Multiple Regression R^2 and Adjusted R^2 Analysis - Project Tasks

Project Name: Multiple Linear Regression R^2 and Adjusted R^2 Comparison with Multicollinearity

Author: Yair Levi

Version: 2.0

Status: In Progress

Last Updated: October 3, 2025

📋 Project Overview

Develop a Python application that:

1. Creates **two regression models**: original (50 independent predictors) and extended (55 predictors with 5 dependent)
2. Calculates **both R^2 and Adjusted R^2** for each model
3. Tests across **20 fixed noise values** (epsilon: -3.5 to 3.5)
4. Visualizes **4 lines in one graph**: R^2 and Adjusted R^2 for both models
5. Demonstrates **multicollinearity effects** and **Adjusted R^2 penalty mechanism**
6. Uses **dot product operations** throughout

✅ Completed Tasks (90%)

Phase 1: Project Setup & Documentation

- ✓ Define project requirements and scope
- ✓ Create comprehensive PRD document (133+ requirements)
- ✓ Create detailed README.md with 4-line visualization guide
- ✓ Define configuration parameters
- ✓ Set up project structure
- ✓ Document R^2 and Adjusted R^2 formulas

Phase 2: Original Model Implementation (50 Predictors)

- ✓ **Coefficient Generation**
- ✓ Generate β_0 from uniform [-0.5, 0.5]
- ✓ Generate β_1 to β_{50} from uniform [-0.9, 0.9]
- ✓ Support random seed for reproducibility

- ✓ **X Data Generation**
- ✓ Generate 100 samples × 50 predictors
- ✓ X ~ Normal(μ=0, σ=1)
- ✓ Ensure statistical independence
- ✓ **Y Calculation**
- ✓ Use dot product for predictions
- ✓ Add fixed epsilon noise
- ✓ **R^2 Calculation**
- ✓ Calculate using dot product
- ✓ SS_res = np.dot(residuals, residuals)
- ✓ SS_tot = np.dot(deviations, deviations)

Phase 3: Extended Model Implementation (55 Predictors)

- ✓ **Dependent Predictor Generation**
- ✓ Implement `add_dependent_predictors()` function
- ✓ Create 5 dependent predictors
- ✓ Each as linear combination of 2-3 original predictors
- ✓ Add small noise (σ=0.1) to avoid perfect collinearity
- ✓ **Extended Coefficient Generation**
- ✓ Generate β_{51} to β_{55} from uniform [-0.9, 0.9]
- ✓ Concatenate with original coefficients
- ✓ **Y Calculation for Extended Model**
- ✓ Use same dot product approach
- ✓ Handle 55 predictors correctly

Phase 4: Adjusted R^2 Implementation

- ✓ **Adjusted R^2 Function**
- ✓ Implement `calculate_adjusted_r_squared()` function
- ✓ Formula: Adj $R^2 = 1 - [(1-R^2) \times (n-1)/(n-p-1)]$
- ✓ Accept n_samples and n_predictors as parameters
- ✓ Calculate adjustment factor correctly
- ✓ Handle edge case: $n \leq p + 1$
- ✓ **Integration with Main Workflow**
- ✓ Calculate Adjusted R^2 for original model (p=50)
- ✓ Calculate Adjusted R^2 for extended model (p=55)
- ✓ Store both R^2 and Adjusted R^2 for each epsilon

Phase 5: Four-Line Visualization

- ✓ **Updated Plotting Function**
- ✓ Rename to `plot_r_squared_comparison()`
- ✓ Accept 4 arrays: r2_orig, r2_ext, adj_r2_orig, adj_r2_ext
- ✓ Increase figure size to (16, 9)
- ✓ **Line Specifications**
- ✓ Blue solid (○): R^2 - Original
- ✓ Blue dashed (△): Adjusted R^2 - Original
- ✓ Green solid (□): R^2 - Extended
- ✓ Green dashed (◇): Adjusted R^2 - Extended
- ✓ **Annotations**
- ✓ Yellow box with all 4 metrics at $\varepsilon \approx 0$
- ✓ Blue box with interpretation guide
- ✓ Reference lines ($R^2=1$, 0.5, 0, $\varepsilon=0$)
- ✓ **Legend**
- ✓ Two-column layout
- ✓ All 4 lines clearly labeled
- ✓ Fontsize 10 for readability

