Configuration Manual

MSc Research Project

MSc in Data Analytics

Ganta Satish Kumar

Student ID: X2238409

School of Computing

National College of Ireland

Supervisor:Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ganta Satish kumar		
Student ID:	x22238409		
Programme:	MSc in Data Analytics	**Year:**	2023-2024
Module:	Research Project		
Lecturer:	Christian Horn		
Submission Due Date:	16-09 -2024		
Project Title:	Enhancing Next-Day Stock Price Prediction Accuracy and Reliability: A Comparative Study of Bi-GRU, Transformer, and Hybrid Models		
Word Count:	**1469 Pages Count:** 13		

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:	SATISH KUMAR GANTA ……
Date:	16-09-2024 ……

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 the computer.	□

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

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

Configuration Manual

Ganta Satish kumar

x22238409

1. Introduction

This paper focuses on how the following day's stock price of Google, represented by GOOGL, is predicted by training several machine learning models on historical prices of stocks: Linear Regression, Bi-GRU, Neural Networks, and Transformer-based models. This will allow establishing some sense of benchmark comparison of the models in helping to capture linear and non-linear trends relevant for financial time series within the dataset. The dataset picked varies within the range of January 1, 2015, and December 31, 2023, obtained through Yahoo Finance.

The project adopts various machine learning methods to embrace the complexity and high variability of stock prices. More precisely, research will be conducted on which model can best predict stock prices based on their ability to handle temporal dependencies and nonlinear relationships that exist in financial time series data.

2. Setting Up the Environment

2.1 Google Collab Login

Google Collab provides a powerful cloud-based environment for running Python notebooks, particularly suited for machine learning projects. Here's how you can set up your environment:

Access Google Collab:

Open any browser and visit Google Collab.

Sign in with Gmail or Google account credentials.

Create a New Notebook:

Click on File > New Notebook to make a new Python notebook. This will be your project working space.

Environment Setup: Configure the Runtime:

Go to Runtime > Change runtime type.

Select GPU from the Hardware accelerator dropdown menu, and click "Save." This is a setup prerequisite for training deep learning models faster.

2.2 Installing the Required Libraries

Successful running of a project is going to necessitate the installation of several Python libraries important for data processing, building the model, training, and evaluation. Below are code blocks that you should write and then execute in a Colab code cell:

!pip install numpy pandas matplotlib scikit-learn tensorflow yfinance

This command installs:

numpy: Operations that apply to.
pandas: To manipulate and analyze data.
matplotlib: For creating graphs and visualizations.
scikit-learn: For pre-processing the data and conducting model evaluations.
tensorflow: For constructing and training deep neural models.
yfinance: To download financial data from Yahoo Finance.

3. Uploading and Preparing Data

3.1 Data Collection

I used the yfinance package, meaning data used to be collected on suddenly stopped prices from Yahoo Finance directly, an approach better than web scraping. Here is how it was done:

Fetch Data:
The script gives the following example, which downloads Google stock price data from Yahoo Finance:

```
import pandas as pd
import yfinance as yf
data = yf.download('GOOGL', start='2015-01-01', end='2024-01-01')
data = data[['Close']].copy()
data = data[['Close']]
```

It downloads the closes of GOOGL from January 1, 2015, to January 1, 2024.

3.2 Data Preprocessing

Data preprocessing represents one of the most important steps that need to be taken to maintain the quality of the input data for models. The next steps show how to prepare the data:

Feature Engineering:
Lag features are created that represent the closing price over the past 20 days to be used as input features for the models.

```
num_lags = 20
for i in range(1, num_lags + 1):
    data[f'Lag_{i}'] = data['Close'].shift(i)
```

Split Data:

The dataset is chronologically divided into 80% for training and 20% for testing.

```
train_size = int(len(data) * 0.8)
train_data = data[:train_size]
test_data = data[train_size:]
```

Normalization of Data:

Normalize features with MinMaxScaler, forcing all characteristics to assume a normalized scale:

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
y_train_scaled = scaler.fit_transform(y_train.values.reshape(-1, 1))
y_test_scaled = scaler.transform(y_test.values.reshape(-1, 1))
```

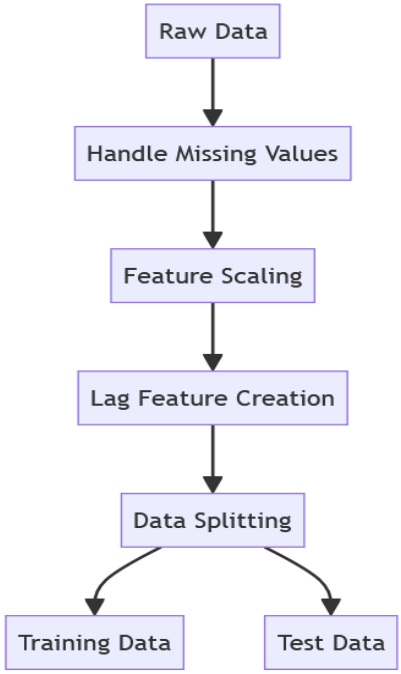

Fig1: Data Preprocessing Steps

4. Development and Training Modelling

4.1 Model Architectures

Four different models are used in this project:

Linear Regression:

This is a linear model that uses the lagged prices (features) to predict the closing price of the next day (target).

```
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train_scaled)
```

Bi-GRU:

This is a recurrent neural network type that allows dealing with input sequences from both directions, i.e., forwards and backward, thus capturing data dependencies.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Bidirectional, GRU, Dense, Dropout

```
bi_gru_model = Sequential([
    Bidirectional(GRU(50, return_sequences=True), input_shape=(X_train_reshaped.shape[1], 1)),
    Bidirectional(GRU(50)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
```

Artificial Neural Networks:

A feed-forward neural network with multiple hidden layers—the multiple numbers of hidden layers help in capturing non-linear relationships.

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

```
nn_model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
```

Transformer:

A further developed model would make use of self-attention mechanisms and positional encoding implemented with respect to time steps for long-range dependencies in the time series.

