Product Requirements Document: Multiple Linear Regression R^2 and Adjusted R^2 Analysis with Multicollinearity Comparison

1. Executive Summary

This document outlines the requirements for developing a Python application that compares R^2 and Adjusted R^2 metrics across two regression models—one with independent predictors and another with multicollinear (dependent) predictors—to demonstrate the importance of Adjusted R^2 in model evaluation and the effects of multicollinearity.

2. Product Overview

2.1 Purpose

Create an educational and analytical tool that demonstrates:

- The difference between R^2 and Adjusted R^2
- How multicollinearity affects model evaluation metrics
- Why Adjusted R^2 is superior for comparing models with different predictor counts
- The effect of fixed noise (systematic bias) on both metrics
- The penalty mechanism in Adjusted R^2

2.2 Target Users

- Statistics and data science students learning regression metrics
- Educators teaching model evaluation and multicollinearity
- Data analysts comparing multiple models
- Machine learning practitioners learning overfitting prevention
- Researchers understanding metric limitations

2.3 Product Author

Yair Levi

2.4 Key Innovation

Four-line comparative visualization showing R^2 and Adjusted R^2 for both independent and multicollinear models simultaneously, with explicit demonstration of how Adjusted R^2 penalizes unnecessary predictors and detects multicollinearity.

3. Functional Requirements

3.1 Model Specifications

3.1.1 Original Model (Independent Predictors)

- **FR-001**: The application SHALL create a regression model with exactly **50 independent predictors**
- **FR-002**: Predictors SHALL follow Normal(μ=0, σ=1) distribution
- **FR-003**: All predictors SHALL be statistically independent
- **FR-004**: Model equation: $\boxed{Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{50} x_{50} + \varepsilon}$

3.1.2 Extended Model (Multicollinear Predictors)

- **FR-005**: The application SHALL create an extended model with **55 predictors total**
- **FR-006**: Extended model SHALL include all 50 original predictors
- **FR-007**: Extended model SHALL add exactly **5 dependent predictors**
- **FR-008**: Dependent predictors SHALL be linear combinations of original predictors
- **FR-009**: Each dependent predictor SHALL combine 2-3 randomly selected original predictors
- **FR-010**: Weights for combinations SHALL be random uniform [-1, 1]
- **FR-011**: Small noise (σ=0.1) SHALL be added to avoid perfect collinearity
- **FR-012**: Model equation: $\boxed{Y = \beta_0 + \beta_1 x_1 + ... + \beta_{50} x_{50} + \beta_{51} x_{51} + ... + \beta_{55} x_{55} + \varepsilon}$

3.1.3 Common Specifications

- **FR-013**: Both models SHALL use exactly **100 data points** (observations)
- **FR-014**: Both models SHALL use the same random seed for reproducibility
- **FR-015**: Both models SHALL be tested across **20 fixed noise values** (epsilon)

3.2 Coefficient Generation Requirements

3.2.1 Intercept Coefficient (β_0)

- **FR-016**: β_0 SHALL be randomly selected from uniform distribution [-0.5, 0.5]
- **FR-017**: Same β_0 SHALL be used for both models

3.2.2 Original Predictor Coefficients (β_1 to β_{50})

- **FR-018**: Each β_i (i=1 to 50) SHALL be randomly selected from uniform [-0.9, 0.9]
- **FR-019**: Same coefficients SHALL be used in both models

3.2.3 Dependent Predictor Coefficients (β_{51} to β_{55})

- **FR-020**: Each β_{51} to β_{55} SHALL be randomly selected from uniform [-0.9, 0.9]
- **FR-021**: These coefficients SHALL only apply to extended model
- **FR-022**: Different random seed SHALL be used for these coefficients

3.3 Data Generation Requirements

3.3.1 Original Predictor Matrix

- **FR-023**: Generate X matrix with shape (100, 50)
- **FR-024**: Each element ~ Normal(0, 1)
- **FR-025**: All columns (predictors) statistically independent

3.3.2 Dependent Predictor Generation

- **FR-026**: Function SHALL be named `add_dependent_predictors()`
- **FR-027**: Input: Original X matrix (100, 50)
- **FR-028**: Output: Extended X matrix (100, 55)
- **FR-029**: For each dependent predictor:
  - Randomly select 2-3 original predictors
  - Create weighted sum with random weights
  - Add small noise (σ=0.1)
- **FR-030**: Dependent predictors SHALL create multicollinearity

