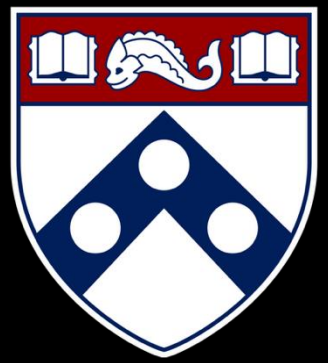
Secure Systems Engineering and Management

A Data-driven Approach



Security analytics: Intro to data analysis (part 2)

Hypothesis Testing, Effect Sizes, and Regression

Michael Hicks
UPenn CIS 7000-003
Spring 2026

Lecture 2: Regression Models — Linear and Logistic

Lecture 2 Overview

Topics:

- Linear regression fundamentals
- Categorical predictors (dummy coding)
- Model comparison (likelihood-ratio tests)
- Logistic regression for binary outcomes
- Odds ratios and predicted probabilities
- Common regression pitfalls

Part 1: Linear Regression Foundations

The Linear Model

Goal: Explain aspects of some empirical data
How? Model a continuous outcome (Y)
as a linear function of predictors (X)

Equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \epsilon$$

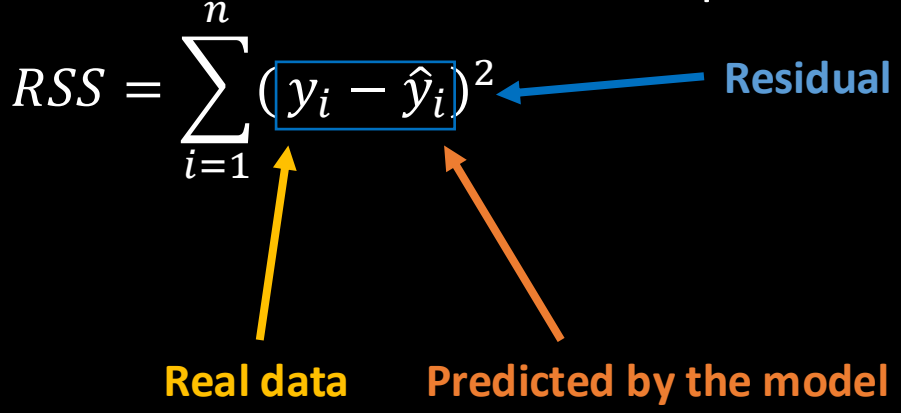
where

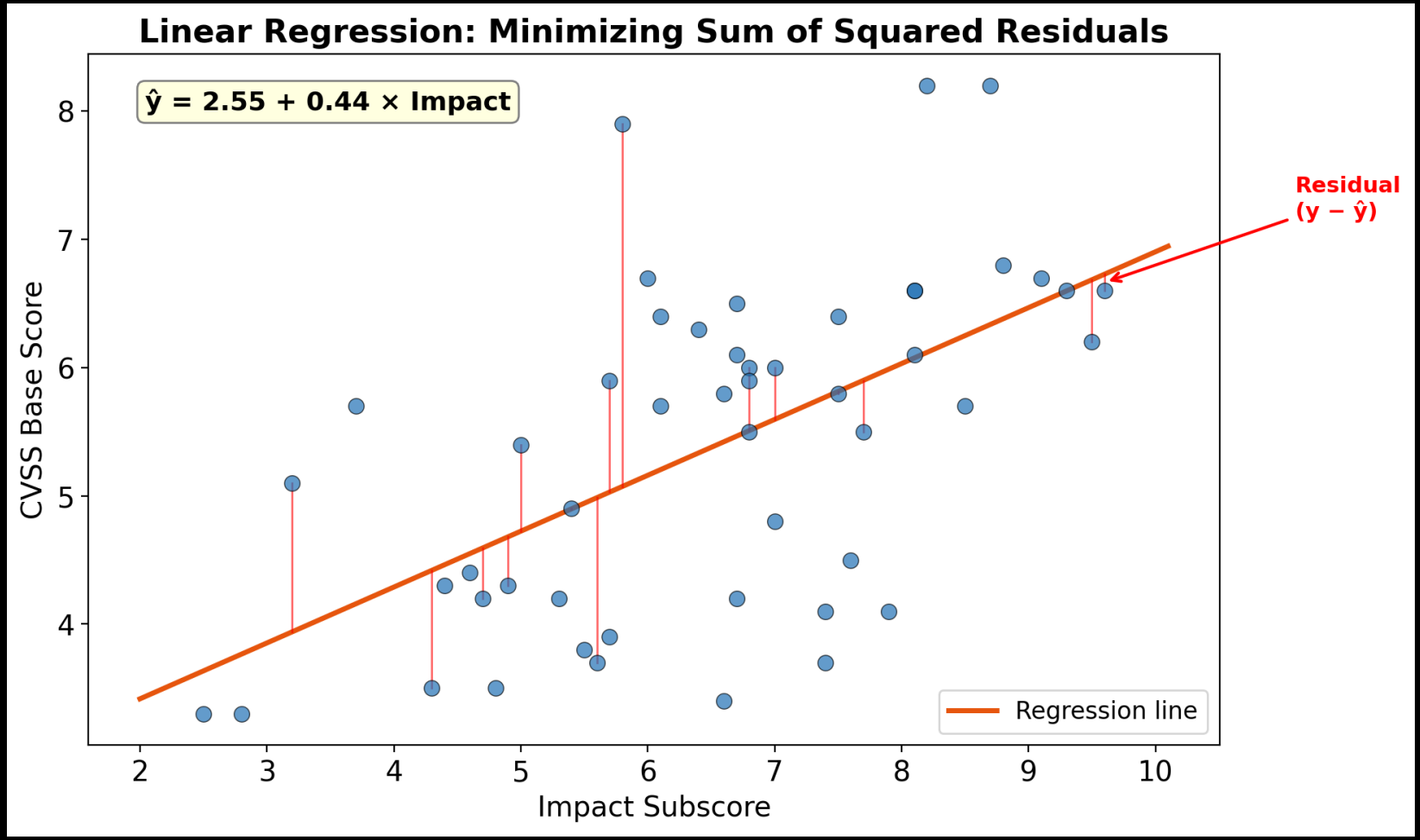
- β_0 = intercept
- β_i = slope (change in Y per unit X)
- ε = residual error

Ordinary Least Squares (OLS)

Objective: Fit data to a linear model

Approach: Find coefficients that minimize the sum of squared residuals




Linear Regression: Minimizing Sum of Squared Residuals
ŷ = 2.55 + 0.44 × Impact
Residual
(y − ŷ)
Regression line
CVSS Base Score
Impact Subscore
2
3
4
5
6
7
8
9
10
4
5
6
7
8

Key Regression Outputs

Output	Meaning
β (coefficients)	Estimated slopes and intercept
Standard errors	Uncertainty in coefficient estimates
t-statistics	β / SE — tests if coefficient ≠ 0
p-values	Significance of each coefficient
R^2	Proportion of variance explained

