Winter in Data Science (WiDS) 2025

Key Learnings and Insights

Jash Taparia
Indian Institute of Technology Bombay

January 2025

*A Comprehensive Reflection on Technical Skills,
Methodological Insights, and Personal Growth*

Contents

1 Executive Summary

The Winter in Data Science (WiDS) 2025 program provided an intensive learning experience across five major projects: linear regression from first principles, salary prediction modeling, neural network implementation, algorithmic trading with Kalman Filters, and deep reinforcement learning. This report synthesizes the key technical skills acquired, methodological insights gained, and personal growth achieved throughout this journey.

Key Insight

The most profound realization from this program: **Data science is both an art and a science**. While mathematical rigor and computational skills form the foundation, creativity in problem formulation, judgment in model selection, and wisdom in interpretation are equally crucial for success.

2 Technical Skills Acquired

2.1 Statistical Foundations

2.1.1 Linear Algebra Mastery

Through implementing linear regression from scratch, I gained deep understanding of:

- **Matrix Operations:** Computing $(X^T X)^{-1}$ for coefficient estimation
- **Projection Matrices:** Understanding the hat matrix $H = X(X^T X)^{-1} X^T$
- **Least Squares:** The geometric interpretation of minimizing residual norm
- **Eigenvalue Analysis:** Used in leverage score calculations

Lesson Learned

Implementation = Understanding
Coding algorithms from scratch reveals subtleties that reading theory alone cannot. For example, understanding why we add a column of ones for the intercept term became intuitive only after manual implementation.

2.1.2 Probability and Inference

Key concepts mastered:

- **Distributions:** Normal, t-distribution for hypothesis tests
- **Maximum Likelihood:** Foundation for loss functions in ML
- **Hypothesis Testing:** p-values, confidence intervals, significance levels
- **Bayesian Thinking:** Prior beliefs + data = posterior understanding

Key Insight

Statistical assumptions (linearity, homoscedasticity, normality) are not just theoretical requirements—they directly impact the reliability of predictions and inference. Always verify, never assume.

2.1.3 Diagnostic Tools

Learned to critically evaluate models using:

1. **Residual Analysis:**
   - Patterns indicate model misspecification
   - Heteroscedasticity violates OLS assumptions
   - Q-Q plots reveal non-normality
2. **Leverage and Influence:**
   - High leverage $\neq$ high influence
   - Cook's distance combines both effects
   - Threshold: $\frac{4}{n}$ for Cook's D, $\frac{2(p+1)}{n}$ for leverage
3. **Multicollinearity Detection:**
   - VIF (Variance Inflation Factor) analysis
   - Correlation matrices
   - Age vs. years_experience showed expected high correlation

Lesson Learned

One Metric Is Never Enough
Outlier detection requires multiple complementary metrics. A point with high leverage but low residual is very different from one with both high—understanding this distinction prevents incorrect data removal.

2.2 Machine Learning Fundamentals

2.2.1 Supervised Learning Techniques

Regression:

- Linear regression: Interpretable baseline
- Multiple regression: Handling multiple predictors
- Feature interactions: Education $\times$ Experience effects
- Polynomial features: Capturing non-linear relationships

Classification:

- Random Forests: Ensemble learning for robust predictions
- Binary classification: Next-day price movement
- Multi-class: Digit recognition (10 classes)

Key Insight

The Bias-Variance Trade-off is Everywhere
Simple models (high bias, low variance) vs. complex models (low bias, high variance). Linear regression for salary prediction offered interpretability; Random Forest captured non-linearities but lost explainability. Choose based on project goals.

2.2.2 Feature Engineering

Learned that features matter more than algorithms:

1. **Encoding Strategies:**
   - Ordinal encoding for education (preserves order)
   - One-hot encoding for nominal variables (gender, industry)
   - drop_first=True to avoid dummy variable trap
2. **Normalization Techniques:**
   - Z-score: $(x - \mu)/\sigma$ for Kalman residuals
   - Min-max scaling: For neural network inputs
   - Rolling window standardization: Adapts to regime changes
3. **Domain-Specific Features:**
   - Technical indicators: RSI, ATR, Bollinger Bands
   - Derived features: Fair Value, Spread, Momentum
   - Interaction terms: Experience $\times$ Education

Lesson Learned

Feature Engineering is 80% of the Work
In the salary prediction project, proper encoding of categorical variables and handling of ordinal education levels had more impact than model selection. Good features make simple models work; bad features doom complex ones.

