Week 8 Assignment:
Deep Learning for Financial Time Series
LSTM Implementation and Performance Analysis

MSc Banking and Finance

Due Date: Final Lecture

Overview

In this last assignment, you will implement and evaluate Long Short-Term Memory (LSTM) networks for cryptocurrency price prediction. You will build LSTM models from scratch, compare their performance against baseline methods, and provide a comprehensive analysis of when deep learning adds value in financial forecasting.

Learning Objectives

By completing this assignment, you will:

- Gain hands-on experience with sequential data preparation
- Implement LSTM architectures
- Understand the importance of baseline comparisons
- Analyze and interpret model performance metrics
- Develop critical thinking about when to use deep learning

Assignment Structure

Part	Description	Marks
Part 1	Data Collection & Preparation	20
Part 2	Baseline Models	20
Part 3	LSTM Implementation	30
Part 4	Performance Analysis	20
Part 5	Written Report	10
Total		**100**

1 Part 1: Data Collection & Preparation (20 marks)

1.1 Task 1.1: Data Collection (5 marks)

Instructions

Collect historical price data for **ONE** of the following cryptocurrencies:

- Bitcoin (BTC)
- Ethereum (ETH)
- Binance Coin (BNB)
- Cardano (ADA)

Requirements:

- Time period: At least 2 years of daily data
- Minimum: 730 observations
- Data source: CoinGecko API, Yahoo Finance, or similar

Hint

You can use the following libraries for data collection:

- `yfinance` for Yahoo Finance data
- `requests` for CoinGecko API
- `pandas_datareader` for various sources

Example code:

```
import yfinance as yf
# Download Bitcoin data
btc = yf.download('BTC-USD', start='2020-01-01', end='2024-01-01')
```

Deliverable

Submit:

1. Python code for data collection
2. CSV file with your collected data
3. Brief description (2-3 sentences) of data source and collection method

1.2 Task 1.2: Exploratory Data Analysis (5 marks)

Instructions

Perform exploratory data analysis on your collected data:

1. Calculate and plot the price series
2. Compute daily returns: $r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$
3. Calculate rolling statistics:
   - 7-day moving average
   - 30-day moving average
   - 7-day rolling volatility (standard deviation of returns)
4. Create visualizations showing:
   - Price over time
   - Daily returns distribution (histogram)
   - Rolling volatility
5. Provide summary statistics (mean, std, min, max, skewness, kurtosis)

Deliverable

Submit:

1. Python code for EDA
2. 3-4 high-quality plots
3. Table of summary statistics
4. Brief commentary (100-150 words) on key patterns observed

1.3 Task 1.3: Sequence Preparation (10 marks)

Instructions

Prepare your data for LSTM training:

Step 1: Data Scaling

- Use MinMaxScaler to scale prices to [0, 1] range
- **Important:** Fit scaler ONLY on training data

Step 2: Create Sequences

- Choose a look-back window: $w \in [20, 60]$ days
- Create sequences: For each time t, input $= [x_{t-w}, \ldots, x_{t-1}]$, target $= x_t$
- Reshape to 3D array: (samples, timesteps, features)
- Features = Just the closing prices (**univariate= for this assignemnt**)
- Features = Price + Volume + Volatility + Moving Average (multivariate)

Step 3: Train/Validation/Test Split

- Training: First 70% of data
- Validation: Next 15% of data
- Test: Final 15% of data
- **No shuffling!** Maintain temporal order

Hint

Key considerations:

- The look-back window (w) is a hyperparameter - justify your choice
- After creating sequences, you'll lose the first w observations
- Ensure no data leakage: validation and test data should never influence scaling

Deliverable

Submit:

1. Python function to create sequences
2. Code showing train/val/test split
3. Print shapes of X_train, y_train, X_val, y_val, X_test, y_test
4. Justification (50-100 words) for your choice of look-back window
5. Visualization showing the temporal split (similar to Week 8 notebook)

2 Part 2: Baseline Models (20 marks)

2.1 Task 2.1: Implement Baseline Models (15 marks)

Instructions

Implement and evaluate THREE baseline models:

1. Naive Forecast (Persistence Model)

- Prediction: $\hat{y}_t = y_{t-1}$ (tomorrow = today)
- Use the last value in each sequence

2. Simple Moving Average

- Prediction: $\hat{y}_t = \frac{1}{k}\sum_{i=0}^{k-1} y_{t-i}$
- Use last k values from sequence (choose $k \in [3, 7]$)

3. Linear Regression

- Use the entire sequence as features
- Flatten 3D array to 2D: (samples, timesteps $\times$ features)
- Train using scikit-learn's LinearRegression

Evaluation Metrics:
For each baseline model, calculate on the **test set**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3}$$

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

Hint

Important: Don't forget to inverse transform predictions back to original scale before calculating metrics!

```
# Example
predictions_scaled = model.predict(X_test)
predictions_original = scaler.inverse_transform(predictions_scaled
    )
y_test_original = scaler.inverse_transform(y_test)
# Now calculate metrics
```

Deliverable

Submit:

1. Python code implementing all three baseline models
2. Table comparing metrics for all three models:

Model	MAE	RMSE	MAPE	R^2
Naive	...	...	...	...
Moving Average	...	...	...	...
Linear Regression	...	...	...	...