Adjusted R^2 penalizes for adding more predictors and is often preferred.

Interpreting Coefficients

Example: Predicting CVSS from subscores

$$CVSS = \beta_0 + \beta_1 \times Exploitability + \beta_2 \times Impact$$

Coefficient	Interpretation
β_0 (Intercept)	Predicted CVSS when Exploitability = 0 and Impact = 0
β_1	Expected change in CVSS for 1-unit increase in Exploitability, holding Impact constant
β_2	Expected change in CVSS for 1-unit increase in Impact, holding Impact constant

Actual CVSS: Not linear

For CVSS v2, the base score is computed as:

$$CVSS = round((0.6 \times Impact + 0.4 \times Exploitability - 1.5) \times f(Impact))$$

where:

- $f(Impact) = 1.176$ if Impact > 0, otherwise 0
- The result is rounded to 1 decimal place

Linear Regression in Python

```python
import statsmodels.formula.api as smf

# Fit linear model using formula interface
model = smf.ols('cvss_base ~ exploitability + impact',
data=data).fit()

# View comprehensive summary
print(model.summary())

# Extract specific components
print("Coefficients:")
print(model.params)             # Coefficients
print("\n95% CIs:")
print(model.conf_int())         # Confidence intervals
print(f"\nR² = {model.rsquared:.3f}")
print(f"Adjusted R² = {model.rsquared_adj:.3f}")
```

Running Linear Regression on Sample Data

Model: cvss_base ~ exploitability + impact

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.4157	0.092	4.50	<0.001	0.235	0.597
exploitability	0.3698	0.011	34.38	<0.001	0.349	0.391
impact	0.3886	0.012	33.78	<0.001	0.366	0.411

$R^2 = 0.588$, Adjusted $R^2 = 0.587$

...

