COS30018- Task 1: Setup

Alvi Hossain Safri Himalya

Student ID

104223717

Attended Lab

Friday- 2.30-4.30

1. Introduction

This report outlines the steps taken to set up the environment, test the provided code bases (v0.1 ,P1 &p2), and summarize the initial code base v0.1. The goal is to prepare a working environment, test the stock price prediction models, and provide documentation for future project development.

2. Environment Setup

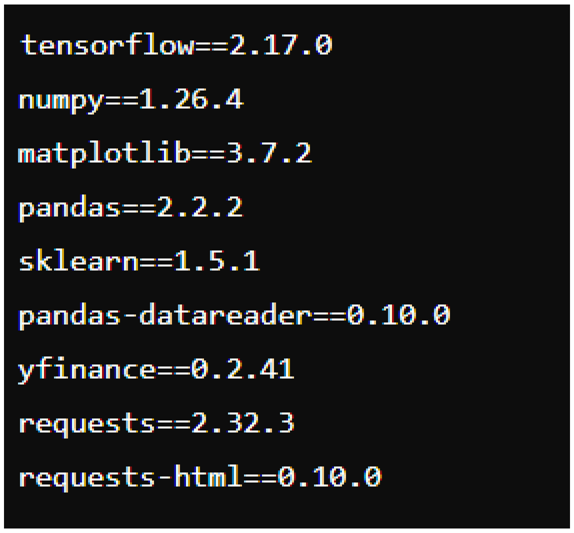
2.1 Virtual Environment Setup

To maintain a clean and isolated environment, a virtual environment was created for running the stock prediction models.

Steps:

1. **Create Virtual Environment:**
   - Command: python -m venv env
2. **Activate Virtual Environment:**
   - On Windows: .\env\Scripts\activate
3. **Install Required Libraries:**
   - The required libraries were installed using the requirements.txt file. The requirements file was generated based on the dependencies in both v0.1 and P1.

Requirements File (requirements.txt):



```
tensorflow==2.17.0
numpy==1.26.4
matplotlib==3.7.2
pandas==2.2.2
sklearn==1.5.1
pandas-datareader==0.10.0
yfinance==0.2.41
requests==2.32.3
requests-html==0.10.0
```

3. Testing of Code Bases

3.1 Testing v0.1

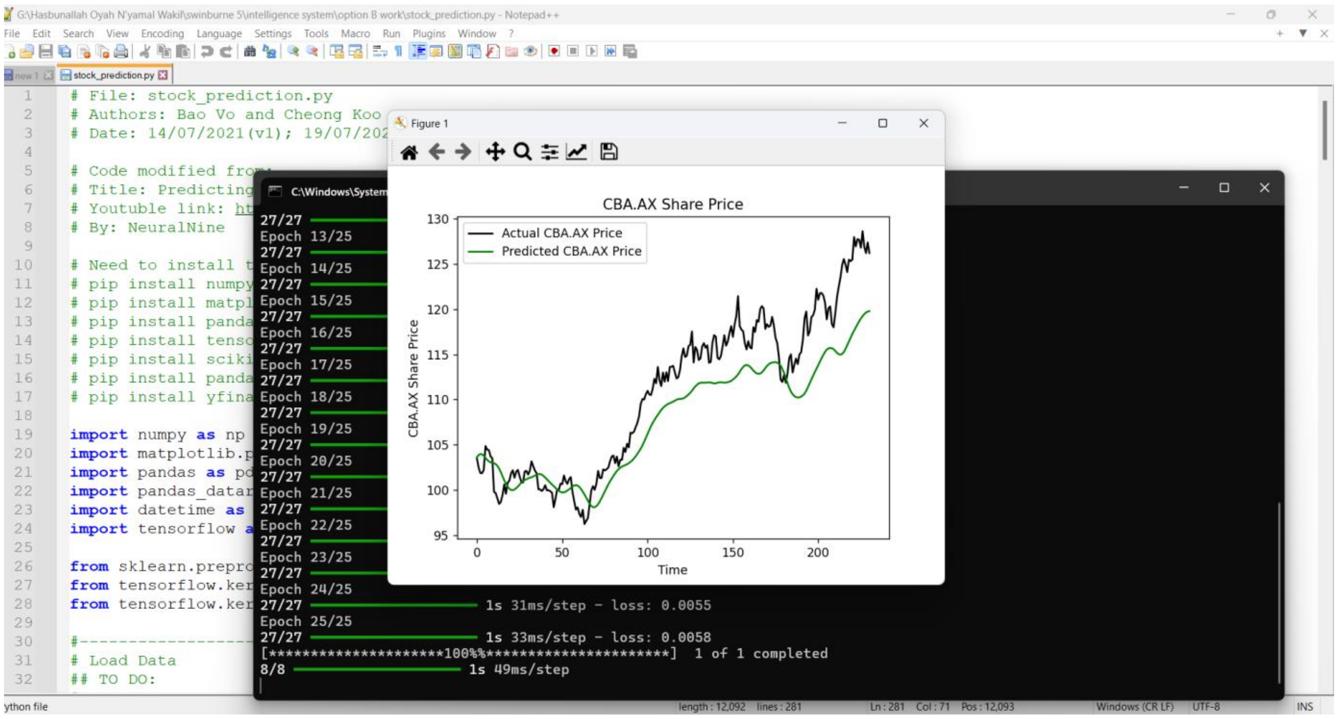
The initial code base v0.1 was obtained from Canvas. The setup and testing followed the instructions provided in the YouTube tutorial.