Phase 6: Comparative Analysis Output

- ✓ **Statistical Summaries**
- ✓ Display mean, min, max for all 4 metrics
- ✓ Show values at $\varepsilon \approx 0$ for both models
- ✓ Calculate penalties (R^2 - Adj R^2)
- ✓ **Comparison Reporting**
- ✓ R^2 difference between models
- ✓ Adjusted R^2 difference between models
- ✓ Penalty comparison
- ✓ Key findings with interpretation

Phase 7: Code Quality

- ✓ Add comprehensive docstrings
- ✓ Document Adjusted R^2 formula
- ✓ Explain penalty mechanism
- ✓ Comment multicollinearity creation
- ✓ Follow PEP 8 style guidelines
- ✓ Use clear variable names

🔄 Current Tasks (In Progress - 10%)

Testing & Validation

- [ ] **Unit Tests for Adjusted R^2** (Priority: HIGH)
- [ ] Test Adjusted R^2 calculation with known data
- [ ] Perfect fit: Adj R^2 should be close to R^2
- [ ] Poor fit: Adj R^2 should be much lower than R^2
- [ ] Verify penalty = R^2 - Adj R^2
- [ ] Test adjustment factor calculation
- [ ] Original model: $(99/49) \approx 2.02$
- [ ] Extended model: $(99/44) \approx 2.25$
- [ ] Test edge cases
- [ ] When $n = p + 1$ (no adjustment possible)
- [ ] When R^2 is negative
- [ ] When $R^2 = 1.0$
- [ ] **Unit Tests for Dependent Predictors** (Priority: HIGH)
- [ ] Verify dependent predictor generation
- [ ] Check shape (100, 5)
- [ ] Verify linear combinations used
- [ ] Check noise added correctly
- [ ] Test multicollinearity creation
- [ ] Calculate correlation between dependent and original
- [ ] Verify high correlation exists
- [ ] Check not perfectly collinear
- [ ] **Integration Tests** (Priority: HIGH)
- [ ] Test complete workflow for both models
- [ ] Verify 4 arrays created (20 values each)
- [ ] Check penalty for extended > penalty for original
- [ ] Validate all 4 lines display correctly
- [ ] Test with different random seeds
- [ ] **Validation Tests** (Priority: MEDIUM)
- [ ] Compare R^2 at ε=0 for both models
- [ ] Verify Adjusted $R^2 < R^2$ for all cases
- [ ] Check symmetry around ε=0 for all 4 lines
- [ ] Validate penalty increases with more predictors
- [ ] Cross-check with manual calculations

Performance & Optimization

- [ ] **Performance Testing** (Priority: LOW)
- [ ] Measure execution time (target: <3 seconds)
- [ ] Profile memory usage (target: <100MB)
- [ ] Benchmark dot product operations
- [ ] Test with larger datasets (200 samples, 100 predictors)

📝 Pending Tasks

Phase 8: Advanced Features

- [ ] **Additional Metrics** (Priority: MEDIUM)
- [ ] Implement AIC (Akaike Information Criterion)
- [ ] Implement BIC (Bayesian Information Criterion)
- [ ] Add comparison with R^2 and Adjusted R^2
- [ ] Visualize AIC/BIC alongside other metrics
- [ ] **VIF Calculation** (Priority: MEDIUM)
- [ ] Implement Variance Inflation Factor
- [ ] Calculate VIF for all predictors
- [ ] Identify predictors with high VIF (>10)
- [ ] Display VIF results in output
- [ ] **F-Statistic** (Priority: LOW)
- [ ] Calculate F-statistic for model significance
- [ ] Compare F-stats between models
- [ ] Add p-value calculation

