Data Science X Logistic Regression (DSLR)

Harry Potter and a Data Scientist

Summary: Write a classifier and save Hogwarts!

School: 42 Paris
Session: 2025

Author: Diego Agudelo
Date: 29/04/2025

TABLE OF CONTENTS

Chapter 1: Describe.py

Mandatory

Yes, this is a common practice in data science roles. Before building predictive models or generating complex visualizations, it is essential to understand the raw structure and quality of the data.

This process is part of what is known as Exploratory Data Analysis (EDA) and relies heavily on concepts from descriptive statistics [1], [2].

The goal is to:

- Detect anomalies or outliers.
- Evaluate data completeness and integrity.
- Understand distributions and tendencies.
- Guide preprocessing steps *(normalization, transformations, etc.)*

This initial descriptive analysis builds the foundation for every data-driven decision that follows.

The describe.py file is used to generate a statistical overview of all numeric columns in a dataset. It conforms to the behavior of pandas describe () method, without ever using it, and goes beyond that with bonus stats [3] .

- **Count:** The number of values that are not missing.
- **Mean:** Arithmetic means. $\mu = \frac{1}{n}\Sigma x_i$ [1], [2]
- **Std:** Standard deviation: a measure of dispersion [2].
- **Min/Max:** Minimum and maximum values.
- **25% / 50% / 75% (25%, 50%, 75%):** Quartiles calculated by interpolation [3].

> **Note: All statistics are calculated manually without a specialized library.**

Standard Deviation (STD)

Standard deviation measures how scattered the data is around the mean. In other words, it tells you if the values are grouped or scattered [2].

$$std = \sqrt{\left(\Sigma \frac{(x_i - mean)^2}{n}\right)} \; or \; std = \sqrt{\left(\frac{\Sigma (x_i - \mu)^2}{n}\right)}$$

- std : **standard**, standard deviation (final value we are looking for)
- Σ : **sum**, we add up all the elements of the sequence.
- x_i : **i-th value**, the i value in your list (e.g. 10, 12, etc.)
- μ (mu): **mean**, the average of all values.
- $(x_i - \mu)^2$: **squared difference**, the difference between xi and the mean, squared.
- n : **count**, the total number of values

For the population

$$std = \sqrt{\left(\frac{\Sigma (x_i - \bar{x})^2}{n}\right)}$$

For a sample (the most common)

$$std = \sqrt{\left(\frac{\Sigma (x_i - \bar{x})^2}{n - 1}\right)}$$

If the standard deviation is relatively large compared to the mean (including max and min values), consider the following actions:

- Transform, delete, or filter the data.
- Normalize or standardize the values.
- Detect outliers (anomalous values, usually due to input errors).
  - 68% of values between: Mean ± 1 * std
  - 95% of values between: Mean ± 2 * std
  - 99.7% of values between: Mean ± 3 * std
- Select features with reasonable standard deviation/mean values.

Interpolation of values Q1, Q2, Q3 (25%, 50%, 75%)

Used to calculate percentiles (25%, 50%, 75%) via linear interpolation [3].

$$k = (n - 1) * p$$

$$d = k - f$$

$$f = int(k)$$

$$c = (f + 1) \; or \; (n - 1)$$

$$interpolated = value_{low} + (value_{high} - value_{low}) * d$$

- n : list size
- p : percentile (0.25, 0.5, 0.75)
- k : position (fractional)
- f : floor(k), Integer part of k, the lower index
- c : ceil(k), Next integer (the upper subscript)

EXAMPLE:

$$list = [2, 4, 6, 8, 10, 12, 14, 16]$$

$$n = 8$$

$$We \; are \; looking \; for \; the \; 25\% \; (Q1) \rightarrow p = 0.25$$

$$k = (8 - 1) * 0.25 = 7 * 0.25 = 1.75$$

$$f = int(k) = 1 \; \# \rightarrow value: list[1] = 4$$

$$c = f + 1 = 2 \; \# \rightarrow value: list[2] = 6$$

Interpolation:

$$result = list[f] + (list[c] - list[f]) * (k - f)$$

$$= 4 + (6 - 4) * (1.75 - 1)$$

$$= 4 + 2 * 0.75 = 4 + 1.5 = \mathbf{5.5}$$

EXAMPLE:

$$list = [10, 20, 30, 40]$$

$$k = (n - 1) * p = (4 - 1) * 0.25 = 0.75$$

$$f = 0 \; (value: 10)$$

$$c = 1 \; (value: 20)$$

$$d = k - f = 0.75$$

EXAMPLE:

$$liste = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

$$n = 10$$

$$60\,\% \rightarrow p = 0.6$$

$$k = (10 - 1) * 0.6 = 9 * 0.6 = 5.4$$

$$f = int(k) = 5$$

$$c = f + 1 = 6$$

$$d = k - f = 5.4 - 5 = 0.4$$

$$valeur = list[f] + (list[c] - liste[f]) * d$$

$$= liste[5] + (liste[6] - liste[5]) * 0.4$$

$$= 6 + (7 - 6) * 0.4$$

$$= 6 + 1 * 0.4$$

$$= 6.4$$

Interpretation of results:

- $k = 5.4$ (Theoretical position of the 60th percentile)
- $f = 5$ (Floor index)
- $c = 6$ (Ceiling index)
- $d = 0.4$ Position between the two, fractional offset between two integer indices
- Outcome 6.4 60th percentile value

Print values like a table in terminal, and to export into a .txt file

```
$> describe.[extension] dataset_train.csv
```

	Feature 1	Feature 2	Feature 3	Feature 4
Count	149.000000	149.000000	149.000000	149.000000
Mean	5.848322	3.051007	3.774497	1.205369
Std	5.906338	3.081445	4.162021	1.424286
Min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
Max	7.900000	4.400000	6.900000	2.500000

$$f"\{val:>15\}"$$

- ": " Indicates that we are going to format
- " $>$ " : Right Aligned
- "15" : Total Allocated Width = 15 Characters
- ":<15" : Left aligned
- ": $>$ 15" : Aligned to the right
- ": ^15" : Centered

Bonus Stats [4], [5], [6]

- Range: Extent of data (max-min).
- Variance: Average dispersion around the mean. $\sigma^2 = \frac{1}{n}\Sigma(x_1 - \mu)^2$
- CV: Coefficient of variance to measure the level of dispersion.
- IQR: Interquartile range (Q3 -Q1), useful to detect outliers.
- Skewness: Distribution asymmetry [5].
  - Skew > 0: long tail on the right.
  - Skew < 0: long tail on the left.
  - Based on the third moment [2].
- Kurtosis: Measures the peakedness or flatness [6], [2].
  - Kurtosis > 0: sharp peak.
  - Kurtosis < 0: flat distribution
  - Based on the fourth moment

Moments

The term **moment** in statistics refers to a mathematical quantity used to describe the **shape of a data distribution**. It originates from mathematics and physics (like the "moment of inertia") and follows a logic of **progressive order.**

Each **moment of order n** is based on a general formula:

$$Moment_n = \frac{1}{N}\sum_{i=1}^{N}\left(X^{(i)} - \mu\right)^n$$

where μ is the mean, and n is the order of the moment.

These moments allow us to analyze different properties of the distribution:

- **Mean:** Central tendency

$$\mu = \left(\frac{1}{n}\right) \cdot \sum x^{(i)}$$

- **Variance:** Dispersion around the mean

$$\sigma^2 = \left(\frac{1}{n}\right)\sum\left(x^{(i)} - \mu\right)^2$$

- **Skewness:** Asymmetry (left or right tail)

$$Skewness = \left(\frac{1}{n}\right)\sum\frac{\left(x^{(i)} - \mu\right)^3}{\sigma^3}$$

- **Kurtosis:** Peakedness and weight of the tails

$$Kurtosis = \left(\frac{1}{n}\right)\sum\frac{\left(x^{(i)} - \mu\right)^4}{\sigma^4} - 3$$

Each higher-order moment adds new information about the shape:

- Odd moments (like order 3) keep the direction of the deviation.
- Even moments (like order 2 or 4) focus on the magnitude of the deviation.

Range

Specifies the **range** of values (the difference between the largest and smallest value).

$$max - min$$

Understanding the extreme limits.

Variance

Measures the **dispersion** of values around the mean (the larger the mean, the more scattered the data).

$$\sigma^2 = \left(\frac{1}{n}\right)\sum(xi - \mu)^2$$

$$\text{Variance} = \frac{1}{n}\sum(x_i - \bar{x})^2$$

- μ = moyenne (mean)
- σ = standard deviation (std)
- x_i = individual value
- n = total number of values

Understand data **dispersion**.

Coefficient of variance

$$CV = \frac{std}{mean} \cdot 100$$

- If CV is higher, the dispersion is important compared to the mean.
- If CV is closer to 0, the data is more concentrated around the mean.
- If CV is more than 100%, there are strong variations.

IQR

It gives the **interquartile range**, very useful for detecting **outliers**.

$$Q3 - Q1\ (75\% - 25\%)$$

$$Low\ IQR = Q1 - 1.5\ IQR$$

$$High\ IQR = Q3 + 1.5\ IQR$$

Identify **outliers**. If a value is outside the set limits (Low and High), it is an outlier.

