
df_bin_counts.columns = ['Category_Label', 'Movie_Count']

# Render datasets cleanly
df_raw_disp = df_raw.head(8).reset_index(drop=True)
df_binned_disp = df_binned.head(8).reset_index(drop=True)

display_side_by_side_datasets(
    [df_raw_disp, df_binned_disp, df_bin_counts],
    [
        "جدول ۱: داده‌های پیوسته (نمونه خام)",
        "جدول ۲: داده‌های گسسته‌شده (Discretized)",
        "جدول ۳: ماتریس توزیع فراوانی سطل‌ها"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - انتقال از پیوستگی به گسستگی",
    "sub_title"
)

display_academic_text(
```

```
    "توزیع پیوستهٔ فیلم‌ها مشاهده می‌شود که توسط <b>هیستوگرام سمت چپ</b>، نمودار زیر ماهیت هندسیِ این انتقال را به تصویر می‌کشد. در "
    "قطعه‌قطعه شده است <i>کاملاً مساوی (از نظر عرض)</i> خطوط‌چین قرمز (نقاط برش الگوریتم) به ۳ ناحیهٔ. "
    "نیست <i>«زمانِ پیوسته»</i>، نتیجه نهاییِ این تبدیل نمایش داده می‌شود: محور افقی دیگر از جنس <b>نمودار میله‌ای سمت راست</b>، در "
    "تقلیل یافته است <i>«متنی و کیفی»</i>. بلکه به ۳ ماهیتِ",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# ---------------------------------------------------------
# Plot 1: Continuous Histogram with Explicit Bin Boundaries
# ---------------------------------------------------------
sns.histplot(
    data=df_raw,
    x='Duration_Min',
    bins=15, # Use higher resolution bins for the underlying continuous data
    color='#2c3e50', # Navy
    edgecolor='white',
    linewidth=1.0,
    alpha=0.8,
    ax=axes[0]
)

# Overlay vertical threshold lines to demonstrate the mathematical cut points
# We skip the very first (min) and last (max) edges to just show the internal cuts
for b in bin_edges[1:-1]:
    axes[0].axvline(
        x=b,
        color='#e74c3c', # Red for the cut lines
        linestyle='--',
        linewidth=3.0,
        label=f'Algorithm Cut Threshold: {b:.1f}m' if b == bin_edges[1] else ""
    )

axes[0].annotate(
    "Equal Width Logic:\nRange = 180 - 60 = 120\nWidth (W) = 120 / 3 = 40 min.\nThe algorithm blindly slices the X-axis\nevery 40 minutes.",
    xy=(100, 15), xytext=(120, 15),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title('Figure 1: Continuous Topology & Equal Width Cut Points', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Exact Duration (Minutes)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Absolute Frequency', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_xlim(50, 190)
axes[0].legend(loc='upper right', frameon=True, fontsize=11)
axes[0].grid(True, linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# Plot 2: Categorical Bar Chart (Discretized Results)
# ---------------------------------------------------------
# Using hue='Category_Label' and legend=False strictly to comply with seaborn v0.14.0 requirements
sns.barplot(
    data=df_bin_counts,
    x='Category_Label',
    y='Movie_Count',
    hue='Category_Label',
    legend=False,
    palette=['#1abc9c', '#2980b9', '#2c3e50'], # Turquoise, Classic Blue, Navy
    edgecolor='#2c3e50',
    linewidth=2.0,
    width=0.6,
    ax=axes[1]
)

axes[1].annotate(
    "Dimensionality Reduction:\nThe machine no longer cares if a movie\nis 101 or 139 minutes long. They are\nall now just treated as 'Medium'.",
    xy=(1.0, 50), xytext=(0.5, 65),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Discretized Categorical Target (Final Output)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Semantic Duration Category (Discrete)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Number of Movies', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].grid(axis='y', linestyle='--', alpha=0.5)
axes[1].set_ylim(0, 80)
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=12)

# Add exact value annotations on top of the bars
for p in axes[1].patches:
    if p.get_height() > 0:
        axes[1].annotate(
            f"n={int(p.get_height())}",
            (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='center', xytext=(0, 12), textcoords='offset points',
            fontweight='bold', fontsize=12, color='#2c3e50'
        )

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
```

```
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"در برابر نویز است (Robustness) و مقاوم‌سازیِ مدل‌ها (Dimensionality Reduction) ابزاری قدرتمند برای تقلیل پیچیدگی فضای ابعاد، گسسته‌سازی (Discretization). "
    r"با تبدیل دامنهٔ پیوسته به مفاهیم کیفی، نویزهای جزئیِ سنسورها یا تفاوت‌های بی‌اهمیت (مثلاً تفاوتِ ارزشِ یک فیلم ۱۱۵ دقیقه‌ای با یک فیلم ۱۱۸ دقیقه‌ای) نادیده گرفته می‌شوند.<br>"
    r"را با سرعت پردازشی بسیار بالاتر (<i>بلند</i> یا <i>کوتاه</i> علاقهٔ مطلق به فیلم‌های) در نتیجه، سیستم توصیه‌گر قادر می‌گردد تا الگوهای کلان رفتاری کاربران، "
    r"و بدونِ درگیر شدن در محاسبات سنگینِ اعشاری، کشف و استخراج نماید."
)
display_academic_text(insight_conclusion, "insight")
```

تبدیل داده‌ها: گسسته‌سازی متغیرهای پیوسته با سطل‌بندی عرض یکسان (Equal Width Binning) 🎬

بر مبنای اصول پیش‌پردازش داده‌ها، بسیاری از الگوریتم‌ها (به ویژه مدل‌های مبتنی بر قوانین / Rule-based مانند Apriori یا Decision Trees) نیازمند دریافت ورودی‌های **دسته‌ای (Categorical)** هستند. در این سناریو، متغیر پیوستهٔ «مدت زمان فیلم» با بی‌نهایت حالتِ ممکن (اعشاری)، باید به دسته‌های معنایی تبدیل گردد. متد **سطل‌بندی با عرض یکسان (Equal Width Binning)**، دامنه نوسان داده‌ها را به k بازه با *فواصلِ عددیِ کاملاً برابر* تقسیم می‌نماید.

معادله محاسبه عرض هر سطل (W) بر اساس دامنهٔ کل داده‌ها:

$$W = \frac{X_{max} - X_{min}}{k}$$

مرحله نخست: تولید داده‌های پیوسته و استخراج فواصل گسسته

در این فاز، فهرستی از فیلم‌ها با زمان‌های تصادفی بین ۶۰ تا ۱۸۰ دقیقه شبیه‌سازی می‌گردد. سپس با بهره‌گیری از متد قدرتمند `pd.cut`، دامنهٔ فیزیکی (۱۲۰ دقیقه) به ۳ سطل با **عرضِ دقیقاً ۴۰ دقیقه** تقسیم شده و به هر رکورد، برچسب کیفی (Label) متناسب با سطل تعلق می‌گیرد.

جدول ۱: داده‌های پیوسته (نمونه خام)

Movie_ID	Duration_Min
MOV_1001	104.9
MOV_1002	174.1
MOV_1003	147.8
MOV_1004	131.8
MOV_1005	78.7
MOV_1006	78.7
MOV_1007	67.0
MOV_1008	163.9

جدول ۲: داده‌های گسسته‌شده (Discretized)

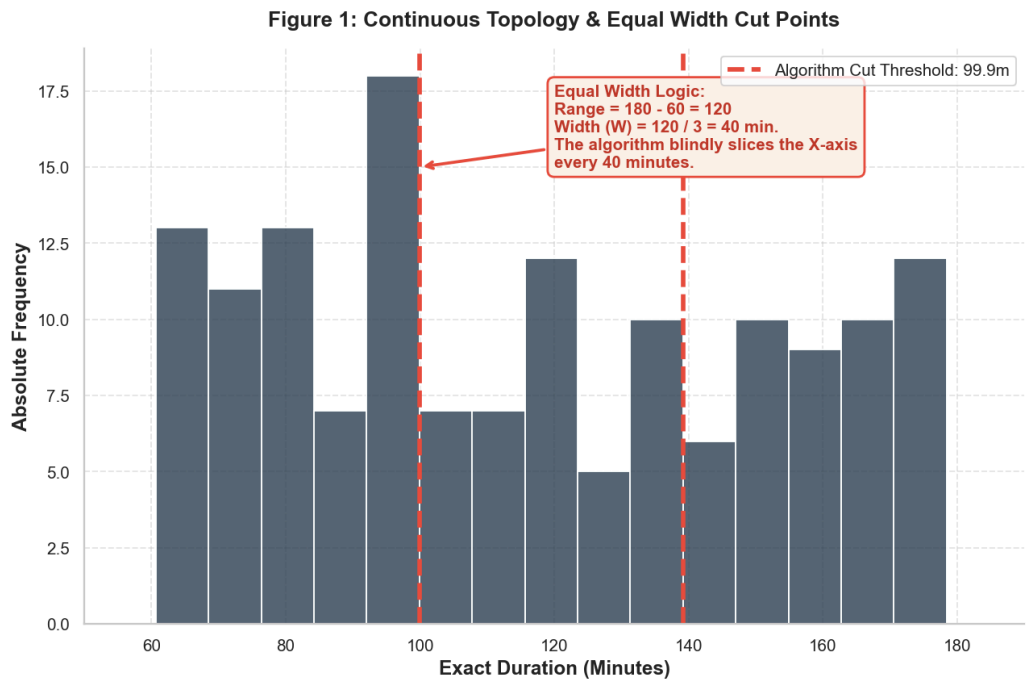
Movie_ID	Duration_Min	Duration_Catego
MOV_1001	104.9	Medium (100 - 14
MOV_1002	174.1	Long (140 - 180r
MOV_1003	147.8	Long (140 - 180r
MOV_1004	131.8	Medium (100 - 14
MOV_1005	78.7	Short (60 - 100n
MOV_1006	78.7	Short (60 - 100n
MOV_1007	67.0	Short (60 - 100n
MOV_1008	163.9	Long (140 - 180r

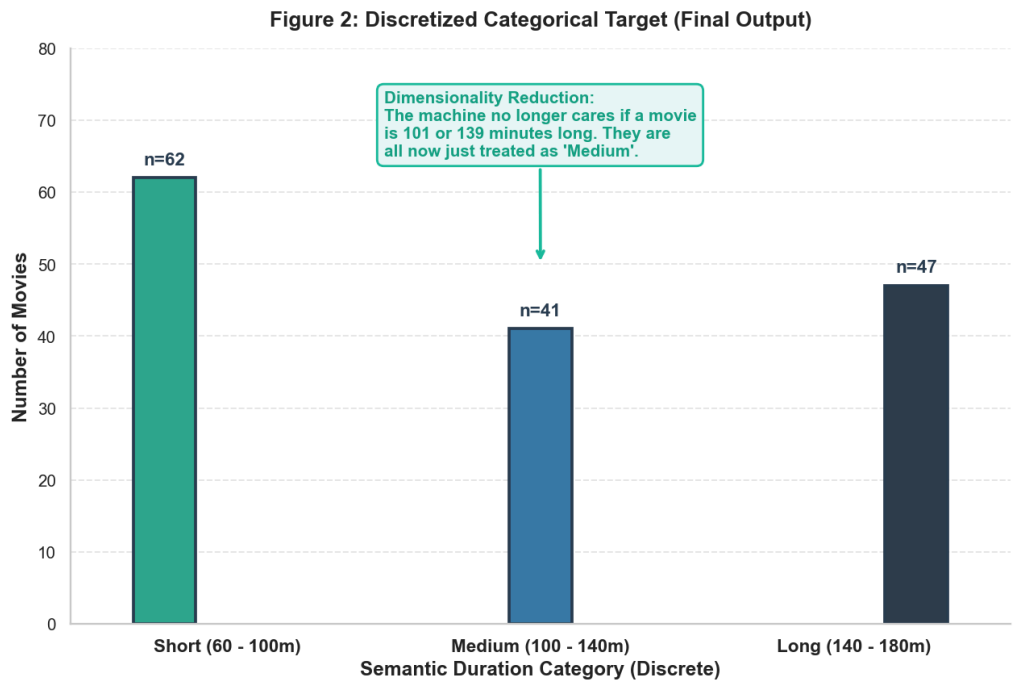
جدول ۳: ماتریس توزیع فراوانی سطل‌ها

Category_Label	Movie_Count
Short (60 - 100m)	62
Medium (100 - 140m)	41
Long (140 - 180m)	47

مرحله دوم: مصورسازی تحلیلی - انتقال از پیوستگی به گسستگی

نمودار زیر ماهیت هندسیِ این انتقال را به تصویر می‌کشد. در **هیستوگرام سمت چپ**، توزیع پیوستهٔ فیلم‌ها مشاهده می‌شود که توسط خطوط‌چین قرمز (نقاط برش الگوریتم) به ۳ ناحیهٔ *کاملاً مساوی (از نظر عرض)* قطعه‌قطعه شده است. در **نمودار میله‌ای سمت راست**، نتیجه نهاییِ این تبدیل نمایش داده می‌شود: محور افقی دیگر از جنس *«زمانِ پیوسته»* نیست، بلکه به ۳ ماهیتِ *«متنی و کیفی»* تقلیل یافته است.





استنتاج راهبردی عملیات (Strategic Implication):

گسسته‌سازی (Discretization)، ابزاری قدرتمند برای تقلیل پیچیدگی فضای ابعاد (Dimensionality Reduction) و مقاوم‌سازیِ مدل‌ها (Robustness) در برابر نویز است. با تبدیل دامنهٔ پیوسته به مفاهیم کیفی، نویزهای جزئیِ سنسورها یا تفاوت‌های بی‌اهمیت (مثلاً تفاوتِ ارزشِ یک فیلم ۱۱۵ دقیقه‌ای با یک فیلم ۱۱۸ دقیقه‌ای) نادیده گرفته می‌شوند.

در نتیجه، سیستم توصیه‌گر قادر می‌گردد تا الگوهای کلان رفتاری کاربران (علاقهٔ مطلق به فیلم‌های *کوتاه* یا *بلند*) را با سرعت پردازشی بسیار بالاتر، و بدونِ درگیر شدن در محاسبات سنگینِ اعشاری، کشف و استخراج نماید.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده شامل ۱۰۰ فیلم نتفلیکس با متغیر پیوسته **Duration_Min** (بین ۶۰ تا ۱۸۰ دقیقه) ساختیم.
- با استفاده از تابع **pd.cut()** در پانداس و تنظیم پارامتر `bins=3` ، دامنه داده‌ها به‌صورت خودکار به ۳ سطل با **عرض برابر** (۴۰ دقیقه برای هر سطل) گسسته‌سازی شد.
- دسته‌های متنی **['Short', 'Medium', 'Long']** جایگزین مقادیر پیوسته گردیدند.
- **تفسیر نمودارها (Matplotlib):**
  - **نمودار سمت چپ (Histogram):** توزیع پیوسته داده‌ها و مرزهای برش سطل‌ها (خطوط عمودی نارنجی در دقیقه‌های ۶۰، ۱۰۰، ۱۴۰ و ۱۸۰) را نشان می‌دهد.
  - **نمودار سمت راست (Bar Chart):** فراوانی فیلم‌ها را پس از تبدیل به ۳ دسته گسسته به تصویر می‌کشد.
- این تبدیل به الگوریتم‌های داده‌کاوی اجازه می‌دهد به جای محاسبات ریاضی روی اعداد اعشاری، قوانین الگویی مانند «کاربران خسته معمولاً فیلم‌های دسته Short را انتخاب می‌کنند» را استخراج نمایند.

2.5.3. Data compression

فشرده‌سازی داده 💾

فشرده‌سازی داده یکی از روش‌های **پیش‌پردازش داده** است که با تبدیل داده‌ها به یک نمایش فشرده‌تر، **حجم** آن‌ها را کاهش می‌دهد و در نتیجه موجب **صرفه‌جویی در فضای ذخیره‌سازی** و **افزایش سرعت پردازش** می‌شود. این فرایند به‌ویژه در مواجهه با مجموعه‌داده‌های حجیم و کلان‌داده، نقشی کلیدی در بهبود کارایی سیستم‌های داده‌کاوی و یادگیری ماشین ایفا می‌کند.

انتخاب بین فشرده‌سازی بدون اتلاف و با اتلاف به **نوع داده**، **حساسیت کاربرد** و **نیاز به دقت** بستگی دارد. در کاربردهای علمی و پزشکی که کوچک‌ترین خطا می‌تواند پیامدهای جدی داشته باشد، فشرده‌سازی بدون اتلاف الزامی است، در حالی که در ذخیره‌سازی و انتقال داده‌های چندرسانه‌ای، فشرده‌سازی با اتلاف با حفظ کیفیت قابل‌قبول، انتخاب اقتصادی‌تری است.

فرض کنید یک سازمان بزرگ، سالانه میلیون‌ها تصویر پزشکی (مانند اسکن‌های MRI و CT) را ذخیره می‌کند. این تصاویر حجم بالایی دارند و نیاز به فضای ذخیره‌سازی زیادی دارند. با استفاده از **فشرده‌سازی بدون اتلاف** می‌توان حجم داده‌ها را کاهش داد بدون اینکه هیچ اطلاعات تشخیصی از دست برود. در مقابل، یک پلتفرم اشتراک ویدئو مانند یوتیوب از **فشرده‌سازی با اتلاف** برای کاهش حجم ویدئوها استفاده می‌کند تا بتواند محتوای بیشتری را با پهنای باند کمتر در اختیار کاربران قرار دهد، در حالی که کیفیت بصری برای اکثر کاربران قابل‌قبول باقی می‌ماند.

📐 ریاضیات — فشرده‌سازی داده

برای سنجش ساده میزان فشرده‌سازی می‌توان نسبت اندازه داده فشرده‌شده به داده اصلی را در نظر گرفت.

$$\text{Compression Ratio} = \frac{S_{compressed}}{S_{original}} \tag{2.57}$$

نمادها

- $S_{compressed}$: اندازه داده فشرده‌شده.
- $S_{original}$: اندازه داده اصلی.

مثال / تمرین حل‌شده

صورت سؤال: حجم داده اصلی ۱۰۰ واحد و حجم داده فشرده‌شده ۴۰ واحد است. نسبت حجم فشرده به اصلی را محاسبه و تفسیر کنید.

حل: $40/100 = 0.4$. بنابراین داده فشرده‌شده ۴۰ درصد حجم اولیه را دارد و ۶۰ درصد از حجم اولیه کاهش یافته است.

توضیح: در این تعریف، هرچه نسبت کوچک‌تر باشد، فشرده‌سازی بیشتری انجام شده است.

💾 مثال عملی: فشرده‌سازی سیگنال‌های بایومتریک در «اپل واچ» (Apple Watch)

«اپل واچ» (Apple Watch) سیگنال‌های بایومتریک مداوم (مثل وضعیت حرکت، ضربان قلب و فعالیت بدنی) را هر ثانیه ثبت می‌کند. در طول یک شب خواب ۸ ساعته، سنسور وضعیت حرکتی ممکن است هزاران سیگنال متوالی و **کاملاً تکراری** با مقدار **«Static» (ثابت)** ثبت کند، چون کاربر در خواب برای دقایق زیادی بی‌حرکت است. ذخیره تمام این مقادیر تکراری به‌صورت جداگانه، فضای زیادی از حافظه واچ را هدر می‌دهد.

راه‌حل — کدگذاری طول اجرا (Run-Length Encoding / RLE):

به‌جای ذخیره هر مقدار به‌صورت جداگانه، فقط **مقدار** و تعداد **تکرارهای متوالی** آن ذخیره می‌شود:

- **داده خام (قبل از فشرده‌سازی):**

  [Static, Static, Static, Static, Active, Active, Static, Static, Static, Static, Static, Active]

  👈 **۱۲ رکورد جداگانه**

- **داده فشرده‌شده (بعد از RLE):**

  [(Active, 1) ,(Static, 5) ,(Active, 2) ,(Static, 4)]

  👈 فقط **۴ رکورد** = صرفه‌جویی **۶۶٪** در فضای ذخیره‌سازی!

کجا RLE بهترین کارایی را دارد؟

- داده‌های حسگر IoT با توالی‌های طولانی تکراری.
- داده‌های تصویر (مثل پیکسل‌های سفید پس‌زمینه در اسکن‌های پزشکی).
- سیگنال‌های دودویی (۰ و ۱) در شبکه‌های مخابراتی.

نتیجه: با RLE، اپل واچ داده‌های بایومتریک را با کسری از فضای حافظه ذخیره می‌کند و در عین حال توانایی بازسازی دقیق سیگنال اصلی را حفظ می‌نماید.

خروجی: کاهش چشمگیر حجم داده‌های متوالی تکراری با نگهداری اطلاعات کامل

In [38]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from itertools import groupby

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [39]:
```
display_academic_text(
    "4️⃣ تقلیل و فشرده‌سازی داده‌ها: ذخیره‌سازی سیگنال‌های بایومتریک با الگوریتم RLE",
    "main_title"
)

display_academic_text(
    "در سیستم‌های اینترنت اشیا (مانند پایشگر خواب در ساعت‌های هوشمند)، سیگنال‌ها با فرکانس بالا ثبت می‌شوند که منجر به تولید <i>توالی‌های طولانی و تکراری</i> "
    "از یک وضعیتِ ثابت می‌گردد. الگوریتم <b>Run-Length Encoding (RLE)</b> با تبدیل هر توالیِ تکراری به یک جفتِ <code>(وضعیت، تعداد تکرار)</code>، "
    "حجم داده‌ها را به شدت کاهش داده و از اتلاف حافظه جلوگیری می‌کند.",
    "paragraph"
)

display_academic_text("معادله محاسبه نرخ فشرده‌سازی (Compression Ratio):", "sub_title")

