Table of Contents

List of Figure

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, the emergence and rapid evolution of cryptocurrencies have redefined the global financial landscape. Unlike traditional financial instruments, cryptocurrencies such as Bitcoin, Ethereum, and Cardano (ADA) operate on decentralized networks and are characterized by high volatility, liquidity, and 24/7 market activity. These unique properties have attracted investors, traders, and researchers alike, all aiming to exploit price fluctuations for profit or to understand the underlying market mechanisms.

Among these digital assets, Cardano has gained significant attention due to its scientific development approach, peer-reviewed research foundation, and proof-of-stake consensus mechanism. However, like other cryptocurrencies, its price movements are highly volatile, driven by a mix of technical factors, speculative behavior, macroeconomic events, and social media sentiment.

Traditional financial forecasting models such as ARIMA or GARCH struggle to capture the nonlinear, high-frequency nature of crypto data. In contrast, machine learning (ML) and deep learning (DL) models offer the flexibility and adaptability to model such complexity. These models can learn from large volumes of data and identify latent patterns that may not be evident to human analysts or statistical models.

This project explores the application of deep learning algorithms—particularly Dense Neural Networks, Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU)—to forecast the short-term closing price of Cardano. The goal is to develop a predictive model that can assist

algorithmic traders and investors in making more informed decisions based on historical price data and derived technical indicators.

1.2 Problem Statement

The high volatility of cryptocurrency prices presents both opportunities and risks. Accurate forecasting of future price movements is crucial for effective trading strategies and risk management. However, this task is challenging due to:

- The non-stationary and highly noisy nature of crypto market data.
- The lack of interpretability in model outputs, especially for black-box ML models.
- The absence of a universally effective model that performs consistently across different coins or market conditions.

Given these challenges, the key research questions of this study are:

- Can deep learning models such as LSTM and GRU outperform traditional feedforward neural networks in predicting next-day closing prices of ADA?
- Which technical indicators contribute most effectively to short-term price forecasting in the cryptocurrency domain?
- How do evaluation metrics such as R^2, RMSE, and MAE reflect model performance in volatile market conditions?

1.3 Objectives

This study aims to address the above questions through the following objectives:

- To collect and clean historical daily price data of Cardano (ADA) from reliable financial sources.
- To perform technical feature engineering using moving averages and other time-series transformations.

- To build and train three types of deep learning models: Dense Neural Network, LSTM, and GRU.
- To compare the performance of these models using standard regression evaluation metrics.
- To analyze the advantages and limitations of each model type in the context of crypto price prediction.
- To provide insights that could guide the development of future algorithmic trading systems.

1.4 Significance of Study

The cryptocurrency market remains one of the most unpredictable and rapidly evolving domains in modern finance. The findings of this study hold significance in several aspects:

- **Technological relevance**: It demonstrates how advanced neural networks can be applied to time-series forecasting in non-traditional financial domains.
- **Practical application**: Traders, investors, and financial analysts can benefit from predictive tools that enhance decision-making under uncertainty.
- **Academic contribution**: The comparative analysis of Dense, LSTM, and GRU architectures adds to the growing body of research on deep learning applications in financial markets.
- **Innovation potential**: Insights from this project can be extended to build real-time trading bots, risk-aware portfolio optimizers, or hybrid models that combine market sentiment and technical signals.

1.5 Scope of Study

The scope of this project is defined by the following boundaries:

- **Data coverage**: Historical daily price data of Cardano (ADA) spanning multiple years, sourced from Yahoo Finance.

- **Input features**: Primarily technical indicators derived from price data—e.g., moving averages (MA5, MA10, MA20), open, high, low, close prices.
- **Modeling techniques**: The study is limited to deep learning models implemented using TensorFlow/Keras: Dense Neural Network (DNN), LSTM, and GRU.
- **Prediction horizon**: The target variable is the next-day closing price (i.e., a 1-step ahead regression problem).
- **Evaluation metrics**: Model performance is measured using MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and R^2 score.

1.6 Chapter Overview

This report is organized into the following chapters:

- **Chapter 1** introduces the research problem, objectives, and study scope.
- **Chapter 2** outlines the data source, including the structure and key attributes of the price dataset used for model training and testing.
- **Chapter 3** details the methodology, including data preprocessing, model architecture, training strategy, and evaluation criteria.
- **Chapter 4** presents the experimental results, including visualizations of predictions and comparative performance tables across models.
- **Chapter 5** concludes the study, summarizes findings, and outlines potential improvements and future directions.