Steps:

1. **Run the Script:**
   - Command: python stock_prediction.py
2. **Observations:**
   - The code successfully fetched stock data from Yahoo Finance.
   - The model was trained using the LSTM neural network.
   - The final output included a graphical representation of the actual versus predicted stock prices.

Screenshots:

- See the attached screenshots showing the setup, training, and output of the v0.1 model.


G:\Hasbunallah Oyah N'yamal Wakil\swinburne 5\intelligence system\option B work\stock_prediction.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
stock_prediction.py
C:\Windows\System
Figure 1
CBA.AX Share Price
Actual CBA.AX Price
Predicted CBA.AX Price
CBA.AX Share Price
Time
length : 12,092 lines : 281 Ln : 281 Col : 71 Pos : 12,093 Windows (CR LF) UTF-8 INS

```
# File: stock_prediction.py
# Authors: Bao Vo and Cheong Koo
# Date: 14/07/2021(v1); 19/07/202

# Code modified fro
# Title: Predicting
# Youtuble link: ht
# By: NeuralNine

# Need to install t
# pip install numpy
# pip install matpl
# pip install panda
# pip install tenso
# pip install sciki
# pip install panda
# pip install yfina

import numpy as np
import matplotlib.p
import pandas as pd
import pandas_datar
import datetime as
import tensorflow a

from sklearn.prepro
from tensorflow.ker
from tensorflow.ker

#-----------------
# Load Data
## TO DO:
```

```
27/27
Epoch 13/25
27/27
Epoch 14/25
27/27
Epoch 15/25
27/27
Epoch 16/25
27/27
Epoch 17/25
27/27
Epoch 18/25
27/27
Epoch 19/25
27/27
Epoch 20/25
27/27
Epoch 21/25
27/27
Epoch 22/25
27/27
Epoch 23/25
27/27
Epoch 24/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0055
Epoch 25/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0058
[*********************100%%**********************]  1 of 1 completed
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step
```