3.3.3 Epsilon (Fixed Noise) Generation

- **FR-031**: Generate exactly 20 epsilon values
- **FR-032**: Values SHALL be uniformly distributed between -3.5 and 3.5
- **FR-033**: Use `np.linspace(-3.5, 3.5, 20)` for even spacing
- **FR-034**: Each epsilon represents a fixed bias added to all predictions

3.4 R^2 Calculation Requirements

3.4.1 Standard R^2 Calculation

- **FR-035**: Function SHALL be named `calculate_r_squared()`
- **FR-036**: Formula: `R² = 1 - (SS_res / SS_tot)`
- **FR-037**: SS_res SHALL be calculated using dot product: `np.dot(residuals, residuals)`

- **FR-038**: SS_tot SHALL be calculated using dot product: `np.dot(deviations, deviations)`
- **FR-039**: Residuals = Y_observed - Y_predicted
- **FR-040**: Deviations = Y_observed - mean(Y_observed)
- **FR-041**: Handle edge case: if SS_tot = 0, return $R^2 = 1.0$

3.4.2 R^2 Properties to Demonstrate

- **FR-042**: R^2 always increases or stays same when adding predictors
- **FR-043**: R^2 does not account for model complexity
- **FR-044**: R^2 can be misleading when comparing models
- **FR-045**: Range: typically [0, 1], but can be negative with very poor fit

3.5 Adjusted R^2 Calculation Requirements

3.5.1 Adjusted R^2 Implementation

- **FR-046**: Function SHALL be named `calculate_adjusted_r_squared()`
- **FR-047**: Formula: `Adj R² = 1 - [(1 - R²) × (n - 1) / (n - p - 1)]`
- **FR-048**: Parameters:
  - n = number of samples (100)
  - p = number of predictors (50 or 55)
  - R^2 = standard R^2 value
- **FR-049**: Adjustment factor = (n - 1) / (n - p - 1)
- **FR-050**: First calculate standard R^2 using `calculate_r_squared()`
- **FR-051**: Then apply adjustment formula

3.5.2 Adjusted R^2 Properties to Demonstrate

- **FR-052**: Adjusted R^2 penalizes for adding predictors
- **FR-053**: Can decrease when adding unhelpful predictors
- **FR-054**: Accounts for model complexity
- **FR-055**: Better for comparing models with different predictor counts
- **FR-056**: Can be negative (indicates very poor model)

3.5.3 Penalty Calculation

- **FR-057**: Calculate penalty = R^2 - Adjusted R^2

- **FR-058**: Original model penalty ≈ $R^2 \times (50/49)$
- **FR-059**: Extended model penalty ≈ $R^2 \times (55/44)$
- **FR-060**: Demonstrate that extended model has larger penalty
- **FR-061**: Display penalty for both models in output

3.6 Y Calculation Requirements Using Dot Product

3.6.1 Prediction Calculation

- **FR-062**: Create augmented design matrix: $[1, x_1, x_2, ..., x_p]$
- **FR-063**: Calculate Y_linear using dot product: `np.dot(X_augmented, coefficients)`
- **FR-064**: Add fixed epsilon: Y = Y_linear + ε
- **FR-065**: Return both Y (with noise) and Y_linear (without noise)

3.6.2 Calculation for Both Models

- **FR-066**: Calculate Y for original model (50 predictors)
- **FR-067**: Calculate Y for extended model (55 predictors)
- **FR-068**: Use SAME epsilon values for both models
- **FR-069**: Calculate R^2 and Adjusted R^2 for both models at each epsilon

3.7 Comparative Analysis Requirements

3.7.1 Metrics to Calculate

For each model and each epsilon value, calculate:

- **FR-070**: Standard R^2
- **FR-071**: Adjusted R^2
- **FR-072**: Penalty (R^2 - Adjusted R^2)
- **FR-073**: Total: 4 metric arrays, each with 20 values

3.7.2 Statistical Comparisons

- **FR-074**: Calculate mean, min, max for each metric array
- **FR-075**: Compare R^2 between models
- **FR-076**: Compare Adjusted R^2 between models
- **FR-077**: Compare penalties between models
- **FR-078**: Identify epsilon value with maximum R^2 difference

- **FR-079**: Identify epsilon value with maximum Adjusted R^2 difference

3.7.3 Key Comparisons at $\varepsilon \approx 0$

- **FR-080**: Display all 4 metrics at epsilon closest to 0
- **FR-081**: Show R^2 difference between models
- **FR-082**: Show Adjusted R^2 difference between models
- **FR-083**: Compare penalty magnitudes
- **FR-084**: Provide interpretation of differences