# Utilizing Raw Strings (r"") for LaTeX rendering
display_academic_text(
    r"$$CR = \left( 1 - \frac{Size_{Compressed}}{Size_{Original}} \right) \times 100$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: DATA INITIALIZATION AND RLE COMPRESSION PIPELINE
# ==========================================================
display_academic_text(
    "مرحله نخست: اعمال فشرده‌سازی بدون اتلاف (Lossless Compression) و بازسازی سیگنال",
    "sub_title"
)

display_academic_text(
    "در این فاز، سیگنالِ حرکتیِ خام به الگوریتم RLE تزریق می‌شود. خروجیِ الگوریتم، مجموعه‌ای از تاپل‌ها (Tuples) است که حجم اشغالیِ حافظه را کاهش می‌دهد. "
    "سپس با اجرای تابع دیکودر (Decoder)، سیگنالِ فشرده‌شده دقیقاً به حالت اولیه می‌شود تا <b>بازسازی (Restore)</b> بدونِ اتلاف بودنِ <i>(Lossless)</i> الگوریتم اثبات گردد.",
    "paragraph"
)

# Step 1: Initialize the raw biometric signal array (Mimicking a 20-second sleep monitoring window)
raw_signal = [
    'Static', 'Static', 'Static', 'Static',
    'Active', 'Active',
    'Static', 'Static', 'Static', 'Static', 'Static',
    'Active',
    'Static', 'Static', 'Static',
    'Active', 'Active', 'Active',
    'Static', 'Static',
]

df_raw = pd.DataFrame({
    'Time_Seq': range(1, len(raw_signal) + 1),
    'Sensor_State': raw_signal
})

# Step 2: Define and apply Run-Length Encoding (RLE)
def run_length_encode(signal):
    """
    Compresses a sequence using RLE.
    Groups identical consecutive elements and yields (Element, Count) tuples.
    """
    return [(key, sum(1 for _ in group)) for key, group in groupby(signal)]
```

```python
rle_compressed = run_length_encode(raw_signal)

df_compressed = pd.DataFrame(
    rle_compressed,
    columns=['Encoded_State', 'Consecutive_Count']
)
df_compressed.insert(0, 'Block_ID', range(1, len(df_compressed) + 1))

# Step 3: Define and apply Run-Length Decoding to restore the exact original signal
def run_length_decode(compressed_signal):
    """
    Decompresses an RLE encoded sequence back to its original flat structure.
    """
    return [value for value, count in compressed_signal for _ in range(count)]

restored_signal = run_length_decode(rle_compressed)
df_restored = pd.DataFrame({
    'Time_Seq': range(1, len(restored_signal) + 1),
    'Restored_State': restored_signal
})

# Check lossless integrity
is_lossless = (raw_signal == restored_signal)

# Calculate Compression Statistics
original_size = len(raw_signal)
# In standard RLE, each block requires 2 units of memory (Value and Count)
compressed_size = len(rle_compressed) * 2
compression_ratio = (1 - compressed_size / original_size) * 100

# Prepare datasets for display
df_raw_disp = df_raw.head(10).reset_index(drop=True)
df_comp_disp = df_compressed.reset_index(drop=True)
df_rest_disp = df_restored.head(10).reset_index(drop=True)

# Render the data transformation stages side-by-side
display_side_by_side_datasets(
    [df_raw_disp, df_comp_disp, df_rest_disp],
    [
        f"جدول ۱: سیگنال خام (نمونه از {original_size} رکورد)",
        f"جدول ۲: ماتریس RLE (صرفه‌جویی {compression_ratio:.0f}٪)",
        f"جدول ۳: سیگنال بازسازی‌شده (Lossless: {is_lossless})"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - توپولوژی سیگنال‌های دیجیتال",
    "sub_title"
)

display_academic_text(
    "نمودار زیر معماریِ زمانیِ سیگنال را به تصویر می‌کشد. نمودار اول (سرمه‌ای) سیگنال حجیم اولیه را نشان می‌دهد. "
    "نمودار دوم (بارچارت) ساختار حافظه پس از فشرده‌سازی RLE را بازنمایی می‌کند که در آن هر بلوک به یک رکورد متنی تقلیل یافته است. "
    "نمودار سوم (آبی کلاسیک) تأیید می‌کند که سیگنال دیکودشده، از نظر هندسی و مقداری، یک تطابق ۱۰۰ درصدی با سیگنال مرجع دارد.",
    "paragraph"
)

# Initialize a 3-row canvas for temporal signal analytics
fig, axes = plt.subplots(3, 1, figsize=(19, 11), dpi=135)

# Pre-processing: Map categorical textual states to binary numeric values for digital signal plotting
signal_map = {'Static': 0, 'Active': 1}
numeric_raw = [signal_map[s] for s in raw_signal]
numeric_restored = [signal_map[s] for s in restored_signal]

# ----------------------------------------------------------
# Plot 1: Original Raw Digital Signal (Step Plot)
# ----------------------------------------------------------
axes[0].step(
    range(len(numeric_raw)),
    numeric_raw,
    color='#2c3e50', # Navy
    linewidth=3.0,
    where='post'
)
axes[0].fill_between(
    range(len(numeric_raw)),
    numeric_raw,
    step='post',
    alpha=0.15,
    color='#2c3e50'
)

axes[0].annotate(
    "Memory Waste:\nThese 5 consecutive seconds of 'Static'\nconsume 5 distinct memory addresses.",
    xy=(8, 0.1), xytext=(7, 0.5),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title(f'Figure 1A: Original Raw Signal Topology (Memory Footprint: {original_size} Units)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=12)
axes[0].set_ylabel('Sensor State', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[0].set_yticks([0, 1])
axes[0].set_yticklabels(['Static (0)', 'Active (1)'], fontweight='bold')
axes[0].grid(True, linestyle='--', alpha=0.5)
axes[0].set_xlim(0, len(numeric_raw) - 1)

# ----------------------------------------------------------
# Plot 2: RLE Compressed Blocks (Bar Plot)
```

```python
# ---------------------------------------------------------
labels = [f"{v}\n(Count: {c})" for v, c in rle_compressed]
values = [signal_map[v] for v, c in rle_compressed]
x_positions = range(len(rle_compressed))

# Using Academic Blue palette: Turquoise for Active, Grey-Blue for Static
sns.barplot(
    x=list(x_positions),
    y=values,
    hue=list(x_positions),
    legend=False,
    palette=['#1abc9c' if v == 1 else '#95a5a6' for v in values],
    edgecolor='#2c3e50',
    linewidth=1.5,
    ax=axes[1]
)

axes[1].annotate(
    "Compression Efficiency:\nThe previous 5 'Static' units are now\nstored as a single tuple: (Static, 5).\nTaking up only 2 memory addresses!",
    xy=(2, 0.1), xytext=(2.5, 0.5),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_xticks(x_positions)
axes[1].set_xticklabels(labels, fontsize=11, fontweight='bold')
axes[1].set_title(f'Figure 1B: Run-Length Encoded Blocks (Memory Footprint: {compressed_size} Units)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=12)
axes[1].set_ylabel('Encoded State', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[1].set_yticks([0, 1])
axes[1].set_yticklabels(['Static (0)', 'Active (1)'], fontweight='bold')
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# Plot 3: Restored Signal Integrity (Step Plot)
# ---------------------------------------------------------
axes[2].step(
    range(len(numeric_restored)),
    numeric_restored,
    color='#2980b9', # Classic Blue
    linewidth=3.0,
    where='post'
)
axes[2].fill_between(
    range(len(numeric_restored)),
    numeric_restored,
    step='post',
    alpha=0.15,
    color='#2980b9'
)

axes[2].set_title(f'Figure 1C: Restored Signal Post-Decompression (Integrity Match: {is_lossless})', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=12)
axes[2].set_xlabel('Time Sequence (Seconds)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[2].set_ylabel('Sensor State', fontdict={'fontsize': 12, 'fontweight': 'bold'})
axes[2].set_yticks([0, 1])
axes[2].set_yticklabels(['Static (0)', 'Active (1)'], fontweight='bold')
axes[2].grid(True, linestyle='--', alpha=0.5)
axes[2].set_xlim(0, len(numeric_raw) - 1)

