# Statistics Cheatsheet For Data Science

December 2, 2025

## Contents

# 1 Descriptive Statistics

## 1.1 1. Mean (Average)

### 1.1.1 Formula

$\bar{x} = \frac{\sum x_i}{n}$

The mean is the average value of a dataset, calculated by summing all values and dividing by the count.

### 1.1.2 Example

For the dataset [10, 20, 30, 40], the mean is (10+20+30+40)/4 = 25.

### 1.1.3 Practical Application

In business, the mean is used to calculate average sales revenue over a period to assess performance. For instance, if daily sales are [100, 150, 200, 120], the mean helps define typical daily revenue.

**Python Code**

```
import numpy as np
data = np.array([10, 20, 30, 40])
mean = np.mean(data)
print("Mean:", mean)
```

Output

```
Mean: 25.0
```

## 1.2 2. Weighted Mean (Weighted Average)

### 1.2.1 Formula

$\bar{x}_w = \frac{\sum w_i x_i}{\sum w_i}$

The weighted mean accounts for varying importance of data points using weights.

### 1.2.2 Example

Grades with weights: scores [80, 90, 70] with weights [0.2, 0.3, 0.5]. Weighted mean = (80*0.2 + 90*0.3 + 70*0.5) / (0.2+0.3+0.5) = 78.

### 1.2.3 Practical Application

In finance, weighted averages are used in portfolio returns where different investments have varying allocations, helping define overall expected performance.

**Python Code**

```
import numpy as np
values = np.array([80, 90, 70])
weights = np.array([0.2, 0.3, 0.5])
weighted_mean = np.average(values, weights=weights)
print("Weighted Mean:", weighted_mean)
```

Output

Weighted Mean: 78.0

## 1.3 3. Median

### 1.3.1 Formula

Middle of ordered data

The median is the middle value in a sorted dataset, robust to outliers.

### 1.3.2 Example

For [10, 20, 30, 40, 500], sorted is the same, median is 30.

### 1.3.3 Practical Application

In real estate, median home prices define market trends without skew from luxury outliers.

**Python Code**

```
import numpy as np
data = np.array([10, 20, 30, 40, 500])
median = np.median(data)
print("Median:", median)
```

Output

Median: 30.0

## 1.4 4. Mode

### 1.4.1 Formula

Most frequent value

The mode is the most common value in the dataset.

### 1.4.2 Example

In [1, 2, 2, 3, 4], mode is 2.

### 1.4.3 Practical Application

In customer surveys, mode identifies the most frequent rating to define common satisfaction levels.

**Python Code**

```
import pandas as pd
from scipy import stats
data = pd.Series([1, 2, 2, 3, 4])
mode = stats.mode(data)
print("Mode:", mode.mode)
```

Output

Mode: 2

## 1.5 5. Range (Spread)

### 1.5.1 Formula

$R = \max(x) - \min(x)$

The range measures the spread between the highest and lowest values.

### 1.5.2 Example

For [10, 20, 30, 40], range = 40 - 10 = 30.

### 1.5.3 Practical Application

In quality control, range defines variability in product dimensions.

**Python Code**

```
import numpy as np
data = np.array([10, 20, 30, 40])
range_val = np.max(data) - np.min(data)
print("Range:", range_val)
```

Output

```
Range: 30
```

## 1.6 6. Variance (Population)

### 1.6.1 Formula

$\sigma^2 = \frac{\sum(x_i-\mu)^2}{N}$

Population variance measures average squared deviation from the mean.

### 1.6.2 Example

For population [1, 2, 3], mean=2, variance = $((1\text{-}2)^2 + (2-2)^2 + (3-2)^2)/3$0.667.

### 1.6.3 Practical Application

In population studies, it defines data dispersion, like income variability across a country.

**Python Code**

```
import numpy as np
data = np.array([1, 2, 3])
variance_pop = np.var(data, ddof=0)
print("Population␣Variance:", variance_pop)
```

Output

```
Population Variance: 0.6666666666666666
```

## 1.7 7. Variance (Sample)

### 1.7.1 Formula

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

Sample variance uses n-1 for unbiased estimation.

