Here's a print-friendly version of the Machine Learning Cheat Sheet:

Machine Learning Cheat Sheet

1. Key ML Algorithms

Supervised Learning - Classification

Algorithm	Use Cases	Key Parameters	Pros	Cons
Logistic Regression	Binary classification, Spam detection, Risk assessment	C, penalty, solver	Simple & fast, Interpretable, Good for linear data	Can't handle non-linear data, Assumes independence
Decision Trees	Multi-class classification, Feature importance, Non-linear data	max_depth, min_samples_split, criterion	Easy to visualize, Handles non-linear data, No scaling needed	Can overfit, Unstable, Biased to dominant classes
Random Forest	Complex classification, Ensemble learning, Feature selection	n_estimators, max_features, bootstrap	Reduces overfitting, Handles missing values, Feature importance	Black box model, Computationally heavy
SVM	High-dimensional data, Text classification, Image recognition	kernel, C, gamma	Works in high dimensions, Memory efficient, Versatile kernels	Sensitive to scaling, Slow training, Hard to interpret

Supervised Learning - Regression

Algorithm	Use Cases	Key Parameters	Pros	Cons
Linear Regression	Simple prediction, Baseline model, Feature importance	fit_intercept, normalize, n_jobs	Simple & interpretable, Fast training, Feature importance	Assumes linearity, Sensitive to outliers
Ridge (L2)	Multicollinearity, Continuous prediction, Feature selection	alpha, solver, normalize	Handles multicollinearity, Reduces overfitting, Stable solutions	Assumes linearity, Keeps all features
Lasso (L1)	Sparse solutions, Feature selection, Automated selection	alpha, selection, normalize	Feature selection, Sparse solutions, Handles high dimensions	Unstable with correlated features, Needs tuning

Unsupervised Learning

Algorithm	Use Cases	Key Parameters	Pros	Cons
K-Means	Clustering, Segmentation, Grouping	n_clusters, init, n_init	Simple & fast, Scalable, Easy to understand	Needs k value, Sensitive to outliers
DBSCAN	Density clustering, Noise detection, Variable shapes	eps, min_samples, metric	Finds any shape, Handles noise, No preset clusters	Sensitive to parameters, Struggles with varying densities
PCA	Dimension reduction, Feature extraction, Visualization	n_components, svd_solver, whiten	Reduces dimensions, Handles multicollinearity, Unsupervised	Linear assumptions, Loss of interpretability

2. Evaluation Metrics

Classification Metrics

Metric	Formula	When to Use	Implementation
Accuracy	(TP + TN)/(TP + TN + FP + FN)	Balanced datasets	metrics.accuracy_score()
Precision	TP/(TP + FP)	Minimize false positives	metrics.precision_score()
Recall	TP/(TP + FN)	Minimize false negatives	metrics.recall_score()
F1 Score	2×(P×R)/(P + R)	Balance precision/recall	metrics.f1_score()
ROC-AUC	Area under ROC curve	Binary classification	metrics.roc_auc_score()

Regression Metrics

Metric	Formula	When to Use	Implementation
MSE	$\Sigma(y\_true - y\_pred)^2/n$	General purpose	metrics.mean_squared_error()
RMSE	$\sqrt{(MSE)}$	Same units as target	np.sqrt(metrics.mean_squared_error())
MAE	$\Sigma\|y\_true - y\_pred\|/n$	Robust to outliers	metrics.mean_absolute_error()
R^2	1 - (MSE/Var(y))	Model fit quality	metrics.r2_score()

3. Essential Python Code Snippets

Data Loading & Preprocessing

```
import pandas as pd
df = pd.read_csv('data.csv')
df.dropna(inplace=True)
df = pd.get_dummies(df)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
te=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Training & Evaluation

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)
```

Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
params = {'C': [0.1, 1, 10], 'kernel': ['rbf', 'linear']}
grid = GridSearchCV(model, params, cv=5)
grid.fit(X_train, y_train)
print("Best params:", grid.best_params_)
```

4. Feature Engineering Techniques

Technique	Purpose	Implementation
Scaling	Normalize features	StandardScaler(), MinMaxScaler()
Encoding	Handle categories	LabelEncoder(), OneHotEncoder()
Selection	Reduce dimensions	SelectKBest(), RFE()
Creation	Make new features	PolynomialFeatures()
Binning	Group continuous data	pd.cut(), pd.qcut()

5. Common Errors & Solutions

Problem	Symptoms	Solutions
Overfitting	High train score, Low test score	More data, Regularization, Reduce complexity
Underfitting	Low train score, Low test score	More features, Less regularization, More complex model
Data Leakage	Unrealistic high scores	Proper CV splits, Feature scaling after split
Class Imbalance	High accuracy, low recall	SMOTE, Class weights, Stratification

6. Best Practices

1. Data Preprocessing:
    - Handle missing values first
    - Scale features appropriately
    - Check for class imbalance
    - Split data before scaling
2. Model Selection:
    - Start simple
    - Use cross-validation
    - Consider computational cost
    - Check assumptions
3. Model Evaluation:
    - Use multiple metrics
    - Check for overfitting
    - Consider business impact
    - Validate on holdout set
4. Production:

- Save preprocessing steps
- Version control models
- Monitor performance
- Plan for updates

7. Key Libraries

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

from sklearn import (
    preprocessing,
    model_selection,
    metrics,
    ensemble,
    linear_model,
    svm,
    tree
)
```

Remember:

- Start with simple models
- Always split data properly
- Use cross-validation
- Check assumptions
- Document everything
- Monitor performance

Citations: [1] https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/15813152/d6066d59-2144-4ba4-acb3-5816e9292679/paste.txt (https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/15813152/d6066d59-2144-4ba4-acb3-5816e9292679/paste.txt)

You're right. Let me reformat the SVM cheat sheet with proper table formatting where all content stays within columns. Here's the corrected version:

Support Vector Machine (SVM) Cheat Sheet

1. Types of SVM

Type	Description	Use Cases	Key Parameters
Linear SVM	Uses linear hyperplane for separation; Maximizes margin between classes	Text classification; High dimensional data; Linear separable data	C: regularization strength; max_iter: iterations; tol: tolerance
Non-linear SVM	Uses kernel trick; Transforms data to higher dimensions	Image classification; Complex patterns; Non-linear data	kernel: kernel type; C: regularization; gamma: coefficient
SVM Regression	Predicts continuous values; Uses epsilon-tube	Price prediction; Time series; Continuous data	epsilon: margin width; C: regularization; kernel: type

2. Kernel Types

Kernel	Formula	Use Case	Parameters
Linear	K(x,y) = x^T y	High dimensional data; Text classification; Simple datasets	None needed
RBF (Gaussian)	K(x,y) = exp(-γ‖x-y‖²)	Non-linear data; Image processing; General purpose	gamma: kernel coefficient
Polynomial	K(x,y) = (γx^T y + r)^d	Image processing; Natural language; Feature interactions	degree: polynomial degree; gamma: scale; coef0: constant
Sigmoid	K(x,y) = tanh(γx^T y + r)	Neural network alternative; Binary classification	gamma: scale; coef0: constant

3. Important Parameters

Parameter	Purpose	Typical Values	Effect
C	Controls regularization strength	0.1 to 100; Default: 1.0	Large C: Less regularization; Small C: More regularization
gamma	Controls influence range	scale, auto, 0.001 to 1	Large: Close influence; Small: Far influence
kernel	Defines transformation type	rbf, linear, poly, sigmoid	Changes data transformation; Affects complexity
degree	Sets polynomial complexity	2 to 5; Default: 3	Higher: More complex; Lower: Simpler

4. Advantages and Disadvantages

Advantages	Disadvantages
Effective in high dimensions; Memory efficient; Versatile kernels; Robust to overfitting	Sensitive to scaling; Slow on large datasets; Needs parameter tuning; Black box model

5. Best Practices

Area	Recommendations
Data Preparation	Scale all features; Handle missing values; Remove outliers; Convert categorical data
Kernel Selection	Start linear; Try RBF for non-linear; Use polynomial for interactions; Cross-validate
Parameter Tuning	Grid search C and gamma; Start with defaults; Use logarithmic scales; Monitor time
Optimization	Use approximation for large data; Consider feature selection; Monitor support vectors