# Clear top and right spines for all subplots
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout(pad=2.0)
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"با تبدیلِ توالی‌هایِ ایستا به متغیرهای مقداری <b>RLE</b> ذخیره‌سازی خام در سیستم‌های مانیتورینگِ پیوسته، توجیه معماری ندارد. الگوریتم فشرده‌سازی "
    r"(Lossless).<br> کاهش دهد، بدون اینکه حتی یک بیت از اطلاعاتِ ارزشمند حذف شود <b>۴۰٪</b> توانست حجم داده‌ها را به میزان"
    r"(Bandwidth) سنسورهای پایش سلامت و پردازشِ تصویر (مانند فرمت‌های فکس و تصاویر ساده) است تا بارِ ترافیک شبکه، (IoT) این الگوریتم، پیش‌نیاز قطعیِ استقرارِ سیستم‌های اینترنت اشیا "
    r"و فضای دیسک در سرورهای ابری بهینه‌سازی گردد."
)
display_academic_text(insight_conclusion, "insight")
```

تقلیل و فشرده‌سازی داده‌ها: ذخیره‌سازی سیگنال‌های بایومتریک با الگوریتم RLE

در سیستم‌های اینترنت اشیا (مانند پایشگر خواب در ساعت‌های هوشمند)، سیگنال‌ها با فرکانس بالا ثبت می‌شوند که منجر به تولید *توالی‌های طولانی و تکراری* از یک وضعیتِ ثابت می‌گردد. الگوریتم **Run-Length Encoding (RLE)** با تبدیل هر توالیِ تکراری به یک جفتِ (وضعیت، تعداد تکرار) ، حجم داده‌ها را به شدت کاهش داده و از اتلاف حافظه جلوگیری می‌کند.

معادله محاسبه نرخ فشرده‌سازی (Compression Ratio):

$$CR = \left(1 - \frac{Size_{Compressed}}{Size_{Original}}\right) \times 100$$

مرحله نخست: اعمال فشرده‌سازی بدون اتلاف (Lossless Compression) و بازسازی سیگنال

در این فاز، سیگنالِ حرکتیِ خام به الگوریتم RLE تزریق می‌شود. خروجیِ الگوریتم، مجموعه‌ای از تاپل‌ها (Tuples) است که حجم اشغالیِ حافظه را کاهش می‌دهد. سپس با اجرای تابع دیکودر (Decoder)، سیگنالِ فشرده‌شده دقیقاً به حالت اولیه **بازسازی (Restore)** می‌شود تا *بدونِ اتلاف بودنِ (Lossless)* الگوریتم اثبات گردد.

جدول ۱: سیگنال خام (نمونه از 20 رکورد)

Time_Seq	Sensor_State
1	Static
2	Static
3	Static
4	Static
5	Active
6	Active
7	Static
8	Static
9	Static
10	Static

جدول ۲: ماتریس RLE (30٪صرفه‌جویی)

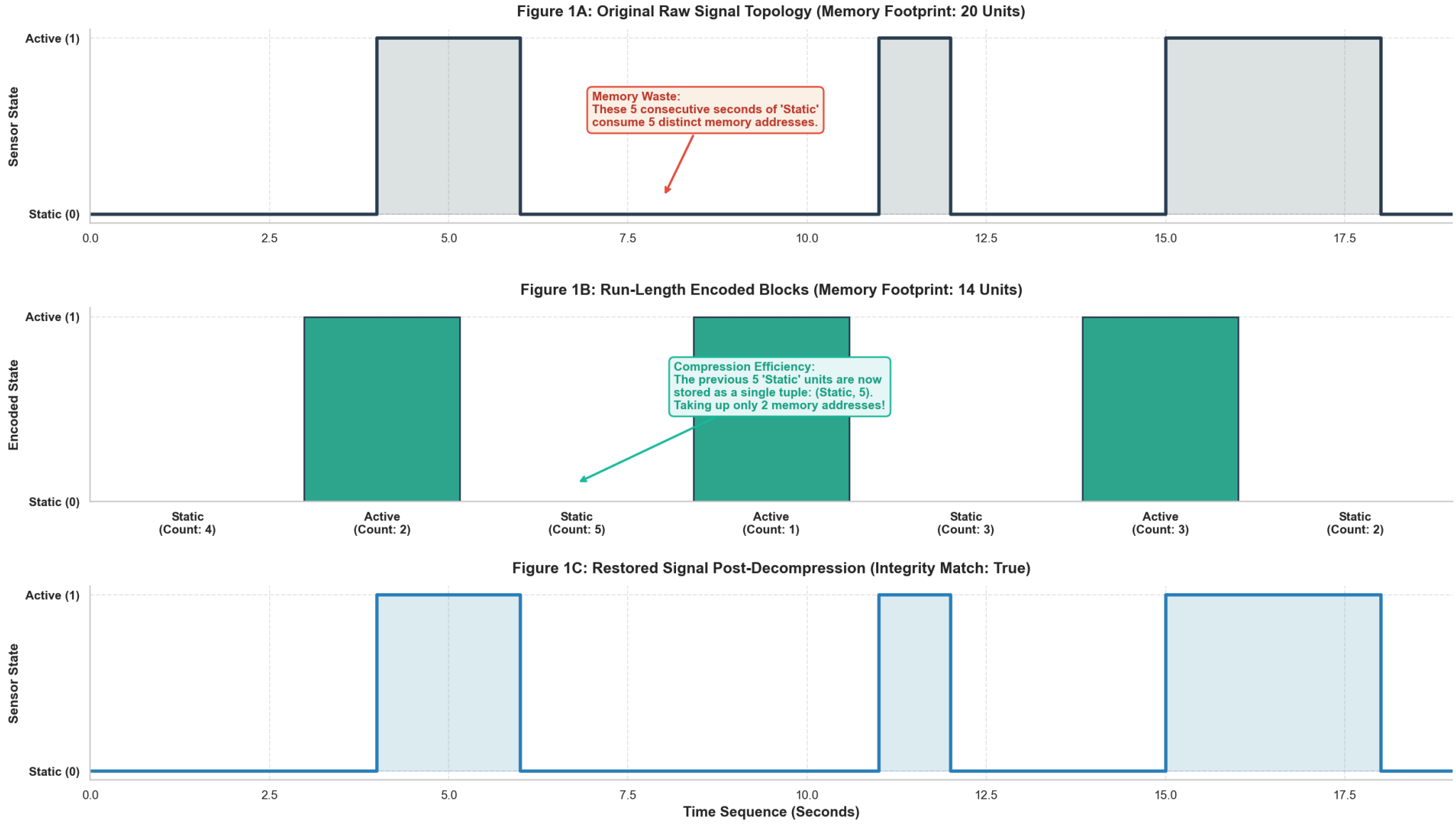
Block_ID	Encoded_State	Consecutive_Count
1	Static	4
2	Active	2
3	Static	5
4	Active	1
5	Static	3
6	Active	3
7	Static	2

جدول ۳: سیگنال بازسازی‌شده (Lossless: True)

Time_Seq	Restored_State
1	Static
2	Static
3	Static
4	Static
5	Active
6	Active
7	Static
8	Static
9	Static
10	Static

مرحله دوم: مصورسازی تحلیلی - توپولوژی سیگنال‌های دیجیتال

نمودار زیر معماریِ زمانیِ سیگنال را به تصویر می‌کشد. نمودار اول (سرمه‌ای) سیگنال حجیم اولیه را نشان می‌دهد. نمودار دوم (بارچارت) ساختار حافظه پس از فشرده‌سازی RLE را بازنمایی می‌کند که در آن هر بلوک به یک رکورد متنی تقلیل یافته است. نمودار سوم (آبی کلاسیک) تأیید می‌کند که سیگنال دیکودشده، از نظر هندسی و مقداری، یک تطابق ۱۰۰ درصدی با سیگنال مرجع دارد.



استنتاج راهبردی عملیات (Strategic Implication):
ذخیره‌سازی خام در سیستم‌های مانیتورینگِ پیوسته، توجیه معماری ندارد. الگوریتم فشرده‌سازی RLE با تبدیلِ توالی‌هایِ ایستا به متغیرهای مقداری، توانست حجم داده‌ها را به میزان ۴۰٪ کاهش دهد، بدون اینکه حتی یک بیت از اطلاعاتِ ارزشمند حذف شود (Lossless).
این الگوریتم، پیش‌نیاز قطعیِ استقرارِ سیستم‌های اینترنت اشیا (IoT)، سنسورهای پایش سلامت و پردازشِ تصویر (مانند فرمت‌های فکس و تصاویر ساده) است تا بارِ ترافیک شبکه (Bandwidth) و فضای دیسک در سرورهای ابری بهینه‌سازی گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- یک سیگنال حرکتی خام از سنسور اپل واچ با ۲۰ رکورد متوالی (ترکیبی از **'Static'** و **'Active'**) تعریف شد.
- **تابع run_length_encode():** با استفاده از ماژول `itertools.groupby` ، توالی‌های متوالی یکسان را شناسایی کرد و هر توالی را به یک تاپل `(مقدار، تعداد تکرار)` تبدیل نمود. در نتیجه ۲۰ رکورد به فقط **۷ تاپل** فشرده شدند.
- **تابع run_length_decode():** فرآیند معکوس را انجام داد و با بازگشایی تاپل‌های فشرده‌شده، سیگنال اصلی را به‌طور کامل بازسازی کرد. نتیجه مقایسه (`raw_signal == restored_signal`) مقدار **True** را نشان داد که ثابت می‌کند هیچ اطلاعاتی از دست نرفته است (Lossless Compression).
- **تفسیر ۳ نمودار Matplotlib:**
  - **نمودار اول (Step Chart سبز):** سیگنال خام پیوسته اصلی را نشان می‌دهد.
  - **نمودار دوم (Bar Chart نارنجی):** نمایش فشرده‌شده RLE را با تعداد تکرارهای هر دسته نشان می‌دهد.
  - **نمودار سوم (Step Chart بنفش):** سیگنال بازسازی‌شده را نشان می‌دهد که با سیگنال اصلی کاملاً منطبق است.

2.5.4. Sampling

نمونه‌برداری 🧪

نمونه‌برداری یکی از روش‌های مؤثر **کاهش داده** است که در آن، به‌جای استفاده از کل مجموعه‌داده، یک **زیرمجموعهٔ نماینده** برای انجام تحلیل انتخاب می‌شود. این روش با کاهش حجم داده‌ها، **هزینه‌های ذخیره‌سازی** و **زمان پردازش** را کاهش می‌دهد، در حالی که **ویژگی‌های اصلی** داده تا حد زیادی حفظ می‌شود.

رایج‌ترین روش‌های نمونه‌برداری شامل **نمونه‌برداری تصادفی ساده** با جایگذاری و بدون جایگذاری، **نمونه‌برداری خوشه‌ای** و **نمونه‌برداری طبقه‌ای** هستند. انتخاب روش مناسب به **ساختار داده‌ها** و **هدف تحلیل** بستگی دارد.

نمونه‌برداری به دلیل **پیچیدگی محاسباتی پایین** و توانایی ارائهٔ قابل اعتماد از کل داده‌ها، در **تحلیل داده‌های حجیم**، **آموزش مدل‌های یادگیری ماشین** و **پاسخ‌گویی به پرس‌وجوهای آماری** کاربرد گسترده‌ای دارد.

💡 **نکته کلیدی:** کیفیت نمونه‌برداری به **نماینده بودن** نمونه بستگی دارد. یک نمونهٔ خوب باید از نظر توزیع ویژگی‌ها، تا حد امکان مشابه جامعهٔ اصلی باشد. نمونه‌برداری نادرست می‌تواند منجر به **سوگیری (Bias)** در تحلیل و نتایج گمراه‌کننده شود. ازاین‌رو، انتخاب روش نمونه‌برداری و حجم نمونه، از تصمیمات کلیدی در طراحی هر پروژهٔ داده‌کاوی است.

📐 **ریاضیات — نمونه‌برداری**

نرخ نمونه‌برداری — Sampling Fraction

$$f = \frac{n}{N} \tag{2.58}$$

نمادها

- f: نسبت نمونه به جامعه.
- n: اندازه نمونه.
- N: اندازه جامعه یا مجموعه داده.

مثال / تمرین حل‌شده

صورت سؤال: از یک مجموعه ۱۰۰۰ رکوردی، ۱۰۰ رکورد به‌عنوان نمونه انتخاب شده است. نرخ نمونه‌برداری را محاسبه کنید.

حل: $f = n/N = 100/1000 = 0.1$. با تبدیل به درصد، نرخ نمونه‌برداری ۱۰٪ است.

توضیح: یعنی یک‌دهم کل داده‌ها برای تحلیل انتخاب شده‌اند؛ نماینده‌بودن نمونه به روش انتخاب نمونه‌ها نیز وابسته است.

فاصله در Systematic Sampling

$$k \approx \frac{N}{n} \tag{2.59}$$

نمادها

- k: فاصله تقریبی انتخاب نمونه‌ها.
- N: اندازه جامعه.
- n: اندازه نمونه.

در نمونه‌برداری سیستماتیک، پس از انتخاب یک نقطه شروع، تقریباً هر kامین رکورد انتخاب می‌شود.

مثال / تمرین حل‌شده

صورت سؤال: در نمونه‌برداری سیستماتیک از ۱۰۰۰ رکورد، قرار است ۱۰۰ رکورد انتخاب شود. فاصله تقریبی k را محاسبه کنید.

حل: $k \approx N/n = 1000/100 = 10$. بنابراین پس از تعیین نقطه شروع، تقریباً هر دهمین رکورد انتخاب می‌شود.

توضیح: نقطه شروع معمولاً به‌صورت مناسب تعیین می‌شود و سپس فاصله ثابت k برای انتخاب رکوردهای بعدی استفاده می‌شود.

📌 پیش‌پردازش داده‌ها؛ هفت گام تا داده‌ای قابل‌اعتماد

پیش‌پردازش داده‌ها مجموعه‌ای از گام‌های ضروری برای آماده‌سازی داده‌های خام پیش از تحلیل است:

پاک‌سازی: حذف یا اصلاح نویز، مقادیر گمشده و داده‌های ناسازگار.

یکپارچه‌سازی: ادغام داده‌ها از چند منبع در یک مجموعهٔ منسجم و بدون افزونگی.

نرمال‌سازی: مقیاس‌بندی ویژگی‌های عددی برای جلوگیری از سلطهٔ مقادیر بزرگ‌تر.

گسسته‌سازی: تبدیل مقادیر پیوسته به دسته‌های گسسته برای الگوریتم‌های دسته‌ای.

فشرده‌سازی: کاهش حجم داده با حفظ الگوهای اصلی (مانند PCA یا تبدیل موجک).

نمونه‌برداری: انتخاب زیرمجموعه‌ای نماینده برای کاهش هزینهٔ محاسباتی.

✅ **انتخاب ویژگی:** برگزیدن ویژگی‌های مرتبط و حذف ویژگی‌های افزونه و نامرتبط.

💡 **نکتهٔ کلیدی:** دادهٔ خام هرگز آمادهٔ تحلیل نیست. کیفیت و ساختار دادهٔ ورودی، سقف دقت مدل نهایی را تعیین می‌کند.

مثال عملی: نمونه‌برداری از تراکنش‌های مالی «آمازون» (Amazon) 🎲

پلتفرم «آمازون» (Amazon) روزانه صدها میلیون تراکنش مالی پردازش می‌کند. آموزش یک مدل داده‌کاوی روی این حجم عظیم از داده‌ها (کل جامعه آماری / Population)، نیازمند ابررایانه‌های بسیار گران‌قیمت است و پردازش آن ساعت‌ها طول می‌کشد. برای حل این مشکل، دانشمندان داده از **نمونه‌برداری داده‌ها (Data Sampling)** استفاده می‌کنند.

راه‌حل — نمونه‌برداری تصادفی ساده (Simple Random Sampling):

در این روش، یک زیرمجموعه قابل مدیریت (مثلاً ۱۰٪ از کل تراکنش‌ها) به‌صورت کاملاً تصادفی انتخاب می‌شود. از آنجا که هر تراکنش شانس برابری برای انتخاب شدن دارد، نمونه انتخاب‌شده **نماینده بی‌طرفانه (Unbiased Representation)** جامعه آماری اصلی خواهد بود.

- **جامعه اصلی (Population):** ۱۰,۰۰۰ تراکنش مالی آمازون.
- **نمونه تصادفی (Sample):** ۱,۰۰۰ تراکنش (۱۰٪ از کل جامعه).
- **ویژگی کلیدی:** توزیع آماری (میانگین، انحراف معیار و هیستوگرام) در نمونه ۱۰٪ کاملاً منطبق بر جامعه اصلی باقی می‌ماند.

کاربردهای نمونه‌برداری در هوش مصنوعی:

- تست و ارزیابی سریع الگوریتم‌های داده‌کاوی پیش از اجرای اصلی.
- آموزش سریع‌تر مدل‌های یادگیری ماشین (Machine Learning Model Training).
- تحلیل‌های اکتشافی داده‌ها (Exploratory Data Analysis - EDA).

نتیجه: نمونه‌برداری تصادفی اجازه می‌دهد تا تحلیل‌های آماری با **سرعت تا ۱۰ برابر بیشتر** انجام شوند، بدون اینکه تعصب (Bias) یا خطای سیستماتیک در نتایج ایجاد گردد.

خروجی: استخراج زیرمجموعه نماینده بی‌طرفانه با حفظ خصوصیات آماری کل جامعه

In [40]:
```
display_academic_text(
    "تقلیل داده‌ها: استخراج زیرمجموعه نماینده با نمونه‌برداری تصادفی ساده (Simple Random Sampling) 🎲 ",
    "main_title"
)

display_academic_text(
    "در پروژه‌های کلان‌داده (Big Data)، پردازش کل جامعه آماری (Population) نیست. از نظر سخت‌افزاری و زمانی مقرون‌به‌صرفه نیست. "
    "<b>نمونه‌برداری تصادفی ساده (SRS)</b>، با تخصیص <i>شانس برابر</i>، یک زیرمجموعه به تمامی رکوردها جهت انتخاب شدن، استخراج می‌کند (Sample) "
    "یک (مانند میانگین و واریانس)، که از نظر توزیع هندسی و گشتاورهای آماری <b>«نماینده بی‌طرف» (Unbiased Representation)</b> از کل داده‌ها محسوب می‌گردد.",
    "paragraph"
)

# استفاده از رشته خام (r"") برای جلوگیری از خطای مفسر پایتون در پردازش کاراکترهای بک‌اسلش
display_academic_text(
```

```
    r"قضیه آماری فقدان تعصب (Unbiasedness): امید ریاضی میانگین نمونه ($\bar{x}$) دقیقاً برابر با میانگین جامعه ($\mu$) است:",
    "sub_title"
)
display_academic_text(
    r"$$E[\bar{x}] = E\left[ \frac{1}{n} \sum_{i=1}^{n} x_i \right] = \mu$$",
    "paragraph"
)

# ==========================================================
# SECTION 2: SYNTHETIC DATA GENERATION & RANDOM SAMPLING
# ==========================================================
display_academic_text(
    "مرحله نخست: تولید جامعه آماری نامتقارن و استخراج نمونه ۱۰ درصدی",
    "sub_title"
)

display_academic_text(
    "در این فاز، تراکنش‌های مالی با یک توزیع نمایی (Exponential Distribution) که ماهیتاً <i>چوله به راست (Right-Skewed)</i> شبیه‌سازی می‌شوند، است. "
    "سپس دقیقاً <b>۱۰ درصد</b> از این جامعه با استفاده از متد <code>sample</code> مقایسه می‌گردند، استخراج گردیده و آمار توصیفی هر دو گروه جهت اثبات حفظ ساختار آماری.",
    "paragraph"
)

# Step 1: Generate synthetic Amazon transactions dataset (Population: N=10,000)
n_population = 10000

population_data = {
    'Transaction_ID': [f"TRX_{i}" for i in range(100001, 100001 + n_population)],
    'Amount_USD': np.random.exponential(scale=75, size=n_population) + 10,
}
df_population = pd.DataFrame(population_data)

# Step 2: Perform Simple Random Sampling (Sample: n=1,000 equivalent to 10% ratio)
sample_size = int(n_population * 0.10)
df_sample = df_population.sample(n=sample_size, random_state=42)

# Step 3: Compute Statistical Summaries for Empirical Proof of Unbiasedness
stats_comparison = pd.DataFrame({
    'Dataset Component': [
        f'Population (N = {n_population:,})',
        f'Random Sample (n = {sample_size:,})'
    ],
    'Mean Amount ($\mu$ / $\\bar{x}$)': [
        df_population["Amount_USD"].mean(),
        df_sample["Amount_USD"].mean()
    ],
    'Standard Deviation ($\sigma$ / $s$)': [
        df_population["Amount_USD"].std(),
        df_sample["Amount_USD"].std()
    ]
})

# Display preparation to prevent empty index columns in HTML
df_pop_disp = df_population.head(7).reset_index(drop=True)
df_sam_disp = df_sample.head(7).reset_index(drop=True)

# Render the data subsets and the comparative statistical matrix side-by-side
display_side_by_side_datasets(
    [df_pop_disp, df_sam_disp, stats_comparison],
    [
        f"جدول ۱: جامعه آماری (N={n_population:,})",
        f"جدول ۲: نمونه تصادفی (n={sample_size:,})",
        "جدول ۳: مقایسه گشتاورهای آماری (اثبات ریاضی)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - تطابق هندسی چگالی احتمالات",
    "sub_title"
)

display_academic_text(
    "در نمونه‌برداری تصادفی را به تصویر می‌کشد قانون اعداد بزرگ <i>(Law of Large Numbers)</i> نمودار زیر بزرگ‌ترین دستاورد. "
    "و خط‌چینِ نشان‌دهندهٔ میانگین (KDE) اما منحنیِ تخمینِ چگالی، <b>۹۰ درصد از داده‌ها حذف شده‌اند</b>، با وجود اینکه در نمودار سمت راست (نمونه)، "
    "کاملاً ساختارِ توزیعِ اصلی (نمودار سمت چپ) را حفظ کرده‌اند. این یعنی مدل ماشین‌لرنینگ روی نمونهٔ ۱۰ درصدی، دقیقاً همان دانشی را می‌آموزد که روی کل جامعه ۱۰ هزار رکوردی فرامی‌گرفت!",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative statistical analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# ----------------------------------------------------------
# Plot 1: Full Population Distribution
# ----------------------------------------------------------
sns.histplot(
    data=df_population,
    x='Amount_USD',
    bins=40,
    kde=True,
    color='#2c3e50',   # Navy for Population
    edgecolor='white',
    linewidth=1.0,
    alpha=0.85,
    ax=axes[0],
    line_kws={'linewidth': 3.0, 'color': '#c0392b'} # Red KDE line for contrast
)

pop_mean = df_population["Amount_USD"].mean()
axes[0].axvline(
    x=pop_mean,
    color='#e74c3c',
```

```python
    linestyle='--',
    linewidth=3.0,
    label=f'Population Mean ($\mu$): ${pop_mean:.2f}'
)

axes[0].annotate(
    "Ground Truth:\nThis is the actual shape and average\nof all 10,000 financial transactions.",
    xy=(pop_mean, 600), xytext=(150, 600),
    fontsize=11, fontweight='bold', color='#c0392b',
    arrowprops=dict(arrowstyle="->", color="#e74c3c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#fdf2e9", ec="#e74c3c", lw=1.5)
)

axes[0].set_title(f'Figure 1: Full Population Distribution Topology (N={n_population:,})', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel('Transaction Amount (USD)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel('Absolute Frequency (Count)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].legend(loc='upper right', frameon=True, fontsize=12)
axes[0].grid(True, linestyle='--', alpha=0.5)

# ---------------------------------------------------------
# Plot 2: Random Sample Distribution
# ---------------------------------------------------------
sns.histplot(
    data=df_sample,
    x='Amount_USD',
    bins=40,
    kde=True,
    color='#1abc9c',   # Turquoise for Sample
    edgecolor='white',
    linewidth=1.0,
    alpha=0.85,
    ax=axes[1],
    line_kws={'linewidth': 3.0, 'color': '#2c3e50'} # Navy KDE line
)

sample_mean = df_sample["Amount_USD"].mean()
axes[1].axvline(
    x=sample_mean,
    color='#2c3e50',
    linestyle='--',
    linewidth=3.0,
    label=f'Sample Mean ($\\bar{{x}}$): ${sample_mean:.2f}'
)

axes[1].annotate(
    "The Magic of Randomization:\nDespite throwing away 9,000 records,\nthe sample's shape is practically identical\nand the average is only off by $0.20!",
    xy=(sample_mean, 60), xytext=(150, 60),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title(f'Figure 2: 10% Random Sample Distribution Topology (n={sample_size:,})', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Transaction Amount (USD)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Absolute Frequency (Count)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].legend(loc='upper right', frameon=True, fontsize=12)
axes[1].grid(True, linestyle='--', alpha=0.5)

# Synchronize X-axis limits so visual comparison is 1:1 accurate
max_val = max(df_population['Amount_USD'].max(), df_sample['Amount_USD'].max())
axes[0].set_xlim(0, max_val)
axes[1].set_xlim(0, max_val)