3.2 Testing P1

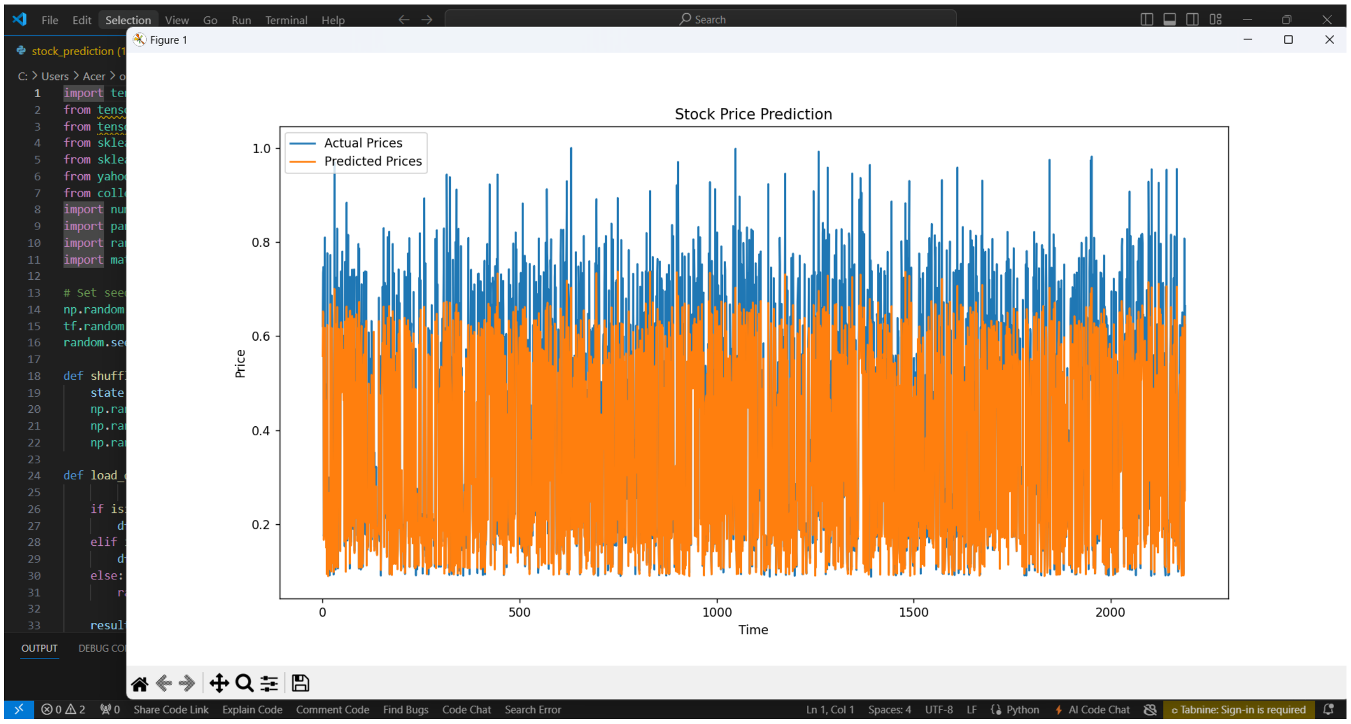
The P1 project was cloned from the provided GitHub repository: https://github.com/x4nth055/pythoncode-tutorials/tree/master/machine-learning/stock-prediction.

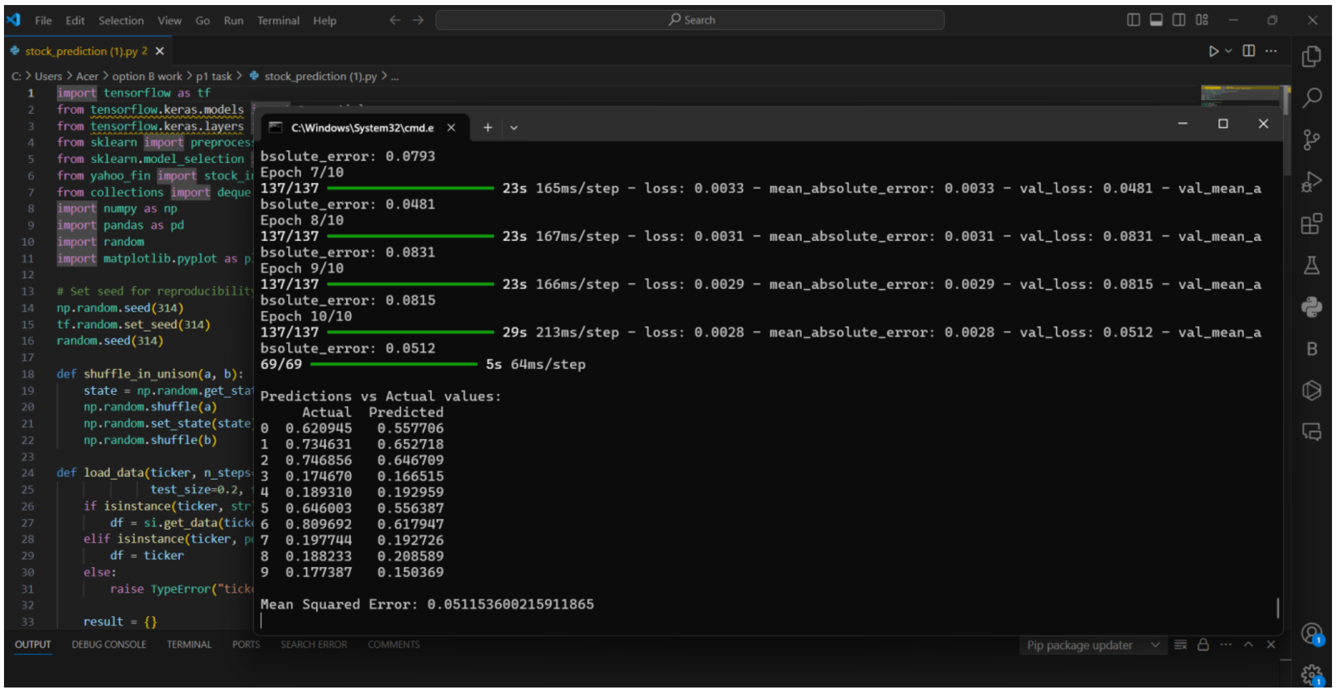
Steps:

1. **Clone the Repository:**
   - Command: git clone https://github.com/x4nth055/pythoncode-tutorials.git
2. **Run the Script:**
   - Command: python stock_prediction(1).py (adjust paths as necessary)
3. **Observations:**
   - Similar to v0.1, P1 fetched stock data and trained an LSTM model.
   - The predictions were outputted in a graph comparing actual and predicted stock prices.

Screenshots:

- See the attached screenshots showing the setup, training, and output of the P1 model





```python
import tensorflow as tf
from tensorflow.keras.models
from tensorflow.keras.layers
from sklearn import preproces
from sklearn.model_selection
from yahoo_fin import stock_i
from collections import deque
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as p

# Set seed for reproducibility
np.random.seed(314)
tf.random.set_seed(314)
random.seed(314)

def shuffle_in_unison(a, b):
    state = np.random.get_stat
    np.random.shuffle(a)
    np.random.set_state(state
    np.random.shuffle(b)

def load_data(ticker, n_steps
              test_size=0.2,
    if isinstance(ticker, str
        df = si.get_data(tick
    elif isinstance(ticker, p
        df = ticker
    else:
        raise TypeError("tick

    result = {}
```

```
bsolute_error: 0.0793
Epoch 7/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 165ms/step - loss: 0.0033 - mean_absolute_error: 0.0033 - val_loss: 0.0481 - val_mean_a
bsolute_error: 0.0481
Epoch 8/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 167ms/step - loss: 0.0031 - mean_absolute_error: 0.0031 - val_loss: 0.0831 - val_mean_a
bsolute_error: 0.0831
Epoch 9/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 166ms/step - loss: 0.0029 - mean_absolute_error: 0.0029 - val_loss: 0.0815 - val_mean_a
bsolute_error: 0.0815
Epoch 10/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 29s 213ms/step - loss: 0.0028 - mean_absolute_error: 0.0028 - val_loss: 0.0512 - val_mean_a
bsolute_error: 0.0512
69/69 ━━━━━━━━━━━━━━━━━━━━ 5s 64ms/step

Predictions vs Actual values:
     Actual  Predicted
0  0.620945   0.557706
1  0.734631   0.652718
2  0.746856   0.646709
3  0.174670   0.166515
4  0.189310   0.192959
5  0.646003   0.556387
6  0.809692   0.617947
7  0.197744   0.192726
8  0.188233   0.208589
9  0.177387   0.150369

Mean Squared Error: 0.051153600215911865
```

4. Summary of Initial Code Base v0.1

4.1 Overview

The v0.1 code base is a Python script that implements a basic stock price prediction model using LSTM. The model is based on a YouTube tutorial and has several limitations, including issues with code structure and performance.

4.2 Key Components

- **Data Collection:** Uses the yfinance library to fetch historical stock price data.
- **Data Preprocessing:** Normalizes data using MinMaxScaler and splits it into training and testing datasets.
- **Model Construction:** Utilizes an LSTM model with sequential layers in TensorFlow/Keras.
- **Prediction and Visualization:** Outputs a graph of actual versus predicted stock prices.

4.3 Issues Identified

- **Code Structure:** The code is not modular and lacks proper function definitions.
- **Performance:** The model performs adequately but struggles with high volatility in stock prices.
- **Dependencies:** The dependencies are not well-documented, which could cause issues during setup.

4.4 Future Improvements

- **Modularization:** Refactor the code into functions or classes to improve readability and maintainability.
- **Data Enrichment:** Include additional features like trading volume or market indicators.
- **Advanced Models:** Explore more sophisticated models or ensembles to improve prediction accuracy.