Phase 9: Enhanced Visualization

- [ ] **Multiple Subplots** (Priority: LOW)
- [ ] Create 2×2 subplot layout
- [ ] R^2 comparison in one subplot
- [ ] Adjusted R^2 comparison in another
- [ ] Penalty comparison in third
- [ ] Statistical summary in fourth
- [ ] **Interactive Visualization** (Priority: LOW)
- [ ] Use plotly for interactivity
- [ ] Add hover tooltips showing exact values
- [ ] Implement zoom and pan
- [ ] Add selector for different metrics

Phase 10: Data Export

- [ ] **CSV Export** (Priority: MEDIUM)
- [ ] Export all metrics to CSV
- [ ] Include epsilon values
- [ ] Add model identifiers
- [ ] Include penalty calculations
- [ ] **JSON Export** (Priority: LOW)
- [ ] Export complete results
- [ ] Include configuration parameters
- [ ] Add statistical summaries
- [ ] Structured format for reuse

Phase 11: Educational Enhancements

- [ ] **Jupyter Notebook** (Priority: MEDIUM)
- [ ] Create interactive tutorial
- [ ] Step-by-step explanation
- [ ] Exercises for students
- [ ] Visualizations embedded
- [ ] **Documentation Expansion** (Priority: LOW)
- [ ] Add theoretical background on Adjusted R^2
- [ ] Explain why it's better than R^2
- [ ] Provide real-world examples
- [ ] Create video tutorial

🐛 Known Issues

Critical

- None currently identified

Medium Priority

- [ ] With very high noise ($|\varepsilon| > 5$), Adjusted R^2 can become very negative
  - **Status**: Documented as expected behavior
  - **Workaround**: Use moderate epsilon range

Low Priority

- [ ] Legend may be crowded with 4 lines

- **Status**: Mitigated with two-column layout
- **Potential improvement**: Make legend draggable

- [ ] Annotations may overlap if metrics are very close
  - **Status**: Rare occurrence
  - **Workaround**: Adjust annotation positions manually

🧪 Testing Checklist

Manual Testing Scenarios

Scenario 1: Default Configuration

- [ ] Run with default parameters
- [ ] Verify 4 lines displayed
- [ ] Check all lines distinguishable
- [ ] Verify extended model has larger penalty
- [ ] Confirm Adjusted $R^2 < R^2$ for all cases

Scenario 2: Reproducibility

- [ ] Run with SEED=42 multiple times
- [ ] Verify identical results
- [ ] Change seed, verify different results
- [ ] Document seed dependency

Scenario 3: Edge Cases

- [ ] Test with very low noise (EPSILON_MIN=-0.1, MAX=0.1)
  - Expected: All metrics high, small penalties
- [ ] Test with very high noise (EPSILON_MIN=-10, MAX=10)
  - Expected: Low/negative metrics, large penalties
- [ ] Test with more dependent predictors (10 instead of 5)
  - Expected: Larger penalty for extended model

Scenario 4: Predictor Variations

- [ ] Test with fewer original predictors (NUM_PREDICTORS=20)
  - Expected: Smaller penalty for original model
- [ ] Test with more original predictors (NUM_PREDICTORS=100)
  - Expected: Larger penalty, closer to extended model

- [ ] Test with many dependent predictors (NUM_DEPENDENT=20)
  - Expected: Much larger penalty for extended model

Automated Testing

Unit Test Suite

- [ ] Create `tests/test_adjusted_r_squared.py`
- [ ] test_adjusted_r_squared_perfect_fit()
- [ ] test_adjusted_r_squared_poor_fit()
- [ ] test_adjustment_factor()
- [ ] test_edge_cases()
- [ ] Create `tests/test_dependent_predictors.py`
- [ ] test_add_dependent_predictors_shape()
- [ ] test_dependent_predictors_correlation()
- [ ] test_multicollinearity_creation()
- [ ] Create `tests/test_four_line_visualization.py`
- [ ] test_plot_accepts_four_arrays()
- [ ] test_plot_displays_correctly()
- [ ] test_annotations_present()