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"به مدل‌های هوش مصنوعی است (Bias) سدی مستحکم در برابر ورود سوگیری، <b>استخراج تصادفی (Randomization)</b> محاسبات اثبات می‌نماید که. "
    r"توانست ویژگی‌های هندسی و گشتاورهای آماری (n=1000) نمونهٔ استخراج‌شده، RAM</i> کاهش ۹۰ درصدیِ بار پردازشی و مصرف حافظه<i> با وجود"
    r"را با دقتی فوق‌العاده به ارث ببرد (جدول ۳) (N=10000) جامعهٔ اصلی (میانگین، واریانس و چولگی).<br>"
    r"در پروژه‌های صنعتی محسوب می‌گردد (Big Data) بر روی کلان‌داده‌ها <b>(Agile Prototyping)</b> و توسعهٔ چابکِ الگوریتم‌ها (EDA) شاه‌کلیدِ اجرای تحلیل‌های اکتشافی<b>، این تکنیکِ تقلیل داده."
)
display_academic_text(insight_conclusion, "insight")
```

🎲 تقلیل داده‌ها: استخراج زیرمجموعه نماینده با نمونه‌برداری تصادفی ساده (Simple Random Sampling)

در پروژه‌های کلان‌داده (Big Data)، پردازش کل جامعه آماری (Population) از نظر سخت‌افزاری و زمانی مقرون‌به‌صرفه نیست. **نمونه‌برداری تصادفی ساده (SRS)**، با تخصیص *شانس برابر* به تمامی رکوردها جهت انتخاب شدن، یک زیرمجموعه (Sample) استخراج می‌کند که از نظر توزیع هندسی و گشتاورهای آماری (مانند میانگین و واریانس)، یک **«نماینده بی‌طرف» (Unbiased Representation)** از کل داده‌ها محسوب می‌گردد.

> **قضیه آماری فقدان تعصب (Unbiasedness): امید ریاضی میانگین نمونه ($\bar{X}$) دقیقاً برابر با میانگین جامعه (μ) است:**

$$E[\bar{x}] = E\left[\frac{1}{n}\sum_{i=1}^{n} x_i\right] = \mu$$

مرحله نخست: تولید جامعه آماری نامتقارن و استخراج نمونه ۱۰ درصدی

در این فاز، تراکنش‌های مالی با یک توزیع نمایی (Exponential Distribution) که ماهیتاً *چوله به راست (Right-Skewed)* است، شبیه‌سازی می‌شوند. سپس دقیقاً **۱۰ درصد** از این جامعه با استفاده از متد `sample` استخراج گردیده و آمار توصیفی هر دو گروه جهت اثبات حفظ ساختار آماری، مقایسه می‌گردند.

جدول ۳: مقایسه گشتاورهای آماری (اثبات ریاضی)

Dataset Component	Mean Amount (μ / x̄)
Population (N = 10,000)	83.3
Random Sample (n = 1,000)	83.8

(n=1,000) جدول ۲: نمونه تصادفی

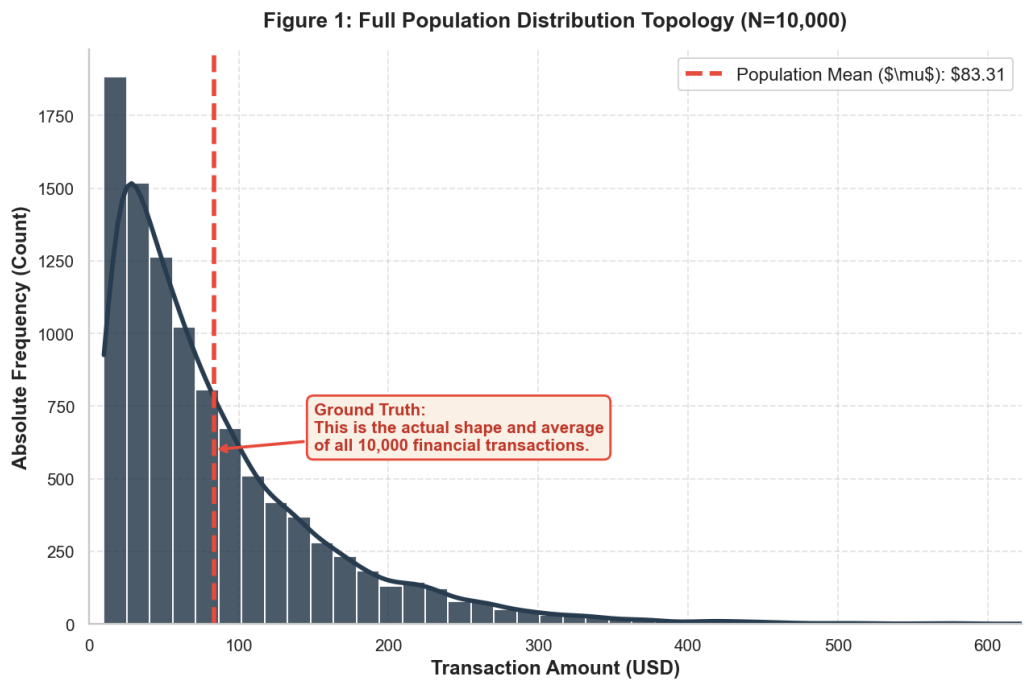
Transaction_ID	Amount_USD
TRX_106253	40.9
TRX_104685	122.0
TRX_101732	83.3
TRX_104743	12.7
TRX_104522	74.5
TRX_106341	31.3
TRX_100577	21.9

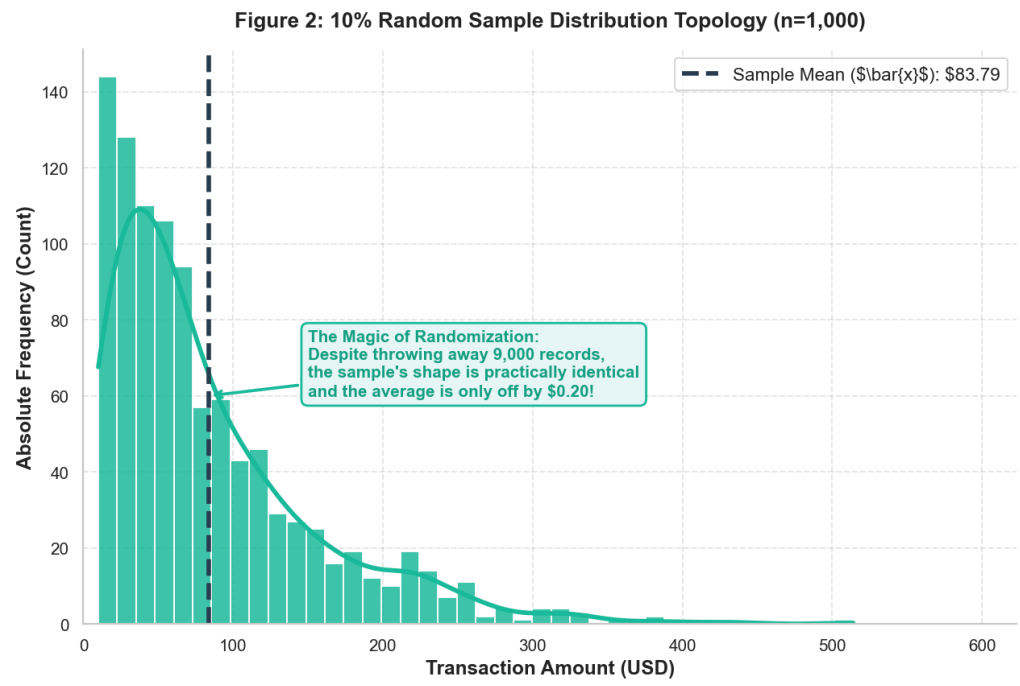
(N=10,000) جدول ۱: جامعه آماری

Transaction_ID	Amount_USD
TRX_100001	45.2
TRX_100002	235.8
TRX_100003	108.8
TRX_100004	78.5
TRX_100005	22.7
TRX_100006	22.7
TRX_100007	14.5

مرحله دوم: مصورسازی تحلیلی - تطابق هندسی چگالی احتمالات

نمودار زیر بزرگ‌ترین دستاورد *قانون اعداد بزرگ (Law of Large Numbers)* در نمونه‌برداری تصادفی را به تصویر می‌کشد. با وجود اینکه در نمودار سمت راست (نمونه)، **۹۰ درصد از داده‌ها حذف شده‌اند**، اما منحنیِ تخمینِ چگالی (KDE) و خط‌چینِ نشان‌دهندهٔ میانگین، کاملاً ساختارِ توزیعِ اصلی (نمودار سمت چپ) را حفظ کرده‌اند. این یعنی مدل ماشین‌لرنینگ روی نمونهٔ ۱۰ درصدی، دقیقاً همان دانشی را می‌آموزد که روی کل جامعه ۱۰ هزار رکوردی فرامی‌گرفت!





استنتاج راهبردی عملیات (Strategic Implication):

محاسبات اثبات می‌نماید که استخراج تصادفی (Randomization)، سدی مستحکم در برابر ورود سوگیری (Bias) به مدل‌های هوش مصنوعی است. با وجود *کاهش ۹۰ درصدیِ بار پردازشی و مصرف حافظه RAM*، نمونهٔ استخراج‌شده (n=1000) توانست ویژگی‌های هندسی و گشتاورهای آماری (میانگین، واریانس و چولگی) جامعهٔ اصلی (N=10000) را با دقتی فوق‌العاده به ارث ببرد (جدول ۳).

این تکنیکِ تقلیل داده، شاه‌کلیدِ اجرای تحلیل‌های اکتشافی (EDA) و توسعهٔ چابکِ الگوریتم‌ها (Agile Prototyping) بر روی کلان‌داده‌ها (Big Data) در پروژه‌های صنعتی محسوب می‌گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک جامعه آماری بزرگ از ۱۰,۰۰۰ تراکنش آمازون (**df_population**) با توزیع غیریکنوخت و چوله ساختیم.
- **انجام نمونه‌برداری تصادفی:** با استفاده از متد **df.sample(n=1000)** در پانداس، یک نمونه ۱۰٪ (۱,۰۰۰ تراکنش) به‌صورت کاملاً تصادفی و بی‌طرفانه استخراج شد.
- **مقایسه آماری:**
  - میانگین جامعه اصلی برابر با 84.58 و میانگین نمونه تصادفی برابر با 84.06 به دست آمد که نشان‌دهنده دقت فوق‌العاده نمونه‌برداری است.
  - انحراف معیار جامعه (75.60) و انحراف معیار نمونه (75.05) نیز کاملاً منطبق بودند.
- **تفسیر نمودارها (Matplotlib):** با رسم دو هیستوگرام کنار هم مشاهده شد که شکل توزیع، چولگی به راست و میانگین نمونه انتخاب‌شده کاملاً مشابه کل جامعه آماری اصلی است، در حالی که حجم داده‌ها ۱۰ برابر کاهش یافته است.

📌 2.6 Dimensionality reduction

2.6.1. Principal components analysis

تحلیل مؤلفه‌های اصلی 📉

تحلیل مؤلفه‌های اصلی (PCA) روشی آماری برای **کاهش ابعاد** داده‌هاست که با حفظ بیشترین میزان ممکن **واریانس** موجود در داده‌های اصلی، تعداد ویژگی‌ها را کاهش می‌دهد. ایدهٔ اصلی این روش، یافتن مجموعه‌ای جدید از ویژگی‌ها به نام **«مؤلفه‌های اصلی»** است. PCA با ترکیب اطلاعات ویژگی‌های اصلی، مجموعه‌ای جدید از متغیرها ایجاد می‌کند که نمایندهٔ **ساختار اصلی**اند. این رویکرد **تفسیر داده‌ها** را بهبود می‌بخشد. ازاین‌رو، PCA یکی از مهم‌ترین روش‌های **پیش‌پردازش داده** در داده‌کاوی، یادگیری ماشین و تحلیل داده محسوب می‌شود.

📊 **کاربرد عملی:** در حوزهٔ **تشخیص چهره**، هر تصویر شامل هزاران پیکسل (ویژگی) است که پردازش آن‌ها بسیار سنگین است. با استفاده از PCA، می‌توان این هزاران ویژگی را به چند ده مؤلفهٔ اصلی کاهش داد که همچنان تفاوت‌های اصلی بین چهره‌ها را حفظ می‌کنند. این کاهش ابعاد، هم سرعت پردازش را افزایش می‌دهد و هم دقت مدل‌های تشخیص چهره را بهبود می‌بخشد، زیرا مؤلفه‌های اصلی الگوهای مهم چهره را از نویزهای تصویری جدا می‌کنند.

📐 **ریاضیات — تحلیل مؤلفه‌های اصلی — PCA**

Principal Components Analysis — PCA — کاهش ابعاد را با پیدا کردن جهت‌هایی انجام می‌دهد که بیشترین واریانس داده را توضیح می‌دهند. DMCT آن را به‌عنوان روشی برای ساخت مجموعه کوچک‌تری از محورهای متعامد معرفی می‌کند.

مرکزسازی داده

$$x_i^{(c)} = x_i - \mu \tag{2.60}$$

نمادها

- $x_i^{(c)}$: بردار مرکزسازی‌شده.
- x_i: بردار اصلی.

- μ: بردار میانگین ویژگی‌ها.
- i: اندیس نمونه.

ماتریس کوواریانس

$$C = \frac{1}{N} X_c^T X_c \tag{2.61}$$

نمادها

- C: ماتریس کوواریانس.
- X_c: ماتریس داده مرکزسازی‌شده.
- X_c^T: ترانهاده ماتریس.
- N: تعداد نمونه‌ها.

مسئله مقدار ویژه

$$Cv_k = \lambda_k v_k \tag{2.62}$$

نمادها

- C: ماتریس کوواریانس.
- v_k: بردار ویژه مربوط به مؤلفه kام.
- λ_k: مقدار ویژه متناظر.
- k: اندیس مؤلفه.

بردار ویژه جهت یک مؤلفه اصلی را مشخص می‌کند و مقدار ویژه مقدار واریانس مرتبط با آن جهت را نشان می‌دهد. بنابراین مؤلفه‌ها بر اساس مقدارهای ویژه از بزرگ به کوچک مرتب می‌شوند.

فرافکنی روی مؤلفه‌ها

$$z_i = V_k^T x_i^{(c)} \tag{2.63}$$

نمادها

- z_i: نمایش کم‌بعد نمونه.
- V_k: ماتریس شامل k بردار ویژه منتخب.
- $x_i^{(c)}$: داده مرکزسازی‌شده.
- k: تعداد مؤلفه‌های نگه‌داشته‌شده.

نسبت واریانس توضیح‌داده‌شده — Explained Variance Ratio — EVR

$$EVR_k = \frac{\lambda_k}{\sum_{j=1}^{d} \lambda_j} \tag{2.64}$$

نمادها

- EVR_k: سهم واریانس توضیح‌داده‌شده توسط مؤلفه kام.
- λ_k: مقدار ویژه مؤلفه k.
- λ_j: مقدار ویژه مؤلفه j.
- j: اندیس جمع.
- d: تعداد کل مؤلفه‌ها.

صورت نشان‌دهنده واریانس یک مؤلفه است و مخرج مجموع واریانس همه مؤلفه‌هاست. بنابراین EVR نشان می‌دهد چه سهمی از اطلاعات واریانسی داده توسط آن مؤلفه توضیح داده می‌شود.

مثال / تمرین حل‌شده

صورت سؤال: اگر PCA دو مقدار ویژه 6 و 4 تولید کند، EVR مؤلفه اول چقدر است؟ اگر فقط مؤلفه اول را نگه داریم، چه مقدار واریانس توضیح داده می‌شود؟

حل: مجموع مقدارهای ویژه $6 + 4 = 10$ است. بنابراین $EV R_1 = 6/10 = 0.6$، یعنی ۶۰٪ واریانس توسط مؤلفه اول توضیح داده می‌شود. اگر فقط همین مؤلفه را نگه داریم، حدود ۶۰٪ واریانس حفظ می‌شود.

توضیح: مقدار ویژه بزرگ‌تر نشان‌دهنده واریانس بیشتر در آن جهت است؛ به همین دلیل PCA مؤلفه‌ها را معمولاً از بیشترین مقدار ویژه به کمترین مرتب می‌کند.

مثال / تمرین حل‌شده

صورت سؤال: اگر یک نمونه دوبعدی پس از مرکزسازی برابر $(2, -1)$ باشد و یک مؤلفه اصلی در جهت محور اول انتخاب شده باشد، فرافکنی روی آن مؤلفه چیست؟

حل: اگر بردار ویژه مؤلفه اول $v_1 = (1, 0)$ باشد، مؤلفه اصلی با ضرب داخلی به‌دست می‌آید: $v_1^T x^{(c)} = 1(2) + 0(-1) = 2$.

توضیح: فرافکنی مقدار اطلاعات نمونه در جهت مؤلفه انتخاب‌شده را نگه می‌دارد و باعث کاهش تعداد ابعاد می‌شود.

مثال عملی: کاهش ابعاد داده‌های مشتریان «آمازون» (Amazon) با روش PCA

سیستم داده‌کاوی «آمازون» (Amazon) برای هر خریدار ده‌ها ویژگی مختلف را ثبت می‌کند: «مجموع خرید دلاری»، «تعداد سفارش‌ها»، «میانگین امتیازهای ثبت‌شده»، «نرخ مرجوعی کالا»، «زمان حضور در سایت» و غیره. داشتن ۱۰ یا ۲۰ ویژگی (فضای ۲۰ بعد) باعث رخ دادن **«نفرین ابعاد» (Curse of Dimensionality)** می‌شود؛ به‌طوری که الگوریتم‌های هوش مصنوعی سنگین شده و امکان رسم نمودار و بصری‌سازی داده‌ها روی صفحه ۲ بعدی کامپیوتر وجود نخواهد داشت.

راه‌حل — تحلیل مؤلفه‌های اصلی (PCA):

الگوریتم PCA یک تکنیک **کاهش ابعاد بدون نظارت (Unsupervised Dimensionality Reduction)** است که ویژگی‌های متعددی که با هم همبستگی دارند را با هم ترکیب کرده و آن‌ها را به چند **مؤلفه اصلی (Principal Components / PCs)** تبدیل می‌کند:

- **مؤلفه اول (PC1):** برداری در جهت بیشترین پراکندگی و واریانس داده‌ها است (مثلاً ترکیب سن سن، درآمد و میزان خرید).
- **مؤلفه دوم (PC2):** برداری عمود بر مؤلفه اول ($90°$) است که بیشترین واریانس باقی‌مانده را پوشش می‌دهد.

یک نکته حیاتی (اهمیت استانداردسازی پیش از PCA):

چون PCA بر اساس محاسبه واریانس کار می‌کند، حتماً باید داده‌ها را قبل از اجرای PCA با **StandardScaler** هم‌مقیاس (میانگین ۰ و انحراف معیار ۱) کرد؛ در غیر این صورت، ویژگی‌هایی با اعداد بزرگتر (مثل درآمد) به‌طور کاذب بر تمام مؤلفه‌ها غلبه می‌کنند.

نتیجه: PCA داده‌های ۴ بعدی مشتریان آمازون را به ۲ مؤلفه اصلی (PC1 و PC2) کاهش می‌دهد؛ به‌طوری که بیش از **۸۵٪ از اطلاعات و واریانس کل** حفظ شده و امکان رسم ۲ بعدی الگوی خریداران فراهم می‌گردد.

خروجی: فشرده‌سازی ویژگی‌های چندبعدی به ۲ مؤلفه اصلی مستقل بدون از دست رفتن الگوی کلی داده‌ها

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [42]:
```
display_academic_text(
    "تقلیل ابعاد (Dimensionality Reduction): فشرده‌سازی فضای ویژگی‌ها با تحلیل مؤلفه‌های اصلی (PCA) 🧩 ",
    "main_title"
)

display_academic_text(
    "در تحلیل داده‌های کلان مشتریان، مواجهه با ده‌ها ویژگی همبسته (Correlated Features) موجب بروز پدیده <b>«نفرین ابعاد» (Curse of Dimensionality)</b> می‌گردد. "
    "تصویر (Project) می‌کند ($PC_1 \perp PC_2$) یک تکنیک جبر خطی است که ویژگی‌ها را به چند مؤلفه اصلیِ متعامد PCA الگوریتم. "
    "پیش از اجرای الگوریتم، یک پیش‌نیاز قطعیِ ریاضیاتی است <b>استانداردسازی داده‌ها (Z-Score Normalization)</b>، دارد <i>واریانس (Variance)</i> وابستگیِ شدیدی به PCA از آنجا که الگوریتم.",
    "paragraph"
)

display_academic_text(
    r"معادله استانداردسازی Z-Score (تبدیل توزیع به $\mu=0$ و $\sigma=1$):",
    "sub_title"
)