### 1.7.2 Example

For sample [1, 2, 3], variance = $((1\text{-}2)^2 + (2-2)^2 + (3-2)^2)/2 = 1$.

### 1.7.3 Practical Application

In surveys, sample variance defines estimate reliability for larger populations.

**Python Code**

```
import numpy as np
data = np.array([1, 2, 3])
variance_sample = np.var(data, ddof=1)
print("Sample Variance:", variance_sample)
```

Output

Sample Variance: 1.0

## 1.8 8. Standard Deviation

### 1.8.1 Formula

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

Standard deviation is the square root of variance, in original units.

### 1.8.2 Example

For variance 4, std dev = 2.

### 1.8.3 Practical Application

In finance, it defines investment risk volatility.

**Python Code**

```
import numpy as np
data = np.array([1, 2, 3])
std_dev = np.std(data, ddof=0)
print("Standard Deviation:", std_dev)
```

Output

Standard Deviation: 0.816496580927726

## 1.9 9. Coefficient of Variation

### 1.9.1 Formula

$CV = \frac{\sigma}{\mu} \times 100$
CV expresses std dev as a percentage of the mean for comparison.

### 1.9.2 Example

Mean 50, std dev 5, CV = (5/50)*100 = 10%.

### 1.9.3 Practical Application

Comparing variability in different datasets, like stock returns.

**Python Code**

```
import numpy as np
data = np.array([45, 50, 55])
cv = (np.std(data, ddof=0) / np.mean(data)) * 100
print("CV:", cv)
```

Output

```
CV: 8.16496580927726
```

## 1.10 10. Quartiles

### 1.10.1 Formula

$Q_1, Q_2, Q_3$ (25%, 50%, 75%)
Quartiles divide data into four equal parts.

### 1.10.2 Example

For [1,2,3,4,5,6,7], Q1=2.5, Q2=4, Q3=5.5.

### 1.10.3 Practical Application

In income analysis, quartiles define distribution brackets.

**Python Code**

```
import numpy as np
data = np.array([1,2,3,4,5,6,7])
quartiles = np.percentile(data, [25, 50, 75])
print("Quartiles:", quartiles)
```

Output

```
Quartiles: [2.5 4. 5.5]
```

# 2 Probability & Distributions

## 2.1 11. Probability

### 2.1.1 Formula

$P(A) = \frac{\text{favorable outcomes}}{\text{total outcomes}}$
Probability measures likelihood.

### 2.1.2 Example

Coin flip heads: $1/2 = 0.5$.

### 2.1.3 Practical Application

In risk assessment, probability defines event occurrence, like machine failure.

**Python Code**

```
import numpy as np
np.random.seed(42)
simulations = np.random.choice(['Heads', 'Tails'], size=1000)
prob_heads = np.sum(simulations == 'Heads') / 1000
print("Simulated␣P(Heads):", prob_heads)
```

Output

```
Simulated P(Heads): 0.49
```

## 2.2 12. Complement Rule

### 2.2.1 Formula

$P(A^c) = 1 - P(A)$
Complement is 1 minus event probability.

### 2.2.2 Example

P(not heads) = 1 - 0.5 = 0.5.

### 2.2.3 Practical Application

In quality control, P(defect-free) = 1 - P(defective).

**Python Code**

```
p_a = 0.5
p_complement = 1 - p_a
print("Complement:", p_complement)
```

Output

```
Complement: 0.5
```

## 2.3 13. Addition Rule

### 2.3.1 Formula

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$
For union of events.

### 2.3.2 Example

P(A or B) where P(A)=0.4, P(B)=0.3, Pboth=0.1: 0.4+0.3-0.1=0.6.

### 2.3.3 Practical Application

In marketing, probability of customer buying product A or B.