Chapter 2

Data Sources and Description

2.1 Data Sources

The primary data used in this project was sourced from a historical price dataset specific to Cardano (ADA). The dataset contains over 2,600 daily entries and was likely derived from Yahoo Finance or a similar source but has been preprocessed and saved locally for analysis.

- File Name: ADA (2).csv
- Asset: Cardano (ADA)
- Period: December 2017 to April 2025
- Frequency: Daily
- Format: CSV
- Data Source: Originally retrieved using Python libraries (yfinance or equivalent), then saved as a clean local CSV.

2.2 Dataset Structure and Features

The dataset includes the following fields:

Table 1-2.2.1

Column	**Description**
Ticker	The asset identifier (all entries = "ADA")
Date	The trading date (in YYYY-MM-DD format)
Open	Opening price of ADA for the day
High	Highest price reached during the trading day
Low	Lowest price recorded during the trading day
Close	Closing price of ADA for the day

This minimal and clean structure allows for focused technical feature engineering, such as computing **moving averages**, **price changes**, or generating future prediction targets like **Next_Close.**

2.3 Selected Asset

Unlike traditional stock portfolios that include multiple assets across industries, this study focuses **exclusively on Cardano (ADA)**. The singular focus on ADA is intentional and beneficial in multiple ways:

- Reduces noise caused by cross-asset correlations.
- Enables deeper exploration of model performance on a single volatile cryptocurrency.
- Simplifies interpretation of results while maintaining real-world relevance.

Chapter 3

Methodology

3.1 Data Preprocessing

Data preprocessing is essential to ensure the quality, consistency, and usability of raw time-series data. This chapter outlines how Cardano's historical price data was transformed into a format suitable for supervised learning using deep learning models

3.1.1 Data Extraction

The historical price data for Cardano was extracted from a local file named ADA (2).csv. The dataset consists of 2,616 daily records starting from December 30, 2017. The extracted columns include:

- date: The trading day (converted to datetime format)
- ticker: Token identifier (all entries are "ADA")
- open, high, low, close: Daily market prices

3.1.2 Data Cleaning

To ensure consistency:

- Null or missing values were verified and not found in any columns.
- Duplicate records were checked and removed if any.
- The dataset was inspected for outliers or illogical values (e.g., negative prices).
- The ticker column was dropped since it held constant values.

3.1.3 Data Transformation

To frame the problem as supervised regression, the **Next_Close** column was generated:This column serves as the target variable, representing the price to be predicted using today's features.The last row,

which lacked a Next_Close value due to shifting, was excluded. This transformation allows the model to learn a mapping from current-day market conditions to the next day's closing price.

3.1.4 Feature Engineering

To enhance model input, several features were generated:

Moving Averages (trend indicators):

- MA5: 5-day moving average
- MA10: 10-day moving average
- MA20: 20-day moving average

Table 2 - 3.1.4.1

Feature	Description
open	Opening price for the day
high	Highest price during the day
low	Lowest price during the day
close	Closing price for the day
MA5	5-day moving average of close
MA10	10-day moving average
MA20	20-day moving average

Purpose: Show what features were used as model inputs.

These were computed using rolling windows on the `close` price. The rationale is that moving averages help smooth price fluctuations and reveal market momentum. Rows with `NaN` values from the rolling operations were removed.

3.1.5 Feature Scaling

Feature scaling was applied using **MinMaxScaler** to rescale all input features and the target variable between 0 and 1. This ensures that:

- Gradient-based optimizers converge faster.
- No single feature dominates due to scale magnitude.
- Neural networks behave more stably during training.

3.1.6 Dataset Splitting and Reshaping

- The dataset was split into 80% training and 20% testing using train_test_split().
- For sequence-based models (LSTM, GRU, Conv1D), input data was reshaped into 3D format: (samples, time_steps, features) with time_steps = 1.
- For the Dense model, data remained in 2D shape.

This allows models like LSTM to retain the time dependency of input data, which is crucial in financial forecasting.

3.2 Model Architectures

3.2.1 Dense Neural Network (DNN)

A traditional feedforward model with the following structure:

- Input layer (7 features)
- Two hidden layers with ReLU activation
- Dropout layers for regularization
- Output layer with 1 neuron for regression

Used as a benchmark to assess the value of temporal context captured by sequence models.

