Machine Learning Engineer Nanodegree

Capstone Proposal

Student Name: Matthew Rivera
Date: 10/12/2025

Domain Background

Algorithmic trading and quantitative finance have become increasingly reliant on machine learning techniques for price prediction and market analysis. The financial technology sector has seen significant growth, with machine learning applications ranging from high-frequency trading to portfolio optimization. According to recent industry reports, the global algorithmic trading market is expected to grow substantially, driven by advances in artificial intelligence and data analytics.

Stock price prediction represents a challenging time series forecasting problem due to market volatility, non-stationarity, and the influence of numerous external factors. Traditional approaches like ARIMA and exponential smoothing have limitations in capturing complex non-linear patterns. Modern machine learning techniques, particularly ensemble methods and deep learning, have shown promise in improving prediction accuracy by leveraging technical indicators and historical patterns.

This project focuses on short-term stock price prediction using multiple machine learning algorithms combined with technical analysis. The goal is to develop a system that can assist investors in making data-driven decisions while acknowledging the inherent unpredictability of financial markets.

Problem Statement

The core problem is to predict the adjusted closing price of stocks at future time points (1, 7, 14, and 28 days ahead) given historical trading data and technical indicators. Specifically, the system must:

1. Accept a training period (start_date, end_date) and list of stock tickers
2. Build predictive models using historical price data and engineered features
3. Generate price predictions for specified future dates
4. Quantify prediction uncertainty and evaluate performance across different time horizons

The problem is quantifiable through metrics like Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and directional accuracy. It is measurable because we can compare predictions against actual future prices. The solution is replicable as it uses publicly available financial data from Yahoo Finance API.

Success criteria: Achieve MAPE < 5% for 7-day predictions and directional accuracy > 60%, demonstrating that the model provides meaningful predictive value over naive baseline approaches.

Datasets and Inputs

Data Source: Yahoo Finance API via the yfinance Python library

Stocks Selected: Five technology sector stocks - AAPL (Apple), GOOGL (Alphabet), MSFT (Microsoft), TSLA (Tesla), and AMZN (Amazon)

Time Period: January 1, 2020 to October 1, 2024 (approximately 1,190 trading days)

Raw Features per Stock:

- Open, High, Low, Close, Volume, Adjusted Close prices
- Daily granularity (one record per trading day)

Engineered Features (30+ total):

- **Trend Indicators:** Simple Moving Averages (7, 21, 50-day), Exponential Moving Averages (12, 26-day)
- **Momentum Indicators:** RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence)
- **Volatility Indicators:** Bollinger Bands (upper, middle, lower), historical volatility
- **Volume Indicators:** Volume moving average, volume rate of change
- **Lag Features:** Previous 1, 2, 3, 5, 7-day closing prices
- **Statistical Features:** Rolling mean, rolling standard deviation

Data Split:

- Training: 80% (2020-01-01 to 2023-12-31)
- Validation: 10% (2024-01-01 to 2024-06-30)
- Testing: 10% (2024-07-01 to 2024-10-01)

The data is publicly accessible, requires no special permissions, and contains no missing values requiring imputation beyond standard forward-fill for technical indicator calculations.

Solution Statement

The proposed solution implements an ensemble machine learning system that combines multiple algorithms to predict stock prices:

Model Architecture:

1. **Baseline:** Linear Regression to establish performance floor

2. **Tree-Based Models:** Random Forest and Gradient Boosting for capturing non-linear relationships
3. **Deep Learning:** LSTM (Long Short-Term Memory) networks for temporal dependencies
4. **Ensemble:** Weighted combination of top-performing models

Implementation Approach:

1. **Data Preprocessing:** Download historical data, handle missing values, calculate technical indicators
2. **Feature Engineering:** Create 30+ features from raw OHLCV data
3. **Feature Scaling:** Standardize features using StandardScaler for neural networks
4. **Model Training:** Train each model independently with time-series cross-validation
5. **Hyperparameter Tuning:** Use GridSearchCV with TimeSeriesSplit to optimize parameters
6. **Ensemble Creation:** Combine predictions using weighted averaging based on validation performance
7. **Evaluation:** Test on hold-out set across multiple prediction horizons

Key Implementation Details:

- Use walk-forward validation to prevent look-ahead bias
- Implement proper temporal ordering (no random shuffling)
- Generate predictions at 1, 7, 14, and 28-day horizons
- Provide confidence intervals for risk assessment

This solution is replicable with clear implementation steps and measurable outcomes at each stage.

Benchmark Model

The benchmark is a **naive persistence forecast**, which predicts that tomorrow's price equals today's price. This is expressed as:

$P(t+h) = P(t)$

where $P(t)$ is the current price and h is the prediction horizon.