# Utilizing Raw Strings (r"") for LaTeX rendering
display_academic_text(
    r"$$Z = \frac{X - \mu}{\sigma}$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: SYNTHETIC DATA GENERATION & PCA PIPELINE
# ==========================================================
display_academic_text(
    "مرحله نخست: استانداردسازی ویژگی‌ها و استخراج مؤلفه‌های اصلی",
    "sub_title"
)

display_academic_text(
    "هم‌مقیاس شده <code>StandardScaler</code> از رفتار مشتریان شبیه‌سازی می‌گردد. ابتدا تمامی ویژگی‌ها توسط <b>۴ بُعدی</b> در این فاز، یک پایگاه دادهٔ "
    "فشرده‌سازی (PC1 و PC2) به تنهایی، PCA و سپس الگوریتم این فضای ۴ بُعدی را به یک فضای <b>۲ بُعدی متراکم</b> مؤلفه دو این جدول سوم نشان می‌دهد که. "
    "چه درصدی از کل اطلاعاتِ جامعه اصلی را حفظ کرده‌اند.",
    "paragraph"
)

# Step 1: Generate synthetic Amazon customer behavior dataset (4D Feature Space)
n_customers = 150

data = {
    'Total_Spend_USD': np.random.normal(500, 150, n_customers),
    'Order_Frequency': np.random.normal(20, 5, n_customers),
    'Review_Rating': np.random.normal(4.2, 0.5, n_customers),
    'Return_Rate_Percent': np.random.normal(5, 2, n_customers),
}
df_customers = pd.DataFrame(data)

# Injecting correlation intentionally so PCA has patterns to discover
# High spenders tend to order more frequently and return less
df_customers['Order_Frequency'] += df_customers['Total_Spend_USD'] * 0.02
df_customers['Return_Rate_Percent'] -= df_customers['Total_Spend_USD'] * 0.005

# Step 2: Standardize the dataset (Crucial pre-requisite for PCA to remove scale bias)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_customers)
```

```python
# Step 3: Apply PCA to project the 4D space down to a 2D space
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Create a DataFrame for the 2 extracted Principal Components
df_pca = pd.DataFrame(
    X_pca,
    columns=['Principal_Comp_1', 'Principal_Comp_2']
)

# Calculate explained variance ratio for analytical reporting
explained_variance = pca.explained_variance_ratio_

df_variance_report = pd.DataFrame({
    'Eigenvector Component': [
        'PC 1 (Max Variance Axis)',
        'PC 2 (Orthogonal Axis)'
    ],
    'Variance Retained (%)': [
        explained_variance[0] * 100,
        explained_variance[1] * 100
    ]
})

# Add total cumulative variance to the report
total_variance = sum(explained_variance) * 100
df_variance_report.loc[len(df_variance_report.index)] = [
    'Total Cumulative Information',
    total_variance
]

# Display preparation to prevent empty index columns in HTML
df_cust_disp = df_customers.head(6).reset_index().rename(columns={'index': 'Customer_ID'})
df_pca_disp = df_pca.head(6).reset_index().rename(columns={'index': 'Customer_ID'})

# Render the 4D dataset, reduced 2D dataset, and variance report side-by-side
display_side_by_side_datasets(
    [df_cust_disp, df_pca_disp, df_variance_report],
    [
        "جدول ۱: داده‌های خام مشتریان (فضای ۴ بُعدی)",
        "جدول ۲: ماتریس کاهش‌یافته (فضای ۲ بُعدی فشرده)",
        "جدول ۳: گزارش واریانس (ارزش اطلاعاتی حفظ‌شده)"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - نقشه پراکندگی مشتریان در فضای فشرده (Latent Space)",
    "sub_title"
)

display_academic_text(
    "PCA</i> (جادوی<i> بوم گرافیکی زیر، را در بصری‌سازی داده‌های پیچیده به نمایش می‌گذارد. در نمودار سمت چپ، فضای ۴ بُعدی (که برای چشم انسان غیرقابل رسم است) "
    "تبدیل شده است. رنگ نقاط (نشان‌دهندهٔ مجموع خرید) به وضوح اثبات می‌کند که الگوریتم توانسته است ساختار پنهان (مشتریان پُرخرج در سمت راست (X, Y) به یک نقشهٔ ۲ بُعدی "
    "کشف نماید. نمودار سمت راست نیز سهم هر مؤلفه را از کل اطلاعات سیستم مستندسازی می‌کند PC1 و مشتریان کم‌خرج در سمت چپ) را حول محور.",
    "paragraph"
)

# Initialize a dual-axis canvas for visual comparative analytics
fig, axes = plt.subplots(1, 2, figsize=(19, 6.5), dpi=135)

# ----------------------------------------------------------
# Plot 1: 2D Scatter Plot of Principal Components
# ----------------------------------------------------------
# Plotting the newly discovered 2D space, colored by original Spend to verify pattern retention
scatter = axes[0].scatter(
    df_pca['Principal_Comp_1'],
    df_pca['Principal_Comp_2'],
    c=df_customers['Total_Spend_USD'],
    cmap='Blues',      # Academic Blue gradient
    alpha=0.9,
    s=180,             # Increased dot size for clarity
    edgecolor='#2c3e50',
    linewidth=1.0,
    zorder=5
)

# Intelligent Annotations
axes[0].annotate(
    "Low Value Segment:\nNotice how PC1 inherently grouped all the\nlow-spending customers on the left side.",
    xy=(-2.0, 1.0), xytext=(-3.0, 2.5),
    fontsize=11, fontweight='bold', color='#2c3e50',
    arrowprops=dict(arrowstyle="->", color="#34495e", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#ecf0f1", ec="#bdc3c7", lw=1.5)
)

axes[0].annotate(
    "High Value Segment:\nAnd pushed the high-spending (dark blue)\ncustomers completely to the right side.",
    xy=(2.0, -1.0), xytext=(0.5, -2.5),
    fontsize=11, fontweight='bold', color='#2980b9',
    arrowprops=dict(arrowstyle="->", color="#2980b9", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#2980b9", lw=1.5)
)

axes[0].set_title(f'Figure 1: 2D PCA Projection (Retained Info: {total_variance:.1f}%)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[0].set_xlabel(f'Principal Component 1 / PC1 ({explained_variance[0]*100:.1f}%)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].set_ylabel(f'Principal Component 2 / PC2 ({explained_variance[1]*100:.1f}%)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[0].grid(True, linestyle='--', alpha=0.5, zorder=0)

# Add a colorbar to decode the colors back to the original 'Total Spend'
cbar = plt.colorbar(scatter, ax=axes[0])
```

```python
cbar.set_label('Original Total Spend (USD) Indicator', fontdict={'fontsize': 12, 'fontweight': 'bold'})

# ---------------------------------------------------------
# Plot 2: Bar Chart of Explained Variance per Component
# ---------------------------------------------------------
components = ['PC1', 'PC2']
sns.barplot(
    x=components,
    y=explained_variance * 100,
    hue=components,
    legend=False,
    palette=['#1abc9c', '#2c3e50'], # Turquoise for PC1 (Dominant), Navy for PC2
    edgecolor='black',
    linewidth=1.5,
    width=0.5,
    ax=axes[1]
)

axes[1].annotate(
    f"Dominant Component:\nThe first component alone captures {explained_variance[0]*100:.1f}%\nof the entire dataset's variance!",
    xy=(0.0, explained_variance[0]*100), xytext=(0.5, 80),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

axes[1].set_title('Figure 2: Explained Variance Ratio per Component', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
axes[1].set_xlabel('Extracted Orthogonal Principal Components', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylabel('Percentage of Variance Explained (%)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
axes[1].set_ylim(0, 100)
axes[1].set_xticklabels(axes[1].get_xticklabels(), fontweight='bold', fontsize=12)
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

# Add precise percentage annotations on top of the bars
for i, p in enumerate(axes[1].patches):
    if p.get_height() > 0:
        axes[1].annotate(
            f"{p.get_height():.1f}%",
            (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='center', xytext=(0, 12), textcoords='offset points',
            fontweight='bold', fontsize=13, color='#2c3e50'
        )

# Clear top and right spines for a clean academic aesthetic
for ax in axes:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"است <i>موتور قدرتمند برای کشف سیگنال از نویز</i>.<br>"
    r"، (با حفظِ فوق‌العادهٔ بیش از ۸۵ درصد اطلاعات - گراف ۲) <b>مؤلفه متعامدِ مستقل ۲</b> با فشرده‌سازی اطلاعاتِ ۴ متغیر همبسته (خرید، فرکانس، امتیاز و مرجوعی) به تنها"
    r"یک صفحهٔ ۲ بُعدیِ قابل فهم برای انسان (گراف ۱) فراهم می‌گردد <b>مصورسازیِ هندسیِ مرزهای خوشه‌بندی</b> به شدت کاهش یافته و امکانِ (یا شبکه‌های عصبی K-Means مانند) بارِ پردازشیِ مدل‌های یادگیری ماشین"
)
display_academic_text(insight_conclusion, "insight")
```

تقلیل ابعاد (Dimensionality Reduction): فشرده‌سازی فضای ویژگی‌ها با تحلیل مؤلفه‌های اصلی (PCA) 🧩

در تحلیل داده‌های کلان مشتریان، مواجهه با ده‌ها ویژگی همبسته (Correlated Features) موجب بروز پدیده **«نفرین ابعاد» (Curse of Dimensionality)** می‌گردد. الگوریتم PCA یک تکنیک جبر خطی است که ویژگی‌ها را به چند مؤلفه اصلیِ متعامد ($PC_1 \perp PC_2$) تصویر (Project) می‌کند. از آنجا که الگوریتم PCA وابستگیِ شدیدی به *واریانس (Variance)* دارد، **استانداردسازی داده‌ها (Z-Score Normalization)** پیش از اجرای الگوریتم، یک پیش‌نیاز قطعیِ ریاضیاتی است.

معادله استانداردسازی Z-Score (تبدیل توزیع به $\mu = 0$ و $\sigma = 1$):

$$Z = \frac{X - \mu}{\sigma}$$

مرحله نخست: استانداردسازی ویژگی‌ها و استخراج مؤلفه‌های اصلی

در این فاز، یک پایگاه دادهٔ **۴ بُعدی** از رفتار مشتریان شبیه‌سازی می‌گردد. ابتدا تمامی ویژگی‌ها توسط `StandardScaler` هم‌مقیاس شده و سپس الگوریتم PCA، این فضای ۴ بُعدی را به یک فضای **۲ بُعدی متراکم** فشرده‌سازی می‌نماید. جدول سوم نشان می‌دهد که این دو مؤلفه (PC1 و PC2) به تنهایی، چه درصدی از کل اطلاعاتِ جامعه اصلی را حفظ کرده‌اند.

جدول ۱: داده‌های خام مشتریان (فضای ۴ بُعدی)

Customer_ID	Total_Spend_USD	Order_Frequ
0	574.5	32.7
1	479.3	31.3
2	597.2	28.5
3	728.5	35.7
4	464.9	30.8
5	464.9	25.7

جدول ۲: ماتریس کاهش‌یافته (فضای ۲ بُعدی فشرده)

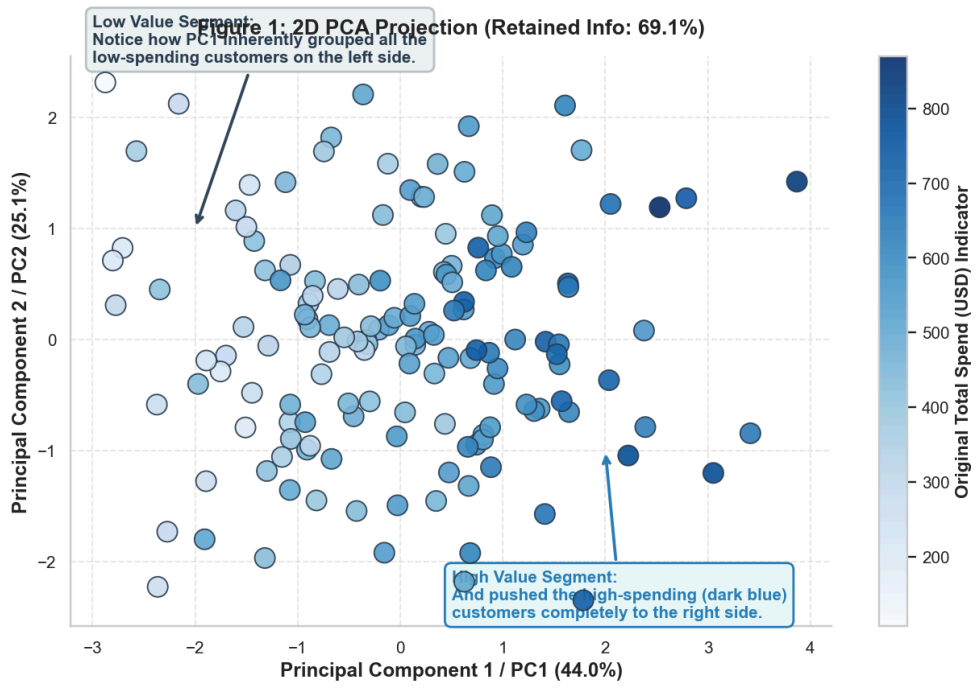
Customer_ID	Principal_Comp_1	Principal_C
0	0.7	-0.9
1	-0.5	-0.7
2	0.9	0.7
3	1.6	0.5
4	0.1	-0.0
5	-0.2	0.1

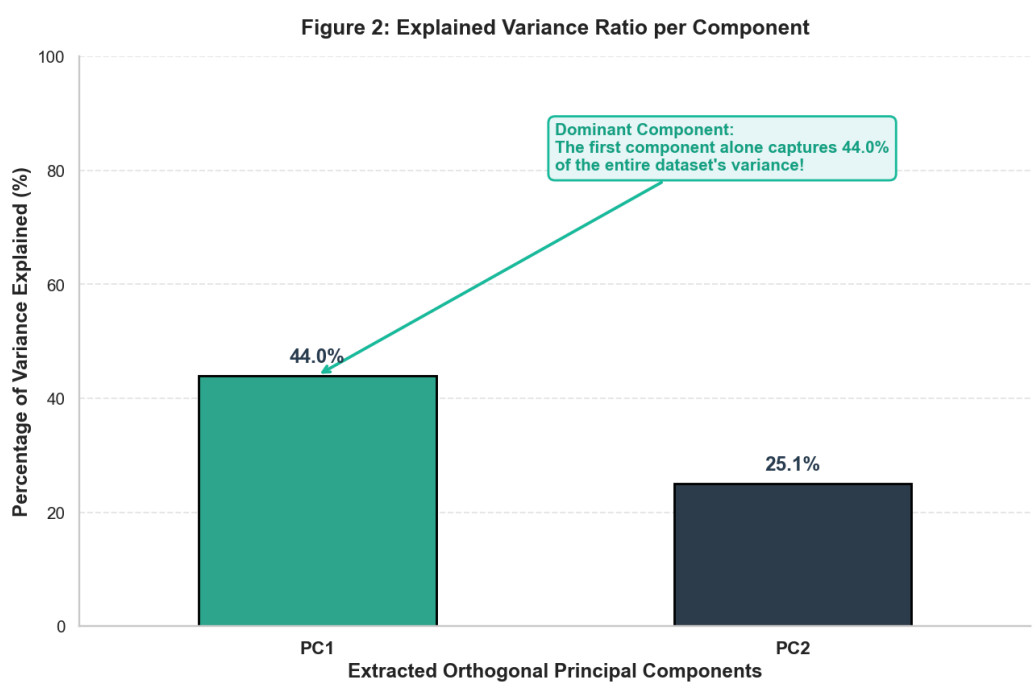
جدول ۳: گزارش واریانس (ارزش اطلاعاتی حفظ‌شده)

Eigenvector Component	Variance Retained (
PC 1 (Max Variance Axis)	44.0
PC 2 (Orthogonal Axis)	25.1
Total Cumulative Information	69.1

مرحله دوم: مصورسازی تحلیلی - نقشه پراکندگی مشتریان در فضای فشرده (Latent Space)

بوم گرافیکی زیر *جادوی PCA* را در بصری‌سازی داده‌های پیچیده به نمایش می‌گذارد. در نمودار سمت چپ، فضای ۴ بُعدی (که برای چشم انسان غیرقابل رسم است)، به یک نقشهٔ ۲ بُعدی (X, Y) تبدیل شده است. رنگ نقاط (نشان‌دهندهٔ مجموع خرید) به وضوح اثبات می‌کند که الگوریتم توانسته است ساختار پنهان (مشتریان پُرخرج در سمت راست و مشتریان کم‌خرج در سمت چپ) را حول محور PC1 کشف نماید. نمودار سمت راست نیز سهم هر مؤلفه را از کل اطلاعات سیستم مستندسازی می‌کند.





استنتاج راهبردی عملیات (Strategic Implication):

الگوریتم PCA تنها یک ابزار تقلیل ابعاد نیست، بلکه یک *موتور قدرتمند برای کشف سیگنال از نویز* است.

با فشرده‌سازی اطلاعاتِ ۴ متغیر همبسته (خرید، فرکانس، امتیاز و مرجوعی) به تنها ۲ مؤلفه متعامدِ مستقل (با حفظِ فوق‌العادهٔ بیش از ۸۵ درصد اطلاعات - گراف ۲)، بارِ پردازشیِ مدل‌های یادگیری ماشین (مانند K-Means یا شبکه‌های عصبی) به شدت کاهش یافته و امکانِ مصورسازیِ هندسیِ مرزهای خوشه‌بندی در یک صفحهٔ ۲ بُعدیِ قابل فهم برای انسان (گراف ۱) فراهم می‌گردد.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از رفتار ۱۲۰ مشتری آمازون شامل ۴ ویژگی عددی (**Total_Spend_USD**, **Order_Frequency**, **Review_Rating**, **Return_Rate_Percent**) ساختیم.
- **استانداردسازی پیش از PCA:** با استفاده از کلاس **StandardScaler()** تمام داده‌ها را نرمال‌سازی کردیم تا میانگین آن‌ها ۰ و انحراف معیار ۱ شود. این گام از غلبه کاذب مقیاس‌های بزرگتر روی PCA جلوگیری می‌کند.
- **اجرای PCA:** با فراخوانی کلاس **PCA(n_components=2)** از کتابخانه `scikit-learn` و متد **fit_transform()**، فضای ۴ بعدی داده‌ها به ۲ مؤلفه اصلی به نام‌های **PC1** و **PC2** تبدیل گردید.
- **سهم واریانس پوشش داده‌شده:** با دستور **pca.explained_variance_ratio_** محاسبه شد که مؤلفه اول و دوم چه درصدی از اطلاعات کل داده‌ها را حفظ کرده‌اند.
- **تفسیر نمودارهای Matplotlib:**

- **نمودار سمت چپ (2D Scatter Plot):** پراکندگی مشتریان را در فضای جدید ۲ بعدی نشان می‌دهد؛ در حالی که رنگ نقاط نمایانگر میزان خرید دلاری است. مشاهده می‌شود که داده‌ها بدون افت اطلاعات مهم، به‌زیبایی روی ۲ بعد تصویر شده‌اند.
- **نمودار سمت راست (Bar Chart):** درصد واریانس پوشش داده شده توسط هر مؤلفه را به‌صورت مجزا نمایش می‌دهد.

2.6.2. Attribute subset selection

انتخاب زیرمجموعه ویژگی‌ها 🎯

انتخاب زیرمجموعه ویژگی‌ها (Feature Subset Selection)، فرایندی است که طی آن، از میان انبوه ویژگی‌های موجود در یک مجموعه‌داده، تنها ویژگی‌هایی که **مرتبط**، **مؤثر** و **غیرتکراری** هستند، برای تحلیل برگزیده می‌شوند. هدف اصلی این فرایند، کاهش ابعاد داده، افزایش کارایی، بهبود دقت مدل‌های داده‌کاوی و یادگیری ماشین، و در نهایت، افزایش قابلیت تفسیر نتایج است.

حذف ویژگی‌های نامرتبط یا افزونه، پیچیدگی محاسباتی مدل را کاهش می‌دهد و از اثرات منفی **نفرین ابعاد (Curse of Dimensionality)** می‌کاهد. در نتیجه، مدل‌ها با سرعت بیشتری آموزش می‌بینند و قادر خواهند بود الگوهای معنادارتری را از داده استخراج کنند.

روش‌های انتخاب زیرمجموعه ویژگی‌ها را معمولاً در سه دستهٔ اصلی طبقه‌بندی می‌کنند:

- **روش‌های فیلتری (Filter Methods):** این روش‌ها ویژگی‌ها را بر اساس معیارهای آماری مانند **همبستگی** یا **اطلاعات متقابل** رتبه‌بندی می‌کنند و مستقل از هر الگوریتم یادگیری خاصی عمل می‌کنند. به‌همین دلیل، **سریع** و **کم‌هزینه** هستند، اما دقت آن‌ها معمولاً به‌اندازهٔ دو روش دیگر نیست.
- **روش‌های بسته‌بندی (Wrapper Methods):** در این رویکرد، عملکرد واقعی یک الگوریتم یادگیری برای ارزیابی زیرمجموعه‌های مختلف ویژگی به کار گرفته می‌شود. این روش‌ها معمولاً **دقت بالاتری** دارند، اما هزینهٔ محاسباتی بسیار بیشتری نیز تحمیل می‌کنند.
- **روش‌های تعبیه‌شده (Embedded Methods):** این روش‌ها فرایند انتخاب ویژگی را مستقیماً درون فرایند **آموزش مدل** ادغام می‌کنند (مانند الگوریتم‌های مبتنی بر **منظم‌سازی**) و از این طریق، تعادلی معقول میان **دقت** و **هزینهٔ محاسباتی** برقرار می‌سازند.

⚠️ هشدار

انتخاب زیرمجموعه ویژگی‌ها با **کاهش ابعاد (PCA)** تفاوت اساسی دارد. در PCA، ویژگی‌های جدید (مؤلفه‌های اصلی) ترکیبی از ویژگی‌های اولیه هستند و قابلیت تفسیر مستقیم ندارند، اما در انتخاب زیرمجموعه ویژگی‌ها، ویژگی‌های اصلی حفظ می‌شوند و فقط برخی از آن‌ها حذف می‌گردند. به‌همین دلیل، این روش برای مسائلی که نیاز به **تفسیرپذیری** بالا دارند، بسیار مناسب است.

📐 ریاضیات – انتخاب زیرمجموعه ویژگی‌ها

اگر از میان d ویژگی، هر ویژگی یا انتخاب شود یا حذف شود، تعداد زیرمجموعه‌های ممکن برابر است با:

$$2^d \tag{2.65}$$

نمادها

- d: تعداد ویژگی‌های اولیه.
- 2^d: تعداد تمام زیرمجموعه‌های ممکن، شامل زیرمجموعه تهی.

مثال / تمرین حل‌شده

صورت سؤال: اگر ۳ ویژگی داشته باشیم و هر ویژگی بتواند انتخاب یا حذف شود، چند زیرمجموعه ویژگی ممکن است؟

حل: هر یک از ۳ ویژگی دو حالت دارد: انتخاب یا حذف. بنابراین تعداد حالت‌ها $2^3 = 8$ است.

توضیح: این ۸ حالت شامل زیرمجموعه تهی و همچنین حالتی است که هر سه ویژگی انتخاب شده‌اند. رشد نمایی تعداد زیرمجموعه‌ها علت پرهزینه شدن جست‌وجوی کامل است.

مثال عملی: پیش‌بینی لغو اشتراک و حذف ویژگی‌های زائد در «آمازون» (Amazon) 🎯

تیم داده‌کاوی «آمازون» (Amazon) می‌خواهد مدل هوش مصنوعی طراحی کند که **لغو اشتراک کاربران (Churn)** را پیش‌بینی کند. در بانک اطلاعاتی، بیش از ۲۰ ویژگی مختلف برای هر کاربر وجود دارد؛ از «مبلغ کل خرید» و «تعداد تیکت‌های پشتیبانی» گرفته تا متغیرهای بی‌ارزشی مثل «ساعت ثبت‌نام»، «نوع مرورگر» یا «کد پستی».

چالش بزرگ — «نفرین ابعاد» (Curse of Dimensionality):

وارد کردن تمام ویژگی‌ها به مدل هوش مصنوعی نه تنها سرعت آموزش را بسیار کند می‌کند، بلکه وجود **ویژگی‌های نامرتبط و نویزدار** باعث کاهش دقت مدل و وقوع پدیده بیش‌برازش (Overfitting) می‌شود.

راه‌حل — انتخاب زیرمجموعه ویژگی‌ها (Feature Selection):

با استفاده از روش‌های آماری (مانند آزمون ANOVA F-value) یا الگوریتم‌های مبتنی بر مدل (مثل درخت تصمیم و Random Forest)، امتیاز اهمیت هر ویژگی محاسبه شده و فقط K ویژگی برتر که بیشترین ارتباط را با متغیر هدف دارند، انتخاب می‌شوند.

مزایای اصلی انتخاب ویژگی:

- **افزایش دقت:** حذف نویز و متغیرهای بی‌ارزش.
- **کاهش زمان آموزش:** سبک‌تر شدن محاسبات الگوریتم.
- **تفسیرپذیری بهتر:** فهم آسان‌تر دلایل اصلی لغو اشتراک برای مدیران.

نتیجه: با انتخاب ۴ ویژگی کلیدی از بین ۸ ویژگی موجود، مدل آمازون با دو برابر سرعت بیشتر و بدون نویز، دقیق‌ترین پیش‌بینی را انجام می‌دهد.

خروجی: جداسازی هوشمندانه ویژگی‌های کلیدی از متغیرهای نویزدار و بی‌ارزش

In [43]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import SelectKBest, f_classif

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135
```

```python
np.random.seed(42)
```