2.2.3 Model Evaluation

Developed rigorous validation practices:

- **Train-Test Split:** 70-30 or 80-20 ratios
- **Stratified Sampling:** Ensures representative distributions

- **Cross-Validation:** k-fold for robust performance estimates
- **Metrics Selection:**
  - RMSE: Penalizes large errors
  - MAE: Interpretable average error
  - R^2: Proportion of variance explained
  - Sharpe Ratio: Risk-adjusted returns

Key Insight

Out-of-Sample Performance is the Only Performance That Matters
Training accuracy is meaningless if the model fails on new data. This lesson was reinforced in every project, especially in trading where overfitting to historical data guarantees future losses.

2.3 Deep Learning

2.3.1 Neural Network Fundamentals

Architecture Design:

- Input layer sizing: 784 neurons for 28×28 images
- Hidden layer depth: 2-3 layers for most tasks
- Output layer: 10 neurons for digit classification
- Activation functions: ReLU for hidden, none for output (softmax in loss)

Forward Propagation:

$$h_1 = \text{ReLU}(W_1 x + b_1), \quad h_2 = \text{ReLU}(W_2 h_1 + b_2), \quad y = W_3 h_2 + b_3 \tag{1}$$

Backpropagation:

- Chain rule application: $\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial W_1}$
- Gradient flow: From loss to weights
- PyTorch autograd: Automatic differentiation

Lesson Learned

Understanding Beats Black-Box Usage
Implementing forward and backward passes manually (even once) demystifies neural networks. When debugging training issues, this foundational knowledge proved invaluable.

Optimizer	Key Feature	When to Use
SGD	Simple, momentum	Well-understood problems
Adam	Adaptive learning rates	Default choice, robust
RMSProp	Adaptive, no momentum	RNNs, online learning
AdaGrad	Rare feature emphasis	Sparse data

Table 1: Optimizer Selection Guide

2.3.2 Optimization Techniques

Optimizers Compared:

Learning Rate Scheduling:

- Step decay: Reduce by factor every N epochs
- Cosine annealing: Smooth reduction
- Warm-up: Start small, gradually increase

Key Insight

Adam is the Default, But Not Always the Best
While Adam worked well for digit classification, SGD with momentum sometimes achieves better final accuracy. The trade-off: Adam converges faster, SGD potentially finds better minima.

2.3.3 Regularization Strategies

Preventing overfitting through:

1. **Dropout:** Randomly deactivate neurons (p=0.5 typical)
2. **Batch Normalization:** Normalize layer inputs
3. **Weight Decay:** L2 regularization on parameters
4. **Early Stopping:** Monitor validation loss
5. **Data Augmentation:** For images (rotation, flip, crop)

Lesson Learned

Regularization is Not Optional
Neural networks' capacity to memorize training data is both strength and weakness. Without regularization, even simple problems lead to overfitting. Always monitor train vs. validation loss.

2.4 Advanced Methods: State-Space Models

2.4.1 Kalman Filter Theory

The Kalman Filter revolutionized my understanding of time series:

State-Space Representation:

$$x_t = Ax_{t-1} + w_t \quad \text{(Hidden state evolution)} \tag{2}$$
$$y_t = Hx_t + v_t \quad \text{(Observation)} \tag{3}$$

Key Components:

- State vector: $x_t = [\text{Level}, \text{Slope}]^T$
- Transition matrix: $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ (local linear trend)
- Process noise: Q controls state flexibility
- Observation noise: R reflects measurement uncertainty

Recursive Algorithm:

1. **Predict:** $\hat{x}_{t|t-1} = A\hat{x}_{t-1}$
2. **Update:** $\hat{x}_t = \hat{x}_{t|t-1} + K(y_t - H\hat{x}_{t|t-1})$
3. **Kalman Gain:** K balances prediction vs. observation

Key Insight

Kalman Filters Separate Signal from Noise Optimally
Unlike moving averages (backward-looking) or exponential smoothing (ad-hoc), Kalman Filters are optimal under Gaussian assumptions. They provide not just estimates but also uncertainty quantification.