Interpretation:

- Each 1-point increase in exploitability → +0.37 CVSS (holding impact constant)
- Each 1-point increase in impact → +0.39 CVSS (holding exploitability constant)
- Model explains 59% of variance in CVSS

Part 2: Categorical Predictors

The Problem

Regression requires numeric predictors.

But CWE category is categorical:

- “Memory”
- “InputValidation”

  . . .
- “Crypto”
- “Auth”

Solution: Convert to numeric using **dummy coding**

Dummy Coding (Treatment Coding)

Convert a k-level categorical variable into k−1 binary indicators:

CWE Category	Is_InputVal	Is_Memory	Is_Crypto
Auth (reference)	0	0	0
InputValidation	1	0	0
Memory	0	1	0
Crypto	0	0	1

Why k−1 Dummies?

Including all k dummies creates perfect multicollinearity:

If Is_InputVal = Is_Memory = Is_Crypto = 0, it *must* be Auth

The reference category is already fully determined by the others

Interpreting Dummy Coefficients

Model:

$$CVSS = \beta_0 + \beta_1 \times Is\_Memory + \beta_2 \times Is\_Crypto + \ldots$$

Example output:

Coefficient	Estimate	p-value
Intercept	5.10	<0.001
Is_Memory	0.42	0.001
Is_Crypto	-0.21	0.082

Coefficient	Estimate	p-value
Intercept	5.10	<0.001
Is_Memory	0.42	0.001
Is_Crypto	-0.21	0.082

Interpretation:

- Auth vulnerabilities (reference) have mean CVSS = 5.10
- Memory vulnerabilities have CVSS 0.42 *higher* than Auth
- Crypto vulnerabilities do not significantly differ from Auth

Choosing the Reference Level

Defaults:

- R: Alphabetically first, or lowest numeric value
- Python (statsmodels): Same

Better approach: *Choose a meaningful baseline*

- Most common category
- Control/baseline condition
- Theoretically neutral category

Setting Reference Level in Python

```
import statsmodels.formula.api as smf

# Method 1: Specify reference in formula using C()
model = smf.ols(
    'cvss_base ~ exploitability + impact + C(cwe_category,
Treatment(reference="Auth"))',
    data=data
).fit()

print(model.summary())

# Method 2: Reorder categories in DataFrame first
data['cwe_category'] = pd.Categorical(
    data['cwe_category'],
    categories=['Auth', 'Crypto', 'InputValidation', 'Memory', 'Other']
)
model2 = smf.ols('cvss_base ~ exploitability + impact + cwe_category',
                 data=data).fit()
```

Running Linear Regression with Categorical Predictor

Model: cvss_base ~ exploitability + impact + C(cwe_category, ref="Auth")

	coef	std err	t	p>|t|	[0.025	0.975]
Intercept	0.4176	0.112	3.73	<0.001	0.198	0.637
cwe_category[Crypto]	-0.0422	0.069	-0.61	0.539	-0.177	0.093
cwe_category[InputVal]	-0.0202	0.060	-0.34	0.738	-0.139	0.098
cwe_category[Memory]	0.0495	0.061	0.82	0.415	-0.070	0.169
cwe_category[Other]	0.0671	0.071	0.95	0.343	-0.072	0.206
exploitability	0.3692	0.011	34.31	<0.001	0.348	0.390
impact	0.3871	0.013	30.71	<0.001	0.362	0.412

$R^2 = 0.589$, Adjusted $R^2 = 0.588$

⚠ Pitfall: Forgetting the Reference Category

Bad:

Predictor	Coefficient
Is_Memory	0.42
Is_Crypto	-0.21

Better:

Predictor	Coefficient
Is_Auth	–
Is_Memory	0.42
Is_Crypto	-0.21

Pitfall: Forgetting the Reference Category

. . .

Wrong: “Memory vulnerabilities have severity 0.42”

Also wrong: “Memory vulnerabilities are 0.63 more severe than Crypto”

. . .

Right: “Memory vulnerabilities have CVSS 0.42 higher than the reference (Auth). Crypto vulnerabilities have CVSS 0.21 lower than Auth.”

