AI-Driven Quantitative Trading Research System

CS4100 Course Project Proposal

Student Name: [Your Name]
Date: October 2025
Project Type: Individual Project

---

1. What is the Problem?

Problem Statement: Quantitative trading research is extremely time-intensive and requires deep analysis of market microstructure, price patterns, and statistical relationships. A human quant researcher analyzing a single asset (like Bitcoin or Gold) can spend days or weeks conducting exploratory data analysis, testing indicator effectiveness, identifying regime patterns, and understanding volume dynamics across multiple timeframes. This process:

- Cannot scale across hundreds of different assets efficiently
- Is limited by human pattern recognition and computational capacity
- Requires repetitive analytical work for each new asset
- Often misses subtle statistical relationships in high-dimensional data

Specific Challenge: Build an AI system that automates the quantitative research process, functioning as an **AI Quantitative Research Assistant** that can:

- Accept ANY asset with historical data (stocks, crypto, forex, commodities) as input
- Conduct comprehensive market microstructure analysis like a professional quant researcher
- Generate detailed research reports with statistical evidence and quantitative insights
- Scale this analysis across multiple assets efficiently

Input → Output Mapping:

- **Input A:** Asset identifier (e.g., BTC-USD, XAU-USD, AAPL) + Historical OHLCV data
- **Output B:** Comprehensive quantitative research report containing:
    - Volume pattern analysis across timeframes (1min, 5min, 1hr, 4hr, daily)
    - Volatility characteristics with statistical measures
    - Trend behavior analysis with confidence intervals
    - Technical indicator effectiveness specific to THIS asset (RSI, MACD, ATR, SMA-50/200, Volume, VWAP)
    - Market regime identification (trending vs. ranging periods)

- Correlation analysis with other major assets
- Trading activity patterns (which timeframes see most volume, peak trading hours)
- Price action characteristics and statistical properties

Key Innovation: Unlike generic trading signals or indicators, this system provides **asset-specific research intelligence** - understanding that Bitcoin behaves differently than Gold, which behaves differently than EUR-USD. The AI learns what makes each asset unique and reports those findings, enabling the human trader to develop tailored strategies.

2. Why is This Problem Interesting and Exciting?

The Core Value Proposition: AI Augmenting Expert-Level Work

The fundamental purpose of AI is to **replace repetitive human work with greater efficiency and accuracy**. Quantitative research is particularly well-suited for AI augmentation because:

- The same analytical process must be repeated for every new asset
- Each analysis requires processing vast amounts of data (years of minute-level price data)
- Pattern recognition across multiple dimensions (time, volume, volatility) is computationally intensive
- Statistical testing and correlation analysis scale poorly with human analysts

What makes this challenging: Quantitative researchers are highly specialized professionals with deep expertise in both **finance domain knowledge** and **statistical/mathematical analysis**. They command high salaries ($150K-500K+) precisely because this combination of skills is rare. Building an AI that can replicate even a portion of their analytical workflow is non-trivial - it's not just pattern matching, it's understanding market microstructure, regime changes, and asset-specific behavior.

Academic Interest (CS4100 Perspective):

Combines Multiple AI Paradigms:

- **Data Science Workflow:** Exploratory data analysis, statistical testing, insight generation
- **Machine Learning Workflow:** Pattern recognition, regime classification, predictive modeling
- **AI Pipeline Architecture:** Multi-stage system (data collection → preprocessing → analysis → reporting)

Real-World Complexity:

- Working with **noisy, non-stationary time series data** (unlike clean academic datasets)
- Handling **high-dimensional feature spaces** (price, volume, indicators across multiple timeframes)
- **Overfitting prevention** is critical - in finance, overfit models lose money in production
- Demonstrates proper train/validation/test methodology in a domain where it truly matters

Applies Core CS4100 Concepts:

- Neural networks for pattern recognition and regime classification
- Regularization techniques to prevent overfitting
- Statistical validation of model outputs
- Understanding what AI can and cannot do (it won't replace human strategic thinking, but augments research capacity)

Personal/Career Motivation:

Differentiating Portfolio Project: Most CS students build toy projects on clean datasets (MNIST, Iris, etc.). This project demonstrates:

- Ability to work with real financial data and domain-specific challenges
- Understanding of both AI/ML techniques AND quantitative finance concepts
- End-to-end system design from data pipeline to actionable insights
- Solo capability to execute a professional-grade project

Foundation for Career Trajectory: My goal is to become a quantitative trader. This project serves as:

1. **Immediate value:** A working tool I can use for actual market research
2. **Technical demonstration:** Shows competency in Python, ML, data analysis, and system architecture
3. **Expandability:** Foundation that can evolve into a full algorithmic trading system with backtesting, strategy development, and eventually live trading
4. **Proof of domain expertise:** Demonstrates understanding of market microstructure, not just coding ability

If successful, this becomes more than a course project - it's the first building block of a production quantitative trading system. The research assistant (this project) → Strategy development → Backtesting framework → Paper trading → Live trading. Each phase builds on the previous.

