Probabilistic Analysis of Open-Source Test-Automation Repositories

Lamhot Jepri Manarihon Siagian

June 1, 2025

Introduction

Probabilistic statistics provides a systematic framework for analyzing datasets that combine categorical (qualitative) attributes with numerical (quantitative) measures. In this research, I examine a curated list of 24 prominent open-source test-automation repositories (e.g., Playwright, Detox, RobotFramework), focusing on two qualitative variables,API Support (Yes/No/Partial) and Mobile Support (Yes/No/Partial), and two quantitative variables, Star Count and Fork Count. By applying fundamental probability rules, conditional probabilities, and Bayes' theorem, I aim to extract actionable insights that inform decision-making processes in software engineering contexts (Ross, 2014). Specifically, I compute category probabilities, evaluate event probabilities involving quantitative thresholds, derive conditional probabilities between features, and apply Bayes' theorem to update beliefs. Ultimately, this research illustrates how probabilistic reasoning can guide practitioners in selecting appropriate tools based on feature availability and popularity metrics.

Chapter 1: Probability Rules and Concepts

1.1 Probability Distribution of a Qualitative Variable

To begin, I consider the qualitative variable API Support, which assumes three categories: "Yes," "No," and "Partial." Out of the 24 repositories under study, the observed frequencies are as follows:

- **API Support = Yes**: 13
- **API Support = No**: 10
- **API Support = Partial**: 1

Employing the frequentist interpretation of probability, I calculate the probability of each category as the relative frequency (Grimmett & Stirzaker, 2001).

Thus:

$$P(\text{API Support} = \text{Yes}) \;=\; \frac{13}{24} \;\approx\; 0.542,$$

$$P(\text{API Support} = \text{No}) \;=\; \frac{10}{24} \;\approx\; 0.417,$$

$$P(\text{API Support} = \text{Partial}) \;=\; \frac{1}{24} \;\approx\; 0.042.$$

This straightforward frequency-based approach ensures the empirical distribution aligns with the observed dataset. In practice, knowing that approximately 54.2% of repositories provide full API support whereas 41.7% do not, and 4.2% only partially, can help developers gauge the likelihood of encountering an API in a randomly selected tool.

1.2 Probability of Events Involving Quantitative Variables

Next, I turn to the quantitative variable **Star Count**, defined as the integer portion of each repository's "stars" metric. I define two events:

- A: "Repository has more than 10,000 stars."
- B: "Repository has more than 2,000 forks."

From the data, repositories satisfying A are indices $\{1, 2, 3, 16, 17, 18, 22\}$, yielding 7 out of 24. Thus,

$$P(A) \;=\; \frac{7}{24} \approx 0.292.$$

Similarly, repositories satisfying B are $\{1, 3, 16, 17, 18, 20\}$, yielding 6 out of 24. Hence,

$$P(B) \;=\; \frac{6}{24} = 0.250.$$

To compute the probability of the union $A \cup B$ (">10,000 stars or >2,000 forks"), I apply the addition rule (Ross, 2014):

$$P(A \cup B) \;=\; P(A) + P(B) - P(A \cap B).$$

The intersection $A \cap B$ corresponds to repositories with both ">10,000 stars" and ">2,000 forks": indices $\{1, 3, 16, 17, 18\}$, or 5 out of 24. Thus,

$$P(A \cap B) \;=\; \frac{5}{24} \approx 0.208,$$

$$P(A \cup B) \;=\; 0.292 + 0.250 - 0.208 = 0.334 \approx 0.333.$$

Therefore, there is roughly a one-third chance that a randomly selected repository from this list has either more than 10,000 stars or more than 2,000 forks. By employing these basic probability rules, decision-makers can quantify the likelihood that a repository meets desired popularity thresholds.

Chapter 2: Conditional Probability

2.1 Conditional Probability for Qualitative Variables

Conditional probability enables an examination of how the presence of one feature influences the probability of another. I define two events based on qualitative attributes:

- Q: "API Support = Yes."
- R: "Mobile Support = Yes."

From the dataset:

$$P(Q) \;=\; \frac{13}{24} = 0.542, \quad P(R) \;=\; \frac{12}{24} = 0.500.$$

Next, I determine $P(R \cap Q)$, the proportion of repositories that support both API and mobile. These are indices $\{3, 7, 21, 24\}$, i.e., 4 out of 24:

$$P(R \cap Q) \;=\; \frac{4}{24} \approx 0.167.$$

Accordingly, the conditional probability of mobile support given API support is:

$$P(R \mid Q) \;=\; \frac{P(R \cap Q)}{P(Q)} \;=\; \frac{0.167}{0.542} \;\approx\; 0.308.$$

Thus, among repositories that provide API support, roughly 30.8% also offer mobile support. This finding indicates that API-enabled frameworks are not always accompanied by mobile capabilities.

