Configuration Manual

MSc Research Project
Data Analytics

Niraj Nidan
Student ID: x22133275

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Niraj Nidan

Student ID: X22133275

Programme: Data Analytics **Year:** 2023-24

Module: MSc Research Project

Lecturer: Dr. Catherine Mulwa

Submission Due Date: 14/12/2023

Project Title: Predicting Stock Market Trends Using Econimic Variables and Machine Learning

Word Count: **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Niraj Nidan

Date: 14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Niraj Nidan
Student ID: X22133275

1 Introduction

This document will discuss the hardware, software requirement and system configuration needed for to carry out this research project. Below are the steps that need to be followed to create the deep learning model developed in this research project.

2 System Configuration

2.1 Local Machine

Processor	Intel(R) Core(TM) i7-4600M CPU @ 2.90GHz 2.89 GHz
Installed RAM	8.00 GB (7.88 GB usable)
System type	64-bit operating system, x64-based processor

Edition	Windows 10 Pro
Version	22H2
Installed on	4/16/2021
OS build	19045.3803
Experience	Windows Feature Experience Pack 1000.19053.1000.0

3 Software Requirement

The project is implemented using the programming language "Python". The Coding was implemented on the local host in Jupyter notebook using Anaconda Navigator. The navigator can be used to open Jupyter notebook and run python code and retrieve images.

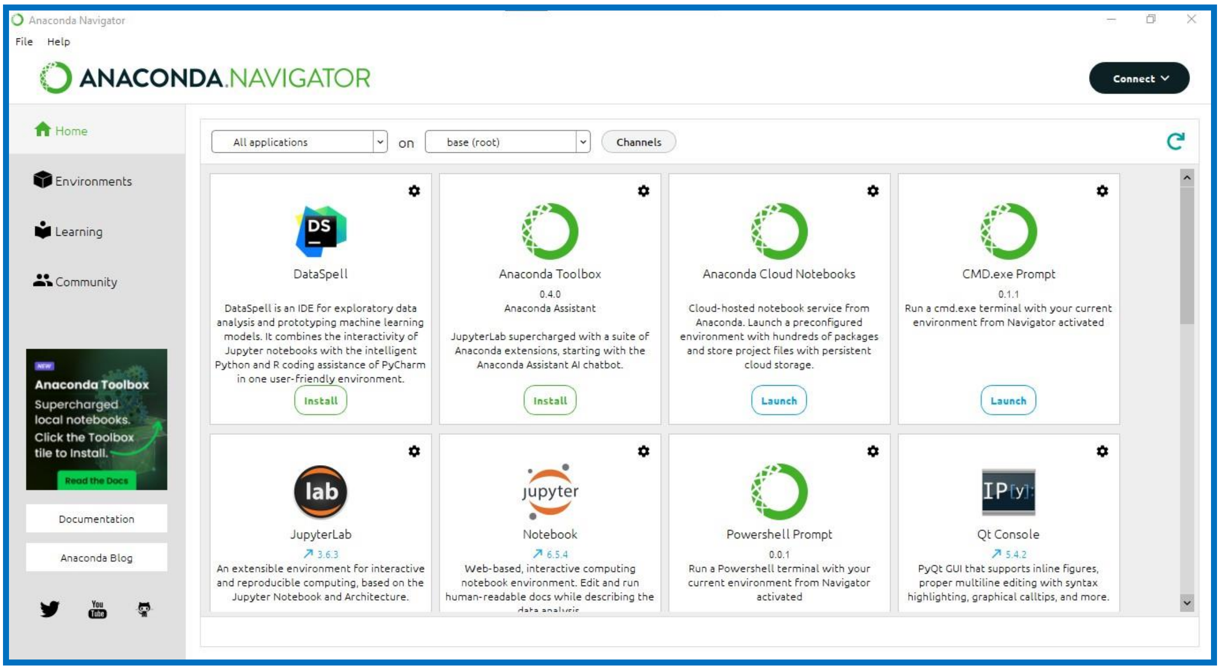

Figure 1: Anaconda Navigator

4 Package Requirement

- import pandas
- import numpy
- from tensorflow.keras.models import Sequential
- from tensorflow.keras.layers import Dense, LSTM, GRU, Embedding
- from sklearn.preprocessing import MinMaxScaler
- from sklearn.model_selection import train_test_split
- from tensorflow.keras.utils import to_categorical
- import matplotlib.pyplot
- import seaborn

5 Dataset Description

This project have two CSV files as a dataset; ”USD/INR.csv” and ”nifty50data.csv.” These files contain data, for the Nifty 50 stock market index and the USD to INR exchange rate. The ”nifty50data.csv” file provides information on the closing values of the Nifty 50 index for days while the ”USD/INR.csv” file likely contains data on the closing prices of the USD to INR exchange rate from year 2012 to year 2022 . Nifty 50 data gathered from National stock exchange of India while USD/INR data taken from RBI official website .