2.4.2 Practical Applications in Finance

Trend Detection:

- Positive slope $\rightarrow$ Uptrend $\rightarrow$ Long position
- Negative slope $\rightarrow$ Downtrend $\rightarrow$ Cash
- Adaptive: Responds to regime changes

Mean Reversion:

- Fair Value $= \beta \times \text{SMA} + \alpha$
- Spread = Actual - Fair Value
- Z-score normalization: $(x - \mu)/\sigma$
- Trade when —Z-score— ¿ threshold

Lesson Learned

Parameter Tuning is Both Art and Science
Process noise Q and observation noise R dramatically affect behavior:

- High Q: Responsive but noisy
- Low Q: Smooth but laggy
- High R: Trust model over data
- Low R: Trust data over model

Optimal values depend on asset characteristics and market regime.

2.5 Reinforcement Learning

2.5.1 Core RL Concepts

Markov Decision Process:

- States S: Environment configurations
- Actions A: Agent choices
- Transitions $P(s'|s,a)$: State dynamics
- Rewards $R(s,a)$: Immediate feedback
- Discount γ: Future value weight

Q-Learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{4}$$

Key Insight

Credit Assignment is the Fundamental Challenge
In Mountain Car, reaching the goal at timestep 200 requires figuring out which of the 200 actions were helpful. Temporal difference learning propagates value backward, but this is slow and sample-inefficient.

2.5.2 Deep Q-Learning (DQN)

Why Neural Networks:
Key Innovations:

1. **Experience Replay:** Store (s,a,r,s') transitions, sample randomly
   - Breaks temporal correlation
   - Improves sample efficiency
   - Enables mini-batch training

Tabular Q-Learning	**Deep Q-Learning**
Discrete states only	Continuous states
$O(\|S\| \times \|A\|)$ memory	$O(\text{params})$ memory
No generalization	Generalizes to similar states
Guaranteed convergence	Not guaranteed
Simple to debug	Complex but powerful

Table 2: Tabular vs. Deep Q-Learning

2. **Target Networks:** Separate network for computing targets
   - Updated periodically (not every step)
   - Stabilizes training
   - Prevents moving target problem
3. **ϵ-Greedy Exploration:**
   - Probability ϵ: Random action
   - Probability $1 - \epsilon$: Greedy action
   - Decay ϵ: From exploration to exploitation

Lesson Learned

RL Requires Extreme Patience
Mountain Car learning curve: Flatline at -200 for many episodes, sudden spike when goal is first reached, gradual improvement. This is normal. The "breakthrough moment" is when the agent accidentally succeeds and the network captures this in the replay buffer.

3 Methodological Insights

3.1 The Data Science Workflow

Through five projects, a clear workflow emerged:

1. **Problem Definition**
   - What exactly are we predicting/optimizing?
   - What constitutes success?
   - What are the constraints?
2. **Data Collection & Cleaning**
   - Source identification (yfinance, CSV files)
   - Missing value handling (imputation strategies)

- Outlier detection and treatment
- Data validation (sanity checks)

3. **Exploratory Data Analysis**
    - Distributions: Histograms, box plots
    - Relationships: Correlation matrices, scatter plots
    - Patterns: Time series plots, seasonality
    - Insights: Summary statistics, domain understanding
4. **Feature Engineering**
    - Transformations: Log, sqrt, polynomial
    - Encodings: One-hot, ordinal, target
    - Derived features: Ratios, lags, rolling statistics
    - Selection: Remove redundant/irrelevant features
5. **Model Development**
    - Baseline: Simple model for comparison
    - Iteration: Gradually increase complexity
    - Validation: Out-of-sample performance
    - Tuning: Hyperparameter optimization
6. **Evaluation & Interpretation**
    - Metrics: Appropriate for problem type
    - Diagnostics: Residual analysis, confusion matrix
    - Interpretation: Feature importance, coefficients
    - Communication: Visualizations, clear explanations
7. **Deployment & Monitoring**
    - Production code: Robust, efficient
    - Error handling: Graceful failures
    - Logging: Track performance
    - Updates: Retrain as needed

Key Insight

Iteration is Not Optional, It's the Process
No project followed a linear path. The workflow is a cycle: EDA reveals data issues $\rightarrow$ back to cleaning; poor model performance $\rightarrow$ back to feature engineering; unexpected results $\rightarrow$ back to EDA. Embrace iteration.