Skewness

Measures the skewness of the distribution:

- Skew > 0: Long tail on the right
- Skew < 0: Long tail on the left

$$\left(\frac{1}{n}\right)\frac{\sum[(xi\ -\ \mu)^3]}{\sigma^3}$$

$$\text{Skewness} = \frac{1}{n}\sum\left(\frac{x_i - \bar{x}}{\sigma}\right)^3$$

- μ = mean
- σ = standard deviation (std)
- x_i = individual value
- n = total number of values

See if the data is biased in one direction.

If mean < median (50% quartiles), the skewness could be < 0. It could deform the value of the mean; the data is not balanced.

So, we need to:

- Confirm if the data is correct.
- Identify errors using logic limits about the values (Example: is it possible that a student has –8 in Divination?)
- If value mean is so deformed, we can use median instead it to calculate the other coefficients and values (not to calculate if the formula asks mean).
- Transform values in log, robust scaler, etc.

Kurtosis

Measures the **flattening** or concentration of data in the distribution:

- Kurtosis = 0: Normal peaks and standard distribution.
- Kurtosis > 0: very concentrated distribution (marked peaks and too many outliers).
- Kurtosis < 0: flat distribution, more spread.

$$\left(\frac{1}{n}\right)\frac{\sum[(xi\ -\ \mu)^4]}{\sigma^4}\ -\ 3\ (excess\ kurtosis)$$

- μ = moyenne (mean)
- σ = standard deviation (std)
- x_i = individual value
- n = total number of values
- -3 = It allows for a normal distribution to 0, kurtosis excess.

Know if the distribution is **sharp or flat** and detect extremes.

Pedagogical note: This measurement comes from statistical moments (order 3 or 4) and allows us to have an intuition about the shape of the distribution. Widely used in data science to detect anomalies or understand trends in a dataset [4]**.**

Results interpretation

Transfiguration

- **Mean**: 1008.21
- **Std**: 154.88
- **Min** / Max: 0.00 / 1098.96
- **Skewness**: -5.85 → very strongly asymmetric to the left (many high values, few weak ones)
- **Kurtosis**: 35.17 → Very precise distribution, with concentrated extreme values

Interpretation:

This material is very biased. Most students have high scores, but a minority have very low scores, close to 0. This can indicate a bug, absences, or a strong pedagogical effect (all or nothing). This material would require analyses or **logarithmic transformation** before it could be used in a model.

Astronomy

- **Mean**: 39.00
- **Std**: 515.10 (!)
- **Skewness**: -0.09 → close to 0, so slightly symmetrical
- **Kurtosis**: -1.68 → Flat distribution, few concentrated values

Interpretation:

Astronomy's scores are widely dispersed, with large variance. The fact that the mean is very small compared to the standard deviation and the amplitude (nearly 2000 range points) suggests extreme values in both directions, and here it would be necessary to **check whether there are any inconsistent scales or input errors.**

Muggle Studies

- **Mean**: -219.68
- **Skewness**: +0.80 → Asymmetrical distribution on the right
- **Kurtosis**: -0.72 → flat distribution
- **IQR**: 813.19 → very wide

Interpretation:

The material has negative grades (!) which seems surprising — is this normal? It is probably a discrepancy in the data (normalization error or transformation). The average is strongly negative while the maximum is positive. The distribution is biased to the right: some students have very high results; the majority are low. **This could distort a pattern if not corrected.**

Charms

- **Mean**: -243.37
- **Std**: 8.78
- **Skewness**: +0.39 -> tail on the right
- **Kurtosis**: -1.08 -> flat distribution

Interpretation:

Another material with a negative average, but this time the dispersion is low. The data is slightly skewed to the right, so the majority is concentrated around the mean. The distribution is flat. **Maybe it's transformed data, but the scale seems more controlled than for Astronomy or Muggle Studies.**

Flying

- **Mean**: 21.96
- **Std**: 97.63
- **Min / Max**: -181.47 / 279.07
- **Skewness**: +0.88 → tail on the right
- **Kurtosis**: -0.15 → flat distribution

Interpretation:

Flying has negative and positive values, a modest mean, and an asymmetric distribution on the right. Extreme values are common (range of 460!). There may be inconsistent scaling between subjects. **Flying could be interesting to model but would require robust standardization (like RobustScaler on IQRs).**

Conclusion

- **Transfiguration**: To watch out for outliers → log transform or winsorization
- **Muggle Studies and Charms**: Possible Data Error → Re-Origin
- **Astronomy and Flying**: high variance → recommended standardization
- **Skewness** > |1| → transformation de variable
- **Kurtosis** > 3→ watch out for peaks → can bias training

Chapter 2: Data Visualization

After computing descriptive statistics, the next natural step is to **visualize the data**.

This is a standard practice in any data science workflow, as visualization helps to **detect patterns, trends, or anomalies** that raw numbers might hide.

Data visualization is a form of **descriptive statistics**, and it is at the core of **Exploratory Data Analysis (EDA)** [1]**,** [2]**.** It helps you:

- Understand the distribution of values (histograms)
- Identify relationships between variables (scatter plots)
- Spot clusters, outliers, or inconsistent data points
- Communicate insights more effectively to others

In this part, we explore different plots, especially **histograms, scatter plots, and pair plots**, and how they relate to **correlation analysis.**

This step is **critical before building any predictive model,** because it gives a visual intuition of the dataset's structure and guides preprocessing choices such as scaling, feature selection, or transformation.

Histogram

A histogram is a graph that represents the distribution of a numerical variable by dividing it into intervals (called "bins") and counting how many observations fall into each interval.

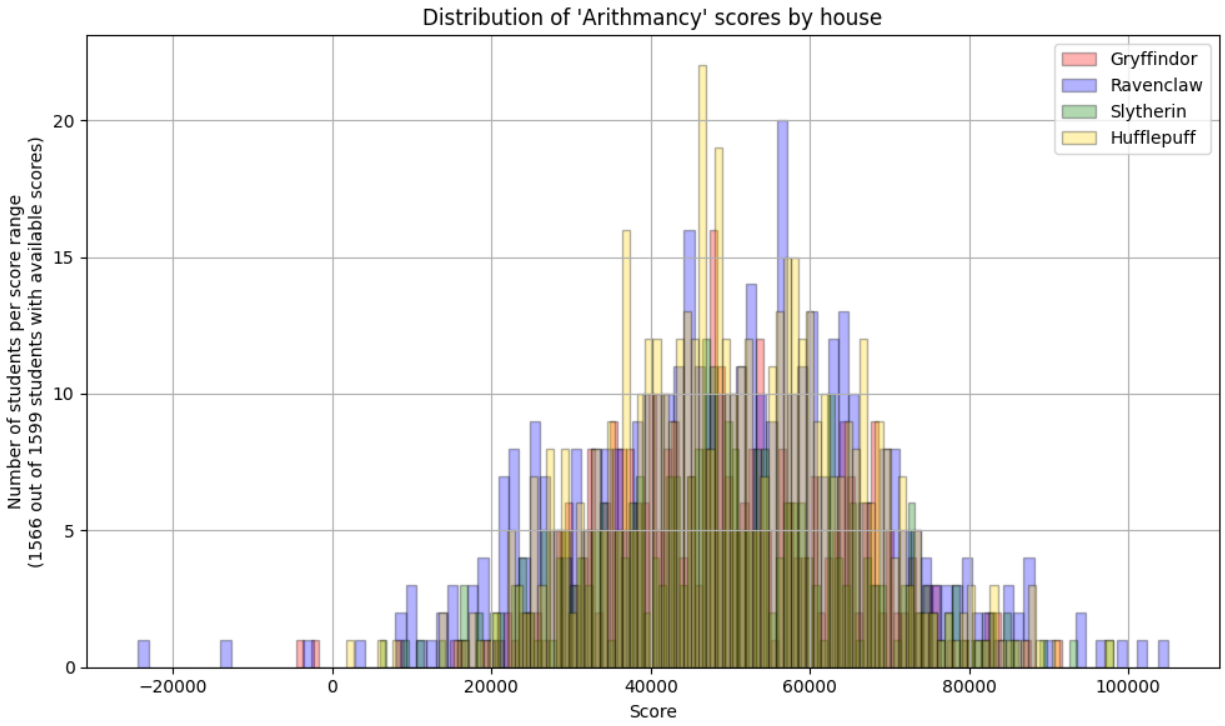
It helps detect:

- If the values are concentrated or dispersed.
- If the distribution is symmetrical, skewed, or multimodal.
- The presence of outliers (e.g. an isolated bar far away).

For example: A tall bar at the far right of the histogram might indicate a small group of students with very high scores – potentially outliers.

