CRYPTOCURRENCY PRICE PREDICTION USING LSTM-GRU MODEL

A PROJECT REPORT

Submitted by

ABDULLAH AMBERKHANI (311619104002)

HEMANTH S (311619104027)

MOHAMMED ABDUL BASIDH P T (311619104041)

in partial fulfillment for the award of

degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE

THORAIPAKKAM, CHENNAI-600097

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report

“CRYPTOCURRENCY PRICE PREDICTION USING LSTM-GRU MODEL “

is the bona fide work of
“ABDULLAH AMBERKHANI, HEMANTH. S, MOHAMMED ABDUL BASIDH. P. T “

who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. N. Saravanan B.E., ME., Ph. D	Mr. Y. Gold Anand M.E., (Ph.D.)
HEAD OF THE DEPARTMENT	**SUPERVISOR / ASSISTANT PROFESSOR**
Department of Computer Science	Department of Artificial
and Engineering	Intelligence and Data Science
Misrimal Navajee Munoth Jain	Misrimal Navajee Munoth Jain
Engineering College,	Engineering College,
Thoraipakkam,	Thoraipakkam,
Chennai-600097	Chennai-600097

Submitted for the Project Viva-Voce Examination held on ____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE

Vision:

Producing competent computer engineers with a strong background in the latest trends and technology to achieve academic excellence and to become pioneers in software and hardware products with an ethical approach to serve the society.

Mission:

- To provide quality education by inculcating strong fundamentals in basic science, mathematics and engineering concepts through the state-of-the-art facilities.
- To provide the learning ambience that helps the students to enhance problem solving skills and to inculcate in them the habit of continuous learning in their domain of interest.
- To serve the society by providing insight solutions to the real-world problems by employing the latest trends of computing technology with strict adherence to professional and ethical responsibilities.

ACKNOWLEDGEMENT

We express our sincere gratitude to our honorable **Secretary** (Administration) **Dr. Harish L Metha** and Secretary (Academic) **Shri L. Jaswant S Munoth** for providing the infrastructure facilities to do this project during our course period.

We thank our **Principal**, **Dr. C. C. Christopher,** for his support and motivation for the development and completion of this project.

We express a profound sense of gratitude and thanks to our **Head of The Department**, **Dr. N. Saravanan,** for his valuable suggestions and guidance for the development and completion of this project.

We thank our project coordinator, **Mrs. A. Jeyanthi , Associate Professor**, for her valuable suggestions and constant encouragement that led us to the successful completion of our project.
We thank our supervisor **Mr. Y. Gold Anand, Assistant Professor,** for his constant help and valuable suggestions.

We thank all the Teaching and Non-Teaching Staff members of our department who helped us to complete our project.

Above all we thank the Almighty, and our parents and for their constant support and encouragement for completing this project.

ABSTRACT

Cryptocurrency price prediction is a proposed system that utilizes deep learning algorithms (LSTM-GRU) to predict the price of user requested cryptocurrency. The system uses the yfinance library to download cryptocurrency data and preprocesses the data using min-max normalization. The deep learning model used in this program is a combination of GRU and LSTM neural network. The program creates sliding windows to format the data and define hyperparameters such as the learning rate, hidden units, batch size, and epochs. The predicted results are shown on a tkinter-based GUI interface. The program also calculates mean squared error (MSE) and mean absolute percentage error (MAPE) to evaluate the accuracy of the model. The accuracy of this system is close to 95% and overall, it provides accurate results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATIONS	EXPANSION
1	ML	MACHINE LEARNING
2	GRU	GATED RECURRENT UNIT
3	LSTM	LONG SHORT-TERM MEMORY
4	RNN	RECURRENT NEURAL NETWORK

CHAPTER 1

INTRODUCTION

Cryptocurrency has gained significant attention in recent years as a decentralized digital currency. Cryptocurrency prices are highly volatile and unpredictable, and this poses a significant challenge for traders and investors in the cryptocurrency market. Therefore, accurate prediction of cryptocurrency prices is a crucial task for traders and investors to make informed investment decisions. With the increasing popularity of deep learning techniques, several studies have applied various deep learning models to predict cryptocurrency prices. In this paper, we propose a hybrid long short-term memory (LSTM) gated recurrent unit (GRU) model for cryptocurrency price prediction.

LSTM is a type of recurrent neural network (RNN) that is designed to handle the vanishing gradient problem. The vanishing gradient problem arises when gradients become extremely small during the training of deep neural networks. This makes it difficult for the model to learn long-term dependencies. LSTM solves this problem by using memory cells that can store information over a long period. GRU is a variant of LSTM that uses fewer parameters and is faster to train. The GRU uses two gates to control the information flow, namely the reset gate and update gate.

The proposed hybrid LSTM-GRU model aims to combine the strengths of both models to improve prediction accuracy. The LSTM component of the model is responsible for learning long-term dependencies, while the GRU component focuses on capturing short-term dependencies. The combination of both models is expected to result in a more robust and accurate prediction model.

The model takes in historical cryptocurrency price data as input and outputs the predicted price. The input data is preprocessed and normalized before being fed into the model. The model is trained using a sequence-to-sequence learning approach. The sequence-to-sequence learning approach involves training the model to predict the next value in the sequence given the previous values in the sequence.

The proposed model is evaluated using several performance metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). The model is compared with GRU and LSTM models separately. The evaluation results show that the proposed hybrid LSTM-GRU model outperforms the models in terms of prediction accuracy.

In conclusion, the proposed hybrid LSTM-GRU model is a promising approach for cryptocurrency price prediction. The model combines the strengths of LSTM and GRU models to improve prediction accuracy. The evaluation results demonstrate the superior performance of the proposed model over other existing models. This study contributes to the development of accurate and reliable cryptocurrency price prediction models, which are essential for traders and investors in making informed investment decisions.

CHAPTER 2

LITERATURE SURVEY

Naila Aslam used a hybrid model (LSTM-GRU) and tweets related to cryptocurrencies from twitter as data and analyzed the sentiment it presented and how it would affect the currency. The proposed LSTM-GRU ensemble shows an accuracy of 0.99 for sentiment analysis, and 0.92 for emotion prediction and outperforms both machine learning and state-of-the-art models.

Patel Jay used the Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) models for Bitcoin, Ethereum, and Litecoin. The results show that the proposed model is superior in comparison to the deterministic models.

Sudeep Tanwar paper used deep-learning-based hybrid model (includes Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM)) to predict the price of Litecoin and Zcash with inter-dependency of the parent coin. Results illustrate that the proposed model forecasts prices with high accuracy compared to existing models.

Zeinab Shahbazi used blockchain framework for secure transaction environment and Reinforcement Learning algorithm for analysis and prediction of price. The focus of this system is on Litecoin and Monero cryptocurrencies. The results show the presented system accurate the performance of price prediction higher than another state-of-art algorithm.

Raj Parekh used hybrid and robust framework, DL-Guess, for cryptocurrency price prediction, that considers its interdependency on other cryptocurrencies and on market sentiments. It considered price prediction of Dash carried out using price history and tweets of Dash, Litecoin, and Bitcoin for various loss functions for validation. It shows the results for price prediction of Bitcoin-Cash with the price history and tweets of Bitcoin-Cash, Litecoin, and Bitcoin.