3.2 Best Practices Discovered

3.2.1 Code Organization

- **Modularity:** Separate functions for data loading, preprocessing, modeling
- **Documentation:** Docstrings for every function
- **Version Control:** Git for tracking changes
- **Reproducibility:** Random seeds, requirements.txt

3.2.2 Experimentation

- **Start Simple:** Linear regression before neural networks
- **One Change at a Time:** Isolate effects
- **Document Everything:** Experiment log with results
- **Visualize:** Plots reveal patterns text cannot

Lesson Learned

Premature Optimization is the Enemy
In the neural network project, I initially spent time optimizing batch size and network architecture. The real bottleneck? Data quality. Fix the biggest issue first, optimize later.

3.2.3 Validation Rigor

- **Never Touch Test Set:** Until final evaluation
- **Stratified Sampling:** For imbalanced data
- **Time Series Care:** No shuffling, forward chaining
- **Cross-Validation:** For robust estimates

Key Insight

Look-Ahead Bias is Subtle and Deadly
In backtesting, using today's close price to make today's trading decision seems innocuous but destroys validity. Always ask: "What information would I have known at decision time?" Shift signals by one day.

3.3 Domain-Specific Insights

3.3.1 Finance and Trading

Key Lessons:

1. **Transaction Costs are Material**

- 10 basis points (0.1%) seems small
- With frequent rebalancing, compounds significantly
- High-frequency strategies need larger edges

2. **Markets are Noisy**
   - Signal-to-noise ratio is low
   - Filtering techniques (Kalman) help
   - Perfect prediction is impossible
3. **Diversification Works**
   - Multi-asset portfolio smooths returns
   - Uncorrelated assets reduce risk
   - BTC + Equity + Gold provided balance
4. **Regime Changes**
   - Markets shift between trending and mean-reverting
   - Static models fail across regimes
   - Adaptive methods (rolling windows) essential

Lesson Learned

Backtesting is Not Reality
Simulated results are best-case scenarios. Real trading involves:

- Slippage (execution price differs from expected)
- Market impact (your orders move prices)
- Liquidity constraints (can't always trade desired size)
- Psychological factors (fear, greed, hesitation)

Add a 20-30% haircut to backtest results for reality.

3.3.2 Labor Economics

From salary prediction project:

1. **Education Matters**
   - Each degree level: +$8,373
   - PhD vs. Bachelor's: ~$16,740 difference
   - Clear returns on educational investment
2. **Industry Choice is Crucial**
   - Finance baseline

- Retail: -\$17,815
- Healthcare: -\$6,183
- Largest single factor

3. **Experience Accumulation**
   - Strong correlation with salary
   - Diminishing returns at higher levels
   - Quality matters more than quantity

4. **Job Mobility Paradox**
   - Previous companies: Low correlation with salary
   - Frequent switching $\neq$ higher pay
   - Strategic moves ¿ random moves

4 Personal Growth and Soft Skills

4.1 Problem-Solving Approach

4.1.1 Structured Thinking

Developed systematic approach:

1. **Decomposition:** Break complex problems into manageable pieces
2. **Prioritization:** Tackle highest-impact issues first
3. **Incremental Progress:** Build and test incrementally
4. **Reflection:** Learn from both successes and failures

Lesson Learned

Stuck? Change Perspective
When debugging the Kalman Filter, I was stuck for hours adjusting parameters. The breakthrough came from visualizing state evolution over time—the problem became immediately obvious. Multiple approaches to the same problem often reveal the solution.