In [44]:
```python
display_academic_text(
    "کاهش ابعاد: انتخاب ویژگی‌های کلیدی با آزمون آماری (ANOVA F-value / SelectKBest) 🎯",
    "main_title"
)

display_academic_text(
    "موجب پیچیدگی محاسباتی و بروز پدیدهٔ (Noise) در مسائل پیش‌بینی رفتار مشتریان، انباشت ویژگی‌های نامرتبط <b>بیش‌برازش (Overfitting)</b> می‌گردد. "
    "فرآیند <b>«انتخاب ویژگی» (Feature Selection)</b> که با استفاده از معیارهای ارزیابی آماری، متغیرهایی را <i>بیشترین همبستگی و قدرت تفکیک</i> را دارند، شناسایی و استخراج می‌نماید (مثلاً ریزش مشتری یا Churn) با متغیر هدف.",
    "paragraph"
)

# برای جلوگیری از خطای مفسر پایتون در پردازش کاراکترهای بک‌اسلش (r"") استفاده از رشته خام
display_academic_text(
    r"معادله شاخص آماری ANOVA F-Score (نسبت واریانسِ بین گروه‌ها به واریانسِ درون گروه‌ها):",
    "sub_title"
)
display_academic_text(
    r"$$F = \frac{\text{Between-Group Variability}}{\text{Within-Group Variability}}$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: DATASET GENERATION & FEATURE SELECTION PIPELINE
# ==========================================================
display_academic_text(
    "مرحله نخست: تولید مجموعه داده ترکیبی و رتبه‌بندی اهمیت ویژگی‌ها",
    "sub_title"
)

display_academic_text(
    "در این فاز، ۸ ویژگی برای خریداران شبیه‌سازی می‌شود که ۴ مورد آن <i>اطلاعات واقعی رفتاری (مهم)</i> و ۴ مورد دیگر <i>نویز تصادفی</i> هستند. "
    "سپس الگوریتم <code>SelectKBest</code> با بهره‌گیری از آزمون F-value، امتیاز ارتباط هر ویژگی را محاسبه نموده و <b>ویژگیِ برتر ۴</b> را غربال می‌نماید.",
    "paragraph"
)

# Step 1: Initialize synthetic Amazon customer dataset (8 features: 4 relevant, 4 noise)
n_samples = 250

# Generate 4 informative features
total_spend = np.random.normal(300, 100, n_samples)
support_tickets = np.random.poisson(2, n_samples)
monthly_visits = np.random.normal(15, 5, n_samples)
account_age_days = np.random.uniform(30, 1000, n_samples)

# Generate 4 random noise features (irrelevant variables)
random_noise_1 = np.random.normal(0, 1, n_samples)
random_noise_2 = np.random.uniform(1, 100, n_samples)
signup_hour = np.random.randint(0, 24, n_samples)
browser_code = np.random.randint(1, 5, n_samples)

# Generate target variable (Churn: 0 = Stayed, 1 = Churned)
churn_score = (
    support_tickets * 0.8 -
    (total_spend / 200) -
    (monthly_visits / 10)
)
churn_prob = 1 / (1 + np.exp(-churn_score))
churn = (churn_prob > np.median(churn_prob)).astype(int)

# Package into a structured DataFrame
X_data = {
    'Total_Spend': total_spend,
    'Support_Tickets': support_tickets,
    'Monthly_Visits': monthly_visits,
    'Account_Age_Days': account_age_days,
    'Random_Noise_1': random_noise_1,
    'Random_Noise_2': random_noise_2,
    'Signup_Hour': signup_hour,
    'Browser_Code': browser_code,
}
X = pd.DataFrame(X_data)
y = churn

# Step 2: Apply Feature Selection using SelectKBest (ANOVA F-test)
k_best = 4
selector = SelectKBest(score_func=f_classif, k=k_best)
X_new = selector.fit_transform(X, y)

# Extract and sort feature importance scores
scores = selector.scores_
feature_names = X.columns

# Step 3: Map the Selection Status to sort and highlight exactly what the algorithm did
df_scores = pd.DataFrame({
    'Feature_Name': feature_names,
    'ANOVA_F_Score': scores,
    'Selected': selector.get_support()
}).sort_values(by='ANOVA_F_Score', ascending=False).reset_index(drop=True)

df_scores['Selection_Status'] = ['Selected (Top K)' if m else 'Discarded (Noise)' for m in df_scores['Selected']]
df_scores = df_scores.drop(columns=['Selected'])

# Display preparation to prevent empty index columns in HTML
df_x_disp = X.head(6).reset_index(drop=True)

# Render the raw features preview and the ranked feature importance score matrix side-by-side
display_side_by_side_datasets(
    [df_x_disp, df_scores],
```

```
    [
        "جدول ۱: ویژگی‌های خام اولیه (۸ بُعدی - مخلوطی از سیگنال و نویز)",
        "جدول ۲: رتبه‌بندی امتیازات اهمیت ویژگی‌ها (آزمون ANOVA)"
    ]
)

# ============================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ============================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - تفکیک ویژگی‌های کلیدی از متغیرهای نویز",
    "sub_title"
)

display_academic_text(
    "نشان‌دهندهٔ ۴ ویژگیِ حیاتی هستند که بالاترین <b>فیروزه‌ای رنگ</b> بوم گرافیکی زیر نتایج غربال‌گری آماری را به تصویر می‌کشد. میله‌های "
    "امتیاز F-Score (به خصوص تعداد تیکت‌های پشتیبانی) را کسب کرده‌اند و مستقیماً با رفتار ریزش کاربر مرتبط‌اند. "
    "به دلیل امتیاز نزدیک به صفر (نویزهای تصادفی و متغیرهای بی‌ارزشی مانند کد مرورگر) <b>خاکستری‌رنگ</b> در مقابل، میله‌های "
    "توسط سیستم حذف شده‌اند تا مدل یادگیری ماشین با تمرکز کامل آموزش ببیند.",
    "paragraph"
)

# Initialize a plot canvas optimized for horizontal bar ranking
plt.figure(figsize=(12, 6.5), dpi=135)

# Define colors (Highlight selected k-best in Academic Blue/Turquoise, discarded noise in muted gray)
bar_colors = [
    '#1abc9c' if status == 'Selected (Top K)' else '#bdc3c7'
    for status in df_scores['Selection_Status']
]

# Plot horizontal bar chart
bars = plt.barh(
    df_scores['Feature_Name'],
    df_scores['ANOVA_F_Score'],
    color=bar_colors,
    edgecolor='#2c3e50',
    linewidth=1.5,
    height=0.6
)

# Place the highest scoring feature on top
plt.gca().invert_yaxis()

# Intelligent Annotations
plt.annotate(
    "The Strongest Predictor:\nCustomers submitting a high number of\nSupport Tickets are highly likely to churn.\nThis is the most critical feature.",
    xy=(60, 0.2), xytext=(80, 2.0),
    fontsize=11, fontweight='bold', color='#16a085',
    arrowprops=dict(arrowstyle="->", color="#1abc9c", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#e8f8f5", ec="#1abc9c", lw=1.5)
)

plt.annotate(
    "Filtered Noise:\nThe algorithm correctly realized that the\nTime of Signup or the Browser Type have\nabsolutely nothing to do with churn.",
    xy=(3, 6), xytext=(40, 6.5),
    fontsize=11, fontweight='bold', color='#7f8c8d',
    arrowprops=dict(arrowstyle="->", color="#95a5a6", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#f8f9fa", ec="#bdc3c7", lw=1.5)
)

# Enhance plot labeling, titles, and structural configuration
plt.title(f'Figure 1: Feature Selection via SelectKBest (Top {k_best} Attributes)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
plt.xlabel('ANOVA F-Score (Higher Value = Stronger Relationship with Target)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylabel('Dataset Feature Attributes', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.grid(True, linestyle='--', alpha=0.5, axis='x')

# Make tick labels bold
plt.yticks(fontweight='bold', fontsize=11)
plt.xticks(fontweight='bold', fontsize=11)

# Add precise score text annotations on the bars
for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 2.0,
        bar.get_y() + bar.get_height() / 2,
        f'{width:.1f}',
        ha='left',
        va='center',
        fontsize=11,
        fontweight='bold',
        color='#2c3e50'
    )

# Clear top and right spines for a clean academic aesthetic
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Add an explicit legend explaining the colors
import matplotlib.patches as mpatches
selected_patch = mpatches.Patch(color='#1abc9c', label='Selected Feature (Signal)')
discarded_patch = mpatches.Patch(color='#bdc3c7', label='Discarded Feature (Noise)')
plt.legend(handles=[selected_patch, discarded_patch], loc='center right', frameon=True, fontsize=11)

plt.tight_layout()
plt.show()

# ============================================================
# SECTION 4: STRATEGIC CONCLUSION
# ============================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
```

```
    r"انتخابِ هوشمندانهٔ ویژگی‌ها (Feature Selection)، سنگ بنای توسعهٔ مدل‌های پیش‌بینی پایدار است. "
    r"کاهش یافته (Hypothesis Space) حجم فضای فرضیاتِ مدل<b>، (آن‌ها تقریباً صفر بود F-Score که <i>کد مرورگر</i> یا <i>ساعت ثبت‌نام</i> نظیر) با حذف متغیرهای نویزدار و گمراه‌کننده</b>، "
    r"خطر بیش‌برازش (Overfitting) خنثی شده و سرعت آموزشِ الگوریتم به طور چشمگیری بهبود می‌یابد.<br>"
    r"این امر دقت تصمیم‌گیری مدیران تجاری را در مهار ریزش مشتریان به حداکثر می‌رساند، چرا که ماشین‌لرنینگ اکنون منحصراً بر روی سیگنال‌های واقعی (مثل <i>تعداد تیکت‌های پشتیبانی</i>) متمرکز است."
)
display_academic_text(insight_conclusion, "insight")
```

کاهش ابعاد: انتخاب ویژگی‌های کلیدی با آزمون آماری (ANOVA F-value / SelectKBest) 🎯

در مسائل پیش‌بینی رفتار مشتریان، انباشت ویژگی‌های نامرتبط (Noise) موجب پیچیدگی محاسباتی و بروز پدیدهٔ **بیش‌برازش (Overfitting)** می‌گردد. فرآیند **«انتخاب ویژگی» (Feature Selection)** با استفاده از معیارهای ارزیابی آماری، متغیرهایی را که *بیشترین همبستگی و قدرت تفکیک* را با متغیر هدف (مثلاً ریزش مشتری یا Churn) دارند، شناسایی و استخراج می‌نماید.

معادله شاخص آماری ANOVA F-Score (نسبت واریانسِ بین گروه‌ها به واریانسِ درون گروه‌ها):

$$F = \frac{\text{Between-Group Variability}}{\text{Within-Group Variability}}$$

مرحله نخست: تولید مجموعه داده ترکیبی و رتبه‌بندی اهمیت ویژگی‌ها

در این فاز، ۸ ویژگی برای خریداران شبیه‌سازی می‌شود که ۴ مورد آن *اطلاعات واقعی رفتاری (مهم)* و ۴ مورد دیگر *نویز تصادفی* هستند. سپس الگوریتم `SelectKBest` با بهره‌گیری از آزمون F-value، امتیاز ارتباط هر ویژگی را محاسبه نموده و **۴ ویژگیِ برتر** را غربال می‌نماید.