Integration Test Suite

- [ ] Create `tests/test_complete_workflow.py`
- [ ] test_both_models_complete()
- [ ] test_all_metrics_calculated()
- [ ] test_penalty_comparison()
- [ ] test_visualization_generation()

Validation Testing

- [ ] **Mathematical Validation**
- [ ] Verify Adjusted R^2 formula manually
- [ ] Check penalty calculation: R^2 - Adj R^2
- [ ] Validate adjustment factors
- [ ] Compare with statistical software (R, Python statsmodels)
- [ ] **Visual Validation**
- [ ] Verify 4 distinct lines visible
- [ ] Check color/style consistency
- [ ] Confirm annotations readable
- [ ] Validate legend completeness

📊 Success Metrics

Functional Metrics

- [x] Original model (50 predictors) implemented ✓
- [x] Extended model (55 predictors) implemented ✓
- [x] Adjusted R^2 calculation working ✓
- [x] 4-line visualization complete ✓
- [ ] All unit tests passing (0/20 written)
- [ ] Integration tests passing (0/5 written)
- [ ] Code coverage >85%

Performance Metrics

- [x] Execution time <3 seconds ✓
- [ ] Memory usage <100MB (need to verify)
- [x] Smooth visualization ✓
- [ ] Scalability tested (pending)

Quality Metrics

- [x] All functions documented ✓
- [x] Code follows PEP 8 ✓
- [x] Dot product usage explicit ✓
- [x] Author attribution present ✓
- [ ] Test coverage >85%
- [ ] All validation tests pass

Educational Metrics

- [x] Demonstrates R^2 vs Adjusted R^2 ✓
- [x] Shows multicollinearity effect ✓
- [x] Penalty mechanism clear ✓
- [x] 4-line visualization informative ✓
- [ ] Suitable for teaching (need user feedback)
- [ ] Learning outcomes achieved (need assessment)

Comparison Metrics

- [x] Extended model shows larger penalty ✓
- [x] Adjusted R^2 corrects for complexity ✓

- [x] All 4 metrics calculated correctly ✓
- [x] Differences clearly visible in graph ✓

🎯 Priority Tasks for Next Session

Immediate (Must Complete)

- [ ] **HIGH**: Write unit tests for Adjusted R^2 calculation
- [ ] **HIGH**: Write unit tests for dependent predictor generation
- [ ] **HIGH**: Verify penalty for extended model > original model
- [ ] **HIGH**: Test complete workflow with multiple seeds
- [ ] **HIGH**: Validate all 4 lines display correctly

Short-term (Should Complete)

- [ ] **MEDIUM**: Implement VIF calculation
- [ ] **MEDIUM**: Add AIC/BIC metrics
- [ ] **MEDIUM**: Create CSV export functionality
- [ ] **MEDIUM**: Measure and document performance
- [ ] **MEDIUM**: Create Jupyter notebook tutorial

Long-term (Nice to Have)

- [ ] **LOW**: Interactive visualization with plotly
- [ ] **LOW**: Multiple subplot layout
- [ ] **LOW**: Video tutorial creation
- [ ] **LOW**: Real-world example datasets
- [ ] **LOW**: Comparison with sklearn/statsmodels

📝 Notes and Observations

Technical Notes

- Adjusted R^2 calculation working correctly
- Penalty for extended model consistently larger (as expected)
- Four-line visualization clear and informative
- Dot product operations verified for all calculations
- Multicollinearity successfully created through linear combinations

Design Decisions

- **Two-column legend**: Chose for better space utilization with 4 lines
- **Figure size (16, 9)**: Larger than original to accommodate more information
- **Two annotation boxes**: Yellow for data, blue for interpretation
- **Dashed lines for Adjusted R^2**: Visual distinction from solid R^2 lines
- **Different markers**: Circles, triangles, squares, diamonds for uniqueness