6. Common Issues and Solutions

Issue	Symptoms	Solutions
Overfitting	High train, low test scores	Decrease C; Use simpler kernel; Add regularization
Underfitting	Low train and test scores	Increase C; Try different kernel; Add features
Slow Training	Long convergence time	Use linear kernel; Reduce dataset; Select features
Poor Performance	Low accuracy, unstable	Scale features; Change kernel; Tune parameters

7. Basic Implementation

```python
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

# Preprocessing
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Basic SVM
svm = SVC(kernel='rbf', C=1.0, gamma='scale')
svm.fit(X_scaled, y)

# Grid Search
params = {
    'C': [0.1, 1, 10],
    'gamma': ['scale', 'auto', 0.1],
    'kernel': ['rbf', 'linear']
}
grid_search = GridSearchCV(SVC(), params, cv=5)
```

Key Points to Remember:

1. Always scale features
2. Start with simple models
3. Use cross-validation
4. Monitor training time
5. Check support vectors

Would you like me to adjust the formatting of any particular section or add more details to any topic?

Complete Naive Bayes Cheat Sheet

1. Basic Concepts

Term	Symbol	Meaning
Posterior	P(A|B)	Probability of A given B
Likelihood	P(B|A)	Probability of B given A
Prior	P(A)	Initial probability of A
Product	∏	Multiply sequence of terms
Mean	μ	Average of distribution
Std Dev	σ	Spread of distribution

2. Types of Naive Bayes

Type	Data Type	Best For	Example Data
Gaussian	Continuous	Physical measurements	Height: 175.5 cm
Multinomial	Count data	Text classification	Word appears 3 times
Bernoulli	Binary data	Presence/absence	Word exists: yes/no

3. Formulas and Smoothing

A. Gaussian NB

```
P(x|class) = 1/(√(2πσ²)) × e^(-(x-μ)²/2σ²)

Where:
x = feature value
μ = class mean
σ = class standard deviation
```

B. Multinomial NB

```
With vocabulary smoothing:
P(word|class) = (count + α)/(total + α|V|)

Where:
count = word occurrences in class
total = all words in class
|V| = vocabulary size
α = smoothing parameter
```

C. Bernoulli NB

```
With class smoothing:
P(word|class) = (count + α)/(total + αk)

Where:
count = documents with word in class
total = documents in class
k = number of classes
α = smoothing parameter
```

4. Practical Examples

A. Gaussian Example (Height Classification)

```
Given:
Male: μ = 175cm, σ = 10
Female: μ = 162cm, σ = 8
New height = 168cm

P(height|male) = 1/(√(2π×10²)) × e^(-(168-175)²/(2×10²))
                = 0.0312

P(height|female) = 1/(√(2π×8²)) × e^(-(168-162)²/(2×8²))
                  = 0.0376

Result: Classify as Female (0.0376 > 0.0312)
```

B. Multinomial Example (Text Classification)

```
Given:
- Word 'money' appears 20 times in spam
- Total spam emails: 100
- Vocabulary size: 1000
- α = 1

P(money|spam) = (20 + 1)/(100 + 1×1000)
               = 21/1100
               ≈ 0.019
```

C. Bernoulli Example (Spam Detection)

```
Given:
- Word 'money' appears in 20 spam emails
- Total spam emails: 100
- Number of classes: 2
- α = 1

P(money|spam) = (20 + 1)/(100 + 1×2)
               = 21/102
               ≈ 0.206
```

5. Implementation in Python

```python
# Gaussian NB
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Multinomial NB
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB(alpha=1.0)
mnb.fit(X_train, y_train)

# Bernoulli NB
from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB(alpha=1.0)
bnb.fit(X_train, y_train)
```

6. When to Use Each Variant

Variant	Use When	Don't Use When
Gaussian	Features are continuous	Data is discrete
Multinomial	Working with word counts	Features are binary
Bernoulli	Features are binary	Need to count occurrences