2.2 Conditional Probability for Quantitative Variables

For the quantitative events A (">10,000 stars") and B (">2,000 forks"), I compute the conditional probability:

$$P(A \mid B) \;=\; \frac{P(A \cap B)}{P(B)} \;=\; \frac{\frac{5}{24}}{\frac{6}{24}} \;=\; \frac{5}{6} \approx 0.833.$$

Hence, if a repository already has more than 2,000 forks, there is an 83.3% chance it also has more than 10,000 stars. This high conditional probability suggests a strong correlation between forks and stars for highly popular projects. In practice, developers looking for high-visibility tools can use ">2,000 forks" as a reliable proxy for ">10,000 stars" with relatively minimal risk.

Chapter 3: Bayes' Theorem

3.1 Scenario Selection and Problem Setup

Bayes' theorem provides a mechanism for updating prior beliefs in light of new evidence (Gelman et al., 2013). I choose a scenario that merges qualitative information about feature availability. Specifically:

- Q: "API Support = Yes" (prior event).
- M: "Mobile Support = Yes" (evidence event).

I seek to compute the posterior probability $P(Q \mid M)$—the probability that a repository supports an API given that it supports mobile. Bayes' theorem for discrete events states:

$$P(Q \mid M) = \frac{P(M \mid Q)\, P(Q)}{P(M)}.$$

From prior analyses:

1. $P(Q) = 13/24 \approx 0.542$.

2. $P(M) = 12/24 = 0.500$.

3. $P(M \mid Q)$ is the probability that a repository supports mobile given it supports an API. From Chapter 2:

$$P(M \mid Q) = \frac{P(M \cap Q)}{P(Q)} = \frac{\frac{4}{24}}{\frac{13}{24}} = \frac{4}{13} \approx 0.308.$$

3.2 Application of Bayes' Theorem

Substituting these values into Bayes' formula:

$$P(Q \mid M) = \frac{0.308 \times 0.542}{0.500} = \frac{0.167}{0.500} = 0.333.$$

Therefore, if a repository supports mobile, there is a 33.3% chance it also supports an API. In Bayesian terms, the prior belief $P(Q) = 0.542$ is updated to the posterior $P(Q \mid M) = 0.333$ once I observe that the repository supports mobile (Gelman et al., 2013).

3.3 Interpretation of Bayesian Results

This posterior probability of 33.3% indicates that mobile-centric frameworks are less likely—relative to the general population—to provide full API support. In practical terms, a developer seeking a mobile framework with a robust API should realize that fewer than one in three mobile-capable tools meet that criterion. Thus, Bayes' theorem translates feature co-occurrence data into actionable insights: the evidence "Mobile=Yes" reduces the belief in "API=Yes" from approximately 54.2% to 33.3%. Such probabilistic updating is crucial when decisions hinge on multiple feature dependencies.

Conclusion

In this research, I selected two qualitative variables (API Support, Mobile Support) and two quantitative variables (Star Count, Fork Count) from a dataset of 24 open-source automation-testing repositories. Chapter 1 computed category probabilities for **API Support** and applied basic probability rules to events ">10,000 stars" and ">2,000 forks," deriving that roughly one-third of projects satisfy at least one of these popularity thresholds. Chapter 2 demonstrated conditional probabilities: among API-supporting libraries, only about 30.8% offer mobile support, and given ">2,000 forks," there is an 83.3% chance of also having ">10,000 stars." Chapter 3 applied Bayes' theorem to update the probability of API support to 33.3% when observing mobile support. Collectively, these analyses highlight how probabilistic statistics—ranging from simple frequency counts to Bayesian updating—can guide informed decisions when selecting tools based on feature availability and community adoption. Future work could extend these methods to larger datasets or include additional features (e.g., CI/CD integrations, platform compatibility) to further refine probability estimates.

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013).
Bayesian data analysis (3rd ed.). Chapman & Hall/CRC.

Grimmett, G., & Stirzaker, D. (2001).
Probability and random processes (3rd ed.). Oxford University Press.

Ross, S. M. (2014).
A first course in probability (9th ed.). Pearson.