**Python Code**

```
p_a = 0.4
p_b = 0.3
p_inter = 0.1
p_union = p_a + p_b - p_inter
print("Union:", p_union)
```

Output

```
Union: 0.6
```

## 2.4 14. Multiplication Rule

### 2.4.1 Formula

$P(A \cap B) = P(A)P(B|A)$
For intersection.

### 2.4.2 Example

P(draw two aces) = (4/52) * (3/51)  0.0045.

### 2.4.3 Practical Application

In reliability, probability of multiple system failures.

**Python Code**

```
p_a = 4/52
p_b_given_a = 3/51
p_inter = p_a * p_b_given_a
print("Intersection:", p_inter)
```

Output

```
Intersection: 0.004524886877828055
```

## 2.5 15. Conditional Probability

### 2.5.1 Formula

$P(A|B) = \frac{P(A \cap B)}{P(B)}$
Probability of A given B.

### 2.5.2 Example

Given disease P=0.01, test positive given disease=0.9, etc. (Use Bayes for full).

### 2.5.3 Practical Application

In medicine, P(disease—positive test).

**Python Code**

```
p_inter = 0.1
p_b = 0.3
p_a_given_b = p_inter / p_b
print("Conditional:", p_a_given_b)
```

Output

```
Conditional: 0.33333333333333337
```

## 2.6 16. Bayes' Theorem

### 2.6.1 Formula

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
Updates probabilities with evidence.

### 2.6.2 Example

P(disease—positive) = [P(positive—disease) * P(disease)] / P(positive).

### 2.6.3 Practical Application

Spam filtering: P(spam—word).

**Python Code**

```
p_b_given_a = 0.9
p_a = 0.01
p_b = 0.05 # Assume
p_a_given_b = (p_b_given_a * p_a) / p_b
print("Bayes:", p_a_given_b)
```

Output

```
Bayes: 0.18000000000000002
```

## 2.7 17. Expected Value

### 2.7.1 Formula

$E[X] = \sum x_i P(x_i)$
Average value over many trials.

### 2.7.2 Example

Lottery: prizes [100,0] probs [0.01,0.99], E = 100*0.01 + 0*0.99 = 1.

### 2.7.3 Practical Application

In insurance, expected claim payout.

**Python Code**

```
import numpy as np
values = np.array([100, 0])
probs = np.array([0.01, 0.99])
expected = np.sum(values * probs)
print("Expected␣Value:", expected)
```

Output

Expected Value: 1.0

## 2.8 18. Binomial Distribution

### 2.8.1 Formula

$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$
For fixed trials, success probability p.

### 2.8.2 Example

10 coin flips, P(exactly 5 heads) = binom(10,5)*$(0.5)^5 * (0.5)^5$0.246.

### 2.8.3 Practical Application

In manufacturing, number of defective items in a batch.

**Python Code**

```
from scipy.stats import binom
n, p = 10, 0.5
k = 5
prob = binom.pmf(k, n, p)
print("Binomial␣P:", prob)
```

Output

Binomial P: 0.24609375000000003

## 2.9 19. Poisson Distribution

### 2.9.1 Formula

$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$
For rare events.

### 2.9.2 Example

=2 calls/hour, P(exactly 3) = $(2^3 * e^-2)/3!$0.18.

### 2.9.3 Practical Application

Modeling customer arrivals in a store.

**Python Code**

```
from scipy.stats import poisson
lambda_ = 2
k = 3
prob = poisson.pmf(k, lambda_)
print("Poisson␣P:", prob)
```

Output

```
Poisson P: 0.1804470443154836
```

## 2.10 20. Normal PDF

### 2.10.1 Formula

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Bell curve distribution.

### 2.10.2 Example

Mean 0, std 1, f(0) = 1/sqrt(2pi)  0.399.

### 2.10.3 Practical Application

Heights in a population.