Technical Challenge:

Why This Is Hard: Quantitative researchers spend years developing intuition about markets. Teaching an AI to replicate their analytical process requires:

- Understanding what makes each asset unique (Bitcoin ≠ Gold ≠ EUR-USD)
- Identifying regime changes automatically (trending vs. ranging markets behave completely differently)
- Separating signal from noise in inherently noisy financial data
- Avoiding data snooping bias and overfitting (the biggest killer of trading strategies)
- Generating insights that are statistically rigorous, not just correlations

CS4100 Relevance: This isn't just applying ML models to data - it's understanding:

- When neural networks are appropriate vs. statistical methods
- How to validate findings in a domain where "future data" is unknowable
- The difference between correlation and causation in market behavior
- How to build AI systems that augment (not replace) human decision-making

The project sits at the intersection of **"What AI Can Do"** and **"What Has Real Value"** - the core framework from CS4100 for choosing meaningful AI projects.

---

3. Proposed Approaches and Methods

Overall Architecture: Hybrid Statistical-ML Pipeline with 4 Stages

This project uses a **hybrid approach** combining traditional statistical analysis (for most research tasks) with machine learning (for regime classification). This design choice balances technical rigor, interpretability, and realistic implementation within the 6-week timeline.

Stage 1: Data Collection & Preprocessing

Data Source: Alpaca Trading API

- Already have API access and tested basic data retrieval
- Free tier provides historical and real-time data for stocks, crypto, and forex

Initial Asset Scope:

- **Primary focus:** XAU-USD (Gold) - high liquidity, well-behaved data
- **Expansion if time permits:** BTC-USD, EUR-USD, SPY

Data Collection:

- **Timeframes:** 1-minute, 5-minute, 1-hour, 4-hour, daily
- **Data types:** OHLCV (Open, High, Low, Close, Volume)
- **Historical depth:** 2-3 years of data (sufficient for statistical analysis)

Technical Indicator Calculation: All indicators calculated using Python libraries (pandas, TA-Lib):

1. **RSI (14-period)** - Relative Strength Index
2. **MACD (12, 26, 9)** - Moving Average Convergence Divergence

3. **ATR (14-period)** - Average True Range
4. **SMA (50-period & 200-period)** - Simple Moving Averages
5. **Volume** - Raw volume data
6. **VWAP** - Volume Weighted Average Price (intraday)

Data Validation:

- Handle missing data (forward fill, interpolation)
- Outlier detection and handling
- Ensure data quality before analysis

Technology Stack:

- Python 3.x with pandas, numpy
- Alpaca Trade API library
- Storage: CSV files initially, can migrate to SQLite/PostgreSQL if needed

---

Stage 2: Statistical Analysis Module

This stage uses **pure statistical methods** (no machine learning) to analyze market behavior. Follows the **Data Science Workflow** from CS4100.

2.1 Volume Pattern Analysis

- Volume distribution across timeframes (which timeframe has most activity?)
- Intraday volume patterns (when is volume highest? UTC hour-by-hour analysis)
- Volume-price relationship (correlation between volume spikes and price moves)
- Statistical measures: mean, median, standard deviation, percentiles
- **Visualizations:**
  - Volume heatmap by hour and day of week
  - Bar chart comparing average volume across timeframes
  - Volume-price correlation scatter plot
- **Output:** Volume analysis with statistical evidence and visual representations

2.2 Volatility Analysis

- Calculate ATR across all timeframes
- Intraday volatility patterns (when is volatility highest?)
- **Volatility clustering analysis:**
  - Test if high volatility periods cluster together (persistence)
  - Measure autocorrelation in volatility (GARCH-style analysis)
  - Quantify how long volatility regimes typically last
  - Research shows volatility clustering is critical for risk management
- Statistical measures: rolling standard deviation, coefficient of variation

- **Output:** Volatility profiles by time of day/week, statistical distributions, volatility clustering metrics

2.3 Trend Characteristics

- Measure trend duration (how long do uptrends/downtrends last?)
- Typical pullback/rally magnitudes during trends
- Trend strength metrics (slope, consistency)
- Statistical tests: t-tests for significance, confidence intervals on measurements
- **Output:** Trend behavior statistics with sample sizes and p-values

2.4 Indicator Effectiveness Testing For each of the 6 indicators, test:

- **Predictive power:** Does indicator signal correlate with future price movement?
- **Optimal parameters:** Do default parameters work, or are asset-specific values better?
- **Timeframe sensitivity:** Which timeframe gives strongest signals?
- **Entropy-based quality scoring:** Measure signal quality using Shannon entropy
  - For each indicator signal (e.g., RSI < 30), calculate entropy of subsequent price movements
  - Low entropy = consistent directional outcomes = high quality signal
  - High entropy = random/ambiguous outcomes = low quality signal
  - This approach based on recent quantitative research on pattern quality identification
- Statistical validation: significance tests, avoiding p-hacking
- **Output:** Ranked indicator effectiveness table with both traditional metrics and entropy-based quality scores

2.5 Correlation Analysis

- Correlation with other major assets (SPY, BTC, DXY, etc.)
- Rolling correlations over time (do relationships change?)
- Lead-lag relationships (does one asset predict another?)
- **Professional market statistics:**
  - Cross-sectional dispersion: measure of return variation across related assets
  - Pairwise correlation metrics: average correlation strength
  - These metrics used by institutional quant research (UBS, Goldman, etc.)
- **Visualizations:**
  - Correlation heatmap matrix
  - Time-series line charts showing rolling correlations
  - Lead-lag relationship visualizations
- **Output:** Correlation matrix, correlation plots, market breadth statistics

2.6 Trading Activity Patterns

- Which timeframes are used most by traders? (inferred from volume concentration)
- Time-of-day analysis: when do traders enter positions vs. manage positions?