Part 3: Model Comparison

The Problem

You've fit two models:

- **Model 1 (simple):** cvss ~ exploitability + impact
- **Model 2 (complex):** cvss ~ exploitability + impact + cwe_category

Question: Does adding CWE category significantly improve the model?

Likelihood-Ratio Test

Logic: Compare how well each model fits the data via their likelihoods

Test statistic:

$$LR = 2 \times (LL_{full} - LL_{reduced})$$

Under H_0 (reduced model is adequate):

$$LR \sim \chi^2_{df}$$

where df = difference in number of parameters

LR Test in Python

```
import statsmodels.formula.api as smf

# Fit nested models
model1 = smf.ols('cvss_base ~ exploitability + impact',
data=data).fit()
model2 = smf.ols('cvss_base ~ exploitability + impact +
C(cwe_category)',
                 data=data).fit()

# Compare R² values
print(f"Model 1 R²: {model1.rsquared:.3f}")
print(f"Model 2 R²: {model2.rsquared:.3f}")

# Likelihood-ratio test
lr_stat, p_value, df_diff = model2.compare_lr_test(model1)
print(f"LR = {lr_stat:.2f}, df = {df_diff}, p =
{p_value:.4f}")
```

Interpreting LR Test Results

Significant ($p < 0.05$):

- The additional predictors improve model fit
- Keep the fuller model

Non-significant ($p \geq 0.05$):

- The simpler model is adequate
- Prefer parsimony (fewer predictors)

LR Test: Does CWE Category Improve Our Model?

Model 1 (reduced): cvss_base ~ exploitability + impact

Model 2 (full): cvss_base ~ exploitability + impact + cwe_category

Model 1 R^2: 0.588 Model 2 R^2: 0.589

LR = 4.68, df = 4, p = 0.3223

Result: Not significant ($p = 0.32 >> 0.05$)

- Adding CWE category does **not** significantly improve model fit
- The simpler model is adequate — prefer parsimony
- R^2 increase is negligible (0.588 → 0.589)

Part 4: Logistic Regression

Predicting Binary Outcomes

Goal: Predict whether a vulnerability is *actively exploited* (in KEV catalog)

Outcome: Binary (0 = not exploited, 1 = exploited)

Why linear regression fails:

- Can predict values outside [0, 1]
- Residuals cannot be normally distributed
- Violates constant variance assumption

The Logistic Model

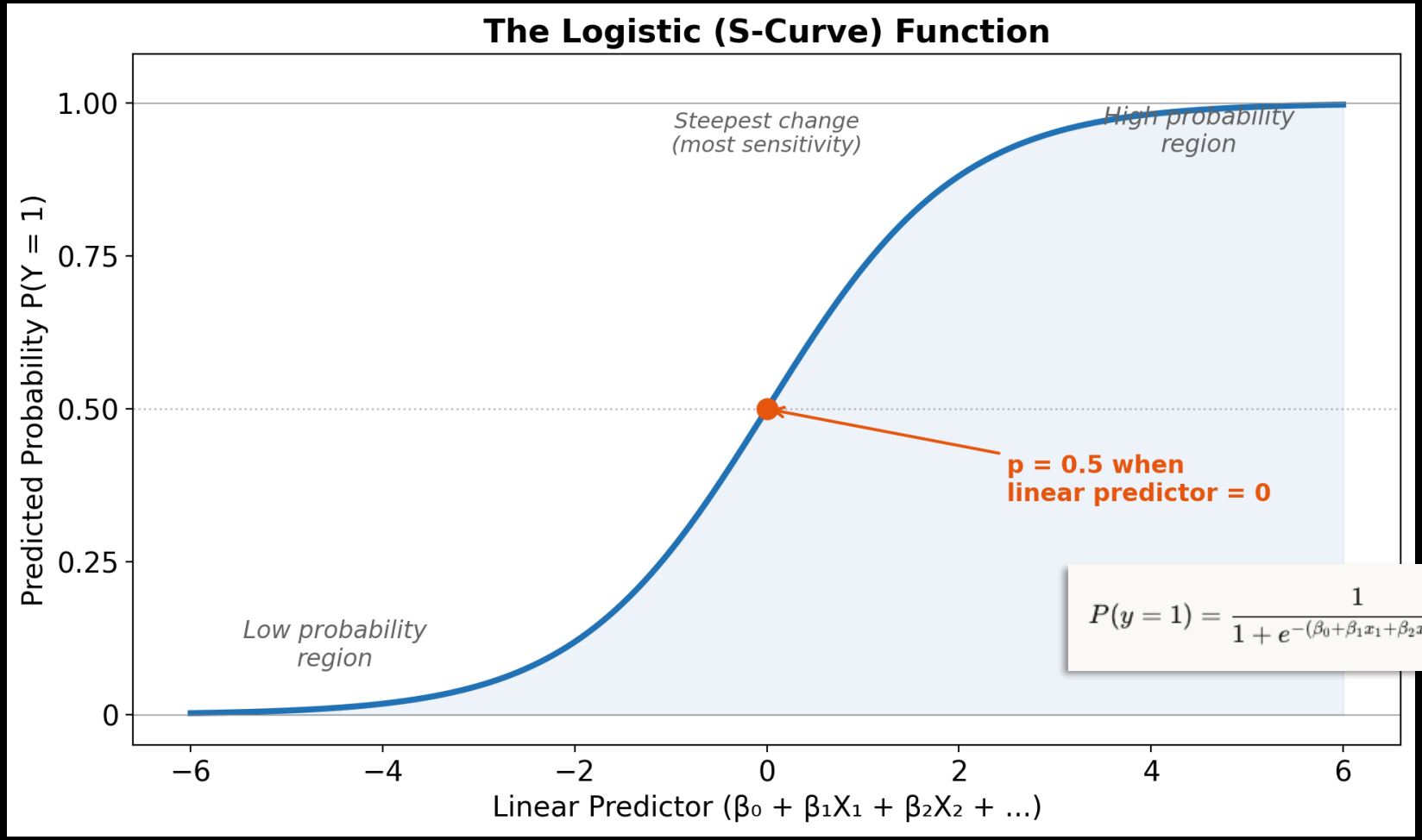
Solution: Model the *log-odds* of the outcome as a linear function:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$$