Key Findings

- Extended model shows R^2 inflation (higher R^2 despite dependent predictors)
- Adjusted R^2 correctly identifies that dependent predictors don't add value
- Penalty difference clearly demonstrates Adjusted R^2 superiority
- Visual comparison makes metric differences immediately obvious
- Educational value significantly enhanced with 4-line comparison

Performance Observations

- Current execution time: ~1-2 seconds (excellent)
- Memory usage appears minimal (<50MB estimated)
- Adding Adjusted R^2 calculation adds negligible overhead
- Four-line plot renders smoothly

Future Considerations

- Could add statistical tests for model comparison
- VIF would quantify multicollinearity numerically
- AIC/BIC would provide alternative comparison metrics
- Interactive version would enhance educational value
- Real dataset examples would validate approach

🔗 Related Documents

- PRD Document
- README.md
- Source Code

- Configuration
- Test Suite (to be created)
- Examples (to be created)
- Notebooks (to be created)

👥 Project Team

- **Author**: Yair Levi
- **Role**: Lead Developer, Documentation, Testing
- **Contact**: [contact information]

📈 Version History

Version	Date	Changes	Completion
1.0	Oct 3, 2025	Initial with R^2 only	75%
1.5	Oct 3, 2025	Added multicollinearity comparison	80%
2.0	Oct 3, 2025	Added Adjusted R^2 and 4-line visualization	90%
2.1	TBD	Add comprehensive testing	Planned
3.0	TBD	Add AIC/BIC, VIF, interactive features	Planned

🏁 Definition of Done

A task is considered complete when:

- [ ] Code is implemented and functional
- [ ] Unit tests written and passing
- [ ] Integration tests passing
- [ ] Documentation updated (docstrings, README, PRD)
- [ ] Code reviewed (if team project)
- [ ] Manually tested with various inputs
- [ ] No known bugs or issues
- [ ] Performance requirements met
- [ ] Educational value validated
- [ ] Committed to version control
- [ ] Changelog updated

📅 Project Timeline

Week 1 (Completed) ✓

- ✅ Project setup and requirements
- ✅ Original model implementation
- ✅ Extended model with multicollinearity
- ✅ R^2 calculation

Week 2 (Completed) ✓

- ✅ Adjusted R^2 implementation
- ✅ Four-line visualization
- ✅ Comparative analysis output
- ✅ Documentation (PRD, README, Tasks)

Week 3 (Current) 🔄

- 🔄 Testing and validation
- 🔄 Unit tests for Adjusted R^2
- 🔄 Integration tests
- ⏳ Performance benchmarking
- ⏳ Bug fixes

Week 4 (Planned)

- ⏳ Advanced features (VIF, AIC/BIC)
- ⏳ Data export functionality
- ⏳ Jupyter notebook tutorial
- ⏳ Final documentation
- ⏳ Release preparation

🎬 Next Actions

Today

- [ ] Run program with SEED=42 and document all 4 metrics

- [ ] Create test file structure (tests/ directory)
- [ ] Write first 3 unit tests for Adjusted R^2
- [ ] Verify penalty calculation manually
- [ ] Take screenshot of 4-line graph for documentation

This Week

- [ ] Complete all unit tests (target: 20 tests)
- [ ] Write integration tests (5 scenarios)
- [ ] Validate with different epsilon ranges
- [ ] Measure performance metrics
- [ ] Update README with test results

This Month

- [ ] Implement VIF calculation
- [ ] Add AIC/BIC metrics
- [ ] Create CSV export feature
- [ ] Develop Jupyter notebook tutorial
- [ ] Prepare for release (v2.0)

Last Review: October 3, 2025

Next Review: October 10, 2025

Status: On Track

Completion: 90% (implementation complete, testing pending)

Blockers: None

Next Milestone: Complete testing suite (Week 3)