CHAPTER 3

SYSTEM OVERVIEW

3.1 EXISTING SYSTEM

3.1.1 OVERVIEW

There are several existing cryptocurrency price prediction systems that were used before the LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) model became popular. Here are a few examples:

1. Autoregressive Integrated Moving Average (ARIMA): This is a commonly used time series model for predicting the price of cryptocurrencies. It assumes that the future values of a time series are a function of its past values, with some randomness added in. ARIMA models have been used to predict the price of Bitcoin, Ethereum, and other cryptocurrencies.
2. Support Vector Regression (SVR): This is a machine learning algorithm that is often used for regression problems, including price prediction. SVR has been used to predict the price of Bitcoin and other cryptocurrencies.
3. Random Forest Regression: This is another machine learning algorithm that is often used for regression problems. Random Forest has been used to predict the price of Bitcoin and other cryptocurrencies.
4. Artificial Neural Networks (ANN): ANN is a popular machine learning algorithm that has been used to predict the price of cryptocurrencies. It is like the LSTM model but does not have the long-term memory feature that LSTM has.

3.1.1 DRAWBACKS

While these existing cryptocurrency price prediction systems can provide useful insights, they also have some limitations and drawbacks. Here are some of the drawbacks of each system:

1. Autoregressive Integrated Moving Average (ARIMA): One major limitation of ARIMA is that it assumes that the time series being analysed is stationary, which means that the statistical properties of the series (such as mean and variance) remain constant over time. However, cryptocurrency prices are highly volatile and exhibit non-stationary behaviour, which can make it difficult to accurately predict future prices using ARIMA.
2. Support Vector Regression (SVR): One limitation of SVR is that it requires a significant amount of data to be trained effectively. Since cryptocurrency prices are highly volatile and can fluctuate rapidly, it can be challenging to collect sufficient data to build an accurate model.
3. Random Forest Regression: While Random Forest can be effective at capturing nonlinear relationships between variables, it may struggle with predicting trends in time series data, especially when the trend is changing rapidly.
4. Artificial Neural Networks (ANN): One limitation of ANN is that it can be prone to overfitting, especially when dealing with small datasets. Additionally, ANNs can be difficult to interpret, which can make it challenging to understand how the model arrived at its predictions.

3.2 PROPOSED SYSTEM

3.2.1 OVERVIEW

- Anyone without prior knowledge in coding can use.
- Uses deep learning model for prediction (LSTM, GRU)
- Supports multiple currencies.
- Desktop application

3.2.2 ADVANTAGES

1. There are several advantages to using the LSTM (Long Short-Term Memory) model over the existing cryptocurrency price prediction systems:
2. Capturing long-term dependencies: One of the key advantages of the LSTM model is its ability to capture long-term dependencies in time series data. This is particularly useful for predicting cryptocurrency prices, which are known for their high volatility and complex patterns.
3. Handling non-stationary data: LSTM models are well-suited for handling non-stationary time series data, which is a common characteristic of cryptocurrency prices. Unlike traditional time series models such as ARIMA, LSTM models do not assume that the statistical properties of the data remain constant over time.
4. Improved accuracy: LSTM models have been shown to outperform traditional time series models such as ARIMA, SVR, and Random Forest in many prediction tasks. This is due to their ability to capture complex patterns and dependencies in the data.
5. Handling high-dimensional data: LSTM models can handle high-dimensional data, which is useful when dealing with multiple features that

may impact cryptocurrency prices, such as transaction volume, market sentiment, and news sentiment.

6. Flexibility: LSTM models can be customized to fit a wide range of prediction tasks and can be trained on a variety of input data types, such as time series data, text data, and image data.

3.3 REQUIREMENT ANALYSIS

The requirement specification is a technical specification of requirements for the software products. The purpose of the software requirement specification is to provide a detailed overview of the software project, its parameters and goal. It describes the project target audience and its user interface, hardware, and software requirements.

3.3.1 SOFTWARE REQUIREMENT

This Python desktop application is platform-agnostic and can be executed on any operating system.

3.3.2 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete engineer as the starting point for the system design.

Ram	:	8GB Ram or more
Processor	:	Any Intel Processor
GPU	:	4GB or more
Hard Disk	:	10GB or more
Speed	:	1.4GHZ or more

3.4 TOOLS USED

3.4.1 PYTHON 3

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, Including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.
Python consistently ranks as one of the most popular programming languages.

The version of python we are using is 3.11and the libraries used are:

- Numpy
- Pandas
- Matplotlib
- Sklearn
- Keras

3.4.2 TENSORFLOW

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

3.4.3 ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

3.4.4 tkinter

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems. Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the _tkinter module for more information about supported versions. Tkinter is not a thin wrapper but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes and refer to the official Tcl/Tk documentation for details that are unchanged.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

The diagram shows the system architecture of the project. Each block shows the structural and functional components of the project.

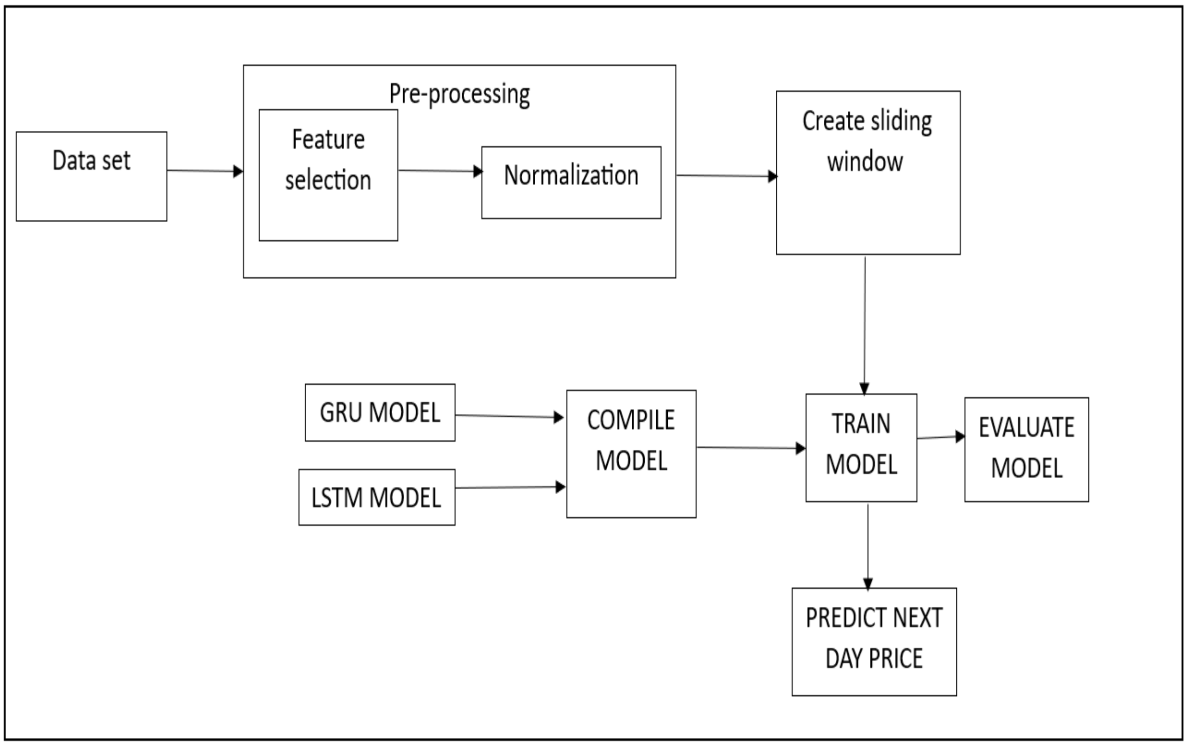


Figure 4.1 System Architecture

Figure 4.1 shows the system architecture diagram for training and testing of the project.