---

5. P2 Task: Exploration of the Project "Using Deep Learning Neural Networks and Candlestick Chart Representation to Predict Stock Market"

5.1 Overview

The P2 project was explored as an additional task to enhance understanding and provide insights for potential improvements in the main project. The project utilizes deep learning and candlestick chart representation to predict stock market trends.

5.2 Setup in a Separate Virtual Environment

To avoid conflicts with the libraries and dependencies of v0.1 and P1, a separate virtual environment was created for P2.

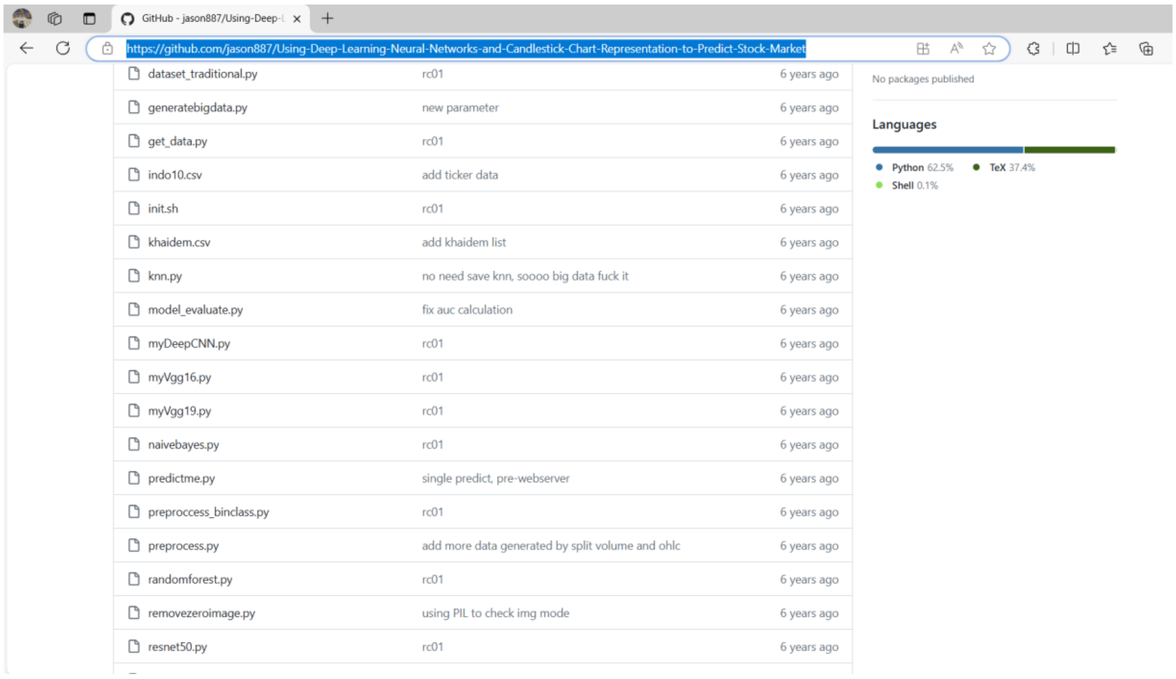
Steps:

1. Create a New Virtual Environment:
   - Command: python -m venv env_p2
2. Activate the Environment:
   - On Windows: .\env_p2\Scripts\activate
3. Install Required Libraries:
   - The requirements.txt file provided by the P2 project was used to install necessary dependencies.

5.3 Testing P2

Steps:

1. Clone the Repository:
   - Command: git clone https://github.com/jason887/Using-Deep-Learning-Neural-Networks-and-Candlestick-Chart-Representation-to-Predict-Stock-Market.git
2. Run the Script:
   - Command: python <P2_Script_Name>.py
3. Observations:
   - The model used deep learning techniques to predict stock market trends based on candlestick chart patterns.
   - The project introduced new methodologies not covered in v0.1 and P1, providing valuable insights for future improvements.