3.8 Visualization Requirements

3.8.1 Four-Line Graph Specifications

- **FR-085**: Create single figure with exactly 4 lines
- **FR-086**: Figure size: (16, 9) for clarity
- **FR-087**: All 4 lines SHALL be distinguishable

3.8.2 Line Specifications

Blue Lines (Original Model - 50 Predictors):

- **FR-088**: R^2 line: Solid, circles (○), lightblue fill, navy edge
- **FR-089**: Adjusted R^2 line: Dashed, triangles (△), cyan fill, navy edge

Green Lines (Extended Model - 55 Predictors):

- **FR-090**: R^2 line: Solid, squares (□), lightgreen fill, darkgreen edge
- **FR-091**: Adjusted R^2 line: Dashed, diamonds (◇), lime fill, darkgreen edge

3.8.3 Reference Lines

- **FR-092**: Horizontal line at $R^2 = 1.0$ (perfect fit) - red dashed
- **FR-093**: Horizontal line at $R^2 = 0.5$ - orange dashed
- **FR-094**: Horizontal line at $R^2 = 0.0$ (no fit) - gray dashed
- **FR-095**: Vertical line at $\varepsilon = 0$ (no noise) - purple dotted

3.8.4 Annotation Requirements

- **FR-096**: Yellow annotation box showing:
  - R^2 values for both models at $\varepsilon \approx 0$

  - Adjusted R^2 values for both models at $\varepsilon \approx 0$
  - R^2 difference between models
  - Adjusted R^2 difference between models
- **FR-097**: Light blue explanation box showing:
  - Line style legend (solid = R^2, dashed = Adjusted R^2)
  - Color legend (blue = original, green = multicollinear)
  - Key insight about penalty
- **FR-098**: Position annotations to avoid overlapping lines

3.8.5 Labels and Legend

- **FR-099**: X-axis label: "Epsilon (Fixed Noise Value)"
- **FR-100**: Y-axis label: "R^2 / Adjusted R^2 (Coefficient of Determination)"
- **FR-101**: Title: "R^2 and Adjusted R^2 Comparison: Independent vs Multicollinear Models"
- **FR-102**: Subtitle: "Effect of Dependent Predictors on Model Performance Metrics"
- **FR-103**: Legend: Two-column layout, fontsize 10
- **FR-104**: Grid: Enabled with alpha=0.3

3.8.6 Axis Limits

- **FR-105**: Y-axis: [-0.1, 1.1] to show full range including potential negative values
- **FR-106**: X-axis: Add padding of 0.2 on each side of epsilon range

3.9 Console Output Requirements

3.9.1 Header Section

- **FR-107**: Display application title
- **FR-108**: Display: "Comparison: Independent vs Multicollinear Predictors"
- **FR-109**: Display author name: "Yair Levi"

3.9.2 Configuration Display

- **FR-110**: Print original predictors count (50)
- **FR-111**: Print dependent predictors added (5)
- **FR-112**: Print total predictors for extended model (55)
- **FR-113**: Print number of samples (100)

- **FR-114**: Print number of epsilon values (20)
- **FR-115**: Print epsilon range [-3.5, 3.5]
- **FR-116**: Print random seed (42)

3.9.3 Original Model Results

- **FR-117**: Section header: "ORIGINAL MODEL (50 independent predictors)"
- **FR-118**: Display R^2 statistics: mean, min, max
- **FR-119**: Display Adjusted R^2 statistics: mean, min, max

3.9.4 Extended Model Results

- **FR-120**: Section header: "EXTENDED MODEL (55 predictors with multicollinearity)"
- **FR-121**: Display R^2 statistics: mean, min, max
- **FR-122**: Display Adjusted R^2 statistics: mean, min, max

3.9.5 Comparison Analysis Output

- **FR-123**: Section header: "COMPARISON ANALYSIS"
- **FR-124**: Display metrics at $\varepsilon \approx 0$ for both models
- **FR-125**: Show R^2 vs Adjusted R^2 difference for each model (penalty)
- **FR-126**: Show between-model R^2 difference
- **FR-127**: Show between-model Adjusted R^2 difference

3.9.6 Key Findings Section

- **FR-128**: Display penalty for original model
- **FR-129**: Display penalty for extended model
- **FR-130**: Compare penalties and explain significance
- **FR-131**: State whether multicollinear model shows R^2 inflation
- **FR-132**: Explain Adjusted R^2 correction mechanism
- **FR-133**: Provide actionable insights about model selection