where p = P(Y = 1)

Logistic function maps log-odds to probabilities:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots)}}$$


The Logistic (S-Curve) Function
Predicted Probability P(Y = 1)
1.00
0.75
0.50
0.25
0
Steepest change
(most sensitivity)
High probability
region
Low probability
region
p = 0.5 when
linear predictor = 0
$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots)}}$
−6
−4
−2
0
2
4
6
Linear Predictor (β₀ + β₁X₁ + β₂X₂ + ...)

Odds and Log-Odds

Odds:

$$\text{odds} = \frac{p}{1-p}$$

If p = 0.75, odds = 0.75/0.25 = 3:1 (“3 to 1”)

Log-odds (logit):

$$\text{logit}(p) = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

If p = 0.5, log-odds = 0
If p = 0.75, log-odds = 1.1

Interpreting Coefficients: Log-Odds

Log-odds coefficient (β):

> "A one-unit increase in X is associated with a β change in the *log-odds* of the outcome."

Example:

$\beta_1 = 0.5$ for CVSS score

> "Each 1-point increase in CVSS is associated with a 0.5 increase in the log-odds of exploitation."

Problem: This is not intuitive!

Interpreting Coefficients: Odds Ratios

Odds ratio (OR):

$$OR = e^{\beta}$$

"A one-unit increase in X *multiplies* the odds by OR."

Odds Ratio	Interpretation
OR = 1	No association
OR > 1	Higher odds (positive association)
OR < 1	Lower odds (negative association)

Odds Ratio Example

Coefficient: $\beta_{CVSS} = 0.5$

Odds ratio: $e^{0.5} = 1.65$

Interpretation:

> "Each 1-point increase in CVSS is associated with 1.65 times the odds of being exploited."

Or equivalently:

> "Each 1-point increase in CVSS increases the odds of exploitation by 65%."