4.1.2 Debugging Skills

- **Print Statements:** Simple but effective
- **Visualize Intermediate Results:** Plot activations, gradients
- **Unit Tests:** Verify components individually
- **Sanity Checks:** Does output make sense?
- **Rubber Duck Debugging:** Explain to someone (or something)

4.2 Communication Skills

4.2.1 Technical Writing

Learned to:

- Structure reports logically (intro, methods, results, conclusion)
- Use visualizations effectively (not decoratively)
- Balance technical depth with accessibility
- Tell a story, not just present facts

4.2.2 Code Documentation

- Clear variable names: `kf_slope` not `x2`
- Docstrings: Explain what, why, and how
- Comments: For complex logic, not obvious statements
- READMEs: Project overview, setup, usage

4.3 Self-Learning Ability

4.3.1 Resources Leveraged

- **Documentation:** NumPy, PyTorch, scikit-learn docs
- **Research Papers:** Original DQN paper, Kalman 1960
- **Online Communities:** Stack Overflow, Reddit, GitHub issues
- **Courses:** Andrew Ng ML, Fast.ai
- **Books:** ISLR, Deep Learning (Goodfellow)

Key Insight

Learning to Learn is the Meta-Skill
Specific algorithms and libraries change rapidly. The ability to:

- Read documentation efficiently
- Understand research papers
- Debug systematically
- Adapt tutorials to your problem

...these skills enable lifelong learning in this field.

4.3.2 Growth Mindset

- **Embrace Failure:** Every error is a learning opportunity
- **Seek Challenges:** Comfort zone is the enemy of growth
- **Ask Questions:** No question is too basic
- **Share Knowledge:** Teaching reinforces understanding

5 Critical Reflections

5.1 What Worked Well

1. **Hands-On Implementation**
   - Building from scratch deepened understanding
   - Mistakes during implementation were valuable lessons
   - Comparing custom vs. library implementations validated correctness
2. **Diverse Project Portfolio**
   - Classical statistics (regression)
   - Applied ML (salary prediction)
   - Deep learning (neural networks)
   - Advanced methods (Kalman Filters, RL)
   - Breadth + depth combination
3. **Iterative Approach**
   - Start simple, add complexity gradually
   - Validate at each step
   - Pivot when needed

5.2 Challenges Faced

1. **Conceptual Complexity**
   - Kalman Filters: Abstract mathematics
   - Solution: Visualized state evolution, implemented simple cases first
2. **Debugging Difficulty**
   - Neural networks: Silent failures
   - Solution: Checked intermediate outputs, validated gradients
3. **Time Management**
   - Balancing depth vs. breadth

- Solution: Time-boxing experiments, accepting "good enough"

4. **Overfitting Temptation**
   - Easy to optimize for training data
   - Solution: Strict train-test separation, focus on generalization

5.3 What I Would Do Differently

1. **More Rigorous Experiment Tracking**
   - Implement systematic logging from day one
   - Use tools like Weights & Biases or MLflow
   - Document all hyperparameter choices
2. **Earlier Focus on Interpretability**
   - Not just "does it work?" but "why does it work?"
   - Feature importance analysis
   - Model explanations (SHAP, LIME)
3. **More Collaboration**
   - Peer code reviews
   - Discussion of approaches
   - Learning from others' mistakes

6 Most Important Lessons

6.1 Top 10 Takeaways

1. **Assumptions Matter More Than Algorithms**

   The best model with violated assumptions performs worse than a simple model with satisfied assumptions. Always check: linearity, independence, homoscedasticity, normality.
2. **Feature Engineering ¿ Model Selection**

   Well-engineered features with linear regression often beat poorly-designed features with deep learning. Invest time in understanding and transforming your data.
3. **Out-of-Sample Testing is Non-Negotiable**

   Training performance is vanity; test performance is reality. In finance, this lesson is especially expensive if ignored.
4. **Simplicity is a Feature, Not a Bug**

   Simple models are interpretable, debuggable, and maintainable. Complexity should be justified by significant performance gains, not assumed to be better.

5. **Visualize Everything**

   Plots reveal patterns, outliers, and relationships that summary statistics hide. Residual plots, learning curves, correlation matrices—visualization is essential.