**Python Code**

```
from scipy.stats import norm
mu, sigma = 0, 1
x = 0
pdf = norm.pdf(x, mu, sigma)
print("Normal␣PDF:", pdf)
```

Output

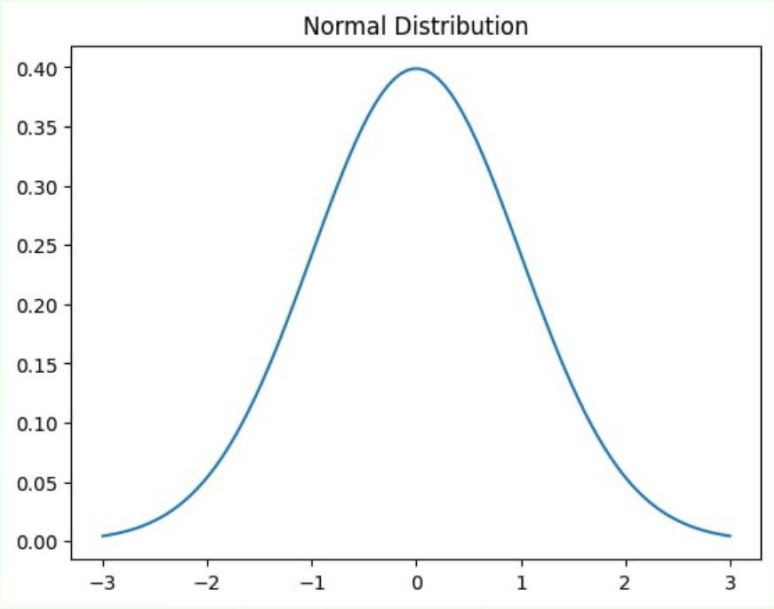
```
Normal PDF: 0.3989422804014327
```

### 2.10.4 For Visualization

**Python Code**

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
x = np.linspace(-3, 3, 100)
pdf = norm.pdf(x, 0, 1)
sns.lineplot(x=x, y=pdf)
plt.title("Normal Distribution")
plt.show()
```

Output




# 3 Inference Statistics

## 3.1 21. Z-score

### 3.1.1 Formula

$Z = \frac{x-\mu}{\sigma}$

Standardizes values.

### 3.1.2 Example

x=85, mu=70, sigma=10, Z=1.5.

### 3.1.3 Practical Application

Comparing test scores across different exams.

**Python Code**

```
x, mu, sigma = 85, 70, 10
z = (x - mu) / sigma
print("Z-score:", z)
```

Output

Z-score: 1.5

## 3.2 22. t-score

### 3.2.1 Formula

$t = \frac{\bar{x}-\mu}{s/\sqrt{n}}$
For small samples.

### 3.2.2 Example

Sample mean 75, mu=70, s=5, n=25, t=(75-70)/(5/sqrt(25))=5.

### 3.2.3 Practical Application

Hypothesis testing for means with unknown variance.

**Python Code**

```
import numpy as np
x_bar, mu, s, n = 75, 70, 5, 25
t = (x_bar - mu) / (s / np.sqrt(n))
print("t-score:", t)
```

Output

t-score: 5.0

## 3.3 23. Confidence Interval (Mean, known )

### 3.3.1 Formula

$\bar{x} \pm Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$
Interval estimate.

### 3.3.2 Example

Mean 100, sigma=15, n=36, Z=1.96, CI=100 ± 1.96*(15/6) [95.1, 104.9].

### 3.3.3 Practical Application

Polling margins in elections.

**Python Code**

```
from scipy.stats import norm
import numpy as np
x_bar, sigma, n, alpha = 100, 15, 36, 0.05
z = norm.ppf(1 - alpha/2)
margin = z * (sigma / np.sqrt(n))
ci = (x_bar - margin, x_bar + margin)
print("CI:", ci)
```

Output

CI: (95.10009003864995, 104.89990996135005)

## 3.4 24. Confidence Interval (Mean, unknown )

### 3.4.1 Formula

$\bar{x} \pm t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$

Uses t-distribution.

### 3.4.2 Example

Similar to above, but t instead of Z.

### 3.4.3 Practical Application

Small sample surveys.

**Python Code**