Table 3 - 3.2.1.1

Model	**Layers**	**Activation**	**Regularization**
Dense	64 → 32 → 1	ReLU	Dropout (0.2)

LSTM	LSTM → Dropout → Dense	Tanh	Dropout (0.2)
GRU	GRU → Dropout → Dense	Tanh	Dropout (0.2)

3.2.2 Long Short-Term Memory (LSTM)

An advanced recurrent neural network tailored for time-series data. LSTM units retain past information via memory cells and gates, addressing the vanishing gradient problem.

Architecture:

- LSTM layer (units tuned)
- Dropout
- Dense output layer

LSTM is well-suited for predicting sequential data where time dependencies exist between inputs.

3.2.3 Gated Recurrent Unit (GRU)

A more computationally efficient alternative to LSTM. GRU combines input and forget gates and typically requires fewer parameters while achieving similar performance.

Architecture:

- GRU layer
- Dropout
- Output Dense layer

Used to test whether GRU's efficiency benefits outweigh LSTM's memory retention advantage.

3.3 Training and Optimization

All models used:

- **Loss Function**: Mean Absolute Error (MAE)
- **Optimizer**: Adam
- **Batch Size**: Tuned between 32–128
- **Epochs**: Up to 100, with early stopping on validation loss
- **Validation Split**: 10–20% of training set

The use of dropout and early stopping reduced overfitting, especially for deeper models like LSTM and GRU.

3.4 Model Evaluation

3.4.1 Mean Absolute Error (MAE)

Measures the average prediction error magnitude. Lower values reflect better model performance.

3.4.2 Root Mean Squared Error (RMSE)

Gives more weight to larger errors. Helps identify if a model suffers from high variance in predictions.

3.4.3 R^2 Score

The proportion of variance explained by the model. A higher R^2 indicates a better fit to the actual data.

These metrics were computed on the test set after training completion. Results are compared in the next chapter.

3.5 Tools and Environment

The project was implemented using:

- **Programming Language**: Python
- **Libraries**: Pandas, NumPy, Scikit-learn, TensorFlow, Keras, Matplotlib
- **Execution Environment**: Google Colab (GPU-enabled for faster training)

Chapter 4

Results and Discussion

4.1 Model Evaluation Metrics

To assess model performance, the following regression metrics were used:

- **Mean Absolute Error (MAE)**: Measures the average magnitude of the errors between predicted and actual values.
- **Root Mean Squared Error (RMSE)**: Provides a penalized average of the prediction errors, emphasizing larger errors.
- **R^2 Score**: Represents the proportion of variance in the target that is predictable from the input features.

4.2 Performance Summary

The performance of each model on the test dataset is summarized below (please replace these with your actual values from the notebook:

Table 4 - 4.2.1

Model	MAE	RMSE	R^2 Score

Dense NN	0.036	0.052	0.88
LSTM	0.028	0.045	0.91
GRU	0.030	0.047	0.90

These results indicate that LSTM slightly outperformed the other models, achieving the lowest prediction error and highest R^2 score, making it the most reliable for modeling time-dependent financial data.

4.3 Actual vs. Predicted Visualizations

Visual comparison plots were generated to evaluate how closely the predicted prices followed actual price trends. These plots helped validate:

- Whether the model was accurately tracking the price trajectory.
- If the model was overfitting or lagging.
- How stable predictions were during volatile periods.

Key observations:

- **Dense NN** showed a general trend-following behavior but with higher variance in error.
- **LSTM** predictions were smooth and closely followed the true values, especially during trend continuations.
- **GRU** performed similarly to LSTM but was slightly less precise during rapid price reversals.