6. **Domain Knowledge Guides Data Science**

   Understanding labor economics improved salary modeling; knowing market dynamics improved trading strategies. Technical skills without domain knowledge is incomplete.

7. **Iteration is the Process**

   First attempt rarely succeeds. EDA $\rightarrow$ modeling $\rightarrow$ back to EDA $\rightarrow$ feature engineering $\rightarrow$ back to modeling. This is normal and productive.

8. **Reproducibility Requires Discipline**

   Random seeds, version control, documentation, environment management—these practices prevent hours of debugging and enable collaboration.

9. **The 80/20 Rule Applies**

   80% of results come from 20% of effort. Identify and focus on high-impact activities: data cleaning, feature engineering, proper validation.

10. **Learning to Learn is the Meta-Skill**

    Specific tools and methods become obsolete. The ability to quickly learn new techniques, read papers, and adapt to changing landscapes is the most valuable skill.

6.2 Fundamental Principles

Key Insight

The Scientific Method in Data Science

1. Observe: EDA, visualizations
2. Hypothesize: Feature relationships, model assumptions
3. Experiment: Train models, test hypotheses
4. Analyze: Evaluate results, diagnostic checks
5. Conclude: Interpret findings, understand limitations
6. Iterate: Refine based on learnings

This cycle mirrors scientific inquiry. Data science *is* science.

Lesson Learned

Humility in the Face of Uncertainty
Models are wrong, but some are useful (Box). We work with:

- Imperfect data (noise, missing values, bias)
- Approximate models (all models make simplifying assumptions)
- Uncertain predictions (confidence intervals, not point estimates)
- Changing systems (non-stationarity, regime shifts)

Acknowledge limitations. Quantify uncertainty. Communicate caveats.

7 Conclusion

7.1 Transformation Summary

The Winter in Data Science 2025 program has been genuinely transformative:

Before: Theoretical understanding of ML algorithms, limited practical experience, uncertainty about implementation details.

After: Hands-on experience across diverse projects, confidence in debugging complex systems, ability to choose and apply appropriate methods, understanding of real-world challenges.

7.2 Core Philosophy Developed

Through this journey, I've internalized a philosophy of data science:

1. **Rigor:** Question assumptions, validate thoroughly, measure properly
2. **Pragmatism:** Perfect is enemy of good; ship working solutions
3. **Curiosity:** Never stop asking ”why?” and ”what if?”
4. **Humility:** Models are approximations; predictions have uncertainty
5. **Ethics:** Consider impact, avoid bias, ensure fairness

Key Insight

The Journey is Just Beginning
WiDS 2025 provided a solid foundation, but it's just the start. The field of AI/ML is evolving rapidly:

- Large language models transforming NLP
- Diffusion models revolutionizing generation
- Reinforcement learning tackling complex control
- Quantum computing approaching practical ML applications

The key is not to know everything now, but to have the tools and mindset to learn anything later.

7.3 Gratitude

This learning journey was enabled by:

- **WiDS Organizers:** For creating this structured program
- **Open Source Community:** NumPy, PyTorch, scikit-learn contributors
- **Researchers:** Who published papers and shared knowledge
- **Online Communities:** Stack Overflow, Reddit, GitHub for support
- **Educators:** Andrew Ng, Jeremy Howard, and countless others who teach freely

7.4 Final Thoughts

Personal Reflection

If I could summarize this entire experience in one sentence:

"Data science is a journey of continuous learning where humility about what we don't know is as important as confidence in what we do know, and where the ability to translate data into actionable insights matters more than knowing every algorithm."

The technical skills—regression, neural networks, Kalman Filters, reinforcement learning—are tools. The real learning was developing judgment: when to use which tool, how to validate results, when to trust models and when to be skeptical, and how to communicate findings effectively.

As I move forward, I carry not just knowledge of specific methods, but a systematic approach to problem-solving, a commitment to rigor, and most importantly, excitement about what's yet to be learned.

The field is vast, the challenges are complex, and the opportunities are immense. I'm ready.

"In God we trust. All others must bring data."
— W. Edwards Deming

End of Report

Winter in Data Science 2025
Key Learnings and Reflections

Jash Taparia
January 2025