Logistic Regression in Python

```
import statsmodels.formula.api as smf
import numpy as np
# Fit logistic regression
log_model = smf.logit('in_kev ~ cvss_base + C(cwe_category)',
                      data=data).fit()
# View summary (log-odds coefficients)
print(log_model.summary())
# Convert to odds ratios
print("\nOdds Ratios:")
print(np.exp(log_model.params))
# 95% CIs for odds ratios
print("\n95% CIs for Odds Ratios:")
print(np.exp(log_model.conf_int()))
```

```
pd.DataFrame({
    'OR': np.exp(log_model.params),
    '2.5%': np.exp(log_model.conf_int()[0]),
    '97.5%': np.exp(log_model.conf_int()[1])
})
```

Running Logistic Regression on Sample Data

Model: in_kev ~ cvss_base + cwe_category (reference = Auth)

	coef	OR	[95% CI OR]	p
Intercept	-8.07	0.00		<0.001
cvss_base	0.60	1.82	[1.39, 2.39]	<0.001
C(cwe_category)[Crypto]	0.74	2.10	[0.49, 8.96]	0.317
C(cwe_category)[InputVal]	0.72	2.06	[0.56, 7.59]	0.278
C(cwe_category)[Memory]	0.99	2.69	[0.77, 9.32]	0.119
C(cwe_category)[Other]	-0.10	0.91	[0.15, 5.52]	0.915

Key interpretation:

Each 1-point increase in CVSS → 1.82× the odds of exploitation ($p < 0.001$)

Converting to Predicted Probabilities

Most interpretable: What's the probability for a specific scenario?

$$\hat{p} = \frac{1}{1 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots)}}$$

Example predictions from our model:

CVSS	CWE Category	P(exploitation)
4.0	Auth	0.3%
6.0	Auth	1.1%
8.0	Auth	3.6%
8.0	Memory	9.2%

Predicted Probabilities in Python

```
# Create scenarios for prediction
new_data = pd.DataFrame({
    'cvss_base': [4.0, 6.0, 8.0, 8.0],
    'cwe_category': ['Auth', 'Auth', 'Auth', 'Memory']
})

# Predict probabilities
predictions = log_model.predict(new_data)

# Display
new_data['prob'] = predictions.round(3)
print(new_data)
```

⚠ Pitfall: Incorrect Scale in Logistic Regression

The coefficient is 0.52…

❌ “Higher CVSS increases exploitation by 0.52”

❌ “Higher CVSS increases exploitation by 52%”

✓ “The log-odds coefficient is 0.52, corresponding to an odds ratio of 1.68”

✓ “Each 1-point CVSS increase is associated with 1.68× the odds of exploitation”

✓ “Predicted probability rises from 1.5% at CVSS=6 to 4% at CVSS=8”

Part 5: Regression Pitfalls

Caution: Class Imbalance in Your Data

KEV catalog: ~1,200 actively exploited CVEs
Total CVEs: ~250,000+

Exploitation rate: < 0.5%

In our sample data: 37 exploited out of 2,000 (1.8%)

Why Rare Events Are Challenging

1. **Accuracy is misleading:** Predicting “not exploited” for everything gives 98%+ accuracy
2. **Coefficients may be unstable:** Few positive cases → high variance
3. **Predicted probabilities may be miscalibrated:** Systematically under- or over-estimated

Better Metrics for Rare Events

Metric	Meaning
Sensitivity (Recall)	Of exploited CVEs, what % did we catch?
Precision	Of CVEs we flagged, what % are actually exploited?
F1 score	Harmonic mean of precision and recall
AUC-ROC	Overall discriminative ability

✓ Practical Guidance

- Report sensitivity and specificity, not just accuracy
- Consider whether the base rate makes prediction meaningful
- Acknowledge class imbalance as a limitation
- Use the logistic model to understand *associations*, not for operational prediction

⚠ Pitfall: Not Reporting Model Fit

The problem: Coefficients can be “significant” even if the model explains almost nothing.

Example of misleading results:

“CWE category significantly predicts CVSS ($p < 0.001$)”

But if $R^2 = 0.02$… the model explains only 2% of variance!

Always report

Model Type	Report
Linear	R^2, Adjusted R^2
Logistic	Pseudo-R^2 (McFadden), comparison to null model

Pitfall: Unclear Model Specification

Ambiguous: “We ran a logistic regression to predict exploitation.”

Clear: “We fit a logistic regression predicting KEV inclusion (1 = in catalog, 0 = not) from:

- CVSS base score (continuous)
- CWE category (dummy-coded, reference = Auth)
- Publication year (continuous, centered at 2020)

No interactions were included.”

⚠ Pitfall: Regression ≠ Causation

The problem: Regression shows association, not causation.

❌ “Our regression shows that higher CVSS scores *cause* vulnerabilities to be exploited.”

✓ “Higher CVSS scores are *associated with* greater likelihood of exploitation. However, this association may reflect confounding factors rather than a causal relationship.”