4.2 USE CASE DIAGRAM

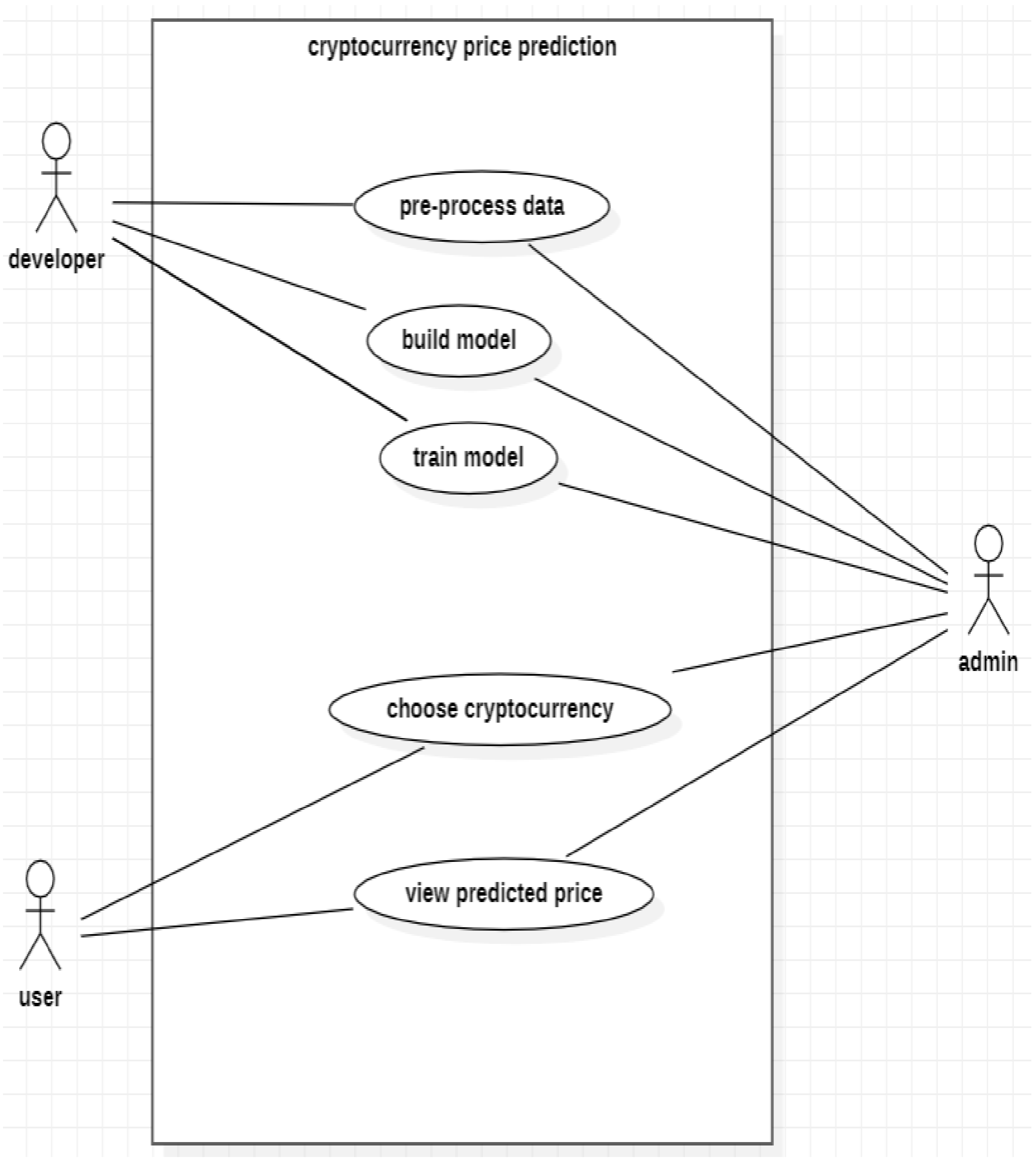


Figure 4.2 Use Case Diagram

Figure 4.2 shows the role of the user, admin, and developer in this project.

4.3 ACTIVITY DIAGRAM

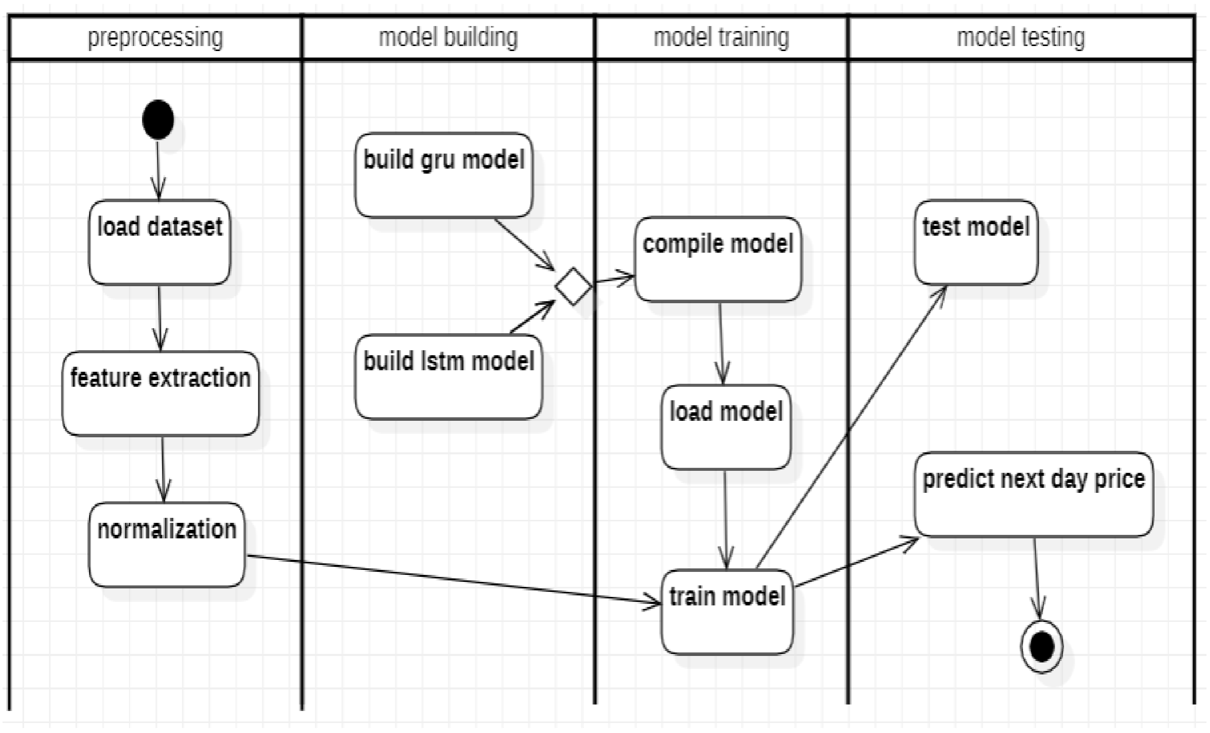


Figure 4.3 Activity diagram for Modelling

Figure 4.3 shows the flow from one activity to another activity.

