The AI Detective: Finding Hidden Patterns in Data with Mean, Median & More

Minor In AI, IIT Ropar

10th Feb, 2025

Welcome, future AI enthusiasts! This module is your guide to understanding some fundamental statistical concepts that form the backbone of data analysis and decision-making in the world of Artificial Intelligence. We'll journey from the vast plains of Africa to the inner workings of Python, uncovering how these seemingly simple ideas can unlock powerful insights from data.

1 The Great Migration and the Curious Case of Numbers

Imagine witnessing the Great Migration of the Masai Mara in Kenya. Millions of wildebeest, zebras, and other animals embark on a perilous journey in search of greener pastures. It's a breathtaking spectacle, a dance of life and survival played out on a grand scale.

Figure 1: The magnificent Great Migration in the Masai Mara, Kenya, showing thousands of wildebeest crossing the plains. Photo credit: Matt Scobel on 500px.

When faced with this immense movement, what questions come to mind? You might wonder:

- **How many** animals participate in this migration each year?
- **What types** of animals make the journey?
- **When** is the peak migration season?
- **Why** do they migrate at all?

These questions are all about data. Data isn't just numbers in a spreadsheet; it's a representation of the real world. And to understand the real world, we need to analyze data.

Think about tracking the number of animals. Are they all moving in a single massive herd, or are they breaking up into smaller groups? How long does the migration take? Gathering this data allows us to understand the rhythm and scale of this natural phenomenon.

But what if we only had a small sample of data about the ratings about the documentary about the migration? Maybe only a few viewers have provided some data? This is where some important concepts come in:

2 Unveiling Central Tendencies: Mean, Median, and Mode

Let's start with a simple example. Suppose we asked ten people to rate a documentary about the Masai Mara migration on a scale of 1 to 5, where 5 is the best. Here are the ratings we received:

4, 4, 4, 4, 4, 5, 5, 5, 1, 4

From just glancing at this data, we can tell that people generally liked the documentary. But how can we summarize this information more precisely?

2.1 The Average Storyteller: Mean

One way is to calculate the *mean*, also known as the average. To find the mean, we add up all the ratings and divide by the number of ratings:

$(4 + 4 + 4 + 4 + 4 + 5 + 5 + 5 + 1 + 4) / 10 = 3.8$

So, the *mean* rating is 3.8. In Python, we can use the `numpy` library to calculate the mean:

```python
import numpy as np

ratings = [4, 4, 4, 4, 4, 5, 5, 5, 1, 4]
mean_rating = np.mean(ratings)
print(f"The mean rating is: {mean_rating}")  # Output: The mean rating is: 3.8
```

Listing 1: Calculating the mean using NumPy

2.2 The Middle Ground: Median

Another way to describe the "center" of our data is to find the *median*. The median is the middle value when the data is sorted. First, let's sort the ratings:

1, 4, 4, 4, 4, 4, 5, 5, 5, 5

Since we have an even number of ratings (10), the median is the average of the two middle values (4 and 4).

$(4 + 4) / 2 = 4$

So, the *median* rating is 4. Let's calculate it in python as well!

```python
import numpy as np

ratings = [4, 4, 4, 4, 4, 5, 5, 5, 1, 4]
median_rating = np.median(ratings)
print(f"The median rating is: {median_rating}") #The median rating is: 4.0
```

Listing 2: Finding the median using NumPy

2.3 The Popular Choice: Mode

Finally, the *mode* is the value that appears most often in the data. In our example, the rating of 4 appears five times, which is more than any other rating. Therefore, the *mode* is 4. Again, let's calculate it in python:

```python
import scipy.stats as stats
ratings = [4, 4, 4, 4, 4, 5, 5, 5, 1, 4]
mode_rating = stats.mode(ratings)
print(f"The mode rating is: {mode_rating[0]}") #The mode rating is: [4]
```

Listing 3: Determining the mode using SciPy

So, in our migration data example, one might also be curious about what the median animal count looks like during October. The mean and mode might also be insightful in their own way.

3 Visualizing Distributions: Histograms

Histograms are powerful tools for visualizing the distribution of data. They show us how frequently different values occur within a dataset.