4. Technical Requirements

4.1 Programming Language and Libraries

- **TR-001**: Python 3.6 or higher
- **TR-002**: NumPy >= 1.19.0 for numerical operations

- **TR-003**: Matplotlib >= 3.2.0 for visualization
- **TR-004**: No other external dependencies required

4.2 Code Structure Requirements

- **TR-005**: Modular functions with single responsibilities
- **TR-006**: All functions SHALL have comprehensive docstrings
- **TR-007**: Follow PEP 8 style guidelines
- **TR-008**: Use meaningful variable names matching mathematical notation

4.3 Required Functions

```python
generate_coefficients(num_predictors, beta_0_range, beta_i_range, seed)
generate_x_data(num_samples, num_predictors, mu, sigma, seed)
add_dependent_predictors(X, num_dependent, seed)
generate_epsilon_values(num_epsilon, epsilon_min, epsilon_max)
calculate_y_with_fixed_epsilon(X, coefficients, epsilon_value)
calculate_r_squared(Y_observed, Y_predicted)
calculate_adjusted_r_squared(Y_observed, Y_predicted, n_samples, n_predictors)
plot_r_squared_comparison(epsilon_values, r2_orig, r2_dep, adj_r2_orig, adj_r2_dep)
main()
```

4.4 Dot Product Requirements

- **TR-009**: ALL matrix-vector multiplications SHALL use `np.dot()`
- **TR-010**: SS_res calculation SHALL use `np.dot(residuals, residuals)`
- **TR-011**: SS_tot calculation SHALL use `np.dot(deviations, deviations)`
- **TR-012**: NO explicit Python loops in numerical calculations
- **TR-013**: All operations SHALL be fully vectorized

4.5 Performance Requirements

- **TR-014**: Total execution time SHALL be < 3 seconds (excluding plot interaction)
- **TR-015**: Memory usage SHALL be < 100 MB
- **TR-016**: Support datasets up to 1000 samples without performance degradation
- **TR-017**: Support up to 200 predictors efficiently

4.6 Error Handling

- **TR-018**: Handle SS_tot = 0 case in R^2 calculation
- **TR-019**: Handle n <= p + 1 case in Adjusted R^2 calculation
- **TR-020**: Validate input dimensions before dot product operations
- **TR-021**: Provide clear error messages for invalid configurations

5. Mathematical Specifications

5.1 R^2 Formula

```
R² = 1 - (SS_res / SS_tot)

Where:
  SS_res = Σ(yᵢ - ŷᵢ)² = dot(residuals, residuals)
  SS_tot = Σ(yᵢ - ȳ)² = dot(deviations, deviations)

Properties:
  - Range: (-∞, 1], typically [0, 1]
  - Always increases with more predictors
  - Does not penalize complexity
```

5.2 Adjusted R^2 Formula

```
Adjusted R² = 1 - [(1 - R²) × (n - 1) / (n - p - 1)]

Where:
  n = number of observations
  p = number of predictors (excluding intercept)

Adjustment Factor:
  Original (p=50):  (100-1)/(100-50-1) = 99/49 ≈ 2.02
  Extended (p=55):  (100-1)/(100-55-1) = 99/44 ≈ 2.25

Properties:
  - Penalizes for adding predictors
  - Can decrease when predictors don't add value
  - Better for model comparison
  - Can be negative
```

5.3 Penalty Calculation

```
Penalty = R² - Adjusted R²
        = R² - [1 - (1 - R²) × (n-1)/(n-p-1)]
        = (1 - R²) × [(n-1)/(n-p-1) - 1]
        = (1 - R²) × p/(n-p-1)

Expected Behavior:
  - Larger p → larger penalty
  - Lower R² → smaller absolute penalty (but larger relative)
  - Extended model should show larger penalty than original
```

5.4 Multicollinearity Effect

```
Dependent Predictor:
  x₅₁ = w₁x₁ + w₂x₂ + noise

Effect on Metrics:
  - R² may increase (more parameters capture noise)
  - Adjusted R² may decrease (penalty > R² gain)
  - Larger gap between R² and Adjusted R²
  - Detection: Compare penalties between models
```

6. Quality Requirements

6.1 Accuracy Requirements

- **QR-001**: R^2 calculations accurate to 6 decimal places
- **QR-002**: Adjusted R^2 calculations accurate to 6 decimal places
- **QR-003**: Dot product results identical to traditional methods (within floating-point precision)
- **QR-004**: Penalty calculations correct for both models