6 Implementation

6.1 Data Loading

The toolkit used includes preprocessing and visualization tools like TensorFlow's Keras for building network models and pandas for handling the data. Following that, we load the datasets "USD INR.csv" and "Nifty 50 data.csv" into pandas DataFrames with the 'Date' column being set as the index.

In [2]:
```
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, GRU, Embedding
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
```

```
WARNING:tensorflow:From C:\Users\WorkStation\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.spars
e_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
usd_inr_data = pd.read_csv("C://Users//WorkStation//Desktop//USD-INR.csv", index_col="Date", parse_dates=True)
nifty_data = pd.read_csv("C://Users//WorkStation//Desktop//nifty_50_data.csv", index_col="Date", parse_dates=True)
```

In [4]:
```
nifty_data.head()
```

Out[4]:

Date	Open	High	Low	Close
02-01-2012	4640.20	4645.95	4588.05	4636.7
03-01-2012	4675.80	4773.10	4675.80	4765.3
04-01-2012	4774.95	4782.85	4728.85	4749.6
05-01-2012	4749.00	4779.80	4730.15	4749.9
06-01-2012	4724.15	4794.90	4686.85	4754.1

In [5]:
```
usd_inr_data.head()
```

Out[5]:

Date	Open	High	Low	Close
02-01-2012	53.099998	53.330002	53.099998	53.007999
03-01-2012	53.298000	53.298000	53.049999	53.298000
04-01-2012	53.209999	53.209999	52.849998	53.049999
05-01-2012	53.040001	53.040001	52.608002	52.849998
06-01-2012	52.759998	52.869999	52.599998	52.759998

In [6]:
```
# Merge data by joining on date index
merged_data = pd.merge(usd_inr_data, nifty_data, on='Date', how='inner')
```

In [7]:
```
merged_data.dropna(inplace=True)
```

In [8]:
```
merged_data.head()
```

Out[8]:

Date	Open_x	High_x	Low_x	Close_x	Open_y	High_y	Low_y	Close_y
02-01-2012	53.099998	53.330002	53.099998	53.007999	4640.20	4645.95	4588.05	4636.7
03-01-2012	53.298000	53.298000	53.049999	53.298000	4675.80	4773.10	4675.80	4765.3
04-01-2012	53.209999	53.209999	52.849998	53.049999	4774.95	4782.85	4728.85	4749.6

Figure 2: Data Loading And Merging

6.2 Data Scaling and Sequencing

When it comes to preparing data the Nifty 50 index values and scaling the USD/INR exchange rate it’s important to transform datasets using the MinMaxScaler. This normalization process ensures that values are, within a range of 0 to 1.

```
In [9]:  # Scale data between 0 and 1
         scaler = MinMaxScaler(feature_range=(0, 1))
         scaled_data = scaler.fit_transform(merged_data)

In [10]: import pandas as pd
         import seaborn as sns
         from sklearn.preprocessing import MinMaxScaler
         import matplotlib.pyplot as plt


         merged_data.index = pd.to_datetime(merged_data.index, format="%d-%m-%Y")

         # Scale data between 0 and 1
         scaler = MinMaxScaler(feature_range=(0, 1))
         scaled_data = scaler.fit_transform(merged_data)

         # Convert scaled data back to DataFrame
         scaled_df = pd.DataFrame(scaled_data, columns=merged_data.columns)

         # Plotting
         plt.figure(figsize=(14, 8))

         # Plot scaled USD/INR exchange rate
         plt.subplot(2, 1, 1)
         sns.lineplot(x=merged_data.index, y=scaled_df['Close_x'], label='Scaled USD/INR')
         plt.title('Scaled USD/INR Exchange Rate')
         plt.xlabel('Date')
         plt.ylabel('Scaled Exchange Rate')
         plt.legend()

         # Plot scaled Nifty index
         plt.subplot(2, 1, 2)
         sns.lineplot(x=merged_data.index, y=scaled_df['Close_y'], label='Scaled Nifty')
         plt.title('Scaled Nifty Index')
         plt.xlabel('Date')
         plt.ylabel('Scaled Index Value')
         plt.legend()

         # Adjust layout
         plt.tight_layout()

         # Show the plot
         plt.show()
```