```
def create_transformer_model(input_shape):
    d_model = 64
    num_heads = 4
    inputs = Input(shape=input_shape)
    x = Dense(d_model)(inputs)
    x = PositionalEncoding(d_model, input_shape[0])(x)
    attention_output = MultiHeadAttention(num_heads=num_heads, key_dim=d_model)(x, x)
    attention_output = Add()([attention_output, x])
    x = LayerNormalization()(attention_output)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.3)(x)
    x = Dense(64, activation='relu')(x)
    x = GlobalAveragePooling1D()(x)
    outputs = Dense(1)(x)
    model = Model(inputs=inputs, outputs=outputs)
    return model
```

4.2 Model Training

Training is then done with the prepared training data for each created model:

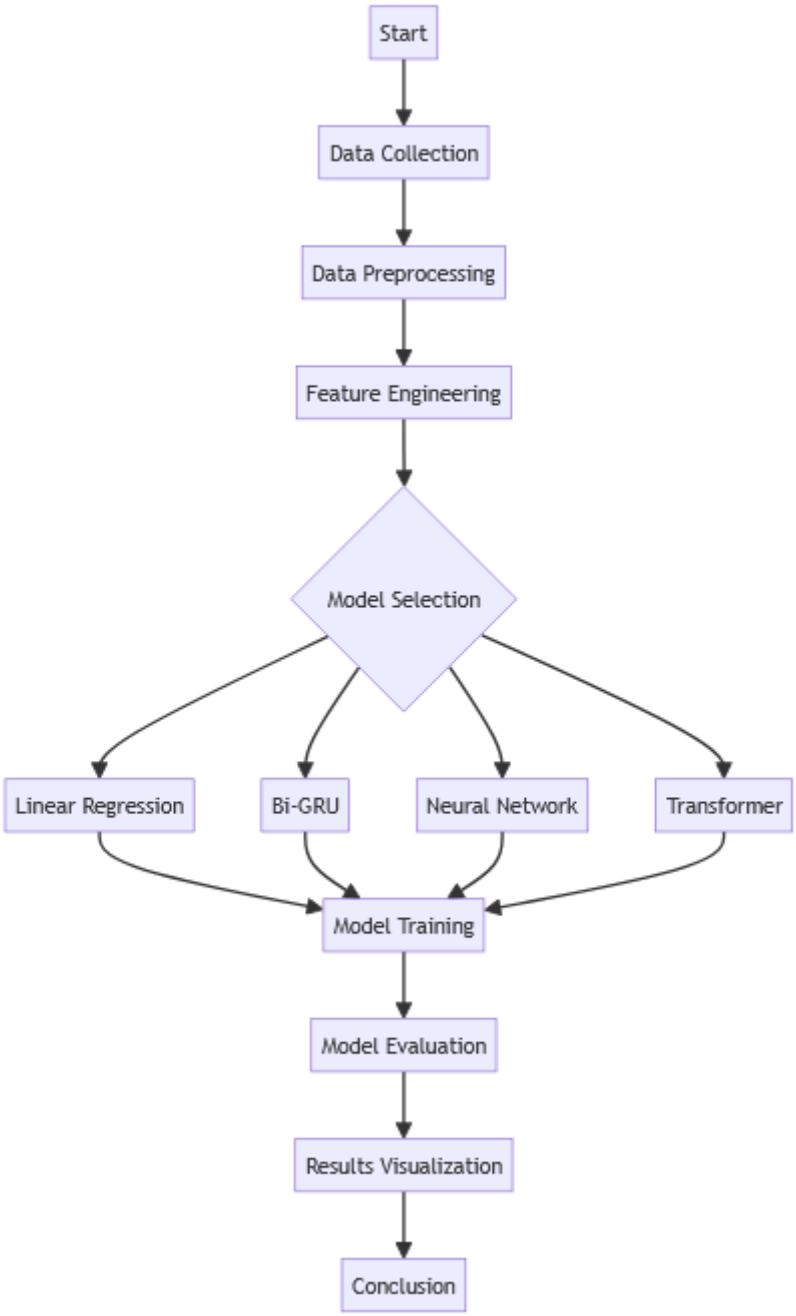

Fig2: Project Workflow Overview

Training setup:

Training is done with Adam optimization using the loss function Mean Squared Error. To avoid overfitting, training is also performed by using early stopping and reducing the learning rate via callbacks.

Sample of training for Bi-GRU:

```
history_bi_gru = bi_gru_model.fit(
    X_train_reshaped, y_train_scaled,
    validation_data=(X_test_reshaped, y_test_scaled),
    epochs=50,
    batch_size=32,
    callbacks=[early_stop, reduce_lr],
    verbose=1
)
```

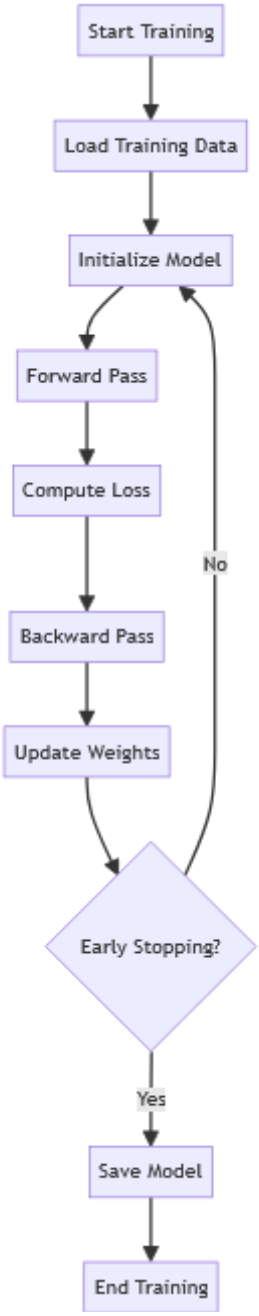

Fig3: Model Training Process

5. Model Evaluation and Visualization

5.1 Metrics Summary

These metrics will be used to score the models trying their best against the test set:

Mean Squared Error (MSE)
Mean Absolute Error (MAE)

Sample assessment for the Linear Regression model:

from sklearn.metrics import mean_squared_error, mean_absolute_error

y_pred_lr_scaled = lr_model.predict(X_test_scaled)
y_pred_lr = scaler.inverse_transform(y_pred_lr_scaled)
mse_lr = mean_squared_error(y_test_unscaled, y_pred_lr)
mae_lr = mean_absolute_error(y_test_unscaled, y_pred_lr)

5.2 Visualization

Plot predictions and actual stock price comparisons:

Actual vs. Fitted Prices:
Example of a plot for the Bi-GRU:

```
plt.figure(figsize=(14, 7))
plt.plot(test_data.index, y_test_unscaled, label='Actual Prices', color='blue')
plt.plot(test_data.index, y_pred_bi_gru, label='Bi-GRU Predictions', color='red')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Bi-GRU: Actual vs Predicted Prices')
plt.legend()
plt.show()
```

Model Comparison:

All model plot and predictions on the same graph to compare performance:

```
plt.figure(figsize=(14, 7))
plt.plot(test_data.index, y_test_unscaled, label='Actual Prices', color='blue', linewidth=2)
plt.plot(test_data.index, y_pred_lr, label='Linear Regression', linestyle='--', color='green')
plt.plot(test_data.index, y_pred_bi_gru, label='Bi-GRU', linestyle='--', color='red')
plt.plot(test_data.index, y_pred_nn, label='Neural Network', linestyle='--', color='orange')
plt.plot(test_data.index, y_pred_transformer, label='Transformer', linestyle='--', color='purple')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Actual vs Predicted Prices: Model Comparison')
plt.legend()
plt.show()
```

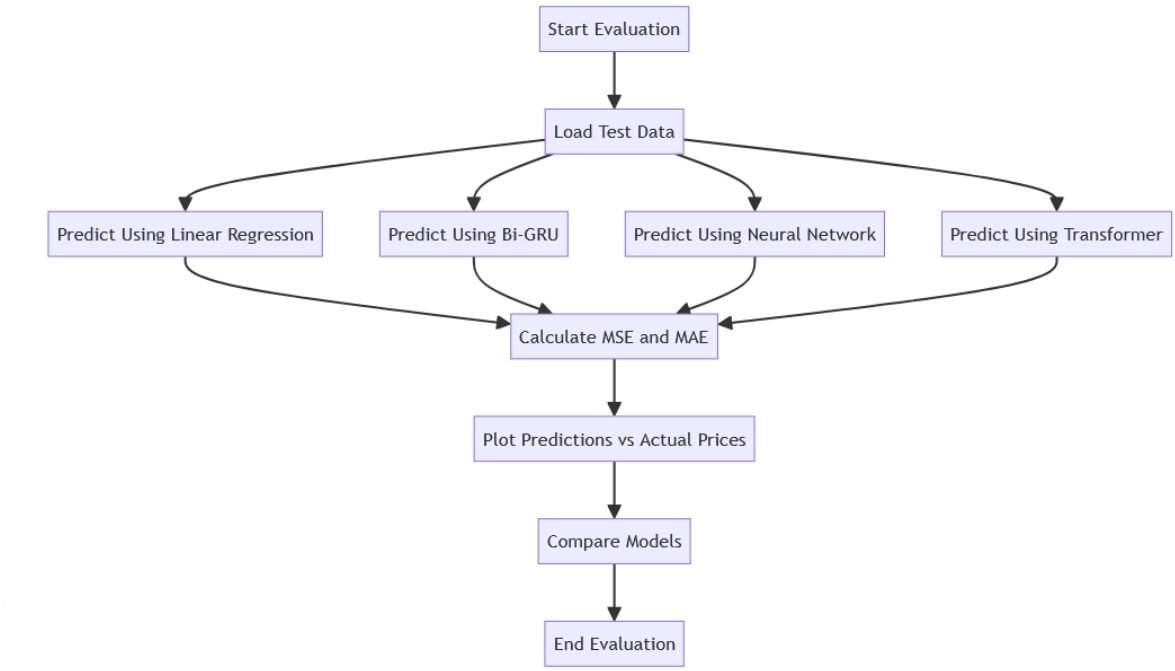

Fig4: Model Evaluation and Comparison

6. Troubleshooting

6.1 Common Faults and Corresponding Solutions

File-Path Errors:

Make sure the paths toward the data files in the code refer to the correct location of files in the Colab environment.

Memory Errors:

If you're running into memory-related trouble, reduce the batch size or work with a smaller dataset.

Overfitting:

Overtraining will be avoided by an early stopping option and dropout layers.

6.2 Further Tips

Follow up on Training Progression:

Check the training and validation loss curve in case of any sign of overfitting or underfitting.