- Day-of-week effects (if any)
- **Output:** Trading behavior insights

Stage 3: Machine Learning - Market Regime Classification

This is the **single focused ML component** of the project, applying the **Machine Learning Workflow** from CS4100.

Problem Formulation:

- **Input A:** Market features at time t
    - Volatility level (ATR percentile)
    - Volume characteristics (relative to average)
    - Price vs. moving averages (above/below SMA-50, SMA-200)
    - RSI level, MACD state
    - Recent return patterns
- **Output B:** Market regime classification
    - Class 0: **Ranging/Consolidation** (low volatility, no clear trend)
    - Class 1: **Trending Up** (clear upward momentum)
    - Class 2: **Trending Down** (clear downward momentum)

Why ML for This Task?

- Regime boundaries are fuzzy, not clear-cut thresholds
- Multiple features interact in complex ways
- Neural networks can learn non-linear regime boundaries
- Human labeling provides ground truth for supervised learning

Regime Labeling Approach (Hybrid Method):

Phase 1: Automatic Heuristic Labeling Instead of manual labeling, use rule-based heuristics to automatically label historical periods:

- ADX > 25 AND price > SMA-50 → Label as "Trending Up"
- ADX > 25 AND price < SMA-50 → Label as "Trending Down"
- ADX < 25 → Label as "Ranging"
- Generates 500-1000+ labeled examples automatically (saves 10-15 hours)
- Interpretable and fast
- Based on established technical analysis principles

Phase 2: Supervised Learning

1. **Feature Engineering:** Normalize features, create rolling window statistics

2. **Train/Validation/Test Split:** 60% / 20% / 20% (time-series aware - no future data in training)
3. **Neural Network Training:** Learn nuanced boundaries beyond simple heuristics

Phase 3: Unsupervised Validation (Optional - if time permits)

- Apply K-Means or GMM clustering to same market features
- Compare clusters to neural network predictions
- If alignment > 80% → regime definitions are robust and validated
- If misalignment → investigate feature engineering or regime definitions
- Research shows K-Means and GMM effective for financial regime detection
- Provides independent validation without confirmation bias

Advantages of Hybrid Approach:

- No time-consuming manual labeling required
- Heuristics provide interpretable baseline
- ML learns complex patterns heuristics miss
- Unsupervised clustering validates regime definitions
- More scalable and reproducible

Model Architecture:

- **Start simple:** Feedforward neural network (3-4 layers) in PyTorch
  - Input layer: 10-15 features
  - Hidden layers: 64 → 32 neurons with ReLU activation
  - Output layer: 3 classes (softmax)
- **Regularization:** Dropout (0.3), L2 regularization to prevent overfitting
- **Loss function:** Cross-entropy loss
- **Optimizer:** Adam

Training Process:

1. Train model on training set
2. Monitor validation accuracy (target: >70% for "good enough")
3. Use techniques from CS4100: backpropagation, gradient descent, regularization
4. Iterate on architecture if needed (but keep it simple)

Evaluation:

- Classification accuracy on test set
- Confusion matrix (which regimes are hardest to distinguish?)
- Avoid overfitting: if train accuracy >> test accuracy, add more regularization

Integration with Pipeline:

- Once trained, model classifies current regime for any asset
- Regime classification informs the research report (e.g., "Currently in Trending Up regime, momentum strategies favored")
- **Visualizations:**
    - Color-coded timeline showing regime changes over analysis period
    - Regime distribution pie chart (% of time in each regime)
    - Regime transition probability matrix

Stage 4: Report Generation & Integration

Combine All Findings:

1. Statistical analysis results from Stage 2
2. ML regime classification from Stage 3
3. Synthesize into comprehensive research report

Report Structure (based on Bitcoin example format):

- Executive Summary
- Volume Analysis (with statistical evidence)
- Volatility Patterns (with confidence intervals)
- Trend Characteristics (with sample sizes, p-values)
- Indicator Effectiveness (ranked with significance tests)
- Current Market Regime (from ML model)
- Correlation Analysis
- Key Takeaways & Statistical Summary

Output Format:

- **Primary:** Markdown format report (easy to read, version control friendly)
- **Visualizations:**
    - Volume heatmaps and distribution charts
    - Correlation heatmap matrix and rolling correlation line charts
    - Regime classification timeline and distribution visualizations
    - All charts saved as PNG files, embedded in markdown
- **Optional:** HTML dashboard if time permits

Technology:

- Report generation: Python with Jinja2 templates or simple string formatting
- Visualizations: matplotlib, seaborn, plotly
- Export: Markdown → can convert to PDF if needed

Statistical Guardrails

To ensure findings remain valid when market conditions change, the following guardrails will be implemented:

1. Time-Series Cross-Validation

- **Walk-forward validation:** Train on months 1-12, test on month 13; then train on months 1-13, test on month 14, etc.
- Never use future data in training (prevents look-ahead bias)
- Validation set always comes after training set chronologically
- Critical for financial time series where temporal ordering matters

2. Regime-Stratified Testing Test every indicator and pattern separately across different market regimes:

- Trending up markets (ADX >25, price rising)
- Trending down markets (ADX >25, price falling)
- Ranging markets (ADX <25)

Only report findings that demonstrate robustness across multiple regimes. Explicitly flag any patterns that are regime-specific. Research shows factor exposures are volatile and regime-dependent - patterns valid in bull markets may fail in bear markets.

3. Stability Analysis via Rolling Windows

- Calculate all metrics using 6-month rolling windows
- Report: mean, standard deviation, minimum, maximum
- Example format: "RSI <30 win rate: 72% ± 8% (range: 58-84%)"
- High standard deviation (>20% of mean) = warning sign of instability
- Visualize metric stability over time

4. Robustness Through Perturbation Testing Following SAFE framework methodology:

- Add random ±5% noise to price data
- Add random ±10% noise to volume data
- Re-run analysis on perturbed data
- If findings still hold (degradation <30%) = robust
- If findings disappear = likely overfit to noise
- Calculate robustness score: correlation between original and perturbed results
- Research shows robust models maintain 0.85+ correlation under perturbation

5. Regularization in Machine Learning Components

- **L2 regularization** on neural network weights (prevents overfitting)
- **Dropout (0.3)** during training (forces redundant learning)

- **Early stopping** based on validation loss (stops before overfitting)
- Target: train accuracy ≈ validation accuracy (within 5%)
- Monitor for train >> validation accuracy = overfitting signal

6. Conservative Statistical Reporting

- Report confidence intervals, not just point estimates
- Require p-value < 0.05 for statistical significance (95% confidence)
- Flag findings with small sample sizes (n < 100 observations)
- Include effect sizes, not just significance tests
- Explicitly acknowledge limitations in report
- Use bootstrap methods for robust confidence intervals

7. Out-of-Sample Validation Protocol

- **Training period:** 2022-2024 data (used for all analysis and model development)
- **Hold-out test period:** 2025 data (never used during development)
- System validation only on completely unseen 2025 data
- If patterns hold on 2025 data = truly predictive
- If patterns fail on 2025 data = overfit to historical period

Success Criteria for Statistical Validity:

- Findings hold on out-of-sample 2025 test data
- Patterns work in at least 2 of 3 market regimes
- Metrics remain stable across rolling windows (standard deviation < 20% of mean)
- Performance degrades gracefully under perturbation (<30% decline)
- All reported findings achieve $p < 0.05$ statistical significance
- Neural network shows train/validation accuracy gap < 5%

Rationale & Justification

Why a Hybrid Statistical-ML Approach?

Statistical Methods (Stage 2):

- **Interpretability:** Statistical results are explainable - you can show exactly why a pattern is significant
- **Proven reliability:** Quantitative researchers have used these methods for decades
- **No overfitting risk:** Well-established statistical tests with clear significance thresholds
- **Faster implementation:** No training required, just computation
- **Best for:** Descriptive analysis, measuring relationships, testing significance

Machine Learning (Stage 3):

- **Handles complexity:** Regime boundaries are fuzzy and multi-dimensional - perfect for neural networks
- **Pattern recognition:** ML excels at finding non-linear relationships in feature combinations
- **Adaptive:** Can learn asset-specific regime characteristics
- **CS4100 relevance:** Demonstrates understanding of when to apply neural networks vs. statistical methods
- **Best for:** Classification tasks with complex, non-linear decision boundaries

Why Not Pure ML?

- Overfitting is the #1 killer of trading systems
- Financial data is noisy and non-stationary
- Statistical methods provide more reliable baseline insights
- Interpretability matters when making real money decisions

Why Not Pure Statistics?

- Some patterns (like regime detection) genuinely benefit from ML
- Demonstrates technical ML competency for CS4100
- Sets foundation for expanding ML components later

Why Regime Classification is the Right ML Task?

1. **Well-defined problem:** Clear input features and output classes
2. **Feasible labeling:** Can manually label 500-1000 historical periods in reasonable time
3. **High impact:** Knowing the regime fundamentally changes how you interpret other indicators
4. **Appropriate complexity:** Not too simple (could solve with thresholds) but not too complex (no need for deep learning or transformers)
5. **Demonstrates CS4100 concepts:** Neural networks, backpropagation, regularization, train/val/test methodology
6. **Actually useful:** Professional quant researchers do care about regime identification

Technical Diligence (CS4100 Framework)

Can AI meet desired performance?