Pitfall: Selectively Reporting Coefficients

The problem: Reporting only “significant” predictors gives incomplete picture.

Wrong:

> “We found that Memory vulnerabilities (p = 0.02) predict higher CVSS.”

Better:

> “We report all coefficients in Table 3. Memory (+0.42, p = 0.001) showed significantly higher CVSS than Auth (reference). Crypto (-0.21, p = 0.08) and InputValidation (-0.08, p = 0.52) did not significantly differ.”

Regression Reporting Checklist

Model Specification:

☐ All predictors listed

☐ Reference categories specified

☐ Transformations described

☐ Interactions noted

Model Fit:

☐ R^2 or pseudo-R^2 reported

☐ Comparison to null model

Coefficients:

☐ All coefficients reported (not just significant ones)

☐ Odds ratios for logistic regression

☐ Confidence intervals for key estimates

Interpretation:

☐ Coefficients interpreted relative to ref.

☐ Association, not causation, language

Complete Regression Reporting Example

“We fit a logistic regression predicting KEV inclusion from CVSS base score and CWE category (dummy-coded, reference = Auth). Results are shown in Table 2.

The model significantly outperformed the null model (LR = 28.5, df = 5, $p < 0.001$), though McFadden’s pseudo-R^2 was modest (0.08). Higher CVSS was associated with increased exploitation odds (OR = 1.68 per point, 95% CI [1.34, 2.12]). CWE categories did not significantly differ from the Auth reference (all $p > 0.10$), though Memory showed a trend toward higher odds (OR = 2.07, 95% CI [0.85, 5.06]).

These associations do not imply causation; unmeasured confounders may explain the observed relationships.”

Lecture 2 Summary

Concept	Key Takeaway	Example Use
Linear regression	Model continuous outcome from predictors	CVSS ~ subscores
Dummy coding	Convert categorical to numeric	CWE category
Reference level	Coefficients are differences from reference	Specify explicitly
LR test	Compare nested models	Does CWE help?
Logistic regression	Model binary outcome via log-odds	Predict exploitation
Odds ratios	e^β — multiplicative effect	Interpret coefficients
Rare events	Accuracy misleading; use sensitivity	KEV prediction

Appendix: Quick Reference

Sample Data Summary

File: `sample_vuln_data.csv`

Variable	Type	Description
cve_id	string	CVE identifier
pub_year	int	Publication year (2018-2024)
cwe_category	categorical	Memory, InputValidation, Crypto, Auth, Other
cvss_base	numeric	CVSS score (0-10)
impact	numeric	Impact subscore
exploitability	numeric	Exploitability subscore
severity	ordered	Low < Medium < High < Critical
in_kev	boolean	TRUE if actively exploited

Python Function Reference

Task	Python Function
Chi-square test	`scipy.stats.chi2_contingency()`
t-test	`scipy.stats.ttest_ind()`
Mann-Whitney U	`scipy.stats.mannwhitneyu()`
Mann-Whitney + effect size	`pingouin.mwu()`
Cohen's d	`pingouin.compute_effsize()`
Linear regression	`statsmodels.formula.api.ols()`
Logistic regression	`statsmodels.formula.api.logit()`
LR test	`model.compare_lr_test()`
Multiple comparison correction	`statsmodels.stats.multitest.multipletests()`

Effect Size Interpretation Guide

Cohen's d (parametric):

d	Interpretation
0.2	Small
0.5	Medium
0.8	Large

Vargha-Delaney A (non-parametric):