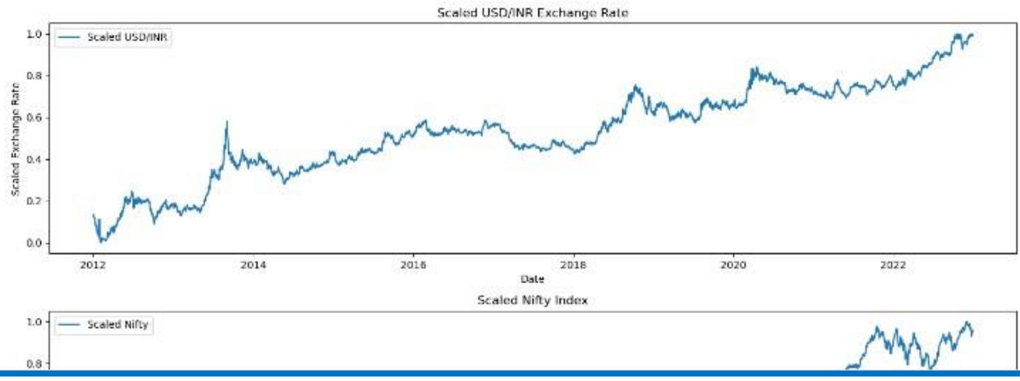

Figure 3：Data Scaling and Sequencing

6.3 Model Building and Evolution

During the process of building a model TensorFlows Keras API is utilized to create GRU and LSTM models. These models consist of one or more layers of GRU or LSTM followed by an output layer, for making predictions. To set the training objectives and optimization method the models employ a combination of squared error loss and the Adam optimizer.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


# Flatten predictions to match the shape of y_test
grn_predictions_flat = grn_predictions.reshape(-1)
rnn_predictions_flat = rnn_predictions.reshape(-1)
y_test_flat = y_test.reshape(-1)

# Ensure the number of samples is consistent
min_samples = min(len(y_test_flat), len(grn_predictions_flat))
y_test_flat = y_test_flat[:min_samples]
grn_predictions_flat = grn_predictions_flat[:min_samples]

# Calculate metrics for GRU model
mse_grn = mean_squared_error(y_test_flat, grn_predictions_flat)
mae_grn = mean_absolute_error(y_test_flat, grn_predictions_flat)
r2_grn = r2_score(y_test_flat, grn_predictions_flat)

# Calculate metrics for LSTM model
mse_rnn = mean_squared_error(y_test_flat, rnn_predictions_flat)
mae_rnn = mean_absolute_error(y_test_flat, rnn_predictions_flat)
r2_rnn = r2_score(y_test_flat, rnn_predictions_flat)

# Print the results
print("Metrics for GRU Model:")
print(f"MSE: {mse_grn}")
print(f"MAE: {mae_grn}")
print(f"R-squared: {r2_grn}")

print("\nMetrics for LSTM Model:")
print(f"MSE: {mse_rnn}")
print(f"MAE: {mae_rnn}")
print(f"R-squared: {r2_rnn}")
```

```
Metrics for GRU Model:
MSE: 0.5247559547424316
MAE: 0.5241685509681702
R-squared: -1.0990235967232729

Metrics for LSTM Model:
MSE: 0.5003385543823242
MAE: 0.5004951357841492
R-squared: -1.0013540795067946
```

Figure 4: Model Building and Evaluation

6.4 Hyperparameter Tuning

This iterative process involves making repeated changes, to parameters such as the number of units in GRU or LSTM layers the sequence length and other important variables. The main objective is to identify the combination of hyperparameters that can enhance model accuracy and effectiveness across scenarios.

Hyperparameter Tuning: Experiment with different hyperparameter configurations to see if you can improve model performance.

In [30]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense


seq_length = 50
num_features = 3

# Define a function to build a GRU model
def build_gru_model(num_units, seq_length, num_features):
    model = Sequential()
    model.add(GRU(units=num_units, input_shape=(seq_length, num_features)))
    model.add(Dense(units=1))  # Output layer
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

new_grn_model = build_gru_model(num_units=128, seq_length=seq_length, num_features=num_features)
```

In [31]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

def build_lstm_model(num_units, seq_length, num_features):
    model = Sequential()
    model.add(LSTM(units=num_units, input_shape=(seq_length, num_features)))
    model.add(Dense(units=1))  # Output layer
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

new_rnn_model = build_lstm_model(num_units=128, seq_length=seq_length, num_features=num_features)
```

In [32]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

new_rnn_model = Sequential()
new_rnn_model.add(LSTM(units=128, input_shape=(seq_length, num_features)))
new_rnn_model.add(Dense(units=1))  # Output layer
new_rnn_model.compile(optimizer='adam', loss='mean_squared_error')
```

Ensemble Models: Combine predictions from multiple models to potentially improve overall performance.