- **Statistical analysis:** Yes - deterministic calculations, no accuracy threshold needed
- **Regime classification:** Target 70%+ accuracy is realistic
    - Even 60-65% would be useful (better than random guessing at 33%)
    - Not aiming for perfect prediction (impossible in markets)
    - Success = model learns meaningful patterns

How much data is needed?

- **Statistical analysis:** 2-3 years of historical data sufficient
    - Minute data: ~500,000+ data points per asset
    - Plenty for robust statistical analysis
- **Regime classification:** Need 500-1000 labeled examples
    - Can label ~50-100 periods per hour of work
    - 10-15 hours of labeling = feasible
- **Data availability:** Alpaca provides years of free historical data

Engineering timeline?

- **Realistic for 6 weeks working solo** (see detailed timeline below)
- Staged approach allows incremental progress
- Can ship "MVP" even if some features incomplete
- Exam period (Nov 20-23) accounted for with buffer time

Business Diligence (CS4100 Framework)

While this is primarily an academic project, applying the business framework:

Value Proposition:

- **Immediate:** Saves hours of manual research per asset analyzed
- **Scalable:** Once built, can analyze unlimited assets quickly
- **Reusable:** Foundation for future quantitative trading system

Future Potential:

- Research assistant (this project) → Strategy development → Backtesting → Live trading
- Could become actual profit-generating system
- Demonstrates real-world value to potential employers

Why This Beats Existing Tools: Most retail trading platforms provide generic indicators, not asset-specific research intelligence. Professional quant research tools (Bloomberg Terminal, FactSet) cost $20K-40K/year. This creates a free, customizable alternative focused on microstructure analysis.

Risk Mitigation in Design Choices

Overfitting Prevention:

- Most analysis is statistical (no training = no overfitting)
- Single ML component with proper train/val/test split
- Regularization (dropout, L2) built into regime classifier
- Manual labeling ensures ground truth quality

Scope Management:

- Start with single asset (XAU-USD)
- Core indicators only (6 total - not trying to test everything)
- One focused ML task (not multiple models)
- Can reduce features if timeline pressures emerge

Data Quality:

- Using Alpaca (reliable, professional data provider)
- Built-in data validation and cleaning steps
- Can verify against free sources (Yahoo Finance) if needed

Technical Feasibility:

- Hybrid approach reduces ML complexity
- PyTorch covered in CS4100 lectures
- Strong Python and math foundation
- Can leverage existing libraries (pandas, sklearn, TA-Lib)

---

4. Project Timeline (Weekly Breakdown)

Timeline Overview: Oct 30 - Dec 8 (6 weeks)
Estimated Weekly Effort: 10-15 hours (adjusted for exam period)
Total Estimated Hours: ~72 hours

Note: This timeline accounts for the exam period (Nov 20-23) with reduced capacity and builds in buffer time. The project is scoped to be achievable as a solo developer with realistic time constraints.

Week 1 (Oct 30 - Nov 5): Data Infrastructure & Exploration

Hours: 12-15 hours
Primary Goal: Get data flowing and understand its structure

Tasks:

- Set up project repository and development environment
- Alpaca API integration and authentication
- Pull historical data for XAU-USD (2-3 years, multiple timeframes)
- Data storage setup (CSV files, organized folder structure)
- Basic data validation and quality checks
- Calculate all 6 technical indicators (RSI, MACD, ATR, SMA-50/200, Volume, VWAP)
- Initial exploratory visualization (price charts with indicators)

Deliverable:

- Working data pipeline that can fetch and process XAU-USD data
- Clean dataset with all indicators calculated
- Basic visualization notebook

Success Criteria: Can load data, calculate indicators, plot charts

---

Week 2 (Nov 6 - Nov 12): Statistical Analysis - Volume & Volatility

Hours: 10-12 hours
Primary Goal: Complete volume and volatility analysis (Stage 2, part 1)

Tasks:

- **Volume Analysis:**
  - Volume distribution across timeframes
  - Intraday volume patterns (hour-by-hour analysis)
  - Volume-price correlation analysis
  - **Create visualizations:**
    - Volume heatmap (hour × day of week)
    - Bar chart of volume by timeframe
    - Volume-price scatter plot
- **Volatility Analysis:**
  - ATR analysis across timeframes
  - Intraday volatility patterns
  - Volatility clustering detection and metrics
  - Statistical measures (std dev, coefficient of variation)

Deliverable:

- Complete volume analysis with visualizations
- Complete volatility analysis with statistical measures
- Document findings in structured format

Success Criteria: Can answer "When is volume highest?" and "When is volatility highest?" with statistical evidence and clear visualizations

---

Week 3 (Nov 13 - Nov 19): Statistical Analysis - Trends, Indicators & Correlations

Hours: 12-15 hours
Primary Goal: Complete trend, indicator, and correlation analysis (Stage 2, part 2)

Tasks:

- **Trend Characteristics:**
  - Measure trend durations (uptrends vs. downtrends)
  - Pullback/rally magnitudes
  - Statistical tests with confidence intervals
- **Indicator Effectiveness:**
  - Test each of 6 indicators for predictive power
  - Calculate entropy-based quality scores
  - Optimal parameter testing
  - Timeframe sensitivity analysis
  - Statistical significance testing
- **Correlation Analysis:**
  - Correlation with other major assets
  - Rolling correlations over time
  - Professional market statistics (cross-sectional dispersion, pairwise correlation)
  - **Create visualizations:**
    - Correlation heatmap matrix
    - Rolling correlation line charts
    - Market breadth statistics charts

Deliverable:

- Trend analysis with statistical evidence
- Ranked indicator effectiveness table with entropy scores
- Correlation analysis with visualizations

Success Criteria: Can answer "Which indicators work best for XAU-USD?" with quantitative evidence and can show how correlations with other assets change over time

---

Week 4 (Nov 20 - Nov 26): Regime Classification - Data Prep & Initial Model

Hours: 3-4 hours (EXAM WEEK Nov 20-23) + 8-10 hours after exams
Primary Goal: Prepare regime classification data and build initial model

Early Week (Nov 20-23) - EXAM PERIOD:

- Light tasks only: Automatic regime labeling using heuristics
  - Implement ADX + SMA-based rules to auto-label regimes
  - Generate labeled dataset (500-1000 examples)
  - No manual work required - fully automated

- Validate labels make intuitive sense (visual spot-check)

Late Week (Nov 24-26) - POST-EXAM:

- Feature engineering for regime classification
- Train/validation/test split (time-series aware, 60/20/20)
- Build simple feedforward neural network in PyTorch
- Initial training runs with regularization (L2, dropout 0.3)
- Basic evaluation (accuracy on validation set)
- Check for overfitting (train vs. validation accuracy gap)

Deliverable:

- Automatically labeled dataset with 500+ examples (complete by end of week)
- Working PyTorch regime classifier (even if accuracy needs improvement)

Success Criteria: Model trains without errors, achieves >50% validation accuracy (better than random), train/val accuracy gap <10%

Week 5 (Nov 27 - Dec 3): Regime Model Refinement & Report Generation

Hours: 12-15 hours
Primary Goal: Improve regime classifier and start building report

Tasks:

- **Model Improvement:**
  - Tune hyperparameters (learning rate, layers, dropout)
  - Add regularization if overfitting
  - Iterate until validation accuracy >70% (or plateau)
  - Final evaluation on test set
  - Error analysis (which regimes are hardest?)
  - **Robustness testing:**
    - Test model on perturbed data (±5% price noise, ±10% volume noise)
    - Calculate robustness score (target: >0.85 correlation)
    - Verify performance degrades gracefully (<30% decline)
  - **Optional: Unsupervised validation**
    - Apply K-Means clustering to market features
    - Compare clusters to NN predictions (target: >80% alignment)
    - Validates regime definitions independently
  - **Create regime visualizations:**
    - Color-coded timeline of regime classifications
    - Regime distribution pie chart
    - Confusion matrix

        - Robustness comparison chart (original vs. perturbed)
- **Report Generation Framework:**
    - Design report template/structure
    - Build code to generate sections automatically
    - Integrate statistical findings from Weeks 2-3
    - Add regime classification results and visualizations
    - Integrate all visualizations (volume heatmaps, correlation charts, regime timelines)
    - Add statistical guardrails to all reported findings (confidence intervals, p-values)

Deliverable:

- Final regime classification model (saved weights)
- Model evaluation report (accuracy, confusion matrix, robustness score, visualizations)
- Draft research report for XAU-USD with most sections complete and all key visualizations
- Statistical validation results showing robustness

Success Criteria: Model achieves 65-70%+ accuracy, robustness score >0.80, draft report contains all main sections with professional visualizations and statistical guardrails

---

Week 6 (Dec 4 - Dec 8): Final Integration, Testing & Documentation

Hours: 10-12 hours
Primary Goal: Polish everything and prepare final submission

Tasks:

- **Report Finalization:**
    - Complete all sections of XAU-USD research report
    - Final visualizations and formatting
    - Executive summary and key takeaways
    - Proofread and polish
- **Code Documentation:**
    - Clean up code, add comments
    - Create comprehensive README
    - Document how to run the system
    - Include requirements.txt for dependencies
- **Final Presentation Prep:**
    - Create slides for final presentation
    - Prepare demo (show live report generation if possible)
    - Practice timing (likely 7-10 minutes)
- **Optional (if time):**
    - Run system on second asset (BTC-USD) to show generalizability

- Additional visualizations

Deliverable:

- Complete, polished XAU-USD research report
- Clean, documented codebase with README
- Final presentation slides
- (Optional) Report for second asset

Success Criteria: Professional-quality submission ready by Dec 8

Timeline Risk Management

Built-in Buffers:

- Week 4 accounts for exam period with minimal work
- Each week slightly underscoped to allow flexibility
- "Optional" tasks in Week 6 provide breathing room

If Running Behind:

- **Priority 1 (Must Have):** Stages 1-2 (statistical analysis) + basic report
- **Priority 2 (Should Have):** Stage 3 (regime classifier with 60%+ accuracy)
- **Priority 3 (Nice to Have):** Polished report, second asset analysis

If Ahead of Schedule:

- Add second asset (BTC-USD or EUR-USD)
- Improve regime classifier architecture
- Create interactive HTML dashboard
- More sophisticated visualizations

Weekly Checkpoint Questions

To stay on track, ask these at the end of each week:

1. Did I complete this week's deliverable?
2. Is my code working and tested?
3. What blockers do I need to address next week?
4. Am I on pace for final deadline?