CHAPTER 5

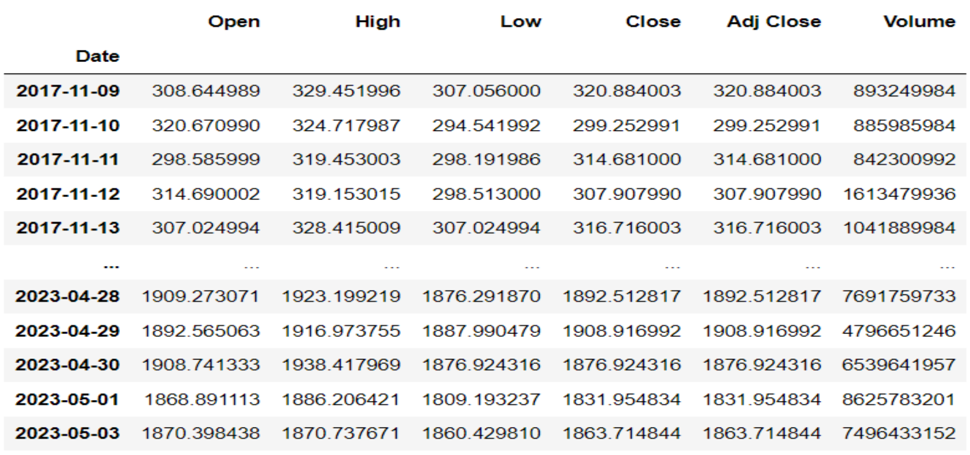
IMPLEMENTATION

5.1 MODULES

- DATASET
- DATASET PREPROCESSING
- MODEL BUILDING
- MODEL TRAINING
- MODEL EVALUATION
- DEPLOYMENT

5.1.1 DATASET

Yahoo Finance Plus is a premium subscription service that provides actionable data and advanced tools for investors to trade with confidence. We use yfinance library in python. It offers a threaded and Pythonic way to download market data from Yahoo!R finance. When the user selects the cryptocurrency from the drop-down menu the library downloads the dataset for that cryptocurrency from yahoo finance and sends it to the preprocessing stage.



	Open	High	Low	Close	Adj Close	Volume
Date						
2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984
...	...	...	...	...	...	...
2023-04-28	1909.273071	1923.199219	1876.291870	1892.512817	1892.512817	7691759733
2023-04-29	1892.565063	1916.973755	1887.990479	1908.916992	1908.916992	4796651246
2023-04-30	1908.741333	1938.417969	1876.924316	1876.924316	1876.924316	6539641957
2023-05-01	1868.891113	1886.206421	1809.193237	1831.954834	1831.954834	8625783201
2023-05-03	1870.398438	1870.737671	1860.429810	1863.714844	1863.714844	7496433152

5.1.2 DATASET PREPROCESSING

The code for predicting the future prices of a selected cryptocurrency goes through a preprocessing stage, which involves several steps. First, the data is downloaded from Yahoo Finance using the yf.download() function from the yfinance library. Then, a Pandas DataFrame is created from the Close column of the downloaded data, which represents the dataset that will be used to train and test the prediction model.

The values in the Close column are then normalized using Min-Max normalization with the MinMaxScaler() function from the sklearn.preprocessing library. The dataset is partitioned into training, validation, and test sets, with the training set containing 70% of the data, the validation set containing 10%, and the test set containing 20%. Sliding windows of size lag are created in the training, validation, and test sets using the create_sliding_windows() function. The x and y arrays are then reshaped to have the shape (samples, timesteps, features), where samples are the number of sequences, timesteps is the size of the sliding window (lag), and features is the number of input features (in this case, there is only one input feature, which is the closing price of the cryptocurrency).

The output of the preprocessing stage is a set of input and output sequences for the prediction model, which have been extracted from the original dataset using sliding windows of size lag and have been normalized using Min-Max normalization. These sequences are used to train and test the GRU and LSTM models for predicting the future prices of the selected cryptocurrency.

5.1.3 MODEL BUILDING

The code for predicting the future prices of a selected cryptocurrency involves building two types of recurrent neural network (RNN) models, namely a Gated Recurrent Unit (GRU) model and a Long Short-Term Memory (LSTM) model.

The first few steps in the model building stage for predicting future prices of a cryptocurrency involve importing the required libraries, defining hyperparameters, and defining the architecture of the models. The necessary libraries for building the models, such as TensorFlow, Keras, and NumPy, are imported into the code. The hyperparameters for the models, including the number of neurons, the number of epochs, the batch size, and the learning rate, are defined. The architecture of the models is defined using the Keras Sequential API, which consists of an input layer, followed by several hidden layers of GRU or LSTM cells, and an output layer. Dropout layers are also added to the models to prevent overfitting. The models are compiled using the Keras compile () function. The loss function, optimizer, and evaluation metric are specified for the models. After these steps, the models are ready to be compiled and trained.

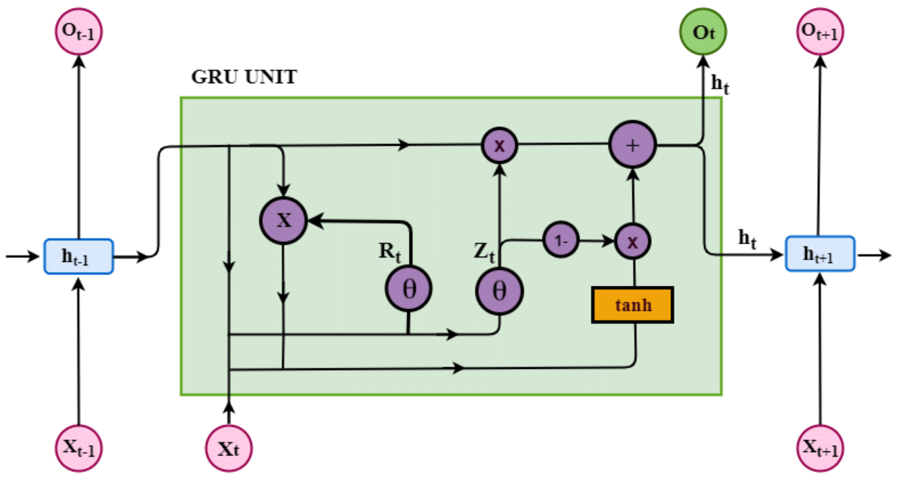
LEARNING PARAMETERS

Learning rate	0.0001
Hidden unit	64
Batch size	256
Epochs	100

5.1.2.1 GATED RECURRENT UNIT (GRU)

A GRU (Gated Recurrent Unit) is a type of neural network used in deep learning for processing sequential data. It is a variation of the RNN architecture that uses gating mechanisms to control the flow of information within the network. The GRU was introduced by Cho et al. in 2014 and has gained popularity due to its simple structure and faster training time compared to other RNNs such as LSTM.

The GRU uses two gates, a reset gate, and an update gate, to selectively reset and update the state of each cell in the network. This allows the GRU to effectively capture long-term dependencies in sequential data.

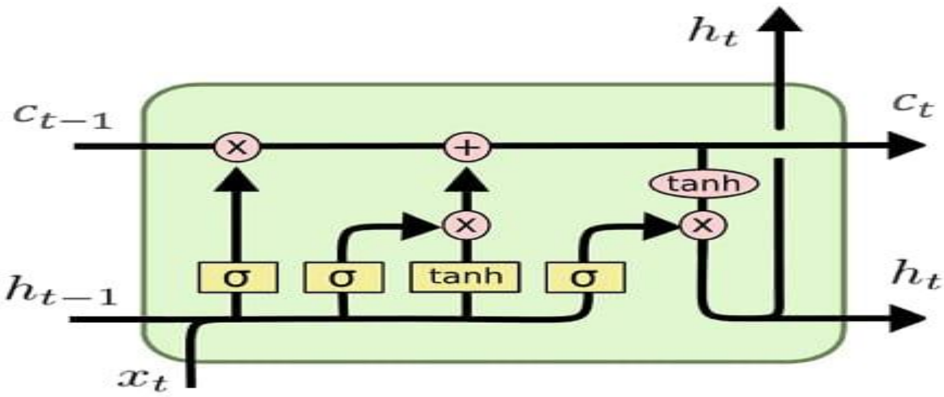


5.1.2.2 LONG SHORT-TERM MEMORY (LSTM)

LSTM (Long Short-Term Memory) layer is a type of recurrent neural network (RNN) layer that is commonly used for processing sequential data such as speech, text, or time-series data.