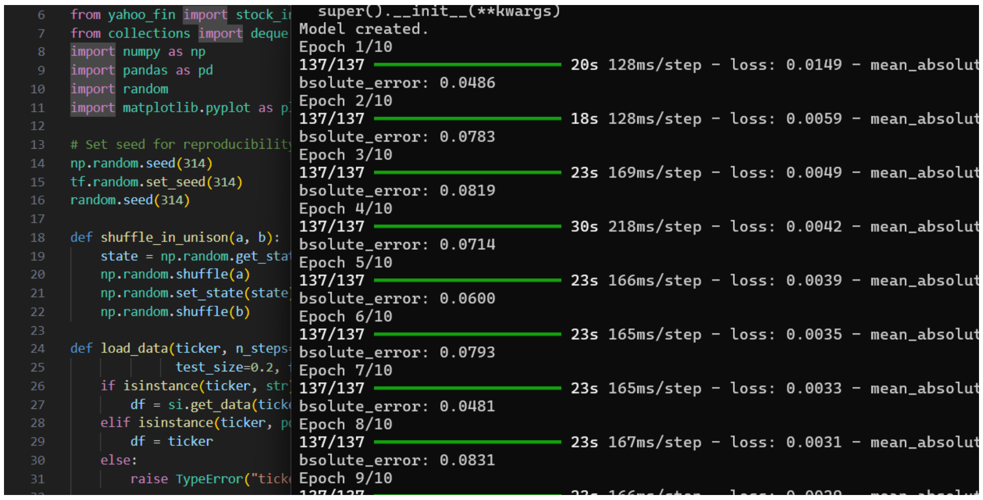

GitHub - jason887/Using-Deep-
https://github.com/jason887/Using-Deep-Learning-Neural-Networks-and-Candlestick-Chart-Representation-to-Predict-Stock-Market

dataset_traditional.py	rc01	6 years ago
generatebigdata.py	new parameter	6 years ago
get_data.py	rc01	6 years ago
indo10.csv	add ticker data	6 years ago
init.sh	rc01	6 years ago
khaidem.csv	add khaidem list	6 years ago
knn.py	no need save knn, soooo big data fuck it	6 years ago
model_evaluate.py	fix auc calculation	6 years ago
myDeepCNN.py	rc01	6 years ago
myVgg16.py	rc01	6 years ago
myVgg19.py	rc01	6 years ago
naivebayes.py	rc01	6 years ago
predictme.py	single predict, pre-webserver	6 years ago
preproccess_binclass.py	rc01	6 years ago
preprocess.py	add more data generated by split volume and ohlc	6 years ago
randomforest.py	rc01	6 years ago
removezeroimage.py	using PIL to check img mode	6 years ago
resnet50.py	rc01	6 years ago

No packages published

Languages

● Python 62.5% ● TeX 37.4%
● Shell 0.1%



```
from yahoo_fin import stock_i
from collections import deque
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as p

# Set seed for reproducibility
np.random.seed(314)
tf.random.set_seed(314)
random.seed(314)

def shuffle_in_unison(a, b):
    state = np.random.get_stat
    np.random.shuffle(a)
    np.random.set_state(state
    np.random.shuffle(b)

def load_data(ticker, n_steps
                test_size=0.2,
    if isinstance(ticker, str
        df = si.get_data(tick
    elif isinstance(ticker, p
        df = ticker
    else:
        raise TypeError("tick
```

```
  super().__init__(**kwargs)
Model created.
Epoch 1/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 20s 128ms/step - loss: 0.0149 - mean_absolut
bsolute_error: 0.0486
Epoch 2/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 18s 128ms/step - loss: 0.0059 - mean_absolut
bsolute_error: 0.0783
Epoch 3/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 169ms/step - loss: 0.0049 - mean_absolut
bsolute_error: 0.0819
Epoch 4/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 30s 218ms/step - loss: 0.0042 - mean_absolut
bsolute_error: 0.0714
Epoch 5/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 166ms/step - loss: 0.0039 - mean_absolut
bsolute_error: 0.0600
Epoch 6/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 165ms/step - loss: 0.0035 - mean_absolut
bsolute_error: 0.0793
Epoch 7/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 165ms/step - loss: 0.0033 - mean_absolut
bsolute_error: 0.0481
Epoch 8/10
137/137 ━━━━━━━━━━━━━━━━━━━━ 23s 167ms/step - loss: 0.0031 - mean_absolut
bsolute_error: 0.0831
Epoch 9/10
```

6. Conclusion

The environment setup for v0.1, P1, was successful and p2 partially successful not showing all outputs but codes running, with all models running as expected. The v0.1 code base was found to have several areas for improvement, and the additional exploration of P1 and P2 provided further insights into potential enhancements. The successful setup and testing of all models provide a solid foundation for future development.

Attachments

- Requirements.txt: List of dependencies used in the project.

Prepared by:

[Alvi Hossain Safri Himalya]

Date: 16 August 2024