COS30018 - Option C - Task 1: Model Setup and Comparative Analysis Report

Chiraath Madahapola - 104834009

August 24, 2025

Contents

Chapter 1

Executive Summary

This report provides an in-depth evaluation of two LSTM-based stock prediction models: the baseline v0.1 model and the advanced P1 model. While both models leverage LSTM neural networks for stock price forecasting, they differ in architecture, data processing, and performance evaluation. The P1 model outperforms the v0.1 model with improved data handling, model persistence, and comprehensive metrics, showcasing enhanced predictive capabilities. # Environment Configuration

1.1 Setting Up the Virtual Environment

1.1.1 Setup Procedure

A Python virtual environment was established using the venv module to ensure dependency isolation and reproducibility:

Step 1: Creating the Virtual Environment

```
# Navigate to the project directory
cd <path-to-project>

# Create a virtual environment
python -m venv venv

# Activate the environment (Windows)
.\venv\Scripts\activate
```

Step 2: Environment Validation

```
# Check Python version
python --version
# Output: Python 3.12

# Verify pip availability
pip --version
```

Step 3: Installing Dependencies

```
# Install dependencies listed in requirements.txt
pip install -r requirements.txt

# Confirm installed packages
pip list
```

1.1.2 Configuration Details

- **Python Version:** 3.12.5 Tool Used: Python's venv module
- **Location:** `<path-to-project>\venv`
- **Status:** Successfully activated and validated
- **Dependencies:** Installed from requirements.txt

1.2 Installed Dependencies

The virtual environment includes the following packages:

```
numpy
pandas
matplotlib
tensorflow
scikit-learn
pandas-datareader
yfinance
joblib
```

1.3 Verification of Setup

The environment was verified through:

- Checking package versions with `pip list`
- Testing imports of key libraries
- Confirming TensorFlow CPU/GPU compatibility
- Validating data retrieval with yfinance

The environment is fully operational for both models, with all dependencies correctly configured. # Overview of Models

1.4 v0.1 Model Description

The v0.1 model is a basic LSTM-based model for stock price prediction:

Key Features:

- **Stock:** META (Meta Platforms Inc.)
- **Training Data:** January 1, 2020, to August 1, 2023
- **Testing Data:** August 2, 2023, to July 2, 2024
- **Architecture:** 3-layer LSTM with dropout layers
- **Lookback Window:** 60 days for next-day predictions
- **Data Source:** yfinance
- **Feature Used:** Closing price only

Model Structure:

- **LSTM Layer 1:** 50 units, return_sequences=True
- **Dropout:** 0.2
- **LSTM Layer 2:** 50 units, return_sequences=True
- **Dropout:** 0.2
- **LSTM Layer 3:** 50 units
- **Dropout:** 0.2
- **Dense Output:** 1 unit

1.5 P1 Model Description

The P1 model is an enhanced version with advanced features:

Key Features:

- **Stock:** META (Meta Platforms Inc.)
- **Training Data:** 5 years of historical data
- **Architecture:** 2-layer LSTM with configurable settings
- **Lookback Window:** 50 days for 15-day-ahead predictions
- **Data Source:** yfinance
- **Features Used:** Adjusted close, volume, open, high, low

Model Structure:

- **LSTM Layer 1:** 256 units, return_sequences=True
- **Dropout:** 0.4
- **LSTM Layer 2:** 256 units
- **Dropout:** 0.4
- **Dense Output:** 1 unit, linear activation

Chapter 2

Training and Testing Workflow

2.1 v0.1 Model Workflow

2.1.1 Training

The v0.1 model was trained with the following command:

```
python v0.1.py
```

Training Settings:

- **Data Source:** yfinance for META
- **Training Period:** January 1, 2020, to August 1, 2023 (3.5 years)
- **Testing Period:** August 2, 2023, to July 2, 2024 (1 year)
- **Preprocessing:** MinMaxScaler (0–1 range)
- **Sequence Length:** 60 days
- **Batch Size:** 32
- **Epochs:** 25
- **Optimizer:** Adam
- **Loss Function:** Mean Squared Error

Training Output:

```
Epoch 1/25: loss: 0.0565 (high initial loss)
Epoch 2/25: loss: 0.0092 (significant improvement)
...
Epoch 25/25: loss: 0.0045 (converged loss)
```

2.1.2 Testing

- **Data Preparation:** Combined training and test data for sequence continuity
- **Prediction:** Generated predictions for test sequences
- **Inverse Scaling:** Converted normalized predictions to actual prices
- **Visualization:** Plotted actual vs. predicted prices using Matplotlib
- **Output:** Predicted next-day price: $468.90

2.2 P1 Model Workflow

2.2.1 Training

The P1 model training involved two stages:

Stage 1: Training Execution

```
python train_p1.py
```

Training Settings:

- **Data Source:** yfinance for META (5 years)
- **Features:** Adjusted close, volume, open, high, low
- **Preprocessing:** MinMaxScaler applied to each feature
- **Sequence Length:** 50 days
- **Prediction Horizon:** 15 days
- **Batch Size:** 64
- **Epochs:** 25
- **Optimizer:** Adam
- **Loss Function:** Huber
- **Validation Split:** 20%

Training Output:

```
Epoch 1/25: loss: 0.0330 - val_loss: 0.0022 (solid initial performance)
Epoch 7/25: val_loss: 0.00198 (checkpoint saved)
Epoch 10/25: val_loss: 0.00166 (further improvement)
...
Epoch 19/25: val_loss: 0.00159 (best performance)
```

Persistence:

- **Model Weights:** Saved via ModelCheckpoint
- **Full Model:** Stored in .keras format
- **Data Storage:** Scalers and training data saved with joblib
- **Logging:** TensorBoard logs for metric visualization