Unlike regular RNNs, which have a single hidden state vector that gets updated at every time step, LSTM has three gates (input, forget, and output) that regulate the flow of information into and out of the memory cell. This makes it easier for LSTM to learn long-term dependencies in the input sequence by selectively remembering or forgetting certain pieces of information.

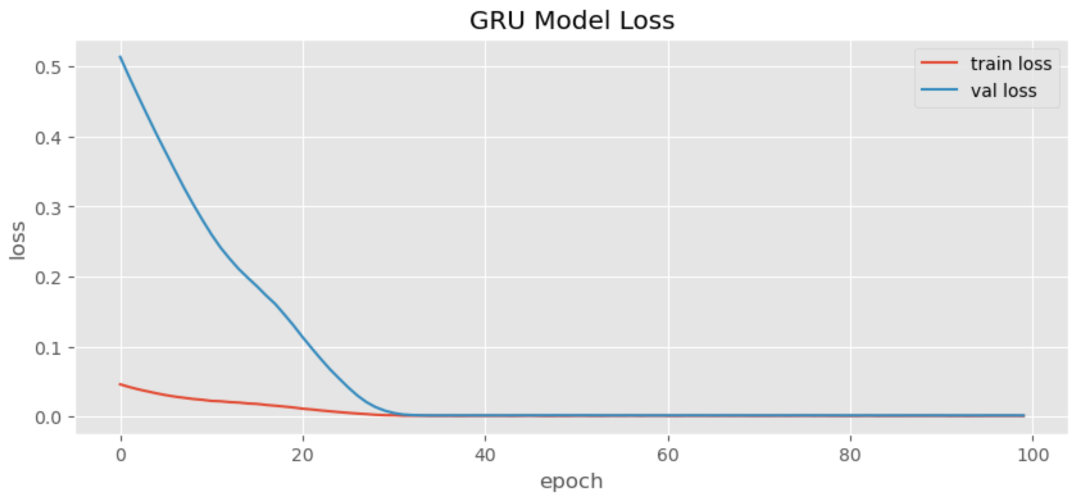
At each time step, the LSTM layer takes in an input vector and the previous hidden state and cell state and computes a new hidden state and cell state. The output of the LSTM layer can be either the hidden state, the cell state, or a combination of both depending on the task at hand.

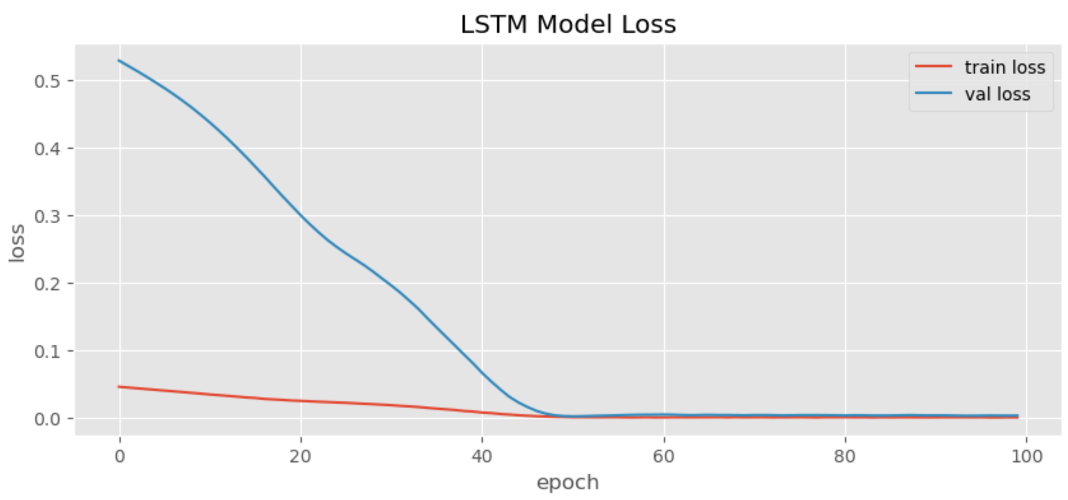


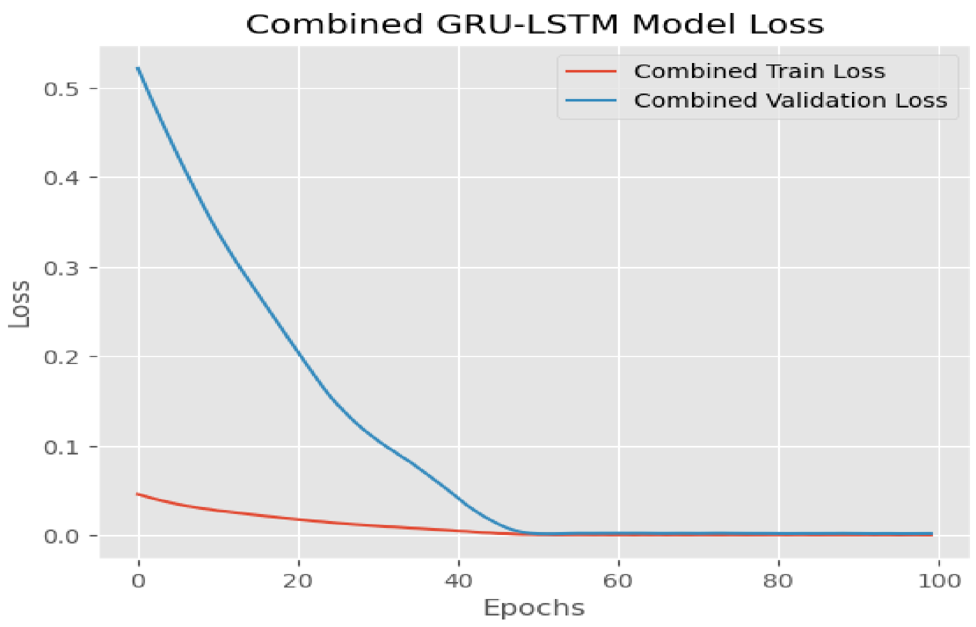
LSTM
(Long−Short Term Memory)

5.1.4 MODEL TRAINING

After the models are compiled the GRU and LSTM models are trained on the training set for 100 number of epochs using the fit () method. During training, the validation set is used to monitor the model's performance and prevent overfitting.







5.1.5 MODEL EVALUATION

Model evaluation is an important part of developing and testing machine learning models. The following evaluation metrics are used:

- Mean squared error (MSE): The mean squared error is used as the loss function in the models, and it is also calculated after the models are trained to evaluate their performance on the test set. The mean squared error is calculated as the average of the squared differences between the predicted and actual values.

- Mean absolute percentage error (MAPE): The mean absolute percentage error is not used in the code, but it is imported from scikit-learn library. It can be used to evaluate the performance of the models, but it is not calculated in this code. The MAPE is calculated as the average of the absolute percentage differences between the predicted and actual values.