Imagine plotting the heights of all the students in a class. A histogram would show how many students fall into different height ranges. Most students might cluster around the average height, with fewer students being very tall or very short. This is a *normal distribution*.

Now, consider the Masai Mara migration. Let's say we track the daily number of animals crossing a particular river. A histogram of this data might show a peak during the main migration season, with fewer crossings at the beginning and end.

Here's an example of how to create a histogram in Python using the `seaborn` library:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Generate synthetic data with one peak (normal distribution)
data = np.random.normal(loc=50, scale=10, size=500)  # Mean=50, Std Dev=10

# Create the histogram using seaborn
sns.histplot(data, bins=30, kde=True, color="royalblue")

# Add labels and title
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.title("Histogram")

# Show the plot
plt.show()
```

Listing 4: Creating a histogram with Seaborn

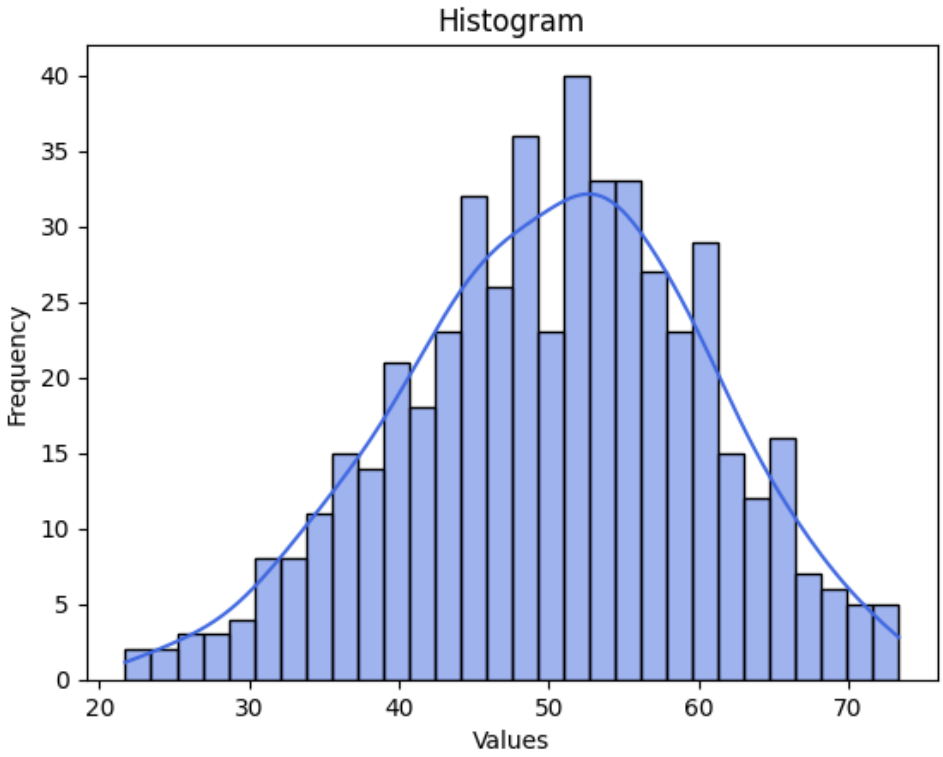

Figure 2: Histogram showing the daily count of animals crossing a river during the Great Migration.

You might also encounter distributions where data is not symetrical, or there might be a few peaks

in the dataset. A visualization with multiple distribution would look like the code below.