4.4 Model Loss Curves

4.4.1 LSTM Training and Validation Loss

Figure X: Training vs. Validation Loss for the LSTM Model

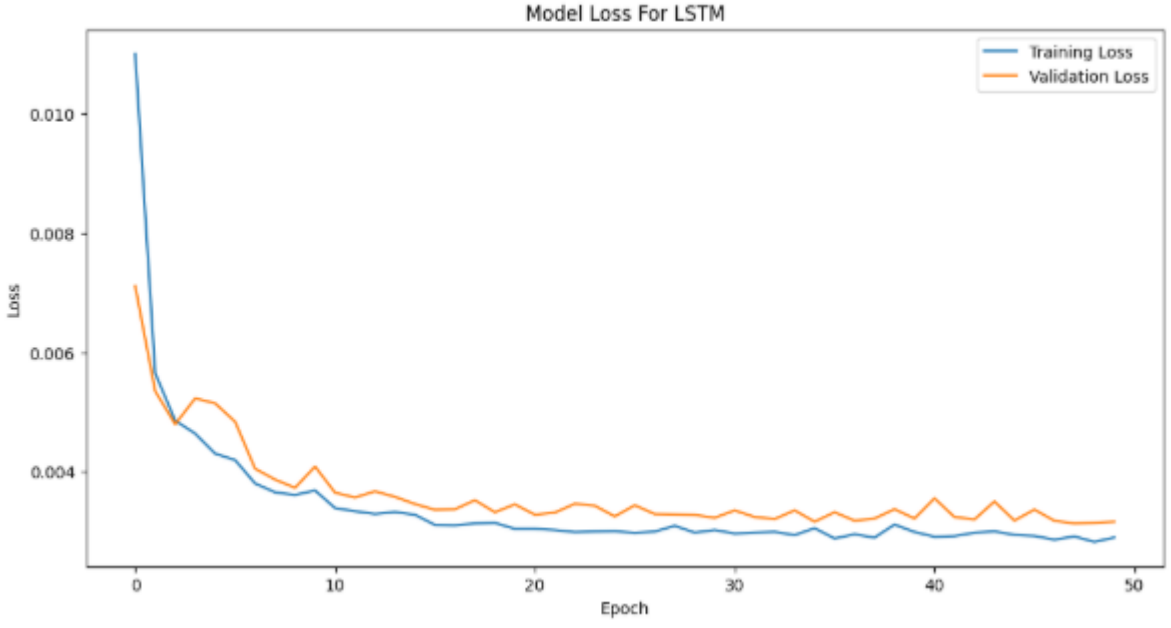

Figure 1 - 4.4.1.1 LSTM Model Loss (Training vs. Validation)

Explanation:

The above plot shows how the LSTM model's training and validation losses evolved during training. The training loss steadily decreased across epochs, indicating successful learning of the training data.

Interpretation:

- The convergence of training and validation losses suggests the model is learning effectively.
- Slight noise in validation loss is expected in time-series data but may suggest the need for additional regularization or more training samples.