If behind by more than one week's work, reassess scope and cut lower-priority features.

---

5. Expected Outcomes & Deliverables

Primary Success Criteria

The project will be considered successful if it achieves these two core outcomes:

1. Working System That Generates Research Reports

- System can ingest any asset ticker (via Alpaca API)
- Automatically performs all analysis stages
- Outputs comprehensive, structured research report
- Process is repeatable and automated (not manual analysis disguised as AI)

2. High-Quality Research Report for XAU-USD

- Provides actionable quantitative insights about Gold trading
- Contains statistically rigorous analysis with confidence intervals, p-values, sample sizes
- Includes clear visualizations and data-driven conclusions
- Would be valuable to an actual quantitative trader researching this asset
- Demonstrates understanding of market microstructure, not just surface-level analysis

---

Academic Deliverables (for CS4100)

1. Comprehensive Research Report for XAU-USD

Content Requirements:

- Executive Summary with key findings
- Volume Analysis:
    - Volume distribution across timeframes with statistical measures
    - Intraday volume patterns (when is trading most active?)
    - Volume-price relationship analysis
- Volatility Analysis:
    - ATR analysis across timeframes
    - Volatility patterns by time of day/week
    - Volatility clustering metrics and persistence analysis
    - Statistical distributions
- Trend Characteristics:
    - Typical trend durations with confidence intervals
    - Pullback/rally magnitudes during trends
    - Statistical significance tests with p-values and sample sizes
- Indicator Effectiveness:

- Ranked effectiveness of all 6 indicators (RSI, MACD, ATR, SMA-50/200, Volume, VWAP)
    - Entropy-based quality scores for each indicator
    - Optimal parameters and timeframes for each
    - Statistical validation (significance tests)
    - Regime-specific performance (which indicators work in which regimes)
- Market Regime Analysis:
    - Current regime classification (from ML model)
    - Regime distribution over analysis period
    - Regime-specific characteristics
    - Robustness validation results
- Correlation Analysis:
    - Relationships with other major assets
    - Time-varying correlations if applicable
    - Professional market statistics (cross-sectional dispersion, pairwise correlation)
- Statistical Guardrails Results:
    - Rolling window stability analysis
    - Perturbation testing results
    - Out-of-sample validation on 2025 data
- Key Takeaways:
    - What makes XAU-USD unique?
    - Highest-probability patterns identified
    - Statistical summary of findings with confidence intervals

Quality Metrics:

- All claims backed by statistical evidence (confidence intervals, p-values < 0.05, sample sizes)
- Clear, professional visualizations (8+ charts)
- Actionable insights (not just "here's some data")
- Would pass scrutiny from actual quant researcher
- Findings validated across multiple market regimes
- Robustness demonstrated through perturbation testing

2. Working Python System

System Capabilities:

- Data collection from Alpaca API
- Technical indicator calculation (6 indicators)
- Statistical analysis automation
- Regime classification using trained ML model
- Automated report generation
- Can be run on new assets (though primary focus is XAU-USD)

Code Organization:

- Modular structure (separate files for data, analysis, ML, reporting)
- Clear documentation and comments
- Requirements file for dependencies
- README with usage instructions

3. Technical Documentation

System Architecture Document:

- Overview of 4-stage pipeline
- Data flow through system
- Key design decisions and rationale
- Technologies used

ML Model Report:

- Regime classification approach
- Training methodology
- Performance metrics (accuracy, confusion matrix)
- Error analysis and limitations

4. Final Presentation

Format: 7-10 minute presentation with slides
Content:

- Problem and motivation
- System architecture overview
- Key findings from XAU-USD analysis (show the actual report)
- ML regime classifier performance
- Demo (if time permits)
- Lessons learned and future work

Evaluation Metrics

Primary Metrics (What Matters Most):

1. **System Functionality (35%)**
    - Does it work end-to-end without errors?
    - Can it successfully analyze XAU-USD and generate report?
    - Is the process automated, or requires manual intervention?
    - Could it theoretically be run on other assets?
2. **Research Report Quality (35%)**
    - Statistical rigor (confidence intervals, significance tests, proper methodology)

- Depth of insights (surface-level vs. genuinely useful findings)
    - Clarity and professionalism of presentation
    - Actionability for actual trading research
    - Validation across market regimes
    - Robustness demonstrated
3. **Quantitative Validation (20%)**
    - **Benchmark comparison:** Do entropy-scored indicators outperform standard usage?
    - **Out-of-sample performance:** Do findings hold on 2025 test data?
    - **Robustness score:** Correlation >0.80 between original and perturbed analysis
    - **Statistical significance:** All major findings achieve $p < 0.05$
    - **Regime stability:** Patterns work in at least 2 of 3 market regimes