Results interpretation

Arithmancy



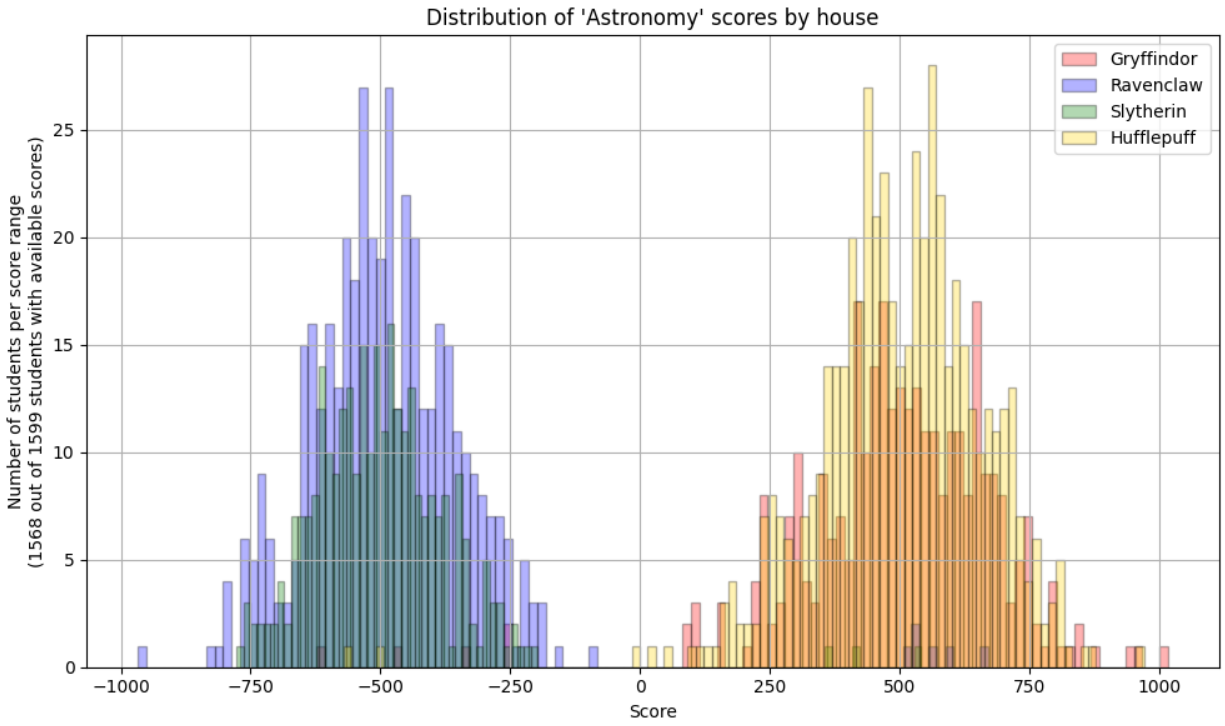
Observation of the graph:

- Distributions are generally centered around 50,000, with a peak between 40,000 and 60,000.
- The four houses are very superimposed.
- Some negative extreme values (< 0) are visible but rare.

Interpretation:

- The subject does not seem to be discriminating for separating the houses: the scores follow similar distributions.
- There is probably some noise or outlier in the negative values, but without massive impact.
- Potentially useful for modeling, but not alone. It will have to be combined with other features.

Astronomy



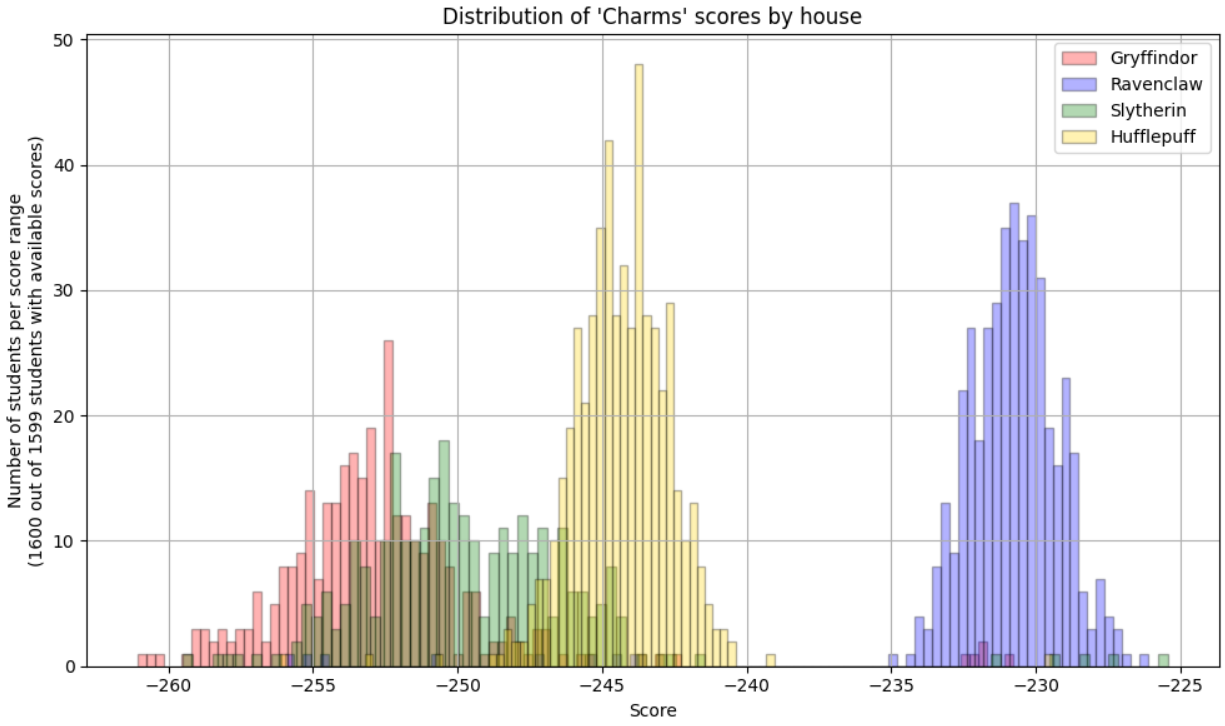
Observation of the graph:

- Two very distinct groups appear:
- Gryffindor + Hufflepuff concentrated around +500
- Ravenclaw + Slytherin around -500
- This forms two well-separated modes, with very little overlap.

Interpretation:

- The variable is extremely useful for classification!
- It would allow Gryffindor/Hufflepuff to be separated from the others with high accuracy.
- Ideal for a supervised model: very informative.

Charms



Observation of the graph:

- Very clear separation into four peaks:
- Gryffindor around -253
- Slytherin around -248
- Hufflepuff around -243
- Ravenclaw around -230
- Almost no overlap: the groups are clearly defined.

Interpretation:

- This material is perfect for identifying the house.
- It is the most discriminating variable among the three.
- It should receive significant weight in your model (logistic regression or other).

Scatter plot

A scatter plot displays the relationship between two numerical variables, each one on an axis (x and y). Each point represents one observation.

It is useful to detect:

- Linear correlation (positive or negative)
- Clusters of data
- Deviating points (potential outliers)

If the points are spread in all directions with no shape → no correlation.

If some dots are far from the main cloud, these may be anomalies.

Correlations: To work with scatter plots.

Pearson correlation is **the most used measure** to analyze **the linear relationship between two numerical variables**.

Intuitive visualization of the linear relationship: A scatter plot allows you to **visualize** if the points align (or not) in a direction:

- Top right→ positive correlation
- Downright→ negative correlation
- Random cloud → no correlation

And Pearson **quantifies** exactly this trend in a single value between **-1** and **+1**.

Pearson's correlation comes from the field of statistics [1], [4].

- It was developed by **Karl Pearson**, a famous British statistician of the nineteenth century.
- It therefore comes from **mathematical statistics**, more precisely from **data analysis** and **descriptive statistics**.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

- x_i : value of observation i in the first variable (e.g. Astronomy)
- y_i : value of observation i in the second variable (e.g. Divination)

- $\bar{x}$: average of x_i values
- $\bar{y}$: average of y_i values
- n : total number of observations (e.g. number of students)
- $\sum$: sum symbol (all the elements are added)