7. Problem-Solving Steps

1. Identify data type:
   - Continuous → Gaussian
   - Count data → Multinomial
   - Binary data → Bernoulli
2. Check assumptions:
   - Feature independence
   - Distribution assumptions
   - Data quality

3. Preprocess data:
   - Handle missing values
   - Scale if needed (Gaussian)
   - Convert to appropriate format
4. Choose smoothing:
   - Multinomial: vocabulary size
   - Bernoulli: number of classes
   - Set α value (typically 1)
5. Calculate probabilities:
   - Use log for numerical stability
   - Apply appropriate formula
   - Compare results

8. Common Issues and Solutions

Issue	Solution
Zero probabilities	Apply Laplace smoothing
Numerical underflow	Use log probabilities
Feature scaling	Standardize for Gaussian
Class imbalance	Adjust prior probabilities

Remember:

- Always scale features for Gaussian NB
- Use log probabilities for stability
- Consider class balance
- Validate independence assumption
- Choose appropriate smoothing

Naive Bayes Detailed Problems and Solutions

1. Gaussian Naive Bayes Problem

Problem: Classify students as Pass/Fail based on study hours and sleep hours.

Given Data:

```
Training Data:
Pass students:
- Study hours: μ = 8, σ = 1
- Sleep hours: μ = 6, σ = 0.5

Fail students:
- Study hours: μ = 4, σ = 1.5
- Sleep hours: μ = 8, σ = 1

Prior probabilities:
P(Pass) = 0.6
P(Fail) = 0.4

New student:
- Study hours = 7
- Sleep hours = 7
```

Solution:

```
1. Calculate P(features|Pass):
   P(study=7|Pass) = 1/(√(2π×1²)) × e^(-(7-8)²/(2×1²))
                   = 0.242

   P(sleep=7|Pass) = 1/(√(2π×0.5²)) × e^(-(7-6)²/(2×0.5²))
                   = 0.107

2. Calculate P(features|Fail):
   P(study=7|Fail) = 1/(√(2π×1.5²)) × e^(-(7-4)²/(2×1.5²))
                   = 0.027

   P(sleep=7|Fail) = 1/(√(2π×1²)) × e^(-(7-8)²/(2×1²))
                   = 0.242

3. Final probabilities:
   P(Pass|features) ∝ 0.6 × 0.242 × 0.107 = 0.0155
   P(Fail|features) ∝ 0.4 × 0.027 × 0.242 = 0.0026

Result: Student likely to Pass (0.0155 > 0.0026)
```

2. Multinomial Naive Bayes Problem

Problem: Classify email as Spam/Not Spam based on word frequencies.

Given Data:

Training Data:
Total emails:
- Spam: 100 emails
- Not Spam: 200 emails

Word frequencies in Spam:
- 'money': 50 occurrences
- 'win': 40 occurrences
- 'free': 60 occurrences

Word frequencies in Not Spam:
- 'money': 10 occurrences
- 'win': 5 occurrences
- 'free': 15 occurrences

Vocabulary size = 1000 words
α = 1 (Laplace smoothing)

New email contains: "free money money"

Solution:

1. Calculate priors:
   P(Spam) = 100/300 = 0.333
   P(Not Spam) = 200/300 = 0.667

2. Calculate P(word|Spam) with vocabulary smoothing:
   P(money|Spam) = (50 + 1)/(150 + 1000) = 0.0444
   P(free|Spam) = (60 + 1)/(150 + 1000) = 0.0530

3. Calculate P(word|Not Spam):
   P(money|Not Spam) = (10 + 1)/(30 + 1000) = 0.0107
   P(free|Not Spam) = (15 + 1)/(30 + 1000) = 0.0155

4. Final calculation:
   $P(\text{Spam}|\text{email}) \propto 0.333 \times 0.0444^2 \times 0.0530 = 3.46 \times 10^{-5}$
   $P(\text{Not Spam}|\text{email}) \propto 0.667 \times 0.0107^2 \times 0.0155 = 1.19 \times 10^{-6}$

Result: Classify as Spam ($3.46 \times 10^{-5} > 1.19 \times 10^{-6}$)

3. Bernoulli Naive Bayes Problem

Problem: Classify document based on presence/absence of keywords.