```
from scipy.stats import t
import numpy as np
x_bar, s, n, alpha = 100, 15, 36, 0.05
t_val = t.ppf(1 - alpha/2, n-1)
margin = t_val * (s / np.sqrt(n))
ci = (x_bar - margin, x_bar + margin)
print("CI:", ci)
```

Output

CI: (94.92465750082785, 105.07534249917215)

## 3.5 25. Margin of Error

### 3.5.1 Formula

$E = Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$

Error bound.

### 3.5.2 Example

Z=1.96, sigma=10, n=100, E=1.96*1=1.96.

### 3.5.3 Practical Application

Survey accuracy.

**Python Code**

```
from scipy.stats import norm
import numpy as np
sigma, n, alpha = 10, 100, 0.05
z = norm.ppf(1 - alpha/2)
e = z * (sigma / np.sqrt(n))
print("Margin of Error:", e)
```

Output

Margin of Error: 1.959963984540054

## 3.6 26. Standard Error

### 3.6.1 Formula

$SE = \frac{s}{\sqrt{n}}$
Variability of sample mean.

### 3.6.2 Example

s=10, n=100, SE=1.

### 3.6.3 Practical Application

Assessing sample representativeness.

**Python Code**

```
import numpy as np
s, n = 10, 100
se = s / np.sqrt(n)
print("Standard Error:", se)
```

Output

Standard Error: 1.0

## 3.7 27. Chi-Square Statistic

### 3.7.1 Formula

$\chi^2 = \sum \frac{(O-E)^2}{E}$
For categorical data.

### 3.7.2 Example

Observed [10,20], Expected [15,15], chi2 = $(10\text{-}15)^2/15 + (20-15)^2/15$3.33.

### 3.7.3 Practical Application

Goodness-of-fit tests in genetics.

**Python Code**

```
from scipy.stats import chisquare
observed = [10, 20]
expected = [15, 15]
chi2, p = chisquare(observed, expected)
print("Chi-Square:", chi2)
```

Output

Chi-Square: 3.333333333333333

## 3.8 28. F-Ratio

### 3.8.1 Formula

$F = \frac{s_1^2}{s_2^2}$
Compares variances.

### 3.8.2 Example

s1=4, s2=2, F=4.

### 3.8.3 Practical Application

Comparing process variabilities.

**Python Code**

```
s1_sq, s2_sq = 16, 4
f = s1_sq / s2_sq
print("F-Ratio:", f)
```

Output

F-Ratio: 4.0

## 3.9 29. ANOVA F-Test

### 3.9.1 Formula

$F = \frac{MS_{between}}{MS_{within}}$
For group means.

### 3.9.2 Example

Computed from group variances.

### 3.9.3 Practical Application

Comparing treatment effects in experiments.

**Python Code**

```
from scipy.stats import f_oneway
group1 = [1,2,3]
group2 = [4,5,6]
f_stat, p = f_oneway(group1, group2)
print("ANOVA␣F:", f_stat)
```

Output

ANOVA F: 13.5

## 3.10 30. Correlation (Pearson)

### 3.10.1 Formula

$r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2 \sum(y_i-\bar{y})^2}}$

Measures linear relationship.

### 3.10.2 Example

For x=[1,2,3], y=[2,4,6], r=1.

### 3.10.3 Practical Application

Relating height and weight.

**Python Code**