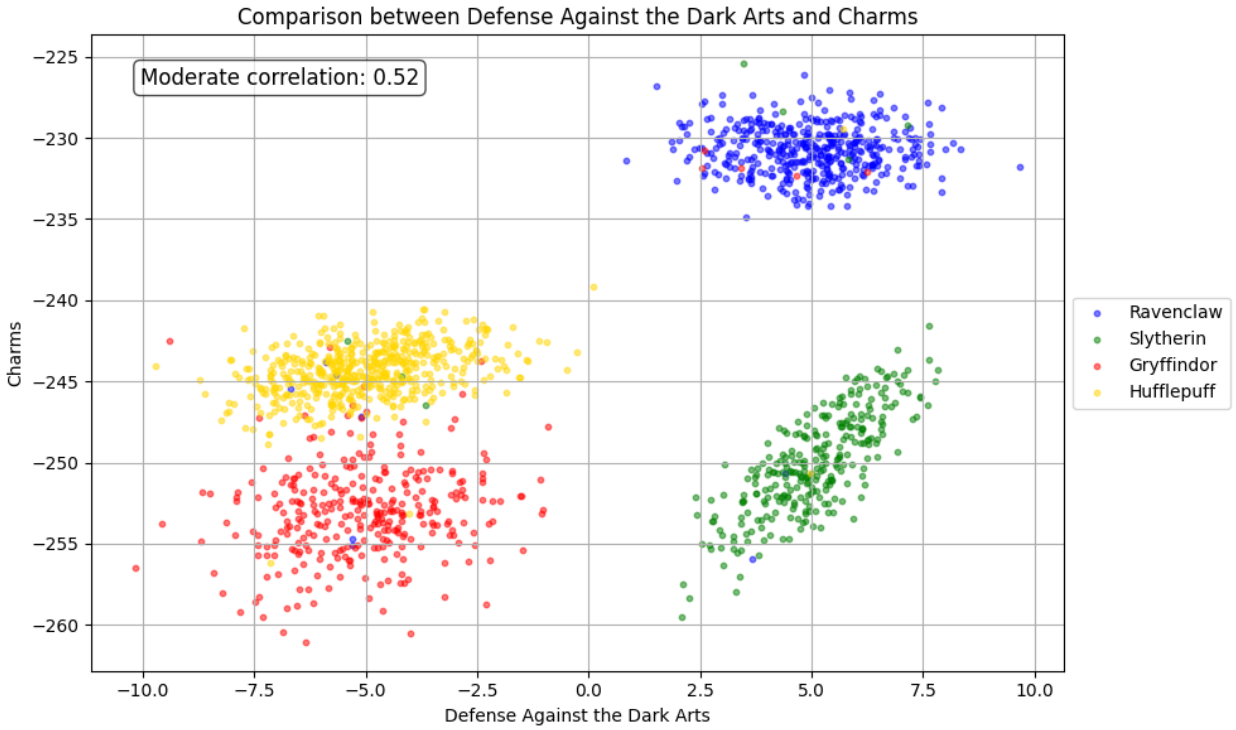
The **numerator** measures the **covariance** between X and Y (how they vary together).

The **denominator** is normalized by the **standard deviation** of X and Y.

The result $r \in [-1,1] r \ in \ [-1, \ 1] r \in [-1,1]$:

- **+1**: perfectly positive correlation (X increases → Y too)
- **0**: No linear correlation
- **-1**: perfectly negative correlation (X increases → Y decreases)

Defense Against the Dark Arts vs Charms



Correlation: 0.52 (moderate)

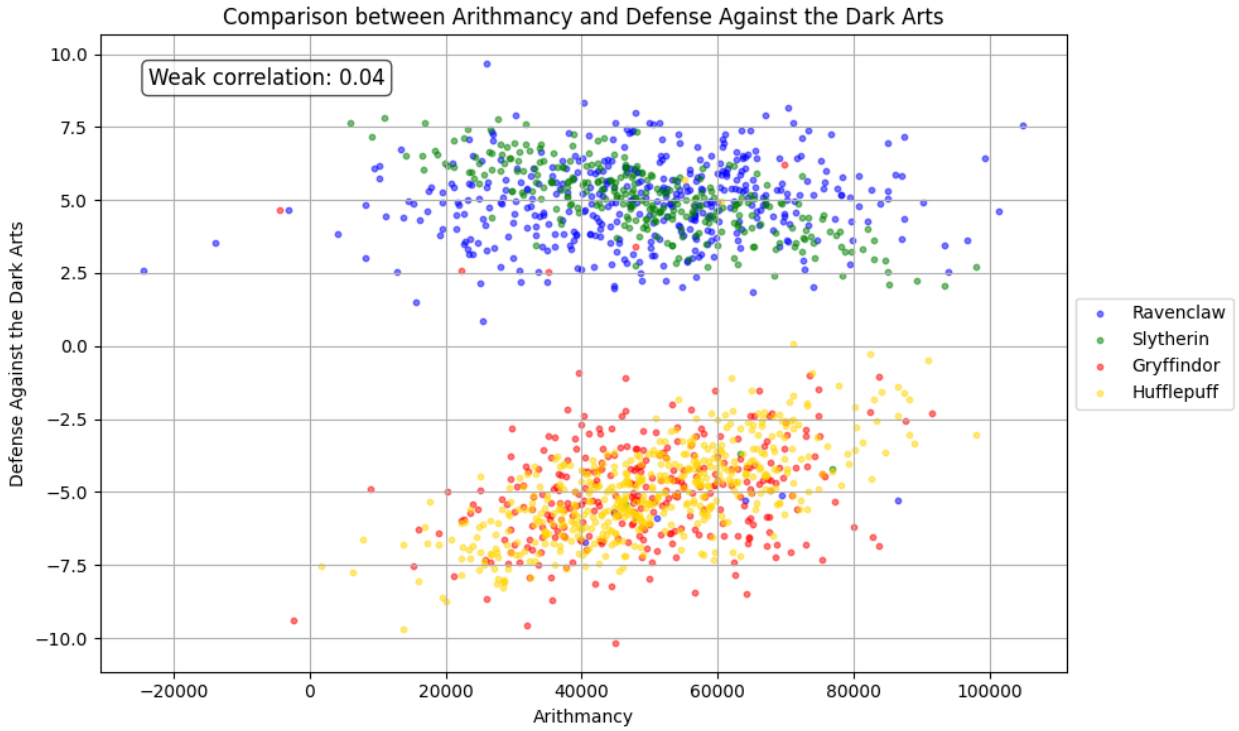
Observations:

- The houses form well-separated groups, especially Ravenclaw (blue, top), Slytherin (green, right), Gryffindor (red, bottom-left), and Hufflepuff (yellow, center).
- The relationship is positive: when the DADA ratings increase, so do the Charms grades (especially for Slytherin).

Interpretation:

- It's a good combination of features to differentiate the houses.
- Slytherin and Ravenclaw clearly stand out, which can be very useful in a supervised model.

Defense Against the Dark Arts vs Charms



Correlation: 0.52 (moderate)

Observations:

The houses form well-separated groups, especially Ravenclaw (blue, top), Slytherin (green, right), Gryffindor (red, bottom-left), and Hufflepuff (yellow, center).

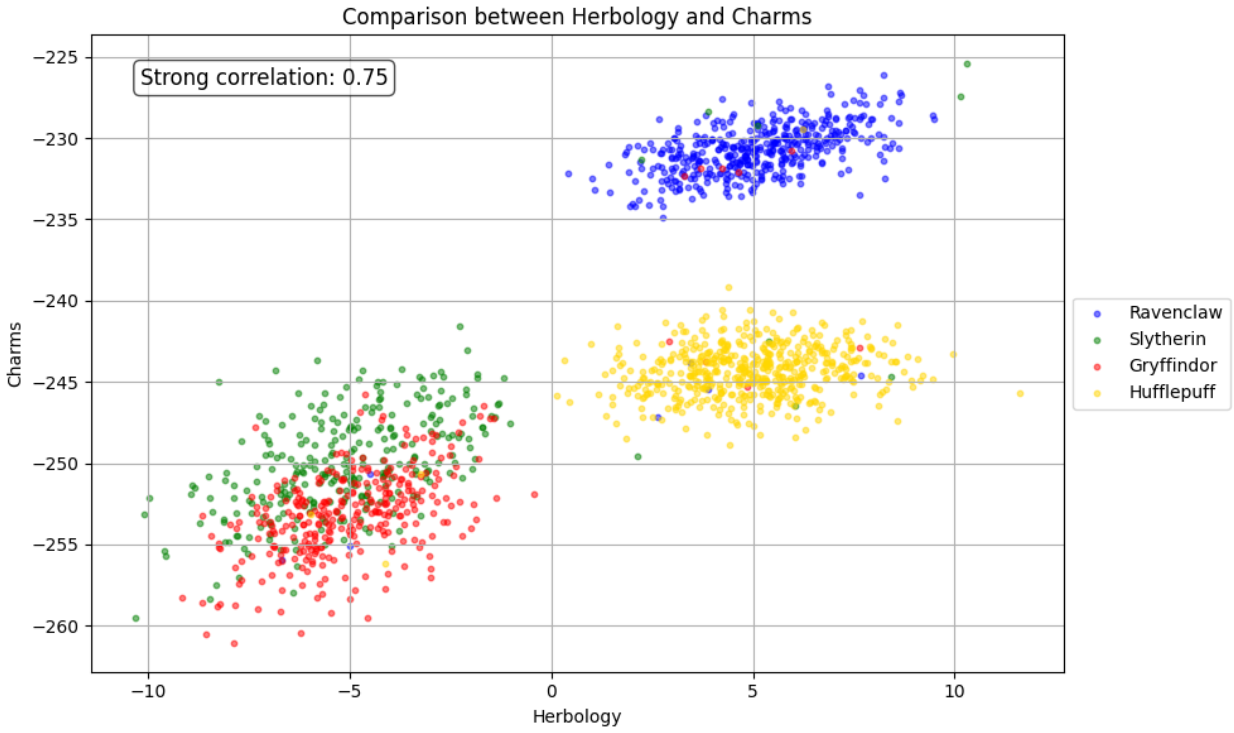
The relationship is positive: when the DADA ratings increase, and so do the Charms grades (especially for Slytherin).

Interpretation:

It's a good combination of features to differentiate the houses.

Slytherin and Ravenclaw clearly stand out, which can be very useful in a supervised model.