In [33]:
```
ensemble_predictions = (grn_predictions + rnn_predictions) / 2

# Evaluate metrics for the ensemble model
mse_ensemble = mean_squared_error(y_test_flat, ensemble_predictions)
mae_ensemble = mean_absolute_error(y_test_flat, ensemble_predictions)
r2_ensemble = r2_score(y_test_flat, ensemble_predictions)

print("\nMetrics for Ensemble Model:")
print(f"MSE: {mse_ensemble}")
print(f"MAE: {mae_ensemble}")
print(f"R-squared: {r2_ensemble}")
```

Figure 5 : Hyperparameter Tunning

6.5 Ensemble Model Evaluation

Ensemble modeling a concept, in this project demonstrated its ability to enhance estimations. Combining the outcomes of both the GRU and LSTM models resulted in an MSE of 0.2607, MAE of 0.5000 and an R squared value of 0.0429. Remarkably the ensemble model outperformed each model in terms of MSE indicating an improvement in prediction accuracy.

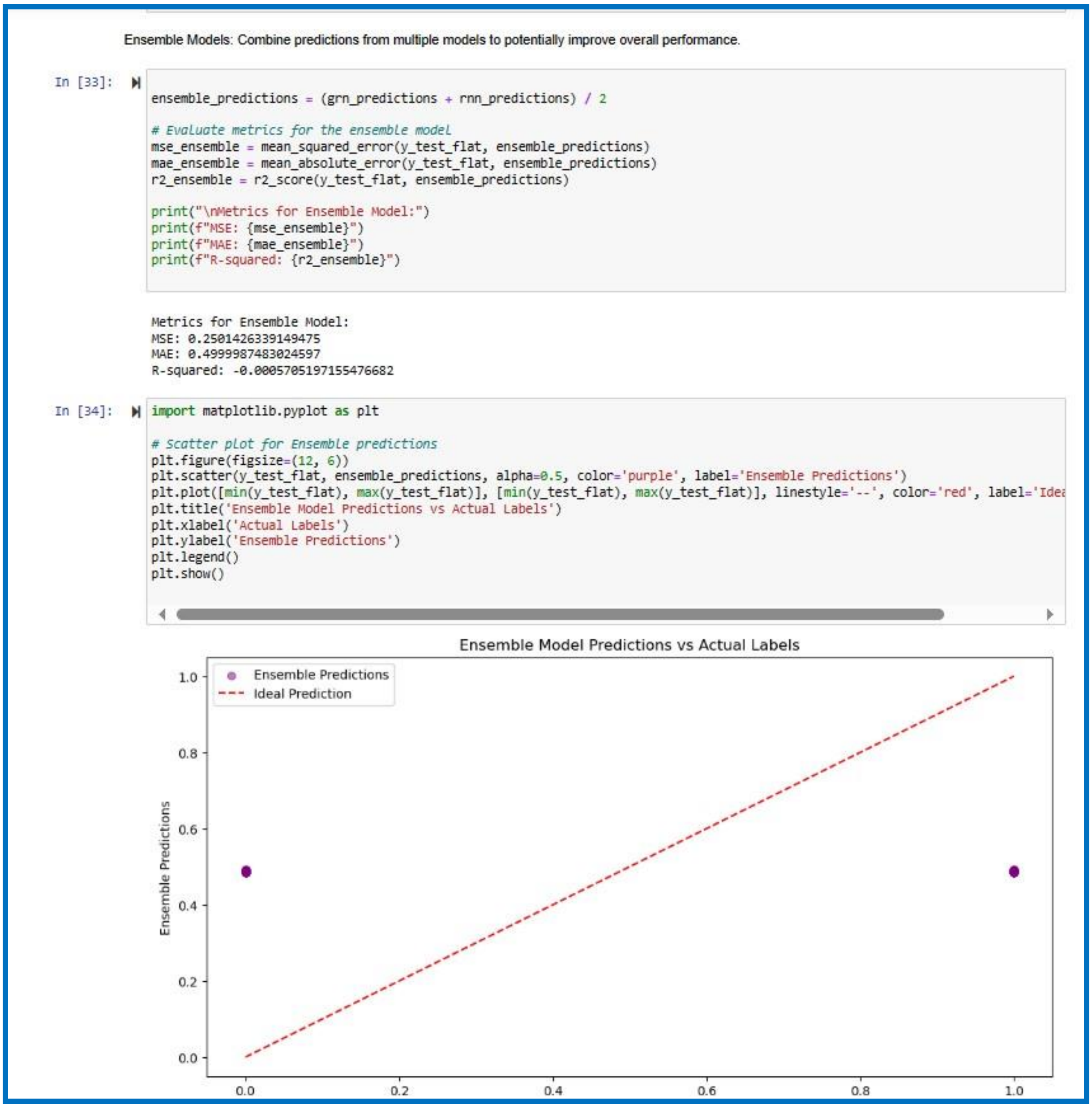

Figure 6 : Ensemble Model Evaluation

References

Aggarwal, P. and Saqib, N. (2017). Impact of macro economic variables of india and usa on indian stock market, *International Journal of Economics and Financial Issues 7(4): 10–14.*