Given Data:

```
Training Data:
Documents:
- Technical: 150 documents
- Non-Technical: 250 documents

Word presence in Technical docs:
- 'code': 120 documents
- 'data': 100 documents
- 'algorithm': 90 documents

Word presence in Non-Technical docs:
- 'code': 20 documents
- 'data': 50 documents
- 'algorithm': 10 documents

α = 1 (Laplace smoothing)
Number of classes (k) = 2

New document contains: 'code' and 'data' (but no 'algorithm')
```

Solution:

```
1. Calculate priors:
   P(Technical) = 150/400 = 0.375
   P(Non-Technical) = 250/400 = 0.625

2. Calculate P(word|Technical) with class smoothing:
   P(code|Tech) = (120 + 1)/(150 + 2) = 0.7894
   P(data|Tech) = (100 + 1)/(150 + 2) = 0.6645
   P(¬algorithm|Tech) = 1 - (90 + 1)/(150 + 2) = 0.4013

3. Calculate P(word|Non-Technical):
   P(code|Non-Tech) = (20 + 1)/(250 + 2) = 0.0833
   P(data|Non-Tech) = (50 + 1)/(250 + 2) = 0.2024
   P(¬algorithm|Non-Tech) = 1 - (10 + 1)/(250 + 2) = 0.9562

4. Final calculation:
   P(Tech|doc) ∝ 0.375 × 0.7894 × 0.6645 × 0.4013 = 0.0791
   P(Non-Tech|doc) ∝ 0.625 × 0.0833 × 0.2024 × 0.9562 = 0.0101

Result: Classify as Technical (0.0791 > 0.0101)
```

Key Points to Remember:

1. Gaussian NB:
   - Use for continuous data
   - Calculate mean and standard deviation
   - Apply Gaussian formula
2. Multinomial NB:

- Use for word frequencies
- Apply vocabulary smoothing
- Count total occurrences

3. Bernoulli NB:

    - Use for presence/absence
    - Apply class smoothing
    - Consider both presence and absence

Common Steps for All:

1. Calculate priors
2. Apply appropriate smoothing
3. Calculate conditional probabilities
4. Multiply probabilities (or add logs)
5. Compare final values

Complete Statistical Tests Guide

Common Acronyms and Terms

Acronym/Term	Full Form	Meaning
ANOVA	Analysis of Variance	Statistical method to analyze differences among group means
SS	Sum of Squares	Measure of variation from the mean
SST	Total Sum of Squares	Total variation in the data
SSB/SSA	Between Groups Sum of Squares	Variation between different groups
SSW/SSE	Within Groups Sum of Squares/Error	Variation within groups
df	Degrees of Freedom	Number of values free to vary
MS	Mean Square	Sum of squares divided by degrees of freedom
SE	Standard Error	Standard deviation of a sampling distribution
H_0	Null Hypothesis	Statement of no effect or difference
H_1	Alternative Hypothesis	Statement of effect or difference
α	Alpha	Significance level (Type I error rate)
μ	Mu	Population mean
σ	Sigma	Population standard deviation
$\bar{x}$	x-bar	Sample mean
s	s	Sample standard deviation

Test Selection Guide

Test	When to Use	Required Assumptions	Example Scenario
One-way ANOVA	Compare means of 3+ groups	Normal distribution, Equal variances	Compare multiple teaching methods
Two-way ANOVA	Compare effects of 2 factors	Normal distribution, Equal variances	Effect of gender & teaching method
F-test	Compare variances	Normal distribution	Compare method variabilities
t-test	Compare means of 2 groups	Normal distribution	Compare control vs treatment
z-test	Compare with known population	Known population σ, Large sample	Compare to population mean

1. One-Way ANOVA

Core Formulas

- SST (Total) = $\Sigma(x - \bar{x})^2$
- SSB (Between) = $\Sigma n_i(\bar{x}_i - \bar{x})^2$
- SSW (Within) = SST - SSB
- F = (SSB/dfb)/(SSW/dfw)
- dfb = k - 1, dfw = N - k