Herbology vs Charms



Correlation: 0.75 (strong)

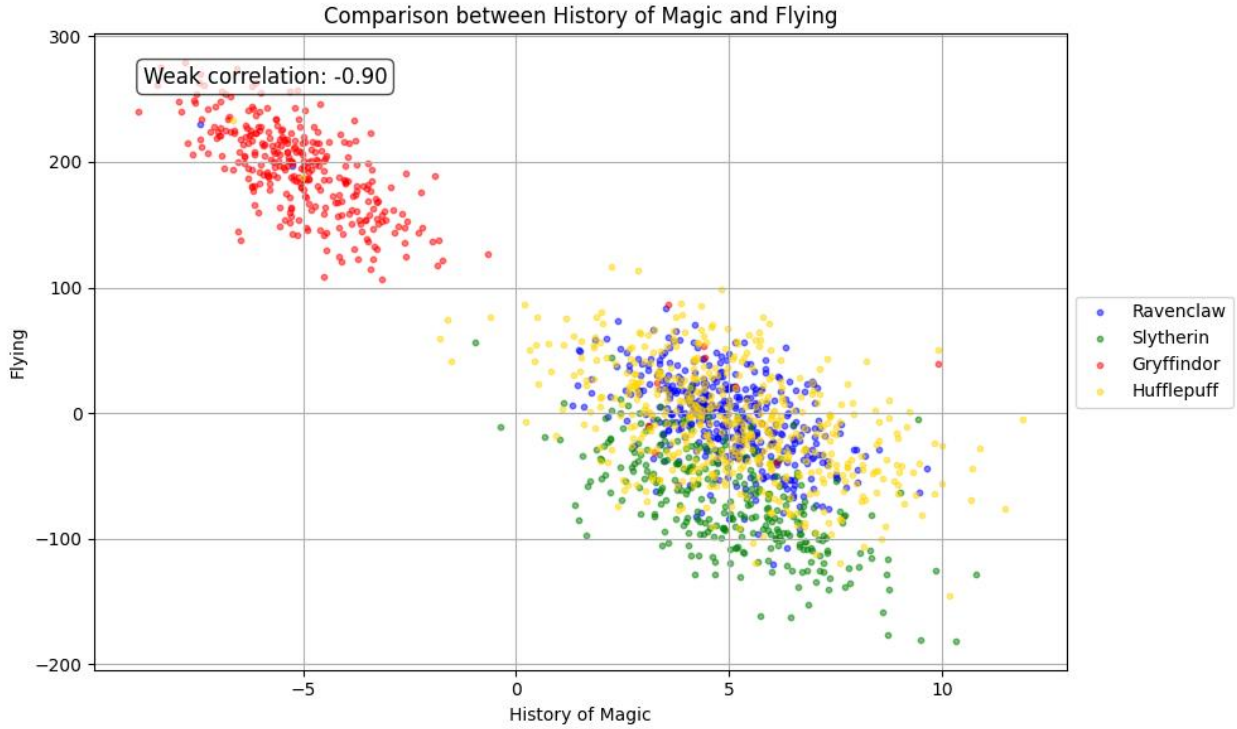
Observations:

- The correlation is visually very clear, with the points forming an ascending diagonal.
- The houses are well separated, especially Ravenclaw (top), Hufflepuff (middle), Gryffindor/Slytherin (bottom).

Interpretation:

- Excellent positive correlation: students who perform well in Herbology also perform well in Charms.
- These two subjects can predict the house very well and are very consistent together.
- To be preferred in the features of the model.

History of Magic vs Flying



Correlation: -0.90 (strong and negative)

Observations:

- Gryffindor (red) is totally isolated at the top left (very high in Flying, very low in History).
- The other houses are more scattered, but less extreme.
- The dots form a nice descending diagonally.

Interpretation:

- A very strong and inverse relationship: those who do well in Flying often fail in History, and vice versa.
- This opposition is very useful for ranking Gryffindor, who seems to be especially strong in Flying.
- Excellent duo of features to use together.

Pair plot == scatterplot matrix

A pair plot (also known as a scatterplot matrix) is a grid of scatter plots showing the relationship between every pair of numerical variables in a dataset.

The pair plot is commonly used in exploration data analysis [3], especially for high-dimensional numerical data.

It allows you to:

- Get a global overview of variable relationships.
- Spot patterns, clusters, and correlated pairs.
- Identify variables that might be redundant.

You can often see diagonal lines, round shapes, or random clouds, depending on the correlation.

The diagonal often shows histograms of each variable, which helps visualize their distribution directly.

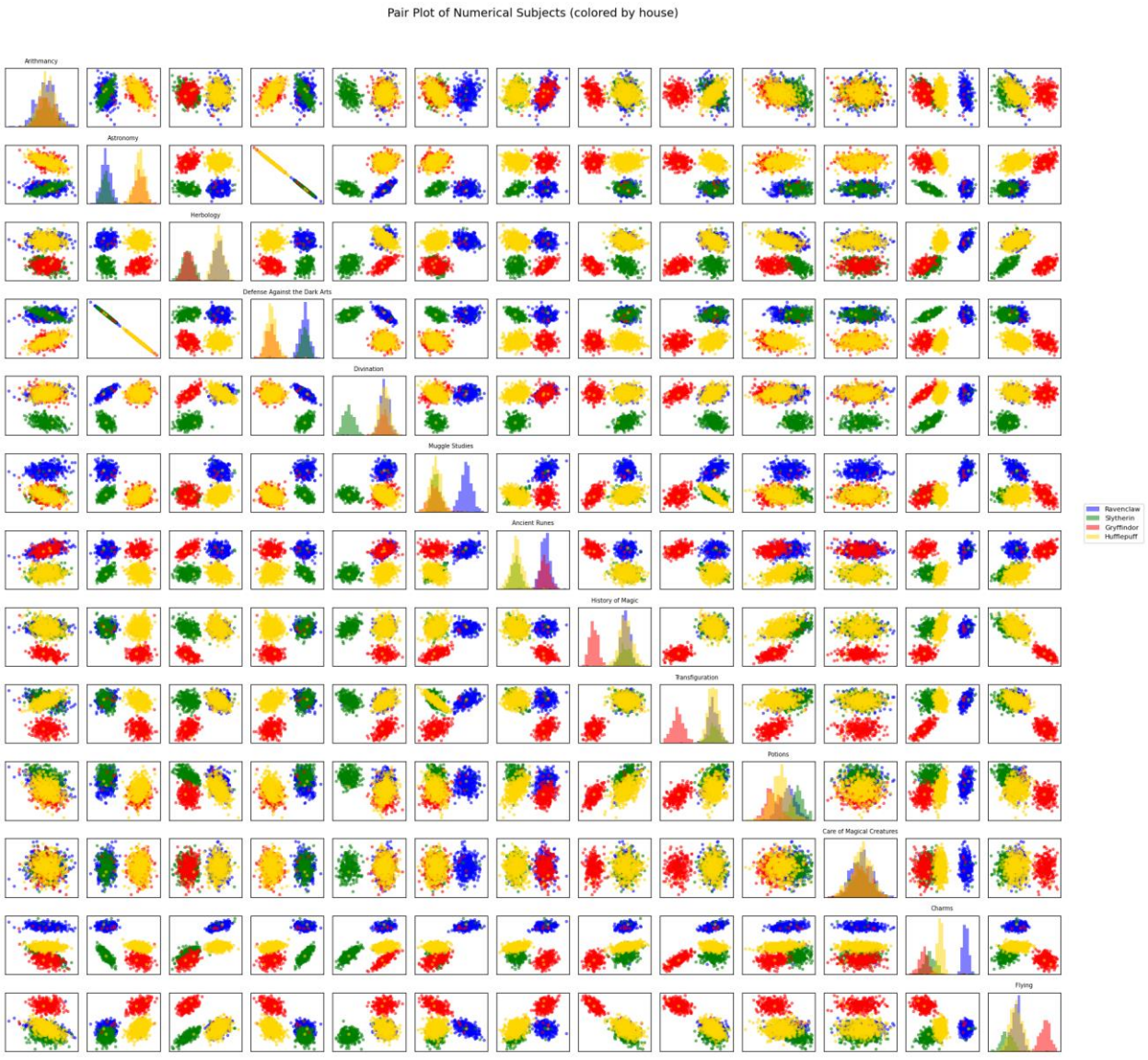
It is a **grid graph** that:

- displays **a scatter plot for each pair of materials** (e.g. Arithmancy vs Astronomy, etc),
- and often, the **diagonal shows the distribution (histogram)** of a single material.

It is widely used in **data science** to quickly detect **correlations**, **patterns**, or **clusters** in data.

Results interpretation

Scatterplot matrix of all vs of subjects



Each square (excluding diagonal) represents a scatter plot between two materials.

For example: the box at the intersection of the "Astronomy" row and the "Herbology" column shows Astronomy in X and Herbology in Y.

The diagonal shows the distribution histograms for each material, colored by house:

- This gives you an idea of the internal distribution by subject, which you have already exploited in your individual histograms.

The colors represent the houses:

- Red: Gryffindor
- Yellow: Hufflepuff
- Blue: Ravenclaw
- Green: Slytherin

Good visual separations:

- Charms vs almost everything else: very clear, the houses group together well.
- Astronomy vs Muggle Studies or vs Defense Against the Dark Arts → very clear diagonal separation.
- Flying vs History of Magic confirms the strong contrast between Gryffindor and the others.