6.2 Reliability Requirements

- **QR-005**: Reproducible results with same seed
- **QR-006**: Handles edge cases without crashes
- **QR-007**: All 4 lines display correctly in graph
- **QR-008**: Annotations readable and non-overlapping

6.3 Educational Quality Requirements

- **QR-009**: Clearly demonstrates R^2 vs Adjusted R^2 differences
- **QR-010**: Multicollinearity effect is obvious in visualization
- **QR-011**: Output explains why Adjusted R^2 is better
- **QR-012**: Suitable for teaching regression metrics

6.4 Code Quality Requirements

- **QR-013**: All functions documented with docstrings
- **QR-014**: Code follows PEP 8 guidelines
- **QR-015**: Variable names match mathematical notation
- **QR-016**: Comments explain complex operations

7. Acceptance Criteria

7.1 Data Generation

- [ ] Original model has 50 independent predictors
- [ ] Extended model has 55 total predictors (50 + 5 dependent)
- [ ] Dependent predictors are linear combinations
- [ ] Both models use 100 data points
- [ ] 20 epsilon values generated correctly

7.2 Metric Calculations

- [ ] R^2 calculated using dot product for both models
- [ ] Adjusted R^2 calculated for both models
- [ ] Penalty calculated for both models
- [ ] Extended model shows larger penalty
- [ ] All calculations across 20 epsilon values

7.3 Visualization

- [ ] Exactly 4 lines displayed
- [ ] Blue lines for original model (solid and dashed)
- [ ] Green lines for extended model (solid and dashed)
- [ ] All lines distinguishable
- [ ] Annotations show correct values
- [ ] Legend clear and complete

7.4 Output

- [ ] Author "Yair Levi" displayed
- [ ] Configuration summary printed
- [ ] Statistics for both models displayed
- [ ] Comparison analysis provided
- [ ] Key findings explained

7.5 Educational Value

- [ ] Demonstrates R^2 inflation with multicollinearity
- [ ] Shows Adjusted R^2 correction mechanism
- [ ] Clear why Adjusted R^2 is better for comparison
- [ ] Penalty differences explained

8. Success Metrics

8.1 Functional Metrics

- R^2 and Adjusted R^2 calculated for all 40 cases (2 models × 20 epsilon)
- All 4 lines visible in single graph
- Penalty for extended model > penalty for original model
- Zero runtime errors

8.2 Performance Metrics

- Execution time < 3 seconds
- Memory usage < 100 MB
- Smooth visualization rendering

8.3 Educational Metrics

- Clearly shows R^2 vs Adjusted R^2 differences
- Multicollinearity effect obvious
- Suitable for teaching (based on user feedback)
- Students understand metric differences after use

9. Future Enhancements

9.1 Phase 2 Features

- AIC/BIC metrics for additional comparison

- F-statistic for model significance
- VIF (Variance Inflation Factor) calculation
- Cross-validation R^2
- Confidence intervals

9.2 Phase 3 Features

- Interactive parameter adjustment
- Multiple model comparison (>2 models)
- Real-time metric updates
- Export to CSV/JSON
- Jupyter notebook version

10. Risk Assessment

Risk	Probability	Impact	Mitigation
Dependent predictors don't create enough multicollinearity	Low	Medium	Use strong linear combinations with minimal noise
Penalties too small to see difference	Low	Medium	Ensure sufficient predictors (50 vs 55)
Lines overlap in visualization	Medium	High	Use distinct styles, colors, markers
Adjusted R^2 concept misunderstood	Medium	High	Comprehensive explanation in output
Performance issues with large datasets	Low	Low	Current size (100×55) is manageable

11. Glossary

- **R^2**: Coefficient of determination; proportion of variance explained
- **Adjusted R^2**: R^2 adjusted for number of predictors
- **Multicollinearity**: High correlation among predictor variables
- **Penalty**: Difference between R^2 and Adjusted R^2
- **Fixed Noise**: Constant bias added to all predictions
- **Dependent Predictor**: Variable that is a linear combination of others
- **Dot Product**: $a \cdot b = \Sigma(a_i b_i)$

Document Version: 2.0

Author: Yair Levi

Last Updated: October 3, 2025

Status: Approved for Implementation

Key Feature: Four-line comparative visualization of R^2 and Adjusted R^2 with multicollinearity demonstration