Evaluation metric	**LSTM-GRU**
RMSE	100.24198402546371
MAPE	5.0267915483030166

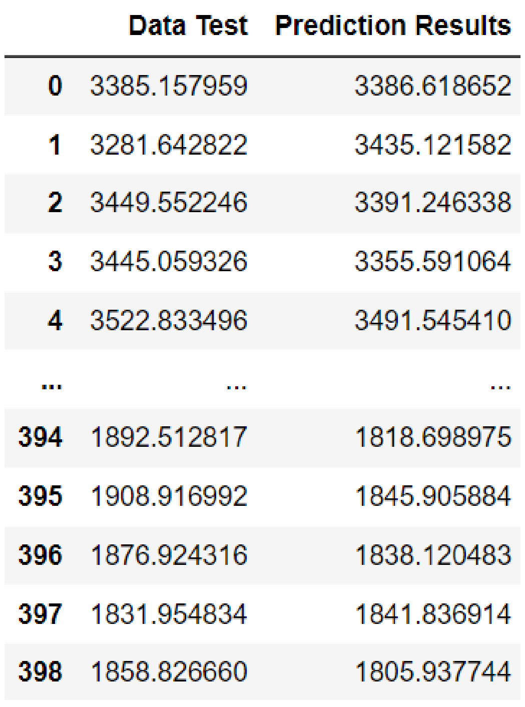
5.1.6 FUTURE PREDICTION

The given code performs a time-series analysis on a dataset using the LSTM and GRU models. The last few data points in the dataset are selected and normalized using a scaler. The normalized data is then reshaped to feed into the models. The predicted price of the next day is obtained by taking the average of the predictions made by the LSTM and GRU models. The predicted price is then transformed back to the original scale using the inverse transform of the scaler. Based on whether the predicted price is higher or lower than the current price, a message is displayed indicating if the price is predicted to increase or decrease tomorrow. The current price and predicted price values are also displayed.

CHAPTER 6

RESULT AND ANALYSIS

To assess the precision of the model, we can compare its predicted values against the actual values. The model was trained on 70% of the available historical data, validation set containing 10% and the remaining 20% was used for testing. The loss function being used by the model is mean squared error, with the Adam optimizer employed to minimize the loss. To measure the accuracy of the model, various metrics such as mean absolute percentage error (MAPE) and root mean squared error (RMSE) can be utilized by comparing the predicted values against the actual values. Additionally, a line chart can be used to visually compare the predicted and actual values. The (MAPE) of most cryptocurrencies is observed to be around 5%.



	Data Test	Prediction Results
0	3385.157959	3386.618652
1	3281.642822	3435.121582
2	3449.552246	3391.246338
3	3445.059326	3355.591064
4	3522.833496	3491.545410
...	...	...
394	1892.512817	1818.698975
395	1908.916992	1845.905884
396	1876.924316	1838.120483
397	1831.954834	1841.836914
398	1858.826660	1805.937744

Evaluation metric	**GRU**	**LSTM**	**LSTM-GRU**
RMSE	85.225081500	127.479746689	100.24198402546371
MAPE	3.8981175167238256	6.91422365224	5.0267915483030166

CHAPTER 7

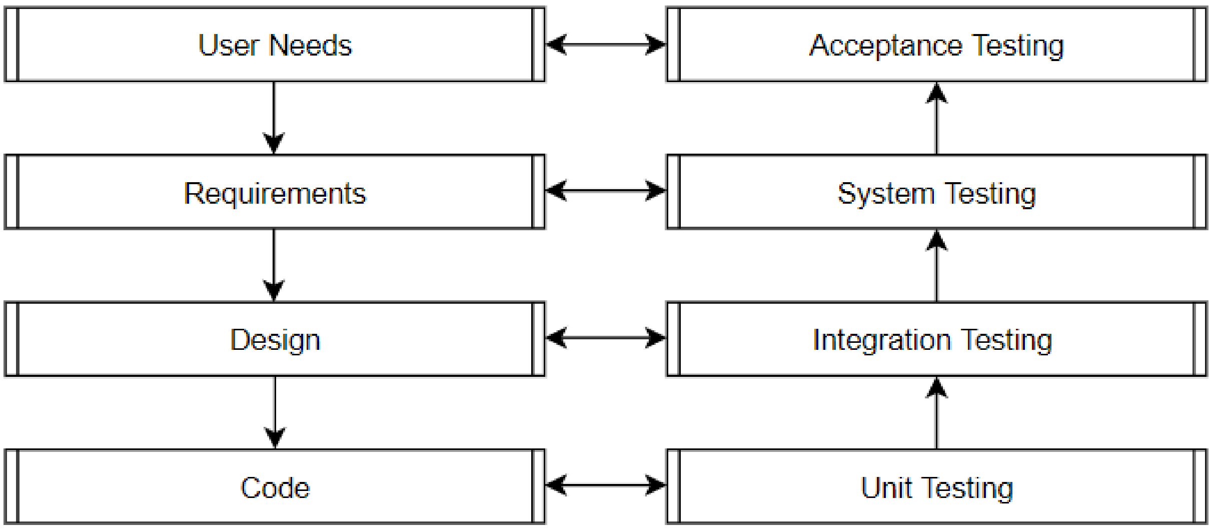
SYSTEM TESTING

7.1 TESTING OBJECTIVES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.2 TYPES OF TESTS

To uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are.



7.2.1 UNIT TEST CASES

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

In this project the unit testing validates the program logic of all modules in range of machine learning.

The testing takes place as:

Input: Select Module.

Output: Show which module is selected and show instructions to work with that module.

7.2.2 FUNCTIONAL TEST CASE

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: dataset

Functions: Inputs must be processed based on the constraints and Deep Learning algorithm must be executed.

Output: Probability of data is carried out.

7.2.3 INTEGRATION TEST CASES

Integration tests are designed to test integrated software and hardware components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing. The combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

In this project the integration takes place in hardware components and software testing is verified.

Input: Proper functional testing of the system (Device), processing speed, Memory, Data usage.

Output: System working well, instantaneous output.

Front End: tkinter.

Back End: Tensorflow, Numpy.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

The code is a functional implementation of a LSTM-GRU model to predict the price of a cryptocurrency based on its historical data using the yfinance library. The user selects the cryptocurrency for which they want to predict the price, and the code returns a prediction for the next day's price along with a graph of the actual and predicted prices.

8.1 FUTURE ENHANCEMENT

This project can be further improved with the following future enhancements:

- Add error handling to deal with incorrect user inputs, such as invalid cryptocurrency names or network errors when downloading data from Yahoo Finance.
- Improve the accuracy of the prediction model. One way to do this is to experiment with different hyperparameters and models to see which one performs the best on the given dataset.
- Add more features to the prediction model, such as volume and open price, to see if they improve the accuracy of the model.
- Add a save function to save the predicted prices and graphs to a file. This can be useful for keeping a record of the predicted prices over time.

APPENDICES

APPENDIX 1

SAMPLE CODING

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, GRU, LSTM
from keras import optimizers
import tkinter as tk
import yfinance as yf
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

root = tk.Tk()
root.title("Cryptocurrency Price Prediction")
root.attributes('-fullscreen', True)

# Create labels
ticker_label = tk.Label(root, text="Enter the cryptocurrency:")
ticker_label.pack()

# Create dropdown menu
crypto_list = ['BTC-USD', 'ETH-USD', 'DOGE-USD']
```

```
selected_crypto = tk.StringVar()
selected_crypto.set(crypto_list[0])
crypto_dropdown = tk.OptionMenu(root, selected_crypto, *crypto_list)
crypto_dropdown.pack()

# Create loading screen
loading_screen = tk.Toplevel()
loading_screen.title("Loading...")
loading_screen.attributes('-fullscreen', True)
loading_label = tk.Label(loading_screen, text="Please wait while the prediction
is being processed...")
loading_label.pack()

# Define function to predict price
def predict_price():
    loading_screen.deiconify()

    seed = 1234
    np.random.seed(seed)
    plt.style.use('ggplot')

    import warnings
    warnings.simplefilter(action='ignore', category=FutureWarning)
    message_text.insert(tk.END, "The Dataset of entered Cryptocurrency is being
downloaded." + "\n")
    ticker = selected_crypto.get()