   Validation Framework (based on quantitative finance research):

    - **Sharpe Ratio:** Risk-adjusted returns of insights-based simple strategies
    - **Win Rate:** Percentage of profitable signals
    - **Maximum Drawdown:** Worst-case loss from peak
    - **Information Ratio:** Excess returns relative to volatility
    - Target: Insights-based strategies achieve Sharpe >1.0 vs. baseline <0.5
4. **ML Component (10%)**
    - Does regime classifier achieve >65% accuracy?
    - Proper train/val/test methodology
    - Demonstrates understanding of PyTorch and neural networks
    - Reasonable error analysis and robustness testing

Note: Emphasis is on building something that *works*, produces *quality output* with *statistical validity*, not just demonstrating ML techniques. The ML component supports the research goal but isn't the main point.

Beyond the Course (Reusable Components)

While the course project focuses on XAU-USD analysis, the system is designed for future expansion:

Immediate Extensions (post-course):

- Analyze multiple assets (BTC, EUR-USD, SPY, etc.)
- Build asset comparison reports ("Which assets are most correlated?")
- Expand indicator library based on asset-specific findings

Medium-term Evolution:

- Strategy development based on research findings

- Backtesting framework to validate hypotheses
- Risk management modules
- Portfolio-level analysis

Long-term Vision:

- Real-time data streaming and analysis
- Automated signal generation (research → signals)
- Paper trading integration
- Eventually: Live trading system

Career Value:

- Demonstrates quantitative research capability
- Shows understanding of market microstructure
- Portfolio piece for quant trading applications
- Foundation for actual profitable trading system

Minimum Viable Project vs. Stretch Goals

Minimum Viable (Must Complete for Passing Grade):

- ✅ Working data pipeline with XAU-USD data
- ✅ Statistical analysis for at least: volume patterns, volatility, indicator effectiveness
- ✅ Basic regime classifier (even if only 60% accuracy)
- ✅ Generated research report with key sections
- ✅ Functional code that runs

Target Outcome (Full Credit):

- ✅ All statistical analysis sections complete with strong evidence
- ✅ Regime classifier >70% accuracy with proper methodology
- ✅ Professional-quality research report
- ✅ Clean, documented code
- ✅ Polished final presentation

Stretch Goals (If Ahead of Schedule):

- Analyze second asset to demonstrate generalizability
- Interactive HTML dashboard for reports
- More sophisticated ML architectures
- Additional analysis types (seasonality, order flow, etc.)
- Open-source publication on GitHub

Success Definition

The project is successful if:

A quantitative trader could pick up the XAU-USD research report, read it in 15-20 minutes, and walk away with 3-5 actionable insights about how Gold behaves that they didn't know before, all backed by statistical evidence.

AND

The system can be pointed at a different asset (e.g., BTC-USD), run the analysis pipeline, and produce a similarly valuable research report without requiring code changes.

This demonstrates both *quality of output* and *generalizability of system* - the two core goals of the project.

Technical Stack

Core Technologies:

- **Language:** Python 3.x
- **ML/DL:** PyTorch (covered in CS4100), scikit-learn
- **Data Processing:** pandas, numpy
- **Visualization:** matplotlib, seaborn, plotly
- **API:** Alpaca Trading API
- **Version Control:** Git/GitHub

Key Libraries:

- **TA-Lib or pandas-ta:** Technical indicator calculations
- **statsmodels:** Statistical analysis
- **backtrader or vectorbt:** Backtesting framework

Risk Mitigation

Potential Challenges & Solutions:

1. **Data Quality Issues**
    - *Risk:* Missing data, outliers in market data

- *Mitigation:* Robust data cleaning pipeline, outlier detection

2. **Overfitting**
    - *Risk:* Model memorizes training data, fails on new data
    - *Mitigation:* Train/validation/test split, regularization techniques from CS4100, cross-validation
3. **Market Regime Changes**
    - *Risk:* Patterns that worked historically may not work currently
    - *Mitigation:* Focus on robust patterns, ensemble methods, clearly document assumptions
4. **Scope Creep**
    - *Risk:* Project becomes too ambitious for 6-week timeline
    - *Mitigation:* Start with single asset (XAU-USD), simple models, expand if time permits

---

Success Criteria

Minimum Viable Project (for course):

- ✅ Working data collection pipeline
- ✅ Pattern analysis identifying at least 3-5 statistical patterns
- ✅ Trained hypothesis generation model (>60% relevance)
- ✅ Basic strategy recommendations
- ✅ Documented codebase and final presentation

Stretch Goals:

- Multi-asset analysis (3+ currency pairs)
- Real-time data streaming
- Web dashboard for visualization
- Advanced model architectures (LSTM, attention mechanisms)
- Published as open-source project

Conclusion

This project sits at the intersection of **"What AI Can Do"** and **"Valuable Application"** (CS4100 framework for choosing AI projects). It applies neural networks, data science workflows, and AI pipeline architecture to a challenging real-world problem with clear evaluation metrics and future extensibility. The 6-week timeline is realistic with staged development, and the deliverables meet both academic requirements and personal career goals in quantitative trading.