```
import matplotlib.pyplot as plt
import numpy as np

# Unimodal distribution
np.random.seed(0)
data_unimodal = np.random.normal(loc=0, scale=1, size=1000)
plt.figure(figsize=(10, 4))
plt.hist(data_unimodal, bins=30, edgecolor='black')
plt.title('Unimodal Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()


# Bimodal distribution
np.random.seed(1)
data_bimodal1 = np.random.normal(loc=-2, scale=1, size=500)
data_bimodal2 = np.random.normal(loc=2, scale=1, size=500)
data_bimodal = np.concatenate((data_bimodal1, data_bimodal2))
plt.figure(figsize=(10, 4))
plt.hist(data_bimodal, bins=30, edgecolor='black')
plt.title('Bimodal Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()


# Multimodal distribution (3 modes)
np.random.seed(2)
data_multimodal1 = np.random.normal(loc=-3, scale=0.8, size=300)
data_multimodal2 = np.random.normal(loc=0, scale=1.2, size=400)
data_multimodal3 = np.random.normal(loc=3, scale=0.9, size=300)
data_multimodal = np.concatenate((data_multimodal1, data_multimodal2,
    data_multimodal3))
plt.figure(figsize=(10, 4))
plt.hist(data_multimodal, bins=30, edgecolor='black')
plt.title('Multimodal Distribution (3 modes)')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

Listing 5: Creating various types of distributions with Matplotlib

4 Box Plots: A Deeper Dive into Data Distribution

Box plots, also known as box-and-whisker plots, offer a concise way to visualize the distribution of data, highlighting key statistics like quartiles and outliers.

4.1 Unpacking the Box Plot

A box plot consists of:

- **A box:** The box represents the interquartile range (IQR), which contains the middle 50% of the data. The left edge of the box corresponds to the 25th percentile (Q1), and the right edge corresponds to the 75th percentile (Q3).
- **A line inside the box:** This line represents the median (Q2), the middle value of the dataset.
- **Whiskers:** The whiskers extend from the edges of the box to the farthest data points within a certain range (typically 1.5 times the IQR).

- **Outliers:** Data points beyond the whiskers are considered outliers and are plotted individually as dots or circles.

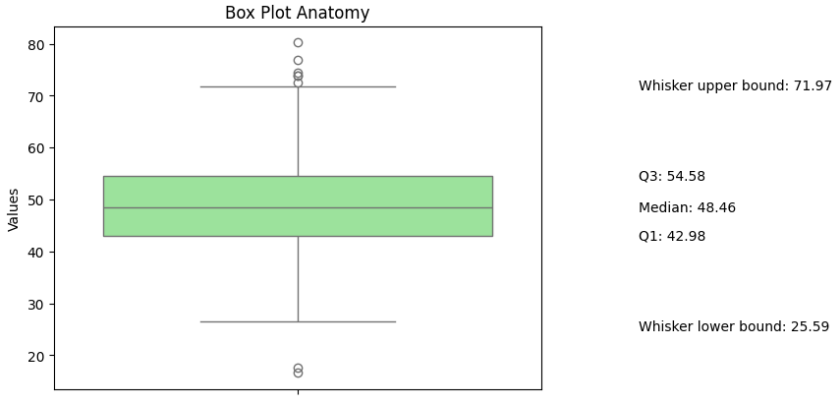

Figure 3: Detailed anatomy of a box plot showing quartiles, median, whiskers, and outliers.

4.2 Creating Box Plots in Python

Here's how to create a basic box plot using Seaborn:

```
#basic Box Plot
import matplotlib.pyplot as plt
import numpy as np
data = np.random.randn(100) * 10 + 50
plt.boxplot(data)
plt.show()
```

Listing 6: Creating a basic box plot with Matplotlib

4.3 Interpreting Box Plots

The beauty of box plots lies in their ability to quickly convey information about the spread, center, and skewness of data.

- **Spread:** A wide box indicates high variability in the middle 50% of the data, while a narrow box suggests low variability.
- **Skewness:** If the median is closer to one edge of the box, the data is skewed. A median closer to the left edge indicates right skewness (positive skew), and a median closer to the right edge indicates left skewness (negative skew).
- **Outliers:** The presence of outliers can indicate unusual or erroneous data points that warrant further investigation.

Let's consider different scenarios to see how box plots help with different datasets

4.4 Case Study: Comparing Student Scores

Imagine comparing the scores of two different classes on the same test. A box plot for each class could reveal:

- Which class has a higher median score.
- Which class has a wider range of scores (more variability).
- Whether either class has any outliers (students who performed exceptionally well or poorly).

By looking at these aspects, we can draw conclusions about the performance of the two classes and determine what additional insights can be extracted.