    # Download data from Yahoo Finance
    dataraw =  yf.download(ticker)
    # use feature 'Date' & 'Close'
    dataset = pd.DataFrame(dataraw['Close'])

    #Min-Max Normalization
    message_text.insert(tk.END, "The Dataset is being preprocessed currently." +
"\n")
    message_text.delete("1.0", "end")
    dataset_norm = dataset.copy()
    dataset[['Close']]
    scaler = MinMaxScaler()
    dataset_norm['Close'] = scaler.fit_transform(dataset[['Close']])

    # Partition data into data train, val & test
    totaldata = dataset.values
    totaldatatrain = int(len(totaldata)*0.7)
    totaldataval = int(len(totaldata)*0.1)
    totaldatatest = int(len(totaldata)*0.2)

    # Store data into each partition
    message_text.insert(tk.END, "Pediction is under process" + "\n")
    message_text.delete("1.0", "end")
    training_set = dataset_norm[0:totaldatatrain]
    val_set=dataset_norm[totaldatatrain:totaldatatrain+totaldataval]
    test_set = dataset_norm[totaldatatrain+totaldataval:]

    # Initiaton value of lag
    lag = 2
    # sliding windows function
```

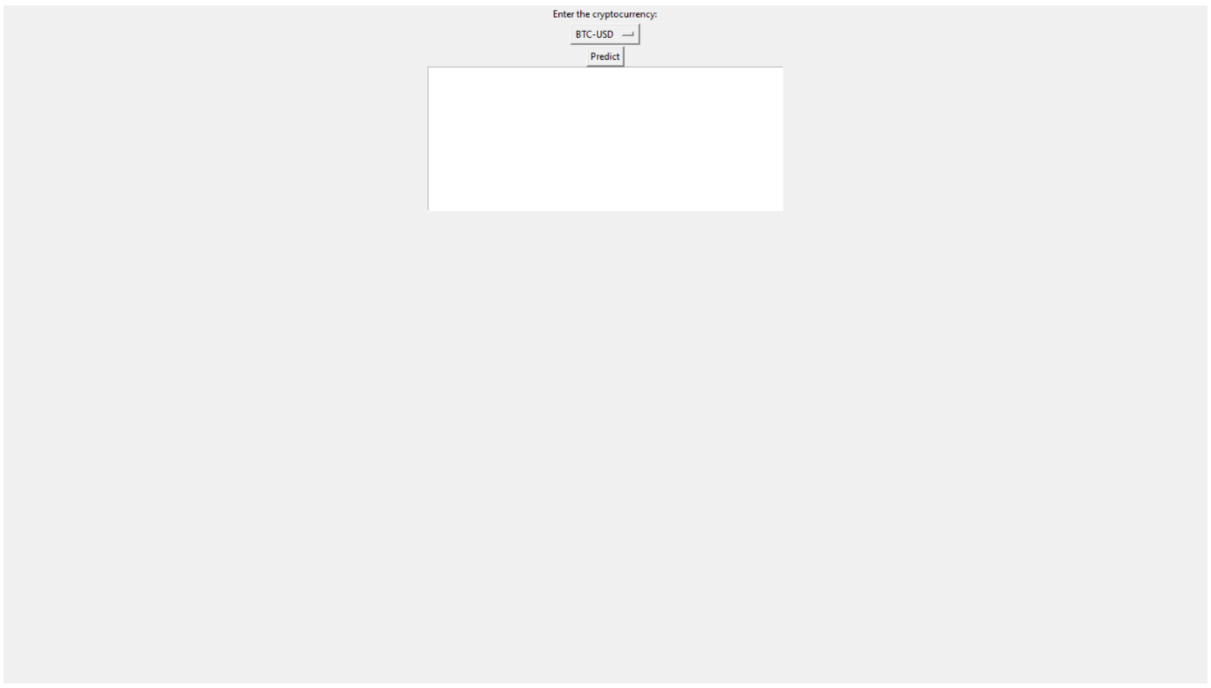
```
def create_sliding_windows(data,len_data,lag):
    x=[]
    y=[]
    for i in range(lag,len_data):
        x.append(data[i-lag:i,0])
        y.append(data[i,0])
    return np.array(x),np.array(y)

# Formating data into array for create sliding windows
array_training_set = np.array(training_set)
array_val_set = np.array(val_set)
array_test_set = np.array(test_set)

# Create sliding windows into training data
x_train, y_train =
create_sliding_windows(array_training_set,len(array_training_set), lag)
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
# Create sliding windows into validation data
x_val,y_val = create_sliding_windows(array_val_set,len(array_val_set),lag)
x_val = np.reshape(x_val, (x_val.shape[0],x_val.shape[1],1))
# Create sliding windows into test data
x_test,y_test = create_sliding_windows(array_test_set,len(array_test_set),lag)
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))

# Hyperparameters
learning_rate = 0.0001
hidden_unit = 64
batch_size = 256
epochs = 100
```

```
    # Define GRU model
    regressorGRU = Sequential()
    regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
input_shape=(x_train.shape[1],1), activation='tanh'))
    regressorGRU.add(Dropout(0.2))
    regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
activation='tanh'))
    regressorGRU.add(Dropout(0.2))
    regressorGRU.add(GRU(units=hidden_unit, return_sequences=False,
activation='tanh'))
    regressorGRU.add(Dropout(0.2))
    regressorGRU.add(Dense(units=1))
    message_text.insert(tk.END, "Pediction is under process" + "\n")
    message_text.delete("1.0", "end")
    regressorGRU.compile(optimizer=optimizers.Adam(learning_rate),
loss='mean_squared_error')

    # Define LSTM model
    regressorLSTM = Sequential()
    regressorLSTM.add(LSTM(units=hidden_unit, return_sequences=True,
input_shape=(x_train.shape[1],1), activation='tanh'))
    regressorLSTM.add(Dropout(0.2))
    regressorLSTM.add(LSTM(units=hidden_unit, return_sequences=True,
activation='tanh'))
    regressorLSTM.add(Dropout(0.2))
    regressorLSTM.add(LSTM(units=hidden_unit, return_sequences=False,
activation='tanh'))
    regressorLSTM.add(Dropout(0.2))
    regressorLSTM.add(Dense(units=1))
    message_text.insert(tk.END, "Pediction is under process" + "\n")
    message_text.delete("1.0", "end")
    regressorLSTM.compile(optimizer=optimizers.Adam(learning_rate),
loss='mean_squared_error')


    # Train the models
    history_gru = regressorGRU.fit(x_train, y_train, validation_data=(x_val,
y_val), batch_size=batch_size, epochs=epochs)
    history_lstm = regressorLSTM.fit(x_train, y_train, validation_data=(x_val,
y_val), batch_size=batch_size, epochs=epochs)
    # Combine predictions from both models
    pred_gru = regressorGRU.predict(x_test)
    pred_lstm = regressorLSTM.predict(x_test)
    pred = (pred_gru + pred_lstm) / 2

    # Get the train and validation loss from the history object
    train_loss_gru = history_gru.history['loss'][-1]
    val_loss_gru = history_gru.history['val_loss'][-1]
    train_loss_lstm = history_lstm.history['loss'][-1]
    val_loss_lstm = history_lstm.history['val_loss'][-1]

    # Print the train and validation loss
    print(f'GRU model train loss: {train_loss_gru:.4f}')
    print(f'GRU model validation loss: {val_loss_gru:.4f}')
    print(f'LSTM model train loss: {train_loss_lstm:.4f}')
    print(f'LSTM model validation loss: {val_loss_lstm:.4f}')
    # Tabel value of training loss & validation loss
    # Create a dataframe to hold the loss values
    loss_df = pd.DataFrame({
        'Model': ['GRU', 'LSTM'],
        'Train Loss': [train_loss_gru, train_loss_lstm],
        'Val Loss': [val_loss_gru, val_loss_lstm]
    })