```
import numpy as np
x = np.array([1,2,3])
y = np.array([2,4,6])
corr = np.corrcoef(x, y)[0,1]
print("Correlation:", corr)
```

Output

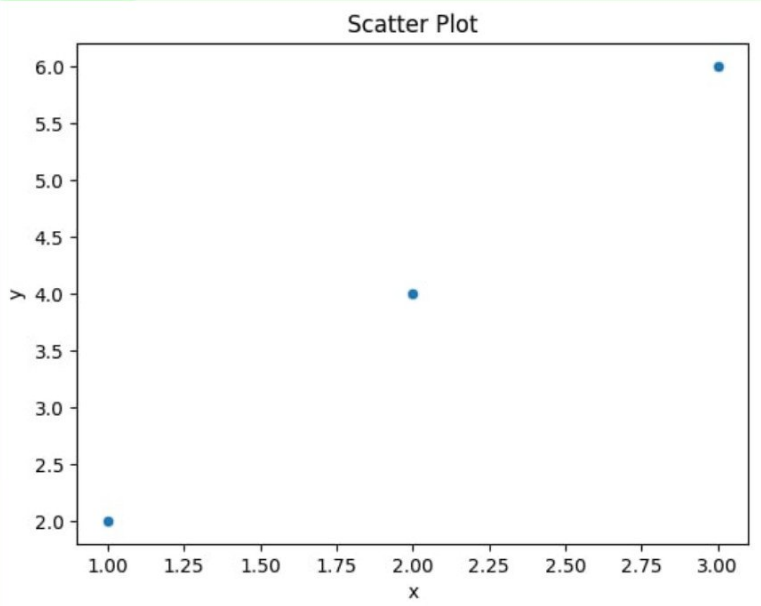
Correlation: 1.0

### 3.10.4 For Visualization

**Python Code**

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({'x': x, 'y': y})
sns.scatterplot(data=df, x='x', y='y')
plt.title("Scatter␣Plot")
plt.show()
```

Output




# 4 Regression & ML Stats

## 4.1 31. Simple Linear Regression

### 4.1.1 Formula

$y = \beta_0 + \beta_1 x + \epsilon$

Predicts y from x.

### 4.1.2 Example

Predicting sales from ad spend.

### 4.1.3 Practical Application

Forecasting revenue based on marketing.

**Python Code**

```
from sklearn.linear_model import LinearRegression
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
print("Slope:", model.coef_[0], "Intercept:", model.intercept_)
```

Output

Slope: 2.0 Intercept: 0.0

## 4.2 32. Slope (1)

### 4.2.1 Formula

$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$
Rate of change.

### 4.2.2 Example

As above, 1=2.

### 4.2.3 Practical Application

Impact of variable increase.

**Python Code**

```
from sklearn.linear_model import LinearRegression
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
print("Slope:", model.coef_[0])
```

Output

Slope: 2.0

## 4.3 33. Intercept (0)

### 4.3.1 Formula

$\beta_0 = \bar{y} - \beta_1 \bar{x}$
Y when x=0.

### 4.3.2 Example

0 in linear example.

### 4.3.3 Practical Application

Baseline value.

**Python Code**

```
from sklearn.linear_model import LinearRegression
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
print("Intercept:", model.intercept_)
```

Output

Intercept: 0.0

## 4.4 34. R-Squared

### 4.4.1 Formula

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

### 4.4.2 Example

Perfect fit R2=1.

### 4.4.3 Practical Application

Model goodness in predictions.

**Python Code**

```
from sklearn.linear_model import LinearRegression
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
r2 = model.score(x, y)
print("R-Squared:", r2)
```

Output

```
R-Squared: 1.0
```

## 4.5 35. Mean Squared Error (MSE)

### 4.5.1 Formula

$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$
Average squared error.

### 4.5.2 Example

For perfect fit, MSE=0.

### 4.5.3 Practical Application

Evaluating regression accuracy.

**Python Code**

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
y_pred = model.predict(x)
mse = mean_squared_error(y, y_pred)
print("MSE:", mse)
```

Output

MSE: 0.0

## 4.6 36. Root Mean Squared Error (RMSE)

### 4.6.1 Formula

$RMSE = \sqrt{MSE}$

Error in original units.

### 4.6.2 Example

sqrt(4)=2.

### 4.6.3 Practical Application

Interpretable error in forecasting.

**Python Code**

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
y_pred = model.predict(x)
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
```

Output

RMSE: 0.0

## 4.7 37. Mean Absolute Error (MAE)

### 4.7.1 Formula

$MAE = \frac{\sum |y_i - \hat{y}_i|}{n}$

Average absolute error.