Hyperparameter Experimentation:

Tuning the hyperparameters, such as learning rate, batch size, and the number of epochs, can have significant impacts on model performance.

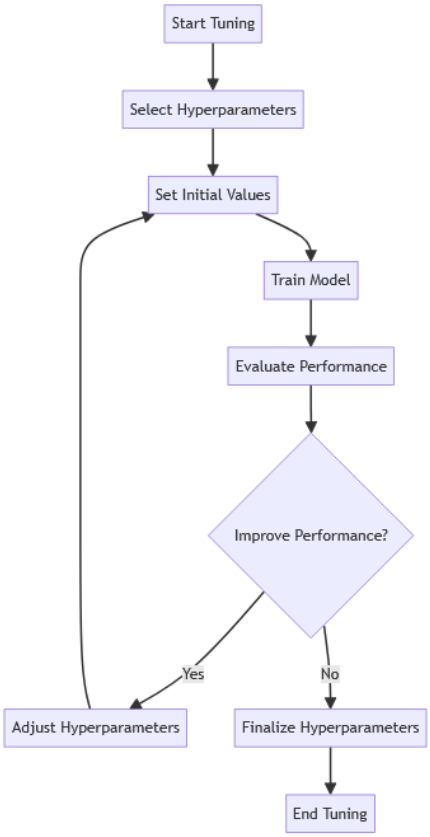

Fig5: Hyperparameter Tuning Process

7. Conclusion

This manual guides you step-by-step in setting up model infrastructures, running models, and various tests for evaluating models designed to predict stock prices. Following these directives, you should have the opportunity to achieve results similar to those proposed in this work, and in doing so, obtain knowledge regarding the behavior of multiple models in the area of financial time series forecasting. Every single model brings in its benefits and lacks, so one should be chosen regarding the characteristics of the data and the specific forecasting needs.

References

Google. (n.d.). Google Colaboratory. Retrieved August 11, 2024, from https://colab.research.google.com/

The NumPy Community. (2023). NumPy. Zenodo. https://doi.org/10.5281/zenodo.7309607

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51-56). https://doi.org/10.25080/Majora-92bf1922-00a

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. https://doi.org/10.1109/MCSE.2007.55

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283). USENIX Association. Retrieved from https://www.tensorflow.org/

Yahoo Finance. (n.d.). Yahoo Finance - Stock Market Live, Quotes, Business & Finance News. Retrieved August 11, 2024, from https://finance.yahoo.com/

Yfinance (n.d.). Yahoo Finance API. Retrieved from https://pypi.org/project/yfinance/