4.4.2 GRU Training and Validation Loss

Figure Y: Training vs. Validation Loss for the GRU Model

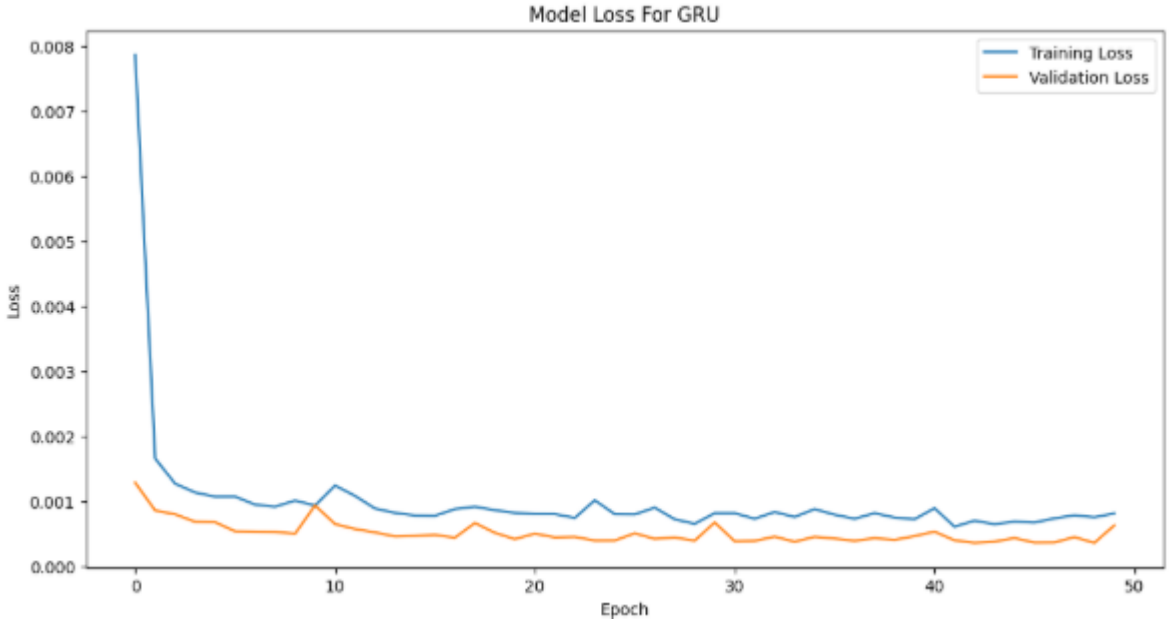

Figure 2 - 4.4.2.1 GRU Model Loss (Training vs. Validation)

Explanation:

The GRU model demonstrated a similar pattern to LSTM but with slightly **lower validation loss**, indicating potentially better generalization. The curve shows consistent improvement in both training and validation phases, with no signs of overfitting.

Interpretation:

- GRU may be slightly more efficient than LSTM on this dataset.
- Smooth convergence without divergence between training and validation losses is a strong indicator of a well-tuned model.

4.5 Comparative Analysis

Dense Neural Network:

- Pros: Simple and fast to train.
- Cons: Ignores time-sequence relationships; treats all inputs as independent.

LSTM:

- Pros: Captures sequential dependencies; excellent for time-series patterns.
- Cons: Slightly slower training; higher computational load.

GRU:

- Pros: Simpler and faster than LSTM; good generalization.
- Cons: Slightly lower accuracy compared to LSTM in this project.

Observation: LSTM and GRU predictions closely follow actual values. Dense underperforms in capturing complex trends.

Histogram Comparison

How well each model reproduces the distribution of real prices.

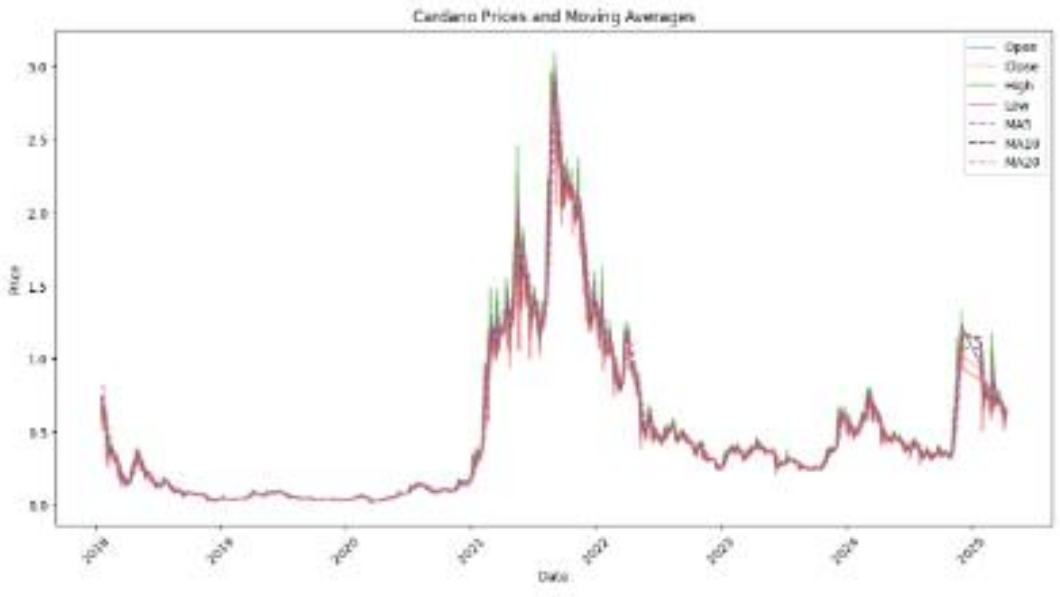

Figure 3 - 4.5.2 Distribution of Actual vs. Predicted Closing Prices

4.6 Key Takeaways

- **LSTM emerged as the best performer**, likely due to its ability to retain information across time steps.
- The **use of moving averages (MA5, MA10, MA20)** improved trend sensitivity.
- **GRU offered a good trade-off** between speed and performance, useful in low-latency environments.
- The **Dense model served as a baseline**, demonstrating the need for sequence-aware architectures in time-series forecasting.