For the validation period (2024-01-01 to 2024-06-30), preliminary analysis shows:

- Naive MAPE: ~4.8%
- Naive Direction Accuracy: ~52% (barely better than random)
- Naive R^2: ~0.73

This benchmark represents the minimum acceptable performance. Any model that cannot beat this simple heuristic fails to add value. Historical literature suggests that well-designed ML models can achieve 15-30% improvement over naive forecasts for short-term predictions.

The benchmark is appropriate because:

1. It requires no training or computation
2. It's commonly used in time series literature

3. It provides a clear, interpretable baseline
4. It represents what would happen with no sophisticated modeling

---

Evaluation Metrics

Primary Metrics:

1. **Mean Absolute Percentage Error (MAPE)**
    - Formula: MAPE = $(1/n) \times \Sigma|((y\_true - y\_pred) / y\_true)| \times 100$
    - Interpretation: Average prediction error as percentage of actual price
    - Target: < 5% for 7-day predictions
    - Justification: Scale-independent, interpretable to non-technical stakeholders
2. **Root Mean Squared Error (RMSE)**
    - Formula: RMSE = $\sqrt{[(1/n) \times \Sigma(y\_true - y\_pred)^2]}$
    - Interpretation: Average magnitude of prediction errors in dollars
    - Target: Minimize relative to baseline
    - Justification: Penalizes large errors more heavily, standard in regression tasks
3. **Directional Accuracy**
    - Formula: DA = $(1/n) \times \Sigma[sign(y\_true(t) - y\_true(t-1)) == sign(y\_pred(t) - y\_true(t-1))]$
    - Interpretation: Percentage of correct up/down movement predictions
    - Target: > 60%
    - Justification: Critical for trading decisions, even small price prediction errors are acceptable if direction is correct

Secondary Metrics:

4. **R^2 (Coefficient of Determination)**
    - Measures explained variance
    - Target: > 0.85
5. **Mean Absolute Error (MAE)**
    - Average absolute error in dollars
    - Less sensitive to outliers than RMSE

Horizon-Specific Evaluation: All metrics will be computed separately for 1-day, 7-day, 14-day, and 28-day predictions to understand how performance degrades with longer horizons. This provides insight into practical applicability for different trading strategies.

These metrics are appropriate because they capture different aspects of prediction quality (accuracy, bias, directional correctness) and are standard in both academic literature and industry practice for financial forecasting.

Project Design

Phase 1: Data Collection and Exploration (Week 1)

- Download historical data for 5 stocks using yfinance API
- Perform exploratory data analysis: price trends, correlation analysis, volatility patterns
- Visualize distributions of returns and identify potential issues
- Deliverable: EDA notebook with key insights documented

Phase 2: Feature Engineering (Week 1-2)

- Implement technical indicator calculations
- Create lag features and rolling statistics
- Analyze feature correlations to identify redundancy
- Validate feature quality on sample data
- Deliverable: Feature engineering module with 30+ engineered features

Phase 3: Baseline Model Development (Week 2)

- Implement train/validation/test split maintaining temporal order
- Train linear regression baseline
- Evaluate baseline performance using all metrics
- Document results for comparison
- Deliverable: Baseline model with performance benchmarks

Phase 4: Advanced Model Implementation (Week 2-3)

- Implement Random Forest with hyperparameter tuning
- Implement Gradient Boosting with hyperparameter tuning
- Implement LSTM architecture with appropriate sequence length
- Conduct time-series cross-validation for each model
- Deliverable: Three trained advanced models

Phase 5: Ensemble Model Creation (Week 3)

- Analyze validation performance of individual models
- Determine optimal weights for ensemble (e.g., 35% RF, 40% GB, 25% Linear)
- Test ensemble on validation set
- Deliverable: Optimized ensemble model

Phase 6: Final Evaluation (Week 3-4)

- Test all models on hold-out test set
- Generate predictions at multiple horizons (1, 7, 14, 28 days)
- Create comprehensive visualizations: prediction vs actual, error distribution, feature importance
- Perform error analysis to understand failure modes
- Deliverable: Complete evaluation report with visualizations

Phase 7: Documentation (Week 4)

- Write technical blog post documenting methodology and results
- Create GitHub repository with clean, documented code

- Include README with setup instructions and usage examples
- Add notebooks for reproducibility
- Deliverable: Complete project package ready for review

Tools and Libraries:

- Python 3.8+
- Data: pandas, numpy, yfinance
- ML: scikit-learn, tensorflow/keras
- Visualization: matplotlib, seaborn, plotly
- Development: Jupyter notebooks, Git/GitHub

Expected Challenges and Mitigation:

1. **Overfitting:** Use regularization, cross-validation, ensemble methods
2. **Data leakage:** Strict temporal validation, careful feature engineering
3. **Computational cost:** Start with smaller datasets, use efficient implementations
4. **Model interpretability:** Focus on feature importance analysis and error patterns

The project design ensures systematic progress with clear deliverables at each phase, making the work manageable and the results reproducible.