    # Print the dataframe
    print(loss_df.to_string(index=False))

    # Implementation model into data test
    y_pred = (regressorGRU.predict(x_test)+regressorLSTM.predict(x_test))/2

    # Rescale the predicted and actual values
    y_pred = scaler.inverse_transform(y_pred)
    y_test = scaler.inverse_transform(y_test.reshape(-1, 1))

    # Calculate the RMSE
    mse = np.mean((y_test - y_pred)**2)
    rmse = np.sqrt(mse)
    message_text.insert(tk.END, "The Root Mean Squared Error is:" + str(rmse)
+ "\n")

    # Calculate the MAPE
    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
    message_text.insert(tk.END, "The Mean Absolute Percentage Error is: " +
str(mape) + "\n")
```

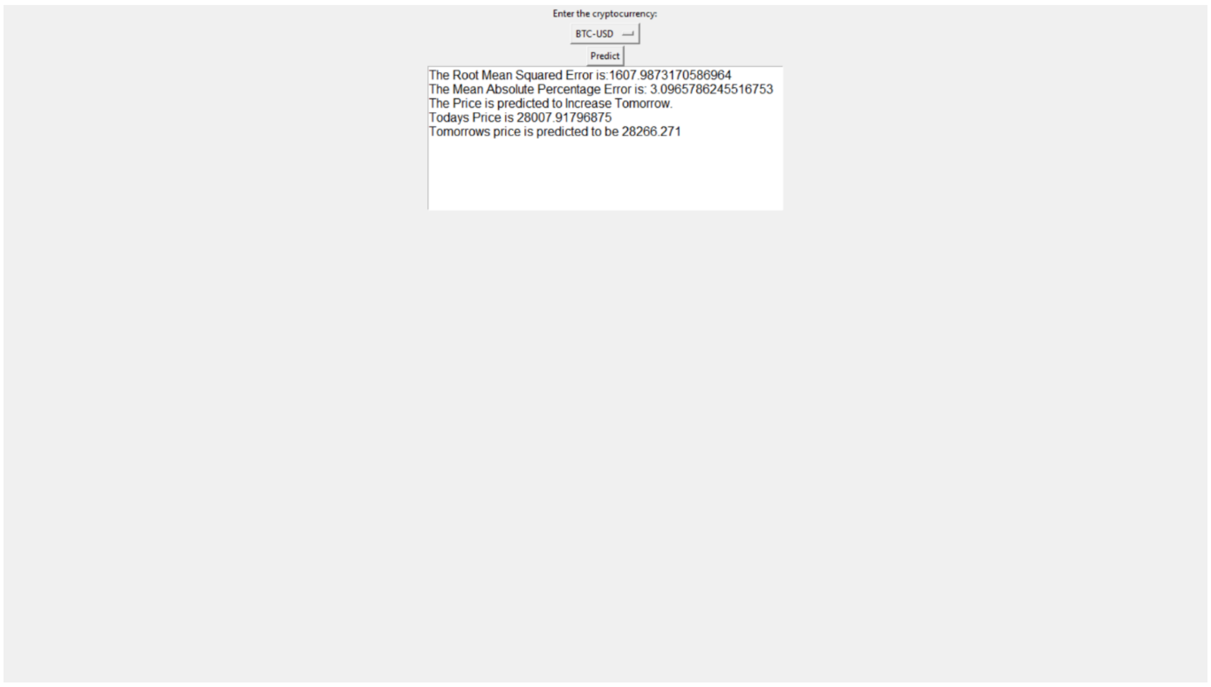
```
    # Comparison data test with data prediction
    datacompare = pd.DataFrame()
    datatest=np.array(dataset['Close'][totaldatatrain+totaldataval+lag:])
    datapred= y_pred

    datacompare['Data Test'] = datatest
    datacompare['Prediction Results'] = datapred
    datacompare

    last_data = dataset[-lag:].values
    last_data_norm = scaler.transform(last_data)
    last_data_norm = last_data_norm.reshape(1, lag, 1)

    # Predict next day's price using the combined model
    predicted_price = (regressorGRU.predict(last_data_norm) +
regressorLSTM.predict(last_data_norm)) / 2

    # Inverse transform the predicted price to get the actual price
    predicted_price = scaler.inverse_transform(predicted_price)[0][0]



    # Determine if the price will increase or decrease
    current_price = dataraw['Close'][-1]
    next_day_price = predicted_price
    if next_day_price > current_price:
        message_text.insert(tk.END, "The Price is predicted to Increase
Tomorrow." + "\n")
    else:
        message_text.insert(tk.END, "The Price is predicted to Decrease
Tomorrow." + "\n")

    message_text.insert(tk.END, "Todays Price is " + str(current_price) + "\n")
    message_text.insert(tk.END, "Tomorrows price is predicted to be " +
str(predicted_price) + "\n")

    loading_screen.withdraw()

# Create predict button
predict_button = tk.Button(root, text="Predict", command=predict_price)
predict_button.pack()

message_text = tk.Text(root, height=10, width=50, font=("Arial", 12))
message_text.pack()

root.mainloop()
```

APPENDIX 2

SCREENSHOTS



Please wait while the prediction is being processed...



```
The Root Mean Squared Error is:1607.9873170586964
The Mean Absolute Percentage Error is: 3.0965786245516753
The Price is predicted to Increase Tomorrow.
Todays Price is 28007.91796875
Tomorrows price is predicted to be 28266.271
```

REFERENCES

1. Sentiment Analysis and Emotion Detection on Cryptocurrency Related Tweets Using Ensemble LSTM - GRU Model

Authors: Naila Aslam; Furqan Rustam; Ernesto Lee; Patrick Bernard Washington; Imran Ashraf

2. Stochastic Neural Networks for Cryptocurrency Price Prediction

Author: Patel Jay; Vasu Kalariya; Pushpendra Parmar; Sudeep Tanwar; Neeraj Kumar; Mamoun Alazab

3. Deep Learning-Based Cryptocurrency Price Prediction Scheme with Inter-Dependent Relations

Author: Sudeep Tanwar; Nisarg P. Patel; Smit N. Patel; Jil R. Patel; Gulshan Sharma; Innocent E. Davidson

4. Improving the Cryptocurrency Price Prediction Performance Based on Reinforcement Learning

Author: Zeinab Shahbazi; Yung-Cheol Byun

5. Improving the Cryptocurrency Price Prediction Performance Based on Reinforcement Learning

Author: Raj Parekh; Nisarg P. Patel; Nihar Thakkar; Rajesh Gupta; Sudeep Tanwar; Gulshan Sharma

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

PO1	**Engineering Knowledge**
	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
PO2	**Problem Analysis**
	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
PO3	**Design/Development of Solutions**
	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
PO4	**Conduct Investigations of Complex Problems**
	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
PO5	**Modern Tool Usage**
	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
PO6	**The Engineer and Society**
	Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7	**Environment and Sustainability** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
PO8	**Ethics** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
PO9	**Individual and Team Work** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
PO10	**Communication** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and receive clear instructions
PO11	**Project Management and Finance** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
PO12	**Life-Long Learning** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1	Specify, design, implement and test digital and analog electronic systems using state of art component and software tools
PSO2	Architect and specify the analog and digital communication systems as per the performance requirement specifications.
PSO3	Understand and specify the components of RF/Wireless communication systems

Thus by doing this project we have attained the above POS and PSOS.