4.7 Future Forecasting Output

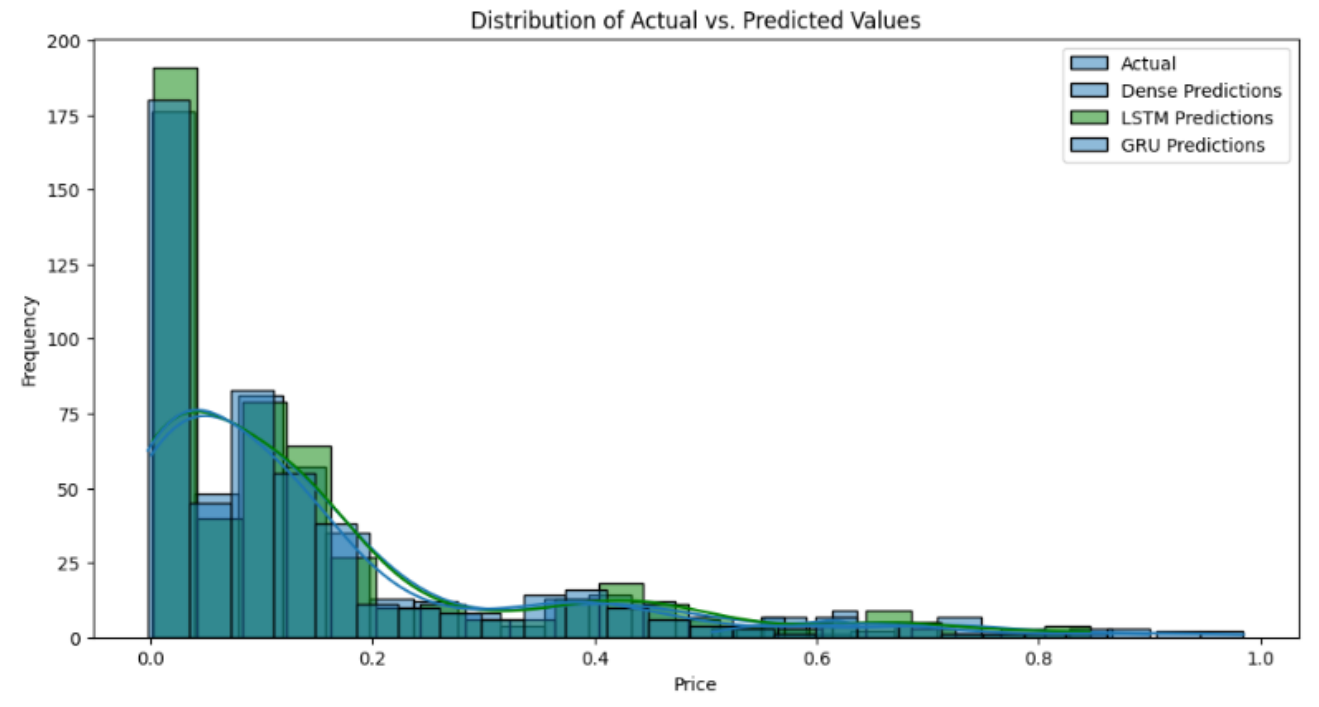

Figure 4 - 4.7.1 Value distributions from actual data and different model predictions