3. Bar chart comparing RMSE across models
4. Line plot showing actual vs. predicted for first 100 test points (all models)

2.2 Task 2.2: Baseline Analysis (5 marks)

Instructions

Answer the following questions:

1. Which baseline model performs best? Why do you think this is?
2. Is the naive forecast competitive? What does this tell you about the predictability of your cryptocurrency?
3. Do you observe any systematic patterns in the prediction errors?
4. Based on baseline performance, do you expect LSTM to show significant improvement?

Deliverable

Submit a brief analysis (200-300 words) addressing all four questions.

3 Part 3: LSTM Implementation (30 marks)

3.1 Task 3.1: Simple LSTM Model (15 marks)

Instructions

Build and train a simple LSTM model:
Architecture:

- Input layer: (timesteps, 1)
- LSTM layer: 50 units
- Dropout: 20%
- Dense output layer: 1 unit

Training Configuration:

- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam with learning rate 0.001
- Batch size: 32
- Epochs: 100 (with early stopping)
- Validation data: Use validation set

Callbacks:

- Early Stopping: patience=15, monitor='val_loss'
- ReduceLROnPlateau: factor=0.5, patience=10

Hint

Example architecture in Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

model = Sequential([
    LSTM(50, input_shape=(timesteps, 1)),
    Dropout(0.2),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Deliverable

Submit:

1. Complete Python code for model architecture and training
2. Model summary (model.summary() output)
3. Training history plots:
   - Training vs. Validation Loss
   - Training vs. Validation MAE
4. Number of epochs trained (before early stopping)
5. Final training and validation loss values
6. Test set metrics (MAE, RMSE, MAPE, R^2)

3.2 Task 3.2: Deep LSTM Model (Optional - 10 bonus marks)

Instructions

Build a deeper LSTM architecture:
Architecture:

- Input layer: (timesteps, 1)
- LSTM layer 1: 50 units, return_sequences=True
- Dropout: 20%
- LSTM layer 2: 50 units
- Dropout: 20%
- Dense output layer: 1 unit

Use the same training configuration as Task 3.1.

Deliverable

Submit:

1. Code for deep LSTM model
2. Training history plots
3. Test set metrics
4. Comparison: Does stacking LSTM layers improve performance?

3.3 Task 3.3: Hyperparameter Experimentation (5 marks)

Instructions

Experiment with at least TWO of the following hyperparameters:

- Number of LSTM units: [32, 50, 100]
- Dropout rate: [0.1, 0.2, 0.3]
- Look-back window: [20, 30, 40, 60]
- Batch size: [16, 32, 64]

For each experiment:

1. Change ONE hyperparameter at a time
2. Train the model
3. Record test set RMSE
4. Compare to your baseline simple LSTM

Deliverable

Submit:

1. Table showing hyperparameter variations and corresponding RMSE:

Configuration	Hyperparameter Value	Test RMSE
Baseline	50 units	...
Experiment 1	32 units	...
Experiment 2	100 units	...
...	...	...

2. Brief discussion (100-150 words) of findings

4 Part 4: Performance Analysis (20 marks)

4.1 Task 4.1: Comprehensive Comparison (10 marks)

Instructions

Create a comprehensive comparison of ALL models:

1. Combine all models (baselines + LSTM(s)) in one comparison table
2. Calculate percentage improvement of LSTM over best baseline:

$$\text{Improvement} = \frac{\text{RMSE}_{\text{baseline}} - \text{RMSE}_{\text{LSTM}}}{\text{RMSE}_{\text{baseline}}} \times 100\%$$

3. Create visualizations:
   - Bar chart of RMSE for all models
   - Line plot: Actual vs. Predictions (first 100 test points, all models)
   - Scatter plot: Actual vs. Predicted (for best model only)

Deliverable

Submit:

1. Complete comparison table with all metrics
2. Calculated percentage improvement
3. All three visualizations
4. Interpretation (150-200 words): Does LSTM provide meaningful improvement?

4.2 Task 4.2: Error Analysis (5 marks)

Instructions

Analyze the prediction errors of your best LSTM model:

1. Calculate prediction errors: $e_t = y_t - \hat{y}_t$
2. Create visualizations:
   - Histogram of errors (check if centered at zero)
   - Time series plot of errors (check for patterns)
   - Q-Q plot (check for normality)
3. Statistical tests:
   - Mean of errors (should be close to 0)
   - Standard deviation of errors
   - Check for autocorrelation in errors

Deliverable

Submit:

1. Error analysis plots
2. Statistical summary of errors
3. Brief interpretation (100-150 words): Are errors random or systematic?