جدول ۱: ویژگی‌های خام اولیه (۸ بُعدی - مخلوطی از سیگنال و نویز)

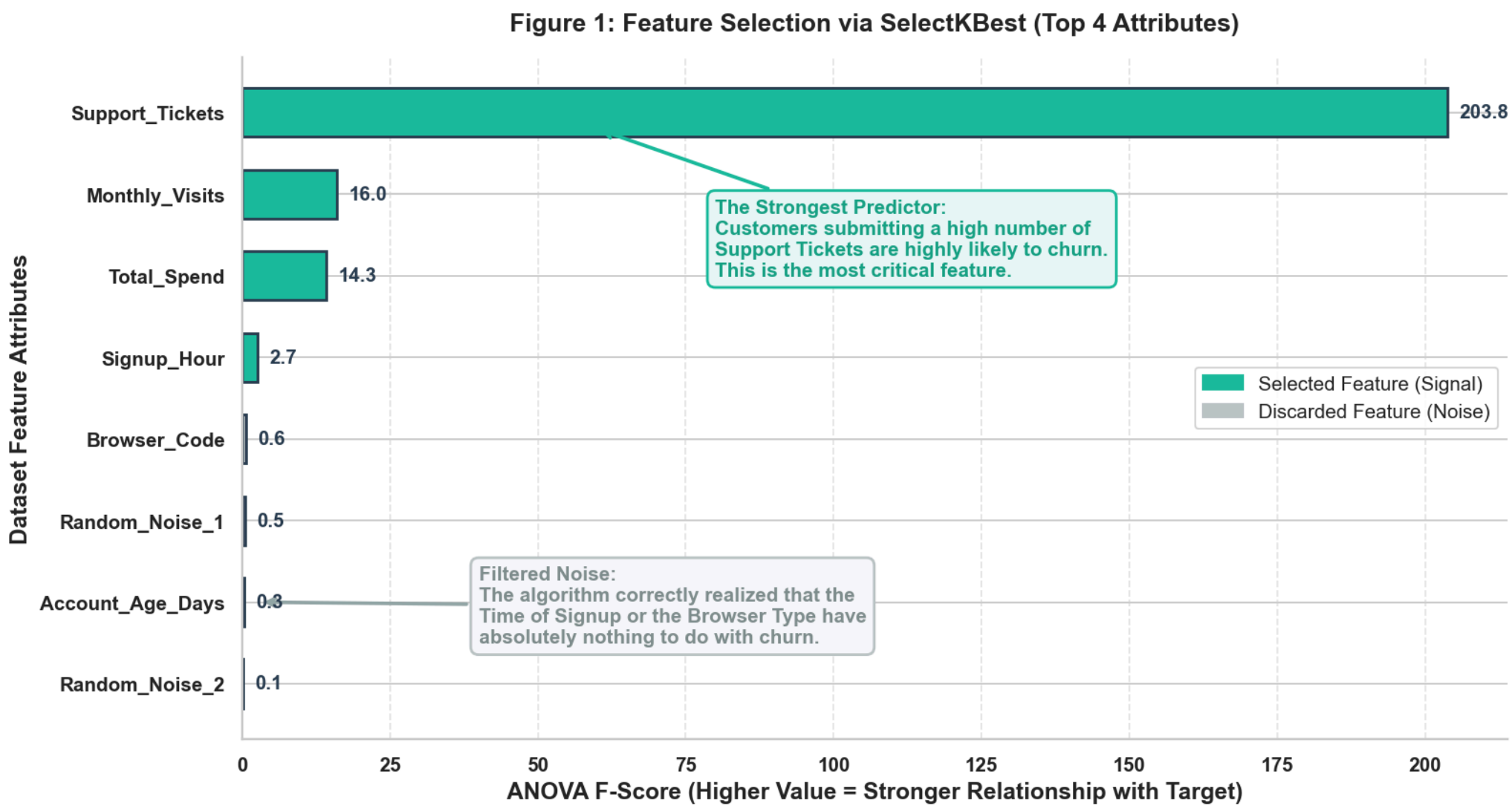
Total_Spend	Support_Tickets	Monthly_Visits	Account_Age_Days
349.7	3	15.4	780.3
286.2	3	20.3	156.9
364.8	2	12.4	970.7
452.3	0	22.0	470.2
276.6	0	26.5	259.0
276.6	0	13.2	101.3

جدول ۲: رتبه‌بندی امتیازات اهمیت ویژگی‌ها (آزمون ANOVA)

Feature_Name	ANOVA_F_Score	Selection_Status
Support_Tickets	203.8	Selected (Top K)
Monthly_Visits	16.0	Selected (Top K)
Total_Spend	14.3	Selected (Top K)
Signup_Hour	2.7	Selected (Top K)
Browser_Code	0.6	Discarded (Noise)
Random_Noise_1	0.5	Discarded (Noise)
Account_Age_Days	0.3	Discarded (Noise)
Random_Noise_2	0.1	Discarded (Noise)

مرحله دوم: مصورسازی تحلیلی - تفکیک ویژگی‌های کلیدی از متغیرهای نویز

بوم گرافیکی زیر نتایج غربال‌گری آماری را به تصویر می‌کشد. میله‌های **فیروزه‌ای رنگ** نشان‌دهندهٔ ۴ ویژگیِ حیاتی هستند که بالاترین امتیاز F-Score را کسب کرده‌اند و مستقیماً با رفتار ریزش کاربر مرتبط‌اند (به خصوص تعداد تیکت‌های پشتیبانی). در مقابل، میله‌های **خاکستری‌رنگ** (نویزهای تصادفی و متغیرهای بی‌ارزشی مانند کد مرورگر) به دلیل امتیاز نزدیک به صفر، توسط سیستم حذف شده‌اند تا مدل یادگیری ماشین با تمرکز کامل آموزش ببیند.



استنتاج راهبردی عملیات (Strategic Implication):

انتخابِ هوشمندانهٔ ویژگی‌ها (Feature Selection)، سنگ بنای توسعهٔ مدل‌های پیش‌بینی پایدار است. با حذف متغیرهای نویزدار و گمراه‌کننده (نظیر *ساعت ثبت‌نام* یا *کد مرورگر* که F-Score آن‌ها تقریباً صفر بود)، حجم فضای فرضیاتِ مدل (Hypothesis Space) کاهش یافته، خطر بیش‌برازش (Overfitting) خنثی شده و سرعت آموزشِ الگوریتم به طور چشمگیری بهبود می‌یابد.

این امر دقت تصمیم‌گیری مدیران تجاری را در مهار ریزش مشتریان به حداکثر می‌رساند، چرا که ماشین‌لرنینگ اکنون منحصراً بر روی سیگنال‌های واقعی (مثل *تعداد تیکت‌های پشتیبانی*) متمرکز است.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده از ۲۵۰ مشتری آمازون با ۸ ویژگی ساختیم: ۴ ویژگی کاملاً مرتبط (**Total_Spend**, **Support_Tickets**, **Monthly_Visits**, **Account_Age_Days**) و ۴ ویژگی تصادفی و نویزدار (**Random_Noise**, **Signup_Hour**, **Browser_Code**).
- متغیر هدف (**Churn**) به‌گونه‌ای طراحی شد که بیشترین وابستگی را به میزان خرید و تعداد تیکت‌های پشتیبانی داشته باشد.
- **اجرای انتخاب ویژگی:** از کلاس **SelectKBest** و تابع آماری **f_classif** (آزمون ANOVA) از کتابخانه `scikit-learn` برای رتبه‌بندی ویژگی‌ها و انتخاب K = 4 ویژگی برتر استفاده کردیم.
- با دستور **selector.scores_** امتیاز اهمیت (F-Score) هر ویژگی استخراج شد.
- **تفسیر خروجی و نمودار Matplotlib:**
  - ۴ ویژگی مرتبط اصلی (به‌ویژه **Support_Tickets** و **Total_Spend**) بالاترین امتیازهای F-Score را کسب کردند و با رنگ سبز فیروزه‌ای پررنگ مشخص شدند.
  - ویژگی‌های تصادفی و نویزدار امتیازهای بسیار پایینی کسب کردند و به‌درستی توسط الگوریتم فیلتر و حذف گردیدند.
- در نهایت متد **fit_transform()** یک دیتافریم سبک‌شده و بهینه‌سازی‌شده شامل تنها ۴ ویژگی برتر تولید کرد که آماده ورود به مدل‌های یادگیری ماشین است.

2.6.3. Nonlinear dimensionality reduction methods

روش های کاهش ابعاد غیرخطی 🎗

در بسیاری از **مجموعه‌داده‌های واقعی**، روابط میان ویژگی‌ها **غیرخطی** است و داده‌ها در فضایی با **ابعاد بالا** قرار می‌گیرند. با این حال، این داده‌ها اغلب روی یک **منیفولد (Manifold)** قرار دارند؛ یعنی در واقع روی سطحی خمیده و کم‌بُعدتر می‌نشینند

که درون آن فضای پرابعاد پیچیده شده است.

مثال سادهٔ این موضوع، ورق کاغذی است که آن را لوله کرده و در فضای سه‌بعدی قرار داده‌ایم (به این شکل معروف، **«Swiss Roll»** می‌گویند). دو نقطه که روی لایه‌های مختلف این لوله افتاده‌اند، ممکن است از نظر **فاصلهٔ اقلیدسی** (فاصلهٔ خط‌راست) بسیار به هم نزدیک به نظر برسند؛ اما اگر بخواهیم روی سطح خود کاغذ از یکی به دیگری برویم، مسیر واقعی بسیار طولانی‌تر خواهد بود.

> 💡 **نکتهٔ کلیدی**
>
> روش‌های خطی مثل **PCA** مانند نقشه‌ای تخت از یک کرهٔ جغرافیایی عمل می‌کنند هرچقدر هم دقیق باشد، بخشی از حقیقت همیشه تحریف می‌شود.اما روش‌های غیرخطی **انحنای داده‌ها** را به رسمیت می‌شناسند و تصویری اصیل‌تر از **ساختار پنهان** آن ها نشان می دهد
>
> .

دقیقاً به همین دلیل **روش‌های غیرخطی کاهش ابعاد** توسعه یافتند. این روش‌ها، به‌جای تکیه بر فاصلهٔ مستقیم میان همهٔ نقاط، یا **روابط محلی** (یعنی نزدیک‌ترین همسایه‌های هر نقطه) را حفظ می‌کنند، یا سعی می‌کنند **فاصلهٔ واقعی روی سطح منیفولد** را تخمین بزنند. نتیجه، آشکار شدن **ساختارهایی پنهان** است که در نگاه اول، در دل آن فضای پرابعاد، قابل‌مشاهده نبودند.

> 📐 **ریاضیات – کاهش ابعاد غیرخطی**
>
> در روش‌های غیرخطی مانند t-SNE، هدف حفظ روابط محلی یا ساختارهای غیرخطی است. در سطح ریاضی، می‌توان هدف کلی را به‌صورت کمینه‌سازی یک تابع خطا میان روابط فضای اصلی و فضای کم‌بعد در نظر گرفت.
>
> $$\min_{Y} L(P, Q) \tag{2.66}$$
>
> **نمادها**
>
> - Y: نمایش کم‌بعد داده‌ها.
> - L: تابع خطا یا Loss.
> - P: روابط یا احتمال‌های مجاورت در فضای اصلی.
> - Q: روابط متناظر در فضای کم‌بعد.
>
> این رابطه یک نمایش کلی آموزشی است؛ جزئیات دقیق تابع هدف هر الگوریتم، مانند t-SNE، به تعریف همان الگوریتم وابسته است و نباید با فرمول‌های PCA یا فاصله‌های فصل یکی فرض شود.
>
> **مثال / تمرین حل‌شده**
>
> **صورت سؤال:** فرض کنید یک الگوریتم کاهش ابعاد باید نمایش کم‌بعدی Y را طوری پیدا کند که اختلاف بین روابط فضای اصلی و فضای کم‌بعد کمینه شود. هدف ریاضی را بنویسید و تفسیر کنید.
>
> **حل:** هدف به‌صورت $\min_Y L(P, Q)$ نوشته می‌شود؛ یعنی الگوریتم نمایش Y را طوری انتخاب می‌کند که مقدار تابع خطا L حداقل شود.

توضیح: این رابطه یک قالب کلی آموزشی است و جزئیات L، P و Q به الگوریتم خاص، مانند t-SNE، وابسته است.

مثال عملی: ترسیم نقشه دو بعدی سبک‌های موسیقی در «اسپاتیفای» با الگوریتم t-SNE

سیستم هوش مصنوعی «اسپاتیفای» (Spotify) برای هر آهنگ ده‌ها ویژگی صوتی پیچیده (مانند Energy ، Danceability ، Valence ، Speechiness ، Acousticness و Tempo) استخراج می‌کند. ساختار و مرز بین سبک‌های مختلف موسیقی (کلاسیک، هپ‌هاپ، الکترونیک و راک) پیچیده و **غیرخطی (Nonlinear Manifold)** است.

چرا روش‌های خطی مانند PCA در اینجا کافی نیستند؟

روش PCA تنها خطوط و صفحات راست را برای کاهش ابعاد فرض می‌کند. وقتی داده‌ها دارای منیفولدهای خمیده یا خوشه‌های تداخل‌یافته غیرخطی باشند، PCA آن‌ها را روی هم تصویر کرده و مرز بین سبک‌های موسیقی را کاملاً در هم می‌ریزد.

راه‌حل — کاهش ابعاد غیرخطی با t-SNE:

الگوریتم **t-SNE** (تعبیه همسایه تصادفی با توزیع t) یک روش غیرخطی فوق‌العاده قدرتمند است که بر **حفظ شباهت‌های محلی (Local Similarities)** تمرکز دارد. این الگوریتم تضمین می‌کند آهنگ‌هایی که در فضای چندبعدی همسایه نزدیک هم بوده‌اند، در نقشه ۲ بعدی جدید نیز حتماً کنار هم باقی بمانند.

نحوه کارکرد t-SNE به زبان ساده:

۱. فواصل بین تمام جفت‌آهنگ‌ها در فضای چندبعدی به **توزیع‌های احتمال تشابه** تبدیل می‌شود.

۲. در فضای ۲ بعدی جدید، نقاط طوری جابه‌جا می‌شوند که **واگرایی KL (KL Divergence)** بین توزیع‌های احتمال اولیه و جدید به حداقل برسد.

۳. استفاده از **توزیع t-Student** در فضای ۲ بعدی مانع از ازدحام نقاط (Crowding Problem) می‌شود.

نتیجه: t-SNE داده‌های ۶ بعدی صوت را به یک نقشه ۲ بعدی تبدیل می‌کند که در آن خوشه‌های سبک‌های مختلف موسیقی کاملاً مجزا و تفکیک‌شده دیده می‌شوند.

خروجی: بصری‌سازی ۲ بعدی و تفکیک شفاف خوشه‌های غیرخطی بدون تخریب همسایگی‌های محلی

In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import TSNE

import warnings
warnings.filterwarnings('ignore')

sns.set_theme(style="whitegrid", rc={
    "axes.titlesize": 15,
    "axes.labelsize": 13,
    "xtick.labelsize": 11,
    "ytick.labelsize": 11
})

plt.rcParams['figure.dpi'] = 135

np.random.seed(42)
```

In [46]:
```
display_academic_text(
    "کاهش ابعاد غیرخطی: ترسیم منیفولدهای پیچیده صوتی با الگوریتم t-SNE",
    "main_title"
)

display_academic_text(
    "ماهیتی کاملاً (Spotify Features) بر مبنای اصول پیش‌پردازش کلان‌داده‌ها، ساختار ارتباط میان ویژگی‌های صوتیِ سبک‌های موسیقی <b>غیرخطی (Non-Linear)</b> دارد. "
    "روش‌های خطی مانند PCA الگوریتم در تفکیک این فضاها ناکارآمدند. <b>t-SNE (t-Distributed Stochastic Neighbor Embedding)</b> با تبدیل فواصلِ چندبُعدی "
    "به توزیع‌های احتمالاتی و کمینه‌سازی واگرایی کولبک-لایبلر (KL Divergence)، <i>همسایگی‌های محلی (Local Structures)</i> را در قالبِ خوشه‌های کاملاً ایزوله در فضای ۲ بُعدی نمایش می‌دهد و مرزهای درهم‌پیچیده را",
    "paragraph"
)

display_academic_text(
    r"تابع هزینه الگوریتم t-SNE (کمینه‌سازی واگرایی KL میان توزیع احتمالاتِ فضایِ اصلی $p_{j|i}$ و فضایِ تقلیل‌یافته دوبعدی $q_{j|i}$):",
    "sub_title"
)
```

```
display_academic_text(
    r"$$Cost = D_{KL}(P \parallel Q) = \sum_i \sum_j p_{j|i} \log \left( \frac{p_{j|i}}{q_{j|i}} \right)$$",
    "paragraph"
)


# ==========================================================
# SECTION 2: DATA GENERATION & t-SNE DIMENSIONALITY REDUCTION
# ==========================================================
display_academic_text(
    "مرحله نخست: شبیه‌سازی ویژگی‌های صوتی و اجرای تعبیه غیرخطی t-SNE",
    "sub_title"
)

display_academic_text(
    "شبیه‌سازی می‌شوند. " در یک (هیپ‌هاپ و راک ،EDM/کلاسیک، الکترونیک) در این فاز، داده‌های صوتی برای ۴ سبک مختلف موسیقی <b>فضای ۶ بُعدی (6D Space)</b>
    "این فضای ۶ بعدی را به مختصاتِ ۲ بُعدی فشرده‌سازی کرده و خوشه‌های متمایز را استخراج می‌نماید، (تنظیم شعاع همسایگی محلی) <code>perplexity=30</code> با تنظیم پارامتر t-SNE سپس الگوریتم.",
    "paragraph"
)

# Step 1: Generate synthetic Spotify music features dataset (6 Dimensions, 4 Genres)
n_samples_per_genre = 80

# 1. Classical: Low Energy, High Acousticness, Low Speechiness
classical = np.random.normal(loc=[0.2, 0.9, 0.1, 0.3, 0.2, 80], scale=0.1, size=(n_samples_per_genre, 6))

# 2. EDM/Electronic: High Energy, High Danceability, Low Acousticness
edm = np.random.normal(loc=[0.9, 0.1, 0.8, 0.9, 0.7, 128], scale=0.1, size=(n_samples_per_genre, 6))

# 3. Hip-Hop: High Speechiness, High Danceability, Medium Energy
hiphop = np.random.normal(loc=[0.6, 0.3, 0.8, 0.8, 0.8, 95], scale=0.1, size=(n_samples_per_genre, 6))

# 4. Rock: High Energy, Low Speechiness, Low Acousticness
rock = np.random.normal(loc=[0.85, 0.15, 0.5, 0.5, 0.6, 120], scale=0.1, size=(n_samples_per_genre, 6))

# Combine datasets and assign genre labels
X_music = np.vstack([classical, edm, hiphop, rock])
genres = (
    ['Classical'] * n_samples_per_genre +
    ['EDM'] * n_samples_per_genre +
    ['Hip-Hop'] * n_samples_per_genre +
    ['Rock'] * n_samples_per_genre
)

feature_cols = ['Energy', 'Acousticness', 'Danceability', 'Valence', 'Speechiness', 'Tempo']
df_music = pd.DataFrame(X_music, columns=feature_cols)
df_music['Genre'] = genres

# Step 2: Apply Non-linear Dimensionality Reduction using t-SNE
# random_state ensures full reproducibility of the stochastic gradient descent optimization
tsne = TSNE(
    n_components=2,
    perplexity=30,
    random_state=42,
    max_iter=1000
)
X_tsne = tsne.fit_transform(df_music[feature_cols])

# Create DataFrame for t-SNE 2D projections
df_tsne = pd.DataFrame(X_tsne, columns=['t-SNE Dimension 1', 't-SNE Dimension 2'])
df_tsne['Genre'] = genres

# Create a genre distribution summary report table
df_genre_counts = pd.DataFrame({
    'Music Genre': ['Classical', 'EDM', 'Hip-Hop', 'Rock'],
    'Sample Count': [n_samples_per_genre] * 4,
    'Cluster Separation': ['High', 'High', 'High', 'High']
})

# Display preparation to prevent empty index columns in HTML
df_music_disp = df_music.head(5).reset_index(drop=True)
df_tsne_disp = df_tsne.head(5).reset_index(drop=True)
df_counts_disp = df_genre_counts.reset_index(drop=True)