Appendix: Source Code

Below are the Python scripts used to perform the probability calculations and generate the empirical results discussed in this research. All code is presented in a clear, well-documented format.

```python
# Appendix: source_code.py

import pandas as pd

# Data setup: Qualitative variable counts
api_support_counts = {
    "Yes": 13,
    "No": 10,
    "Partial": 1
}

# Total number of repositories
total_repos = sum(api_support_counts.values())

# 1.1 Probability Distribution of API Support (Qualitative)
prob_api_yes = api_support_counts["Yes"] / total_repos
prob_api_no = api_support_counts["No"] / total_repos
prob_api_partial = api_support_counts["Partial"] / total_repos

print("P(API Support = Yes) =", round(prob_api_yes, 3))
print("P(API Support = No) =", round(prob_api_no, 3))
print("P(API Support = Partial) =", round(prob_api_partial, 3))

# Data setup: Quantitative thresholds (Star Count and Fork Count)
# For demonstration, assume we have a list of tuples: (stars, forks)
repos = [
    (73034, 4192),
    (11483, 1926),
    (10715, 2438),
    (9032, 554),
    (8616, 563),
    (8554, 1982),
    (6658, 1464),
    (5647, 739),
    (3503, 283),
    (3047, 345),
    (2063, 462),
    (2038, 243),
    (1919, 328),
    (1828, 322),
    (1620, 860),
    (28000, 9000),
    (17000, 7000),
    (47000, 3800),
    (2100, 700),
    (7500, 4200),
    (1300, 300),
    (27000, 1900),
    (1700, 1400),
    (1500, 400)
]

# 1.2 Events A and B
threshold_stars = 10000
threshold_forks = 2000

event_A = [1 for stars, _ in repos if stars > threshold_stars]
event_B = [1 for _, forks in repos if forks > threshold_forks]

P_A = sum(event_A) / len(repos)
P_B = sum(event_B) / len(repos)

print("\nP(A) (stars > 10000) =", round(P_A, 3))
print("P(B) (forks > 2000) =", round(P_B, 3))

# Intersection A   B
event_A_and_B = [1 for stars, forks in repos if (stars > threshold_stars and forks > thresho
P_A_and_B = sum(event_A_and_B) / len(repos)

print("P(A   B) =", round(P_A_and_B, 3))

# Union A   B
P_A_union_B = P_A + P_B - P_A_and_B
print("P(A   B) =", round(P_A_union_B, 3))
```

```python
# 2.1 Conditional Probability for Qualitative Variables
# Given API Support = Yes, what is P(Mobile Support = Yes)?
# Assume I have counts for both features (manually derived)
api_and_mobile_count = 4  # Number of repos where API=Yes and Mobile=Yes
mobile_support_count = 12  # Number of repos where Mobile=Yes

P_Q = api_support_counts["Yes"] / total_repos  # P(API=Yes)
P_R = mobile_support_count / total_repos       # P(Mobile=Yes)
P_R_and_Q = api_and_mobile_count / total_repos # P(Mobile=Yes and API=Yes)

P_R_given_Q = P_R_and_Q / P_Q
print("\nP(Mobile=Yes | API=Yes) =", round(P_R_given_Q, 3))

# 2.2 Conditional Probability for Quantitative Variables
P_A_and_B = P_A_and_B  # from earlier calculation
P_A_given_B = P_A_and_B / P_B
print("P(Stars>10000 | Forks>2000) =", round(P_A_given_B, 3))

# 3 Bayes' Theorem Application
P_M_given_Q = P_R_and_Q / P_Q  # P(Mobile=Yes | API=Yes)
P_M = P_R                     # P(Mobile=Yes)
P_Q_given_M = (P_M_given_Q * P_Q) / P_M

print("\nP(API=Yes | Mobile=Yes) =", round(P_Q_given_M, 3))
```

Appendix: Repository Data CSV

```
Repository/Tool,Stars,Forks,API,WEB,Desktop,Mobile,Link
microsoft/playwright,73034,4192,Yes,Yes,Yes,No,https://github.com/microsoft/playwright
wix/Detox,11483,1926,No,No,No,Yes,https://github.com/wix/Detox
robotframework/robotframework,10715,2438,Yes,Yes,No,Yes,https://github.com/robotframework/ro
web-infra-dev/midscene,9032,554,No,Yes,No,Yes,https://github.com/web-infra-dev/midscene
babalae/better-genshin-impact,8616,563,No,No,Yes,Yes,https://github.com/babalae/better-gensh
karatelabs/karate,8554,1982,Yes,Yes,No,No,https://github.com/karatelabs/karate
atinfo/awesome-test-automation,6658,1464,Yes,Yes,Yes,Yes,https://github.com/atinfo/awesome-t
google/EarlGrey,5647,739,No,No,No,Yes,https://github.com/google/EarlGrey
garden-io/garden,3503,283,Yes,No,No,No,https://github.com/garden-io/garden
getgauge/gauge,3047,345,Yes,Yes,No,No,https://github.com/getgauge/gauge
Blazemeter/taurus,2063,462,Yes,Yes,No,No,https://github.com/Blazemeter/taurus
MaaXYZ/MaaFramework,2038,243,No,No,Yes,Yes,https://github.com/MaaXYZ/MaaFramework
laravel/dusk,1919,328,No,Yes,No,No,https://github.com/laravel/dusk
AirtestProject/Poco,1828,322,No,No,Yes,Yes,https://github.com/AirtestProject/Poco
githublitao/api_automation_test,1620,860,Yes,No,No,No,https://github.com/githublitao/api_aut
Selenium,28000,9000,No,Yes,No,Partial,https://github.com/SeleniumHQ/selenium
Appium,17000,7000,No,Partial,Partial,Yes,https://github.com/appium/appium
```

```
Cypress,47000,3800,Partial,Yes,No,No,https://github.com/cypress-io/cypress
Selendroid,2100,700,No,No,No,Yes,https://github.com/selendroid/selendroid
Apache JMeter,7500,4200,Yes,Yes,No,No,https://github.com/apache/jmeter
OpenTest,1300,300,Yes,Yes,No,Yes,https://github.com/mcdcorp/opentest
Postman,27000,1900,Yes,No,No,No,https://github.com/postmanlabs/newman
SoapUI,1700,1400,Yes,No,No,No,https://github.com/SmartBear/soapui
Carina,1500,400,Yes,Yes,No,Yes,https://github.com/zebrunner/carina
```