4.3 Task 4.3: When Does LSTM Excel? (5 marks)

Instructions

Analyze when your LSTM makes good vs. poor predictions:

1. Identify the 20 best predictions (lowest absolute errors)
2. Identify the 20 worst predictions (highest absolute errors)
3. Examine the characteristics of these periods:
   - Was volatility high or low?
   - Were there strong trends or choppy markets?
   - Any special events (if you know the dates)?
4. Hypothesize why LSTM performs well/poorly in these conditions

Deliverable

Submit:

1. Analysis of best/worst prediction periods
2. Discussion (200-250 words) of patterns and hypotheses

5 Part 5: Written Report (10 marks)

Instructions

Write a concise report (800-1000 words) addressing:

1. **Introduction (10%)**
   - Brief motivation: Why predict cryptocurrency prices?
   - Your chosen cryptocurrency and rationale
2. **Methodology (30%)**
   - Data preparation approach
   - Baseline models chosen and why
   - LSTM architecture design decisions
   - Training strategy and hyperparameters
3. **Results (40%)**
   - Baseline performance summary
   - LSTM performance summary
   - Key visualizations (2-3 figures maximum)
   - Statistical comparison
4. **Critical Discussion (20%)**
   - Does LSTM justify its complexity?
   - Limitations of your approach
   - Real-world applicability
   - What would you do differently?
5. **Conclusion**
   - Key findings
   - Recommendations for practitioners

Deliverable

Submit:

1. PDF report (800-1000 words)
2. Embedded figures should be clear and labeled
3. Professional formatting

Submission Requirements

What to Submit

Jupyter Notebook + PDF Report + Data Files + 5-10 min presenation

Naming Convention

Important: Use the following naming convention:

- Notebook: `StudentID_W8_LSTM.ipynb`
- Report: `StudentID_W8_Report.pdf`
- Data: `StudentID_W8_Data.csv`

Replace `StudentID` with your actual student ID number.

Submission Method

email: tsomig@gmail.com (able to support large emails..)

Tips for Success

Top Tips

1. **Start Early:** LSTM training takes time. Don't wait until the last minute.
2. **Document as You Go:** Add markdown cells explaining your reasoning while coding.
3. **Save Your Work:** Use version control or regularly save your notebook.
4. **Test Incrementally:** Don't write all code at once. Test each part before moving on.
5. **Manage Resources:** If training is slow, use Google Colab with GPU or reduce epochs.
6. **Interpret Results:** Don't just report numbers. Explain what they mean.
7. **Compare Properly:** Always use the same test set for all models.
8. **Be Honest:** If LSTM doesn't beat baselines, that's okay! Explain why.
9. **Proofread:** Check for typos, code errors, and broken plots before submitting.
10. **Ask Questions:** Use office hours if you're stuck. Don't struggle in silence.

Common Pitfalls to Avoid

Avoid These Mistakes

- **Data Leakage:** Fitting scaler on entire dataset instead of training only
- **Random Splitting:** Shuffling time series data destroys temporal structure
- **Comparing Scaled Metrics:** Calculate metrics on original scale, not normalized
- **Ignoring Validation:** Using test set during model development
- **Overfitting:** Training too long without early stopping
- **Cherry-picking:** Only showing good results, hiding poor performance
- **Unjustified Complexity:** Using deep models without justification
- **Missing Baselines:** Not comparing to simple methods

Frequently Asked Questions

Q: Can I use a different cryptocurrency?
A: Yes, but it must be a major cryptocurrency with at least 2 years of reliable daily data.

Q: My LSTM is worse than the naive forecast. Is that okay?
A: Yes! This is a valid finding. Discuss why this might happen and what it tells you about market predictability.

Q: How long should LSTM training take?
A: On CPU: 5-15 minutes. On GPU: 1-3 minutes. If much longer, reduce epochs or use Google Colab.

Q: Can I work in groups?
A: No, this is an individual assignment. However, discussing concepts (not code) is encouraged.

Q: What if I can't get TensorFlow to work?
A: Use Google Colab which has TensorFlow pre-installed, or seek technical support early.

Q: Should I tune all hyperparameters?
A: No, just experiment with 2-3 as specified. Exhaustive tuning is not required.

Q: Can I use external libraries like sklearn for preprocessing?
A: Yes, standard libraries (sklearn, numpy, pandas) are encouraged.

Good Luck!

Remember: The goal is to learn, not just to get perfect predictions.
Thoughtful analysis of imperfect results is more valuable than perfect models without understanding.