Problem Example

Compare three teaching methods:
Method A: 75, 82, 78, 85, 81
Method B: 65, 71, 68, 73, 70
Method C: 85, 88, 90, 87, 86
$\alpha = 0.05$

Detailed Solution Steps

1. Calculate Group Means:

$$\text{Method A: } \bar{x}A = (75 + 82 + 78 + 85 + 81)/5 = 80.2$$
$$\text{Method B: } \bar{x}B = (65 + 71 + 68 + 73 + 70)/5 = 69.4$$
$$\text{Method C: } \bar{x}C = (85 + 88 + 90 + 87 + 86)/5 = 87.2$$
$$\text{Grand Mean: } \bar{x} = (80.2 + 69.4 + 87.2)/3 = 78.93$$

2. Calculate SSB:

$$\begin{aligned} SSB &= \Sigma n_i(\bar{x}_i - \bar{x})^2 \\ &= 5(80.2 - 78.93)^2 + 5(69.4 - 78.93)^2 + 5(87.2 - 78.93)^2 \\ &= 5(1.27^2 + (-9.53)^2 + 8.27^2) \\ &= 5(1.61 + 90.82 + 68.39) \\ &= 804.31 \end{aligned}$$

3. Calculate SST:

$$\begin{aligned} SST &= \Sigma(x - \bar{x})^2 \\ &= (75 - 78.93)^2 + (82 - 78.93)^2 + ... + (86 - 78.93)^2 \\ &= 894.11 \end{aligned}$$

4. Calculate SSW:

$$SSW = SST - SSB = 894.11 - 804.31 = 89.8$$

5. Calculate Degrees of Freedom:

$$dfb = k - 1 = 3 - 1 = 2$$
$$dfw = N - k = 15 - 3 = 12$$

6. Calculate Mean Squares:

$$MSB = SSB/dfb = 804.31/2 = 402.16$$
$$MSW = SSW/dfw = 89.8/12 = 7.48$$

7. Calculate F-statistic:

$$F = MSB/MSW = 402.16/7.48 = 53.76$$
$$F\text{-critical}(0.05,2,12) = 3.89$$

Conclusion

Since F(53.76) > F-critical(3.89), reject H_0. Teaching methods have significantly different effects on performance.

2. Two-Way ANOVA

Core Formulas

- SST = SSA + SSB + SS(AB) + SSE
- FA = MSA/MSE
- FB = MSB/MSE
- FAB = MSAB/MSE

Problem Example

```
Effect of Gender and Teaching Method:
                Traditional     Online
Male:           72,75,71       65,68,63
Female:         78,82,80       70,73,71
α = 0.05
```

Detailed Solution Steps

1. Calculate Cell Means:

```
Male Traditional (MT): x̄MT = (72+75+71)/3 = 72.67
Male Online (MO): x̄MO = (65+68+63)/3 = 65.33
Female Traditional (FT): x̄FT = (78+82+80)/3 = 80.00
Female Online (FO): x̄FO = (70+73+71)/3 = 71.33
```

2. Calculate Main Effect Means:

```
Males: x̄M = (72.67+65.33)/2 = 69.00
Females: x̄F = (80.00+71.33)/2 = 75.67
Traditional: x̄T = (72.67+80.00)/2 = 76.33
Online: x̄O = (65.33+71.33)/2 = 68.33
Grand Mean: x̄ = (69.00+75.67)/2 = 72.33
```

3. Calculate Sum of Squares:

```
SSG (Gender) = 135.37
SSM (Method) = 192.67
SSI (Interaction) = 4.17
SSE (Error) = 313.12
SST = 645.33
```

4. Calculate F-ratios:

```
F_Gender = MSG/MSE = 135.37/39.14 = 3.46
F_Method = MSM/MSE = 192.67/39.14 = 4.92
F_Interaction = MSI/MSE = 4.17/39.14 = 0.11
F-critical(0.05,1,8) = 5.32
```