### 4.7.2 Example

Errors [1,1,1], MAE=1.

### 4.7.3 Practical Application

Robust to outliers in evaluation.

**Python Code**

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(x, y)
y_pred = model.predict(x)
mae = mean_absolute_error(y, y_pred)
print("MAE:", mae)
```

Output

MAE: 0.0

## 4.8 38. Log-Loss

### 4.8.1 Formula

$-\frac{1}{n}\sum[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$
For binary classification.

### 4.8.2 Example

Perfect predictions, log-loss=0.

### 4.8.3 Practical Application

Evaluating probabilistic classifiers.

**Python Code**

```
from sklearn.metrics import log_loss
y_true = [0, 1]
y_prob = [[0.9, 0.1], [0.2, 0.8]]
loss = log_loss(y_true, y_prob)
print("Log-Loss:", loss)
```

Output

Log-Loss: 0.16425203348658793

## 4.9 39. Odds

### 4.9.1 Formula

$\text{Odds} = \frac{p}{1-p}$
Ratio of success to failure.

### 4.9.2 Example

p=0.8, odds=4.

#### 4.9.3 Practical Application

Betting and risk in logistics.

**Python Code**

```
p = 0.8
odds = p / (1 - p)
print("Odds:", odds)
```

Output

Odds: 4.0

### 4.10 40. Logit

#### 4.10.1 Formula

$\log\left(\frac{p}{1-p}\right)$
Log of odds.

#### 4.10.2 Example

For p=0.8, logit=log(4)1.386.

#### 4.10.3 Practical Application

In logistic regression.

**Python Code**

```
import numpy as np
p = 0.8
odds = p / (1 - p)
logit = np.log(odds)
print("Logit:", logit)
```

Output

Logit: 1.3862943611198906

## 5 Advanced & Data Science Applications

### 5.1 41. Entropy

#### 5.1.1 Formula

$H(X) = -\sum p(x) \log p(x)$
Measures uncertainty.

#### 5.1.2 Example

Fair coin: -2*(0.5*log0.5)0.693.

### 5.1.3 Practical Application

In decision trees, choosing splits.

**Python Code**

```
import numpy as np
probs = np.array([0.5, 0.5])
entropy = -np.sum(probs * np.log(probs))
print("Entropy:", entropy)
```

Output

Entropy: 0.6931471805599453

## 5.2 42. Gini Index

### 5.2.1 Formula

$G = 1 - \sum p_i^2$
Impurity measure.

### 5.2.2 Example

For [0.5,0.5], 1-2*(0.25)=0.5.

### 5.2.3 Practical Application

Node purity in classification trees.

**Python Code**

```
probs = np.array([0.5, 0.5])
gini = 1 - np.sum(probs**2)
print("Gini:", gini)
```

Output

Gini: 0.5

## 5.3 43. Information Gain

### 5.3.1 Formula

$IG = H(\text{parent}) - \sum^n H(\text{child}_i)$
Reduction in entropy.

### 5.3.2 Example

Parent entropy 1, children 0.5 each, IG=0.5.

### 5.3.3 Practical Application

Feature selection in ML.

**Python Code**

```
import numpy as np
parent_h = 1.0
child_h1 = 0.5
child_h2 = 0.5
weight1 = 0.5
weight2 = 0.5
ig = parent_h - (weight1 * child_h1 + weight2 * child_h2)
print("Information␣Gain:", ig)
```

Output

Information Gain: 0.5

## 5.4 44. KL Divergence

### 5.4.1 Formula

$D_{KL}(P||Q) = \sum P(x) \log \frac{P(x)}{Q(x)}$
Distribution difference.

### 5.4.2 Example

P=[0.5,0.5], Q=[0.6,0.4], KL0.02.

### 5.4.3 Practical Application

Model comparison in NLP.

**Python Code**