Poorly informative or redundant variables:

- Some subjects have scatters without a clear reason (e.g. Divination vs Arithmancy).
- If two materials have very overlapping dots with no identifiable shape → probably not very correlated or not very useful combined.

Overall conclusion of the pair plot

- The pair plot offers a valuable overview of the relationships between all digital materials, simultaneously visualizing their distributions (diagonal) and cross-correlations (off-diagonal), color-coded by Hogwarts house. It reveals several interesting trends that can guide the selection of features for a classification model.
- Some materials allow for a very clear separation between the houses, including Charms, Astronomy, Flying and History of Magic. The color groups are distinct, suggesting a strong predictive value for the home.
- Clear linear relationships appear between certain pairs of materials, such as Herbology and Charms, or Defense Against the Dark Arts and Charms. This indicates that they are highly correlated and could be redundant in a model (beware of multicollinearity).
- Other subjects, such as Divination or Arithmancy, show little structure or separation per house, which limits their isolated interest in classification, although they may contribute marginally to combination with others.

Conclusion – Exploratory Data Analysis Summary

The combination of statistical analysis (from describe.py) and visual exploration has allowed us to obtain a comprehensive and insightful overview of the dataset. We were able to:

Detect outliers, anomalies, and possible data inconsistencies.

- Understand the distribution and variance of each feature.
- Identify the most discriminative variables for classification.
- Detect strong correlations between pairs of variables.
- Observe the structure and separation of Hogwarts houses through color-coded plots.

This exploratory phase is essential before any modeling [1], [2], [3], as it:

- Guides feature selection.
- Suggests possible transformations (logarithmic, normalization, winsorization)
- Reveals redundant or irrelevant variables.
- Highlights potential risks such as multicollinearity.

In summary, the EDA confirms that some features (like Charms, Astronomy, Flying) are highly informative for predicting a student's house, while others (like Divination or Arithmancy) may be less useful.

These insights will be directly leveraged in the next phase: building a predictive model using logistic regression.

Note: The methods and plots used in this section are based on standard practices in exploratory data analysis, as described in [1], [2], [3]. Pearson's correlation is a foundational concept in mathematical statistics [1], [4], and visual tools such as scatter plots and pair plots are widely used for discovering patterns and guiding feature selection [4].

Chapter 3: Logistic Regression and Machine Learning

Type of regressions models.

Here is an overview of the most used regression models in machine learning and data analysis, with their objectives and typical use cases:

Linear Regression

- **Objective:** Predict a continuous numerical value.
- **Example:** Predicting a student's score based on study hours.
- **Cost Function:** Mean Squared Error (MSE).

Logistic Regression

- **Objective:** Binary classification (0/1).
- **Example:** Predict whether an email is spam or not.
- **Cost Function:** Log-loss (binary cross-entropy).
- **Extension:** Multi-class classification using "One-vs-All" or "Multinomial Logistic Regression."

Polynomial Regression

- **Objective:** Model nonlinear relationships by adding polynomial terms (e.g., x^2, x^3).
- **Example:** Modeling temperature changes over time.
- **Equation:** ($y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n$).

Regularized Regressions (Ridge, Lasso, Elastic Net)

- **Objective:** Prevent overfitting by penalizing large coefficients.
- **Ridge (L2):** Penalizes the squared values of the coefficients.
- **Lasso (L1):** Penalizes the absolute values, encourages feature selection.
- **Elastic Net:** Combines both L1 and L2 penalties.

Multinomial Logistic Regression

- **Objective:** Multi-class classification.
- **Example:** Predicting a student's Hogwarts house (4 classes).
- **Approach:** either train 4 binary classifiers (One-vs-All) or use a direct multinomial formulation (softmax).

Other Models (for reference only)

- **Poisson Regression:** Predicts count-based outcomes.
- **Probit Regression:** Variant of logistic with a Gaussian link.
- **Quantile Regression:** Predicts medians or other quantiles.

- **Log-Log Regression:** Used when variables follow a power-law relationship.

Summary:

- **Continuous output** → Linear regression.
- **Binary output** → Logistic regression.
- **Multi-class output** → Multinomial logistic.
- **Nonlinear data** → Polynomial regression.
- **Regularization needed** → Ridge, Lasso, or ElasticNet.

Key Concepts to Master in Machine Learning

Building and understanding machine learning models requires mastering the following conceptual blocks:

Regression & Prediction

- Linear regression
- Logistic regression
- Ridge, Lasso, ElasticNet
- Polynomial regression
- Multinomial regression
- Loss functions: MSE vs Log-loss
- Concepts: overfitting, underfitting

Theory & Optimization

- Bias-variance tradeoff
- Gradient descent & adaptive methods (Adam, RMSProp)
- Information theory (entropy, information gain)
- Bayesian inference

Descriptive Statistics

- Mean, median, mode
- Variance, standard deviation
- Quartiles, percentiles
- Skewness, kurtosis
- Visualization: histograms, box plots

Probability & Inference

- Probability distributions: normal, binomial, Bernoulli
- PDF / CDF, expected value

- Statistical inference: hypothesis tests, p-value, t-test, confidence intervals

Correlation & Relationships

- Pearson and Spearman correlation
- Covariance
- Multicollinearity detection

Classification & Evaluation

- Confusion matrix, accuracy, recall, precision, F1-score
- ROC curve, AUC
- Log-loss, cross-entropy

Model Validation & Generalization

- Train/test/validation split
- Cross-validation (K-fold)
- Bootstrap sampling

Types of Learning

- Supervised learning: classification, regression
- Unsupervised learning: clustering
- Algorithms to explore: k-NN, SVM, decision trees

These blocks form the foundation for building, training, and evaluating machine learning models reliably and scientifically.

Binary logistic regression

After describing and visualizing the dataset, the next logical step is to **build a predictive model** capable of classifying students into the correct Hogwarts house based on their course results.

In this part, we apply a **supervised learning algorithm**, more specifically a **logistic regression classifier.** Logistic regression is one of the simplest and most interpretable models in machine learning [1], [2]. It is widely used for **binary classification** and can be extended to **multi-class classification** using strategies like ***One-vs-All*** (OvA).

This section will:

- Explain the **objective of logistic regression.**
- Clarify the **differences with linear regression.**
- Introduce concepts like **cost function (log-loss), gradient descent,** and **regularization.**
- Discuss **One-vs-All strategy** for multi-class classification.
- Prepare you to evaluate and improve your model using **loss curves, metrics,** and **cross-validation.**

To understand and implement logistic regression efficiently, you must combine concepts from **statistics, optimization, and data analysis.** This part marks the transition from **descriptive to predictive** modeling.

- **One-vs-all** (recommended): Train one classifier per house (binary).
- **One-hot encoding**: each house becomes a vector (not necessarily here).

Steps:

1. **Logistic regression: Calculation of probability with sigmoid**
2. Cost (loss): Quantify the error
3. Gradient: Slope to minimize error

You want to predict if a student belongs to a **target house (e.g. Gryffindor)** based on their scores (numerical features).

So, we want a model that says:

"Looking at this student's grades, how **likely is** it that he is Gryffindor?"

Logistic Regression Calculation (Prediction)

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{\{-z\}}} \; with \; z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- $\hat{y}$: probability that the student belongs to a given house (e.g. Gryffindor).
- x_i: value of the i-th feature (a note, e.g. Arithmancy, Potions...).
- θ_i : weight learned by the model for the feature x_i.
- θ_0 : model bias (intercept).
- $\sigma(z)$: sigmoid function, brings the output back to 0 to 1.
- z : weighted sum of inputs, this is the argument of the sigmoid.

We need to convert the linear combination of features into a probability between 0 and 1 [2]. That's what the sigmoid does.

Cost function (log-loss)

It is called **a cost function** because its "costs" more when the model gets it wrong.

We want our model to give a **probability close to 1** when y=1 and **close to 0** when y=0.

So, we measure the error with **log-loss**:

We want to quantify **how bad our predictions are**. For binary logistic regression:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right) \right]$$

- $J(\theta)$: total cost (average error on all students)
- m : total number of students
- $y^{(i)}$: **true value** (0 or 1) for student *i*
- $\hat{y}^{(i)}$: **Prediction of the model** for student *i*

The measurement of error depends on the type of problem (classification vs. regression) [1], [4].

Other cost functions calculations

MSE (Mean Squared Error) – used in linear regression

- Measures the **mean squared deviation** between the true value and the prediction.
- **Not adapted** to classification because it does not penalize certain errors enough.

$$J(\theta) = \frac{1}{m} * \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- $J(\theta)$: total cost (mean squared error between prediction and truth).
- m: Total number of examples (students).
- $\hat{y}^{(i)}$: **model prediction** for student *i.*
- $y^{(i)}$: **real value** for students *i.*

MAE (Mean Absolute Error)

- Also used in regression.
- Less sensitive to **large errors** than MSE, but not differentiable in 0 (harder to optimize).

$$J(\theta) = \frac{1}{m} * \sum_{i=1}^{m} \left| \hat{y}^{(i)} - y^{(i)} \right|$$

- $J(\theta)$: total cost (average absolute error between prediction and reality).
- m: Total number of students.
- $\hat{y}^{(i)}$: **model prediction** for student *i.*
- $y^{(i)}$: **real value** for students *i.*
- $| a |$: absolute value of a.