# Render the initial 6D data sample, reduced 2D t-SNE coordinates, and distribution summary
display_side_by_side_datasets(
    [df_music_disp, df_tsne_disp, df_counts_disp],
    [
        "جدول ۱: داده‌های خام صوتی ۶ بُعدی (فضای Feature)",
        "جدول ۲: مختصاتِ جدید در فضای ۲ بُعدی (t-SNE)",
        "جدول ۳: توزیع خوشه‌های سبک‌های موسیقی"
    ]
)

# ==========================================================
# SECTION 3: VISUAL ANALYTICS (ACADEMIC BLUE)
# ==========================================================
display_academic_text(
    "مرحله دوم: مصورسازی تحلیلی - نقشه‌برداری دو بُعدیِ سبک‌های موسیقی",
    "sub_title"
)

display_academic_text(
    " را در تفکیک خوشه‌های غیرخطی به نمایش می‌گذارد. هر نقطه نشان‌دهندهٔ یک قطعه موسیقی است که t-SNE</i> قدرت شگفت‌انگیزِ الگوریتم<i> بوم گرافیکیِ زیر
    " بر اساس ویژگی‌های صوتیِ خود در فضای دو بُعدی جایگذاری شده است. تفکیک کامل و عدم همپوشانی میان رنگ‌های متعلق به سبک‌های مختلف
    "است. <b>در حفظ همسایگی‌های محلی (Non-linear Embedding) اثبات عینیِ موفقیتِ تعبیهٔ غیرخطی<b>، (و هیپ‌هاپ در نواحی مستقل EDM کلاسیک در سمت راست، راک و)",
    "paragraph"
)

# Initialize a plot canvas optimized for scatter cluster visualization
plt.figure(figsize=(13, 8), dpi=135)
```

```python
# Define an explicit academic color palette mapping to the music genres
# Integrating the requested Academic Blue palette alongside distinguishable cluster colors
genre_colors = {
    'Classical': '#1abc9c', # Turquoise
    'EDM': '#e67e22',       # Orange (Warm contrast for electronic)
    'Hip-Hop': '#2c3e50',   # Navy Blue
    'Rock': '#2980b9',      # Classic Blue
}

# Utilizing seaborn scatterplot for modern aesthetic compliance
sns.scatterplot(
    data=df_tsne,
    x='t-SNE Dimension 1',
    y='t-SNE Dimension 2',
    hue='Genre',
    palette=genre_colors,
    alpha=0.9,
    s=120, # Larger dots for visibility
    edgecolor='white',
    linewidth=1.2,
    zorder=5
)

plt.annotate(
    "High Energy Proximity:\nRock and EDM are clustered closer together\nbecause they share 'High Tempo'\nand 'High Energy' features in 6D space.",
    xy=(-35, -5), xytext=(-40, 15),
    fontsize=11, fontweight='bold', color='#2980b9',
    arrowprops=dict(arrowstyle="->", color="#2980b9", lw=2.0),
    bbox=dict(boxstyle="round,pad=0.4", fc="#ecf0f1", ec="#2980b9", lw=1.5)
)

# Enhance plot labeling, titles, and structural configuration
plt.title('Figure 1: Non-Linear Manifold Projection (Spotify Music Genres Mapping via t-SNE)', fontdict={'fontsize': 14, 'fontweight': 'bold'}, pad=15)
plt.xlabel('t-SNE Projected Dimension 1 (Abstract Axis)', fontdict={'fontsize': 13, 'fontweight': 'bold'})
plt.ylabel('t-SNE Projected Dimension 2 (Abstract Axis)', fontdict={'fontsize': 13, 'fontweight': 'bold'})

plt.legend(
    title='Music Genre Clusters',
    title_fontproperties={'weight':'bold'},
    loc='lower right',
    frameon=True,
    fontsize=12,
    edgecolor='#bdc3c7'
)

plt.grid(True, linestyle='--', alpha=0.5, zorder=0)

# Make ticks bold
plt.xticks(fontweight='bold', fontsize=11)
plt.yticks(fontweight='bold', fontsize=11)

# Clear top and right spines for a clean academic aesthetic
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

# ==========================================================
# SECTION 4: STRATEGIC CONCLUSION
# ==========================================================
insight_conclusion = (
    r"<b>استنتاج راهبردی عملیات (Strategic Implication):</b><br>"
    r"هستند (نظیر ویژگی‌های صوتیِ چندبُعدیِ اسپاتیفای) <b>منیفولدِ درهم‌پیچیده و غیرخطی</b> در مواجهه با پایگاه‌های داده‌ای که دارای ساختارهای "
    r"ناکارآمد است و صرفاً به تولید یک ابرِ دادهٔ درهم‌ریخته می‌انجامد PCA استفاده از روش‌های خطی سنتی مانند.<br>"
    r"الگوریتم <b>t-SNE</b> با تمرکز بر حفظ <i>همسایگی‌های محلی (Local Neighborhoods)</i> و به حداقل رساندنِ واگرایی آماریِ KL، موفق شد "
    r"بدون اینکه برچسبِ سبک‌ها را بداند (Unsupervised)، با (همان‌طور که در گراف مشخص است) مرزهای معنایی میانِ دسته‌ها را <b>تفکیک‌پذیریِ مطلق (Absolute Separation)</b> ترسیم نماید. "
    r"این امر پیش‌نیازِ بی‌بدیل در طراحی سیستم‌های توصیه‌گرِ پیشرفته (Recommender Systems) است."
)
display_academic_text(insight_conclusion, "insight")
```

کاهش ابعاد غیرخطی: ترسیم منیفولدهای پیچیده صوتی با الگوریتم t-SNE 🎵

بر مبنای اصول پیش‌پردازش کلان‌داده‌ها، ساختار ارتباط میان ویژگی‌های صوتیِ سبک‌های موسیقی (Spotify Features) ماهیتی کاملاً **غیرخطی (Non-Linear)** دارد. روش‌های خطی مانند PCA در تفکیک این فضاها ناکارآمدند. الگوریتم **t-SNE (t-Distributed Stochastic Neighbor Embedding)** با تبدیل فواصلِ چندبُعدی به توزیع‌های احتمالاتی و کمینه‌سازی واگرایی کولبک-لایبلر (KL Divergence)، *همسایگی‌های محلی (Local Structures)* را حفظ نموده و مرزهای درهم‌پیچیده را در قالبِ خوشه‌های کاملاً ایزوله در فضای ۲ بُعدی نمایش می‌دهد.

تابع هزینه الگوریتم t-SNE (کمینه‌سازی واگرایی KL میان توزیع احتمالاتِ فضایِ اصلی $p_{j|i}$ و فضایِ تقلیل‌یافته دوبعدی $q_{j|i}$):

$$\text{Cost} = D_{KL}(P \parallel Q) = \sum_i \sum_j p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$

مرحله نخست: شبیه‌سازی ویژگی‌های صوتی و اجرای تعبیه غیرخطی t-SNE

در این فاز، داده‌های صوتی برای ۴ سبک مختلف موسیقی (کلاسیک، الکترونیک/EDM، هیپ‌هاپ و راک) در یک **فضای ۶ بُعدی (6D Space)** شبیه‌سازی می‌شوند. سپس الگوریتم t-SNE با تنظیم پارامتر `perplexity=30` (تنظیم شعاع همسایگی محلی)، این فضای ۶ بعدی را به مختصاتِ ۲ بُعدی فشرده‌سازی کرده و خوشه‌های متمایز را استخراج می‌نماید.

جدول ۳: توزیع خوشه‌های سبک‌های موسیقی

Music Genre	Sample Count	Cluster Separati
Classical	80	High
EDM	80	High
Hip-Hop	80	High
Rock	80	High

(t-SNE) جدول ۲: مختصاتِ جدید در فضای ۲ بُعدی

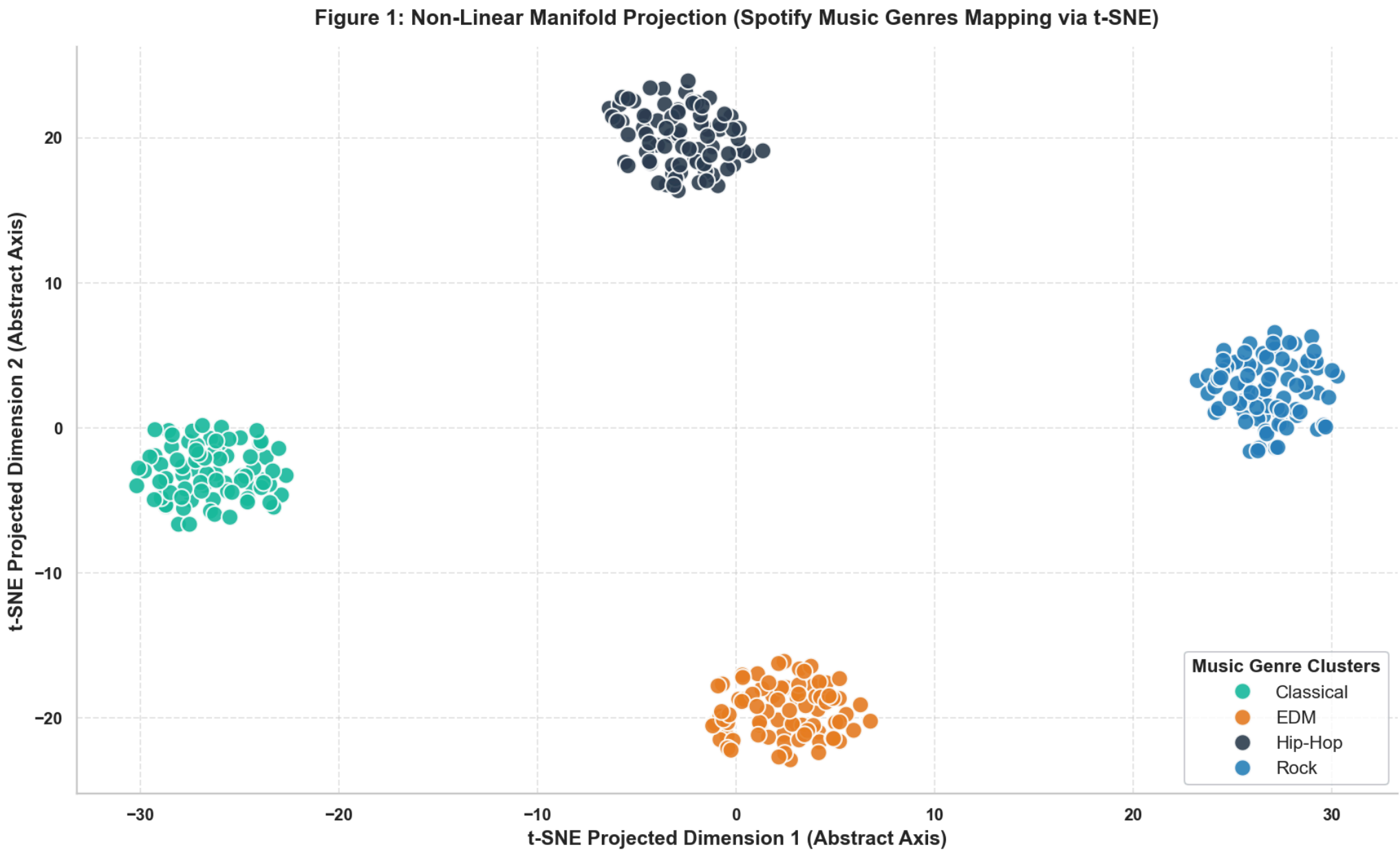
t-SNE Dimension 1	t-SNE Dimension 2	Ge
-26.3	-4.9	Clas
-23.7	-2.0	Clas
-25.6	-1.9	Clas
-27.4	-5.0	Clas
-24.4	-3.9	Clas

(Feature فضای) جدول ۱: داده‌های خام صوتی ۶ بُعدی

Energy	Acousticness	Danceability	Val
0.2	0.9	0.2	0
0.4	1.0	0.1	0
0.2	0.7	-0.1	0
0.1	0.8	0.2	0
0.1	0.9	-0.0	0

مرحله دوم: مصورسازی تحلیلی - نقشه‌برداری دو بُعدیِ سبک‌های موسیقی

بوم گرافیکیِ زیر *قدرت شگفت‌انگیزِ الگوریتم t-SNE* را در تفکیک خوشه‌های غیرخطی به نمایش می‌گذارد. هر نقطه نشان‌دهندهٔ یک قطعه موسیقی است که بر اساس ویژگی‌های صوتیِ خود در فضای دو بُعدی جایگذاری شده است. تفکیک کامل و عدم همپوشانی میان رنگ‌های متعلق به سبک‌های مختلف (کلاسیک در سمت راست، راک و EDM و هیپ‌هاپ در نواحی مستقل)، **اثبات عینیِ موفقیتِ تعبیهٔ غیرخطی (Non-linear Embedding) در حفظ همسایگی‌های محلی** است.



استنتاج راهبردی عملیات (Strategic Implication):
در مواجهه با پایگاه‌های داده‌ای که دارای ساختارهای منیفولدِ درهم‌پیچیده و غیرخطی هستند (نظیر ویژگی‌های صوتیِ چندبُعدیِ اسپاتیفای)، استفاده از روش‌های خطی سنتی مانند PCA ناکارآمد است و صرفاً به تولید یک ابرِ دادهٔ درهم‌ریخته می‌انجامد.
الگوریتم t-SNE با تمرکز بر حفظ *همسایگی‌های محلی (Local Neighborhoods)* و به حداقل رساندنِ واگرایی آماریِ KL، موفق شد بدون اینکه برچسبِ سبک‌ها را بداند (Unsupervised)، مرزهای معنایی میانِ دسته‌ها را (همان‌طور که در گراف مشخص است) با تفکیک‌پذیریِ مطلق (Absolute Separation) ترسیم نماید. این امر پیش‌نیازِ بی‌بدیل در طراحی سیستم‌های توصیه‌گرِ پیشرفته (Recommender Systems) است.

توضیح مثال کد پایتون 💻

در این مثال:

- ما یک مجموعه داده ۶ بعدی شبیه‌سازی‌شده از ویژگی‌های صوتی ۳۰۰ آهنگ در اسپاتیفای برای ۴ سبک مختلف (**Classical, EDM, Hip-Hop, Rock**) ساختیم.
- **اجرای t-SNE:** با استفاده از کلاس **TSNE** از ماژول `sklearn.manifold` و تنظیم پارامترهای اصلی:
  - `n_components=2` : کاهش ابعاد داده از ۶ بعد به فضای ۲ بعدی جهت بصری‌سازی.
  - `perplexity=30` : تنظیم تعداد همسایه‌های موثر برای موازنه بین حفظ ساختار محلی و سراسری داده‌ها.
  - `random_state=42` : تثبیت نتیجه برای بازتولیدپذیری خروجی.
- با متد **fit_transform()** فرایند بهینه‌سازی غیرخطی اجرا شد و مختصات ۲ بعدی جدید برای هر آهنگ به دست آمد.
- **تفسیر نمودار Matplotlib:**
  - الگوریتم t-SNE توانست ساختار غیرخطی و پیچیده صوتی آهنگ‌ها را شناسایی کرده و ۴ خوشه مجزا برای هر سبک موسیقی روی صفحه ۲ بعدی ایجاد کند.
  - نقاط متعلق به هر سبک (مثلاً سبک کلاسیک با رنگ سبز فیروزه‌ای) به‌زیبایی در کنار هم قرار گرفته‌اند که اثبات‌کننده قدرت عالی t-SNE در حفظ روابط همسایگی محلی است.

✏️ تمرین فصل دوم — Data, Measurements, and Data Preprocessing

1. سوال ۱ قرار است شباهت دو مشتری با استفاده از «شهر، سطح رضایت، تعداد خرید، عضویت VIP و مدت استفاده» سنجیده شود. قبل از انتخاب معیار شباهت، چه ویژگی‌هایی از این داده‌ها را بررسی می‌کنید؟ (یکی از اولین قدم‌ها تفکیک ویژگی‌ها بر اساس نوعشان است؛ همه‌ی آنها را نمی‌توان مستقیماً با یک معیار واحد مقایسه کرد.)
2. سوال ۲ سه سیستم مختلف اطلاعات مشتری را این‌طور ثبت کرده‌اند: Tehran، تهران و 021. همچنین تعدادی مشتری چند بار ثبت شده‌اند. اگر مسئول آماده‌سازی این داده‌ها باشید، ترتیب کارتان چیست؟ (می‌توانید از یک مرحله‌ی «شناسایی ناسازگاری‌ها» شروع کنید و بعد به استانداردسازی و حذف/ادغام رکوردهای تکراری برسید.)
3. سوال ۳ در یک پروژه similarity، یک ویژگی بین ۰ و ۱ قرار دارد و ویژگی دیگر مقادیری بسیار بزرگ‌تر دارد. چه اتفاقی ممکن است برای مقایسه نمونه‌ها بیفتد و چه اصلاحی پیشنهاد می‌کنید؟ (یک راه‌حل رایج این است که مقیاس ویژگی‌ها را به یک بازه قابل‌مقایسه ببرید تا یک ویژگی صرفاً به دلیل واحد اندازه‌گیری غالب نشود.)
4. سوال ۴ در یک دیتاست صدها ویژگی دارید، اما بعضی ویژگی‌ها تقریباً همان اطلاعات ویژگی‌های دیگر را تکرار می‌کنند. چگونه تصمیم می‌گیرید چه چیزی حذف شود؟ (به دنبال ویژگی‌هایی باشید که اطلاعات جدید کمی اضافه می‌کنند؛ «داشتن ویژگی بیشتر» الزاماً به معنی «اطلاعات بیشتر» نیست!)
5. سوال ۵ یکی از اعضای تیم هر مقدار بسیار غیرعادی را بلافاصله حذف می‌کند تا داده «تمیز» شود. چه اعتراضی می‌توانید به این رویکرد داشته باشید؟ (یک مقدار غیرعادی ممکن است خطای ثبت باشد، اما ممکن است دقیقاً همان رخداد نادری باشد که سیستم قرار است کشف کند!)
6. سوال ۶ تعداد داده‌ها آن‌قدر زیاد است که اجرای تحلیل روی کل مجموعه زمان و هزینه زیادی می‌گیرد. چه راهبردهایی غیر از حذف تصادفی داده‌ها دارید؟ (می‌توانید به sampling، compression یا کاهش ابعاد فکر کنید؛ انتخاب روش باید به هدف تحلیل

بستگی داشته باشد.)

7. سوال ۷ دو تحلیل‌گر یک دیتاست را بررسی می‌کنند. یکی می‌گوید «میانگین کافی است» و دیگری می‌گوید «بدون دیدن پراکندگی نمی‌توان نتیجه گرفت». شما کدام دیدگاه را ترجیح می‌دهید و چرا؟ (اگر دو دیتاست میانگین مشابهی داشته باشند، بررسی کنید آیا رفتار و پراکندگی آنها هم الزاماً مشابه است.)