```
from scipy.stats import entropy
p = [0.5, 0.5]
q = [0.6, 0.4]
kl = entropy(p, q)
print("KL␣Divergence:", kl)
```

Output

KL Divergence: 0.020400616622643057

## 5.5 45. Covariance

### 5.5.1 Formula

$\text{Cov}(X,Y) = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{n-1}$
Joint variability.

### 5.5.2 Example

x=[1,2], y=[3,4], cov=0.5.

### 5.5.3 Practical Application

Portfolio diversification in finance.

**Python Code**

```
import numpy as np
x = np.array([1,2])
y = np.array([3,4])
cov = np.cov(x, y)[0,1]
print("Covariance:", cov)
```

Output

Covariance: 0.5

## 5.6 46. Standardization

### 5.6.1 Formula

$z = \frac{x - \bar{x}}{s}$

To mean 0, std 1.

### 5.6.2 Example

Data [1,2,3], z=[-1.22,0,1.22] approx.

### 5.6.3 Practical Application

Preprocessing for ML algorithms.

**Python Code**

```
from sklearn.preprocessing import StandardScaler
import numpy as np
data = np.array([[1],[2],[3]])
scaler = StandardScaler().fit_transform(data)
print("Standardized:", scaler.flatten())
```

Output

Standardized: [-1.22474487 0. 1.22474487]

## 5.7 47. Normalization (Min-Max)

### 5.7.1 Formula

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

To [0,1].

### 5.7.2 Example

[1,2,3] -¿ [0,0.5,1].

### 5.7.3 Practical Application

Image processing.

**Python Code**

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np
data = np.array([[1],[2],[3]])
scaler = MinMaxScaler().fit_transform(data)
print("Normalized:", scaler.flatten())
```

Output

```
Normalized: [0. 0.5 1. ]
```

## 5.8 48. Principal Component (PCA Eigenvector)

### 5.8.1 Formula

Maximize $\frac{w^T S w}{w^T w}$

Variance maximization.

### 5.8.2 Example

### 5.8.3 Practical Application

Dimensionality reduction in data viz.

**Python Code**

```
from sklearn.decomposition import PCA
import numpy as np
data = np.array([[1,2], [3,4], [5,6]])
pca = PCA(n_components=1).fit_transform(data)
print("PCA:", pca.flatten())
```

Output

```
PCA: [-2.82842712 0. 2.82842712]
```

## 5.9 49. Mahalanobis Distance

### 5.9.1 Formula

$D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$

Multivariate distance.

### 5.9.2 Example

For identity cov, same as Euclidean.

### 5.9.3 Practical Application

Outlier detection in multivariate data.

**Python Code**

```
import numpy as np
from scipy.spatial.distance import mahalanobis
x = [0,0]
mu = [1,1]
cov = np.eye(2)
d = mahalanobis(x, mu, np.linalg.inv(cov))
print("Mahalanobis:", d)
```

Output

Mahalanobis: 1.4142135623730951

## 5.10 50. Silhouette Score

### 5.10.1 Formula

$s = \frac{b-a}{\max(a,b)}$ (where a = intra-cluster distance, b = nearest-cluster distance)
Cluster quality.

### 5.10.2 Example

Perfect clustering s=1.

### 5.10.3 Practical Application

Evaluating k-means clusters.

**Python Code**

```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
import numpy as np
data = np.array([[1,2], [1,4], [5,6], [5,8]])
kmeans = KMeans(n_clusters=2, random_state=42)
labels = kmeans.fit(data).labels_
score = silhouette_score(data, labels)
print("Silhouette:", score)
```

Output

Silhouette: 0.6471221207743068

### 5.10.4 For Visualization of Clusters

**Python Code**

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame(data, columns=['x', 'y'])
df['cluster'] = labels
sns.scatterplot(data=df, x='x', y='y', hue='cluster')
plt.title("Cluster Plot")
plt.show()
```

Output