Conclusions

1. Gender Effect ($F = 3.46 < 5.32$): Not significant
2. Method Effect ($F = 4.92 < 5.32$): Not significant
3. Interaction ($F = 0.11 < 5.32$): No significant interaction

3. F-Test

Core Formula

$F = s_1^2/s_2^2$ (larger variance/smaller variance)

Problem Example

```
Compare machine variances:
Machine 1: 10.2, 10.4, 10.1, 10.3, 10.2
Machine 2: 10.3, 10.1, 10.4, 10.2, 10.3
α = 0.05
```

Detailed Solution Steps

1. Calculate Means:

```
x̄₁ = (10.2 + 10.4 + 10.1 + 10.3 + 10.2)/5 = 10.24
x̄₂ = (10.3 + 10.1 + 10.4 + 10.2 + 10.3)/5 = 10.26
```

2. Calculate Variances:

```
s₁² = [(10.2-10.24)² + ... + (10.2-10.24)²]/4 = 0.0130
s₂² = [(10.3-10.26)² + ... + (10.3-10.26)²]/4 = 0.0115
```

3. Calculate F-statistic:

```
F = 0.0130/0.0115 = 1.13
F-critical(0.05,4,4) = 6.39
```

Conclusion

Since $F(1.13) <$ F-critical(6.39), cannot reject H_0. No significant difference in variances.

4. t-Test

Core Formula

$t = (\bar{x}_1 - \bar{x}_2)/\sqrt{s^2p(1/n_1 + 1/n_2)}$ where $s^2p = [(n_1-1)s_1^2 + (n_2-1)s_2^2]/(n_1+n_2-2)$

Problem Example

```
Compare treatments:
Control: 68, 72, 70, 71, 65
Treatment: 75, 82, 78, 80, 76
α = 0.05
```

Detailed Solution Steps

1. Calculate Means:

```
Control: x̄₁ = (68 + 72 + 70 + 71 + 65)/5 = 69.2
Treatment: x̄₂ = (75 + 82 + 78 + 80 + 76)/5 = 78.2
```

2. Calculate Sample Variances:

```
s₁² = [(68-69.2)² + ... + (65-69.2)²]/4 = 7.7
s₂² = [(75-78.2)² + ... + (76-78.2)²]/4 = 8.7
```

3. Calculate Pooled Variance:

```
s²p = [(4×7.7) + (4×8.7)]/8 = 8.2
```

4. Calculate t-statistic:

```
t = (69.2 - 78.2)/√[8.2(2/5)] = -4.97
t-critical(0.05,8) = ±2.306
```

Conclusion

Since $|t|$ > t-critical, reject H_0. Treatment has significant effect.

5. z-Test

Core Formula

$z = (\bar{x} - \mu)/(\sigma/\sqrt{n})$

Problem Example

```
Population: μ = 100, σ = 15
Sample (n=36): mean = 96
α = 0.05
```

Detailed Solution Steps

1. Calculate Standard Error:

```
SE = σ/√n = 15/√36 = 2.5
```

2. Calculate z-statistic:

```
z = (96 - 100)/2.5 = -1.6
z-critical(0.05) = ±1.96
```

Conclusion

Since $|z|$ < z-critical, cannot reject H_0. Sample mean not significantly different from population mean.

Key Points to Remember

1. Test Selection:
   - $n \geq 30$: Consider z-test
   - Compare 2 groups: t-test
   - Compare 3+ groups: ANOVA
   - Compare variances: F-test
2. Critical Values:
   - $\alpha = 0.05$ (common)
   - Two-tailed vs One-tailed
   - Consider degrees of freedom
3. Assumptions:
   - Normality
   - Equal variances (when applicable)
   - Independence
   - Random sampling
4. Decision Rules:
   - If test statistic > critical value: Reject H_0
   - If p-value < α: Reject H_0
   - Consider practical significance
5. Effect Size Measures:
   - ANOVA: $\eta^2 = SSB/SST$
   - t-test: Cohen's $d = (\bar{x}_1 - \bar{x}_2)/s\_pooled$
   - z-test: $d = (\bar{x} - \mu)/\sigma$