R^2 (coefficient of determination)

- A measure of the **quality of a linear regression** (0 to 1, or even negative).
- Not for use in classification.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

1. R^2: coefficient determination, measures the proportion of the variance explained by the model.
2. SS_{res}: sum of the squares of the residuals (errors between the actual and predicted values).
3. SS_{tot}: sum of total squares (differences between actual values and their mean).

Log-loss / Binary Cross-Entropy – used in logistic regression

- Penalizes **bad predictions** very heavily.
- Perfect for **probability results**.
- Value close to 0 = good model. The higher it is = bad predictions.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[\, y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right]$$

- $J(\theta)$: total cost (average error on all students)
- m : total number of students
- $y^{(i)}$: **true value** (0 or 1) for student *i*
- $\hat{y}^{(i)}$: **Prediction of the model** for student *i*

> **Note:** Log-loss strongly penalizes confident but wrong predictions, making it ideal for probabilistic classification.

Gradient descent

Gradient descent is a core optimization algorithm used in machine learning [1], and its stochastic and mini-batch variants are commonly applied for large datasets [3].

It is called **a gradient** because it is a **multidimensional slope**. This is what allows us to **determine which direction to adjust θ to** θ **reduce the error**.

The gradient is the **vector of partial derivatives** of the cost function with respect to each parameter θ_j.

To **minimize the loss J(θ)**, we adjust the weights (theta) with the gradient descent:

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}$$

- θ_j : model weight for feature *j.*
- α : **Learning rate**, controls update speed.
- $\frac{\partial J(\theta)}{\partial \theta_j}$: partial derivative of cost with respect to θ_j.
- $:=$: means "becomes" → we update θ_j with each iteration.

This derivative is (for each feature j):

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \times \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}$$

- $\frac{\partial J}{\partial \theta_j}$: partial derivative of the cost function with respect to θ_j.
- m: Total number of examples (students).
- $\hat{y}^{(i)}$: model prediction for student i.
- $y^{(i)}$: true value for student i.
- $x_j^{(i)}$: value of feature j for student i.

In other words: if the prediction is too high, we decrease θ_j, and if it is too low, we increase it.

$$\theta_j := \theta_j - \alpha \cdot \left(\frac{1}{m}\right) * \sum_{i=i}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

- θ_j: model weight for feature *j.*
- α : learning rate.
- m: Total number of students.
- $\hat{y}^{(i)}$: model prediction for student *i.*
- $y^{(i)}$: true value for students *i.*
- $x_j^{(i)}$: value of feature *j* for student *i.*
- $\sum$: sum over all students (from *i = 1* to *m*).

Normalizing methods

Min-max

- **Usefulness**: Brings values within a range [0, 1]
- **Advantage**: preserves the shape of the distribution

- **Borderline**: Outliers sensitive

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$

- x: Original value to normalize.
- x_{norm}: Normalized value of x between 0 and 1.
- $min(x)$: The smallest value of x in the dataset.
- $max(x)$: The largest value of x in the dataset.

These normalization techniques are essential before applying linear models, particularly logistic regression [3], [4].

To use if the data:

- If we want to save the proportions or ratios.
- Not to use if there are outliers.

Standardization (z-score) (reduced centered values)

- **Advantage**: makes all features comparable with mean = 0 and standard deviation = 1.
- **Widely used in logistic regression.**

$$x_{std} = \frac{x - \mu}{\sigma}$$

- x: Original value to standardize.
- x_{std}: **Standardized value** (z-score).
- μ: **Average** of x values.
- σ: **Standard deviation** of the values of x.

To use if the data:

- No outliers.
- Normal distribution
- Used by default in regression, SVM or PCA.

Other methods

- Log Transformation. $x' = \log(x + 1)$
  - To use of skewness positive.
  - It reduces the impact of huge values.
  - To use for log (x +1) but not for log (0)
- Robust Scaler. $x' = \frac{(x - median)}{IQR}$
  - Helpful for so extreme values.
  - It uses median and IQR, so strong for skewness forms

Bonus

Batch Gradient Descent

$$J(\theta) = \left(\frac{1}{m}\right)\sum log - loss$$

$$\theta_j \leftarrow \theta_j - \alpha \cdot \left(\frac{1}{m}\right) \times \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j{}^{(i)}$$

- You use all the examples at once to make a single update of the θ.
- That's what you do in your job ft_calculate_log_loss()
- It uses **all students in each iteration** to:
- Calculate the sigmoid(z) for each student (ft_forward_pass_and_sigmoid).
- Accumulate errors (gradients).
- Update weights **only once per iteration**.
- Il appelle donc : ft_forward_pass_and_sigmoid() → ft_calculate_log_loss()

for i in range(m): m = Total number of samples

Gradient calculation on all xi

Stochastic Gradient Descent (SGD)

- You update the θ **with each example** !
- It speeds up training, with a more "chaotic" behavior.

For each i from 0 to m:

$$\theta_j \leftarrow \theta_j - \alpha \cdot \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j{}^{(i)}$$

- θ_j: model weight for feature *j.*
- α: **learning rate.**
- $h_\theta\left(x^{(i)}\right)$: model prediction for student *i.*
- $y^{(i)}$: true value for students *i.*
- $x_j{}^{(i)}$: value of feature *j* for student *i.*
- ←: Updates (replaces the value of θj).

Operation:

- It makes **several epochs** (passes) on the whole data,
- Each **student**:
- Calculates its prediction sigmoid(zi),
- Calculates its error h(xi) - yi,
- Updates θ **immediately**.
- This is the definition of **the OMS**.

$$\theta_j \leftarrow \theta_j - \alpha * (h(x_i) - y_i) * x_j \ (Immediate\ update)$$

Mini-Batch Gradient Descent

$$\theta_j := \theta_j - \alpha \cdot \left(\frac{1}{B}\right) \times \sum_{i=1}^{B} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$

- θ_j : weight associated with feature *j*.
- α: learning rate.
- B: **Mini-batch size** (number of examples in the subsample).
- $h_\theta\left(x^{(i)}\right)$: model prediction for example *i*.
- $y^{(i)}$: true value for example *i*.
- $x_j^{(i)}$: value of feature *j* for example *i* in the batch.
- $\sum$: sum on the B examples of the mini-batch.
- $:=$: update (θ_j becomes…)

Comparability of gradients

Algorithm	Iteration basis	Key variables for convergence	Idea of equivalence
BGD	1 update / **epoch**	iterations	1 epoch = 1 update
SGD	1 update / **sample**	epochs, alpha	epochs * m updates
Mini-batch GD	1 update / **batch**	epochs, batch_size, alpha	epochs * m/batch_size updates

Epoch definition

An epoch is **a complete pass** over **the entire dataset**.

- Epoch: 1 **complete** pass on **the entire dataset**
- Iteration: 1 parameter update (θ)

Let's imagine that you have **1600 students** (so 1600 examples):

- **1 epoch =** you look at the **1600 examples once** (integers).
- If you make **20 epochs**, you go over **all the examples 20 times**.

BGD

- 1 epoch = 1 iteration
- Because we are waiting to have all the data to make 1 update.
- iterations = 1000
- Only 1 update with **the 1600**

SGD

- 1 epoch = m iterations (where m is the number of samples).
- m = 1600
- Because we do 1 update per sample.
- epochs = 20 # 20 * 1600 = 32000 updates
- 1600 updates (1 update / example)

Mini-Batch

- 1 epoch = m / batch_size iterations
- m = 1600, batch_size = 32 → 1 epoch = 50 iterations
- batch_size = 32
- epochs = 20 # 20 * (1600 / 32) = 1000 updates
- 1600 / 32 = **50 updates** (1 by mini lot).

Conclusion - Logistic Regression Results and Comparison

The evaluation results confirm what is described in practical machine learning literature: SGD is fast and efficient, while MBGD offers a solid compromise [3], [4].