A	Interpretation
0.56	Small
0.64	Medium
0.71	Large

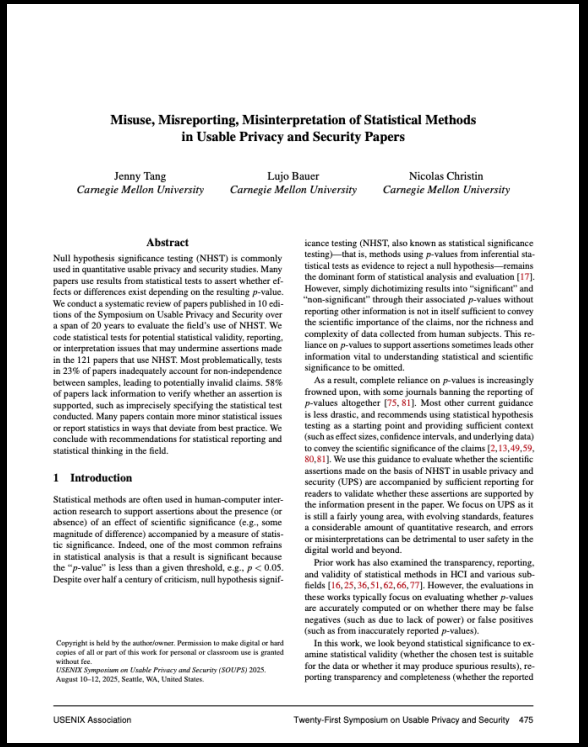
Recommended Readings

Primary Textbook (Franke):

- Section 16.2 — p-values
- Section 16.6.1 — Chi-square
- Section 12.1 — Linear regression
- Section 15.2 — Logistic regression
  (Seltman better)

Secondary Textbook (Seltman):

- Chapter 6.2 — Hypothesis testing
- Chapter 9 — Linear regression
- Chapter 16.2-16.3 — Chi-square and logistic regression



Misuse, Misreporting, Misinterpretation of Statistical Methods in Usable Privacy and Security Papers

Jenny Tang
Carnegie Mellon University

Lujo Bauer
Carnegie Mellon University

Nicolas Christin
Carnegie Mellon University

Abstract

Null hypothesis significance testing (NHST) is commonly used in quantitative usable privacy and security studies. Many papers use results from statistical tests to assert whether effects or differences exist depending on the resulting p-value. We conduct a systematic review of papers published in 10 editions of the Symposium on Usable Privacy and Security over a span of 20 years to evaluate the field's use of NHST. We code statistical tests for potential statistical validity, reporting, or interpretation issues that may undermine assertions made in the 121 papers that use NHST. Most problematically, tests in 23% of papers inadequately account for non-independence between samples, leading to potentially invalid claims. 58% of papers lack information to verify whether an assertion is supported, such as imprecisely specifying the statistical test conducted. Many papers contain more minor statistical issues or report statistics in ways that deviate from best practice. We conclude with recommendations for statistical reporting and statistical thinking in the field.

1 Introduction

Statistical methods are often used in human-computer interaction research to support assertions about the presence (or absence) of an effect of scientific significance (e.g., some magnitude of difference) accompanied by a measure of statistic significance. Indeed, one of the most common refrains in statistical analysis is that a result is significant because the "p-value" is less than a given threshold, e.g., $p < 0.05$. Despite over half a century of criticism, null hypothesis significance testing (NHST, also known as statistical significance testing)—that is, methods using p-values from inferential statistical tests as evidence to reject a null hypothesis—remains the dominant form of statistical analysis and evaluation [17]. However, simply dichotimizing results into "significant" and "non-significant" through their associated p-values without reporting other information is not in itself sufficient to convey the scientific importance of the claims, nor the richness and complexity of data collected from human subjects. This reliance on p-values to support assertions sometimes leads other information vital to understanding statistical and scientific significance to be omitted.

As a result, complete reliance on p-values is increasingly frowned upon, with some journals banning the reporting of p-values altogether [75, 81]. Most other current guidance is less drastic, and recommends using statistical hypothesis testing as a starting point and providing sufficient context (such as effect sizes, confidence intervals, and underlying data) to convey the scientific significance of the claims [2, 13, 49, 59, 80, 81]. We use this guidance to evaluate whether the scientific assertions made on the basis of NHST in usable privacy and security (UPS) are accompanied by sufficient reporting for readers to validate whether these assertions are supported by the information present in the paper. We focus on UPS as it is still a fairly young area, with evolving standards, features a considerable amount of quantitative research, and errors or misinterpretations can be detrimental to user safety in the digital world and beyond.

Prior work has also examined the transparency, reporting, and validity of statistical methods in HCI and various subfields [16, 25, 36, 51, 62, 66, 77]. However, the evaluations in these works typically focus on evaluating whether p-values are accurately computed or on whether there may be false negatives (such as due to lack of power) or false positives (such as from inaccurately reported p-values).

In this work, we look beyond statistical significance to examine statistical validity (whether the chosen test is suitable for the data or whether it may produce spurious results), reporting transparency and completeness (whether the reported

USENIX Symposium on Usable Privacy and Security (SOUPS) 2025.
August 10–12, 2025, Seattle, WA, United States.