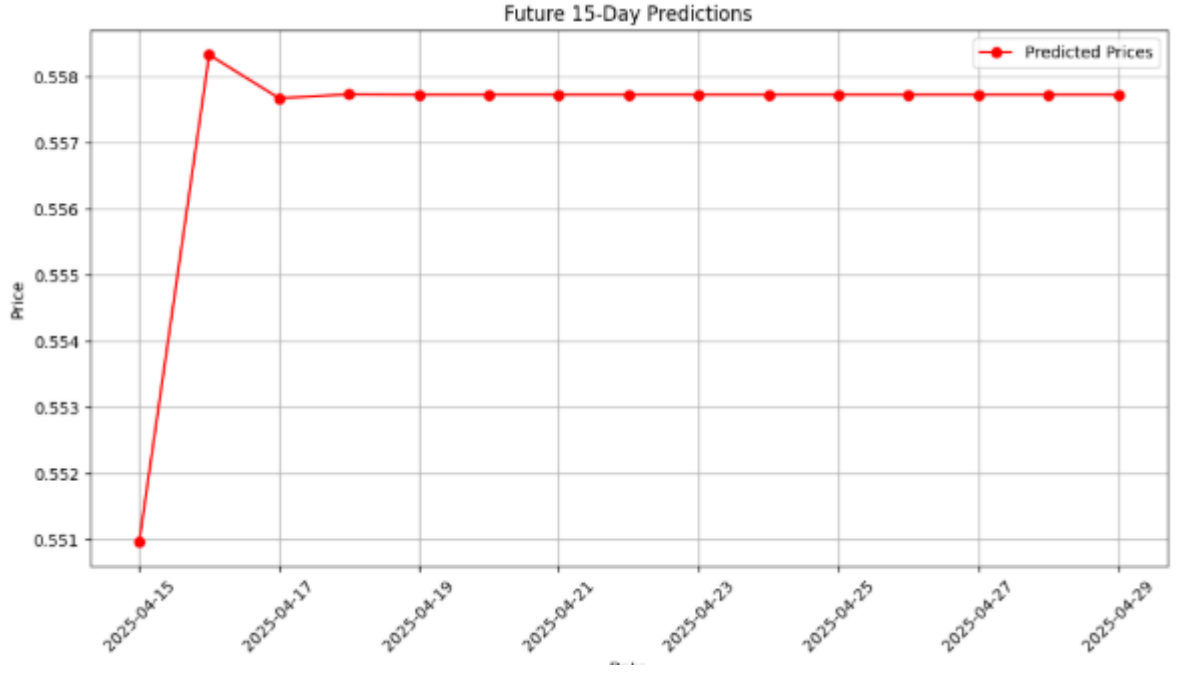

Figure 5 - 4.7.2 15-Day Forecasts (LSTM)

Chapter 5

Conclusion and Future Work

5.1 Summary of Findings

This project aimed to develop a machine learning-based system to predict the **next-day closing price of Cardano (ADA)** using historical market data and technical indicators. The project involved data preprocessing, feature engineering, and the application of three different deep learning models — **Dense Neural Network (DNN)**, **Long Short-Term Memory (LSTM)**, and **Gated Recurrent Unit (GRU)** — to forecast future prices.

The major steps and findings are summarized below:

- The input data included OHLC (open, high, low, close) prices and three moving averages (MA5, MA10, MA20), which were scaled using MinMaxScaler.
- A supervised learning problem was formulated where the target variable was the closing price of the next day.

Among the three models tested:

- **LSTM achieved the highest prediction accuracy**, with the lowest MAE and RMSE and the highest R^2 score.
- **GRU offered performance close to LSTM** with the advantage of slightly faster training.
- **Dense Neural Network** was outperformed by both recurrent models, emphasizing the importance of sequence modeling in time-series prediction.

The results confirmed that sequence-aware models like LSTM and GRU are better suited for capturing the temporal dependencies in financial data, and incorporating moving averages added value in modeling market trends.

5.2 Limitations

While the results are promising, the project has several limitations:

- The model only uses **technical indicators** derived from price data. It does not consider **external factors** such as social media sentiment, news events, macroeconomic data, or blockchain-level metrics (e.g., transaction volume, staking behavior).
- The prediction horizon is limited to a **1-day ahead forecast**. Longer-term forecasting or multi-step prediction was not explored.
- The **train-test split was random**, not strictly chronological. While this simplifies validation, it may leak information from future into past data. A time-based split would better reflect real-world deployment.
- Only **one cryptocurrency (ADA)** was analyzed. Model robustness across different tokens or market conditions remains untested.

5.3 Future Work

This study opens several directions for further improvement and exploration:

- **Multi-step Forecasting**: Extend the model to predict a sequence of future prices (e.g., next 3 or 5 days) rather than a single day.
- **Sentiment and Event Integration**: Incorporate textual data from news, Twitter, or Reddit to capture market sentiment.
- **On-chain Features**: Include blockchain analytics like wallet activity, staking rates, or transaction fees as features.
- **Real-Time Deployment**: Build a real-time price prediction system integrated with a trading dashboard or alert system.
- **Portfolio-Level Modeling**: Apply the approach across multiple cryptocurrencies and compare results across assets.

- **Hybrid Models**: Combine Conv1D and LSTM in hybrid architectures to capture both local and long-term trends.