2.2.2 Testing

Stage 2: Evaluation

```
python test_p1.py
```

Testing Steps:

- **Model Retrieval:** Loaded saved model and data
- **Data Preprocessing:** Applied saved scalers to test data
- **Prediction:** Generated predictions for test sequences
- **Metrics:** Calculated comprehensive performance metrics
- **Profit Analysis:** Evaluated buy/sell signals and profits
- **Visualization:** Plotted actual vs. predicted prices
- **Export:** Saved results to CSV

Testing Features:

- **Fallback:** Loads weights if full model is unavailable
- **Metrics:** Includes loss, MAE, accuracy, and profit
- **Forecasting:** Predicts prices 15 days ahead
- **Validation:** Ensures model reliability

2.3 Infrastructure Details

2.3.1 Directory Organization

Both models generate structured directories:

- **v0.1 Model:** Basic output structure
- **P1 Model:** Detailed structure:

```
p1/
├── data/           # Stores raw data
├── logs/           # TensorBoard logs
├── results/          # Model weights
├── model_checkpoints/  # Full models and data
└── csv-results/      # Evaluation outputs
```

2.3.2 Hardware and Performance

- **Optimization:** TensorFlow utilized CPU instructions (SSE3, SSE4.1, SSE4.2, AVX, AVX2, AVX_VNNI, FMA)
- **Memory:** Efficient batch processing

Chapter 3

Performance Metrics

3.1 v0.1 Model Performance

Training Metrics:

- **Epochs:** 25
- **Final Loss:** ∼0.0045
- **Training Time:** ∼35 seconds
- **Next-Day Prediction:** $468.90

Observations:

- Simple design with limited functionality
- Single-feature prediction
- Minimal evaluation metrics
- Basic visualization

3.2 P1 Model Performance

Training Metrics:

- **Epochs:** 25
- **Training Loss:** 0.0018 (Huber)
- **Validation Loss:** 0.0016
- **Mean Absolute Error:** 0.0432
- **Training Time:** ∼90 seconds

Testing Metrics:

- **15-Day Prediction:** $768.91
- **Huber Loss:** 0.0016128835268318653
- **Mean Absolute Error:** $116.62753188573774
- **Accuracy:** 56.90%
- **Buy Profit:** $1038.2820281982422
- **Sell Profit:** $168.84672546386716
- **Total Profit:** $1207.13
- **Profit per Trade:** $5.0507479

Chapter 4

Code Development Analysis

4.1 Key Enhancements

4.1.1 Data Source Transition

Original (old/p1):

```python
from yahoo_fin import stock_info as si
df = si.get_data(ticker)
```

Current (p1.py):

```python
import yfinance as yf
ticker_data = yf.download(ticker, period="5y")
```

4.1.2 Improved Data Handling

Enhancements:

- Robust MultiIndex column management
- Standardized column names
- Fallback for adjusted close
- Enhanced validation and error handling

4.1.3 Model Persistence

Original:

```
checkpointer = ModelCheckpoint(os.path.join("results", model_name + ".h5"),
                               save_weights_only=True, save_best_only=True, verbose=1)
```

Current:

```
# Save full model in .keras format
model_checkpoint_path = os.path.join("p1/model_checkpoints", model_name + ".keras")
save_model(model, model_checkpoint_path)

# Save data for inference
data_path = os.path.join("p1/model_checkpoints", model_name + "_data.pkl")
joblib.dump(data, data_path)
```

4.1.4 Parameter Management

- **Original:** Hardcoded parameters
- **Current:** Centralized in parameters.py for easy tuning and consistency

4.1.5 Loss Function Update

Original:

```
LOSS = "huber_loss"  # Deprecated
```

Current:

```
LOSS = "huber"  # Modern TensorFlow syntax
```

4.1.6 Directory Structure

- **Original:** Basic flat structure
- **Current:** Organized hierarchy:

```
p1/
├── data/
├── logs/
├── results/
├── model_checkpoints/
└── csv-results/
```

4.2 Architectural Improvements

4.2.1 Model Design

- **v0.1:** 3 layers, 50 units each
- **P1:** 2 layers, 256 units each (higher capacity)

4.2.2 Feature Set

- **v0.1:** Close price only
- **P1:** Multiple features (adjusted close, volume, open, high, low)

4.2.3 Metrics

- **v0.1:** Basic predictions
- **P1:** Comprehensive metrics, including profit and accuracy

Chapter 5

Model Comparison

5.1 Performance Metrics

Metric	v0.1 Model	P1 Model	Winner
Architecture	3x50 units	2x256 units	P1
Features	1 (close)	5 (multi-feature)	P1
Epochs	25	25	Tie
Loss Function	MSE	Huber	P1
Persistence	Weights only	Full model + data	P1
Metrics	Basic	Comprehensive	P1
Profit Analysis	None	$1,207.13 total	P1
Code Structure	Monolithic	Modular	P1

5.2 Technical Advancements

- **Data Stability:** yfinance offers reliable data access
- **Robustness:** Improved error handling
- **Scalability:** Modular design for experimentation
- **Maintainability:** Organized code structure
- **Monitoring:** Detailed logging and metrics

Chapter 6

Insights and Recommendations

6.1 Key Insights

- **P1 Superiority:** Outperforms v0.1 with better metrics
- **Feature Impact:** Multi-feature inputs improve accuracy
- **Architecture:** Fewer, larger layers perform better

Chapter 7

Conclusion

The transition from the v0.1 model to the P1 model marks significant progress in predictive performance and code quality. The P1 model offers superior accuracy, comprehensive metrics, and a robust codebase. With a total profit of $1,207.13 and 56.90% accuracy, it demonstrates practical utility. The adoption of yfinance, modern TensorFlow practices, and modular design ensures a scalable, maintainable solution ready for future enhancements.