We implemented and trained a logistic regression classifier using three different optimization methods:

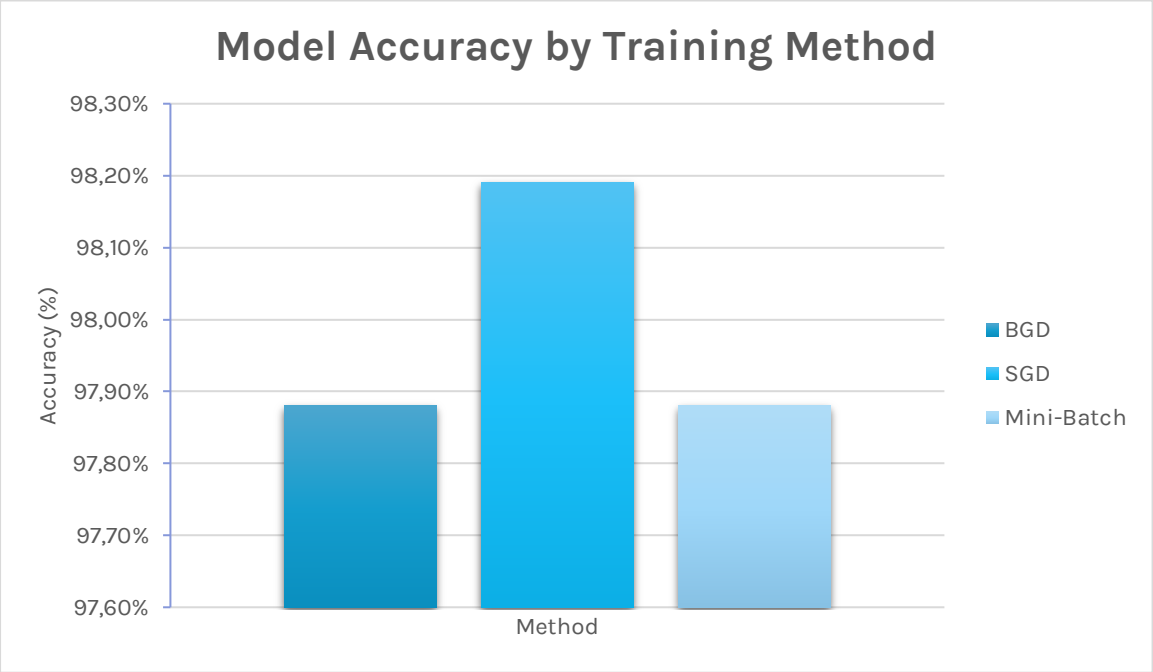
- **Batch Gradient Descent (BGD)**
- **Stochastic Gradient Descent (SGD)**
- **Mini-Batch Gradient Descent (MBGD)**

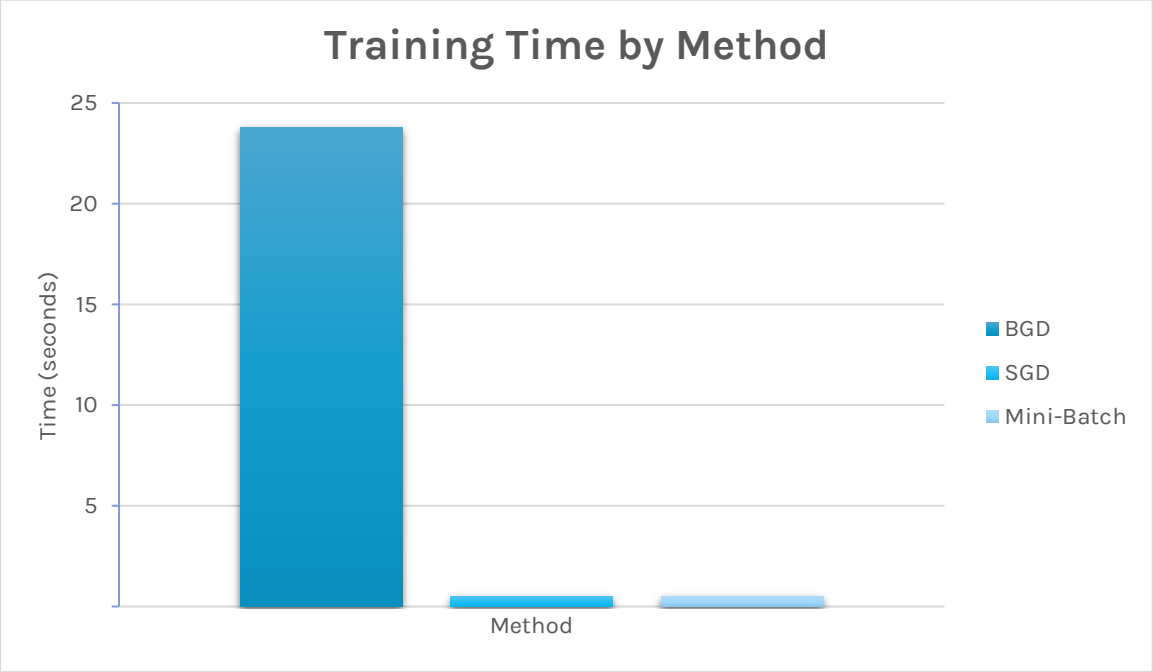
Each model was trained with a **learning rate of 0.01,** and the same features selected after the data visualization phase.

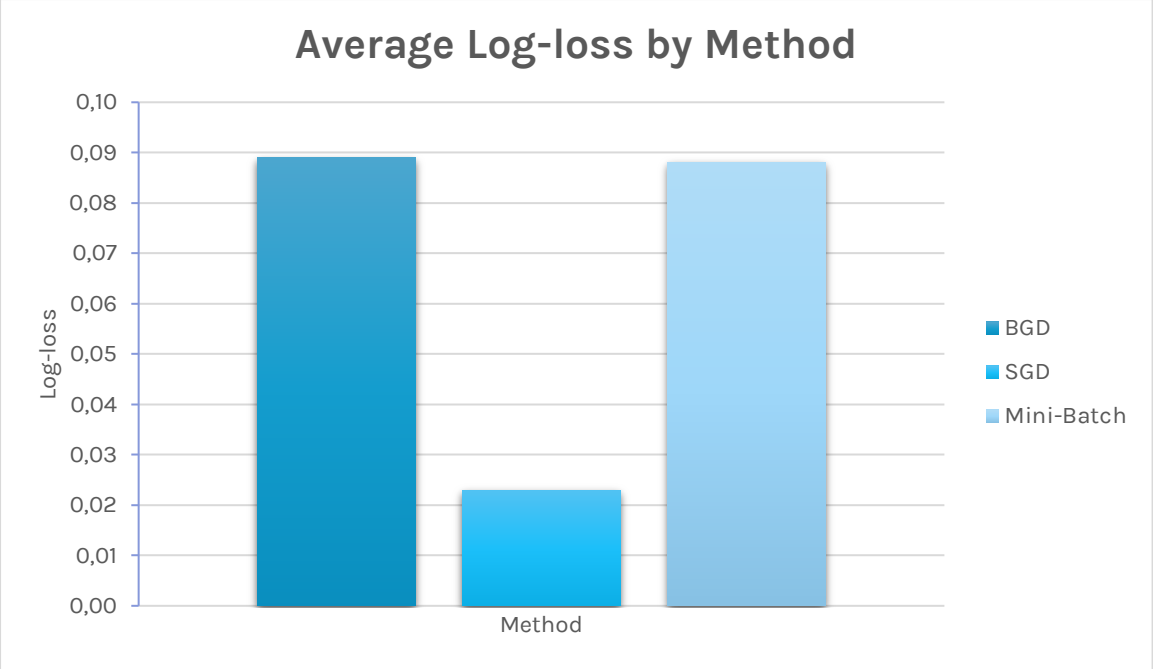
Training Summary

Method	Iterations / Epochs	Batch Size	Log-loss (avg)	Accuracy	Training Time
BGD	1000 iterations	Full batch	~0.11381	97.88%	**23.78 sec**
SGD	20 epochs	1 sample	~0.0557	98.19%	**0.50 sec**
MBGD	20 epochs	32 sample	~0.11535	97.88%	**0.49 sec**

Observations & Insights







- **SGD achieves the best accuracy** (98.19%) and **lowest log-loss,** and trains in under 1 second.
    - It introduces randomness, which can help escape local minima and converge faster.
- **BGD** is the slowest by far (nearly 24 seconds), though stable and precise.
    - It's more deterministic but not scalable for large datasets.
- **Mini-Batch GD** finds a good trade-off between stability and speed.
    - Slightly better than BGD in speed, but same accuracy in this case.

Conclusion

All three methods led to **high-performing classifiers** thanks to:

- Proper feature selection (from the EDA)
- Data normalization
- Tuned learning rate (0.01)

However, **SGD** is clearly the most efficient in this context — delivering **the best trade-off between speed, convergence, and accuracy**.

In larger-scale real-world projects, **mini-batch** is often preferred for its balance of stability and scalability.

This concludes the training phase. The next step is to **evaluate the model on unseen data,** and explore further enhancements such as:

- Regularization (L2)
- Cross-validation
- Model interpretability

Bibliography

The following sources were consulted to guide the statistical methods, data visualization practices, and machine learning implementation presented in this project.

[1] J. L. Devore, Probability and Statistics for Engineering and the Sciences, Boston: Cengage Learning, 2011.

[2] D. Freedman, R. Pisani and R. Purves, Statistics, New York: W. W. Norton & Company, 2007.

[3] W. McKinney, Python for Data Analysis, Sebastopol: O'Reilly Media, 2017.

[4] NIST/SEMATECH, "e-Handbook of Statistical Methods," National Institute of Standards and Technology, 1 October 2012. [Online]. Available: https://www.itl.nist.gov/div898/handbook/. [Accessed 7 April 2025].

[5] W. contributors, "Kurtosis," Wikipedia, 10 March 2024. [Online]. Available: https://en.wikipedia.org/wiki/Kurtosis. [Accessed 7 April 2025].

[6] W. contributors, "Skewness," Wikipedia, 10 March 2024. [Online]. Available: https://en.wikipedia.org/wiki/Skewness. [Accessed 7 April 2025].