# Configuration Manual

MSc Data Analytics

Research Project

Dineshkiumar Lingapandiyan

X22225498

School of Computing

National College of Ireland

Supervisor: Mr. Hicham Rifai

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

**Student Name:** Dineshkumar Lingapandiyan

**Student ID:** X22225498

**Programme:** MSc Data Analytics **Year:**2024

**Module:** Research Project

**Supervisor**: Hicham Rifai

**Submission Due Date:12/12/24**

**Project Title:** Enhancing Time Series Forecasting Accuracy and Resilience in High-Frequency Data Environments through Hybrid Deep Learning Smoothing Models

**Word Count: 10** **page Count:**748

**Signature:** Dineshkumar Lingapandiyan

**Date:12/12/24**

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

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.

| | |
|---|---|
| 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

Dineshkumar Lingapandiyan

X22225498

## Introduction:

High-frequency data forecasting presents challenges due to noise and complexity. This research focuses on hybrid deep learning smoothing models to enhance prediction accuracy and robustness in dynamic environments.

## 1. System Requirements:

Having the right hardware and software resources is essential to ensuring effective model processing and reducing the amount of time needed.

## Hardware Requirements:

An Acer Aspire 7 is used for the implementation, and it has the following configuration.

1. Processor: Intel(R) Core (TM) i5-12300H CPU @ 3.30GHz
2. RAM: 32.00 GB (19.84 GB usable)
3. Hard Disk: 256GB SSD, 1 TB HDD
4. OS Windows 11 Pro 64 – bit

**Software Requirements**: The software, libraries, and tools listed below were installed and set up on the system prior to the start of the model construction phase.

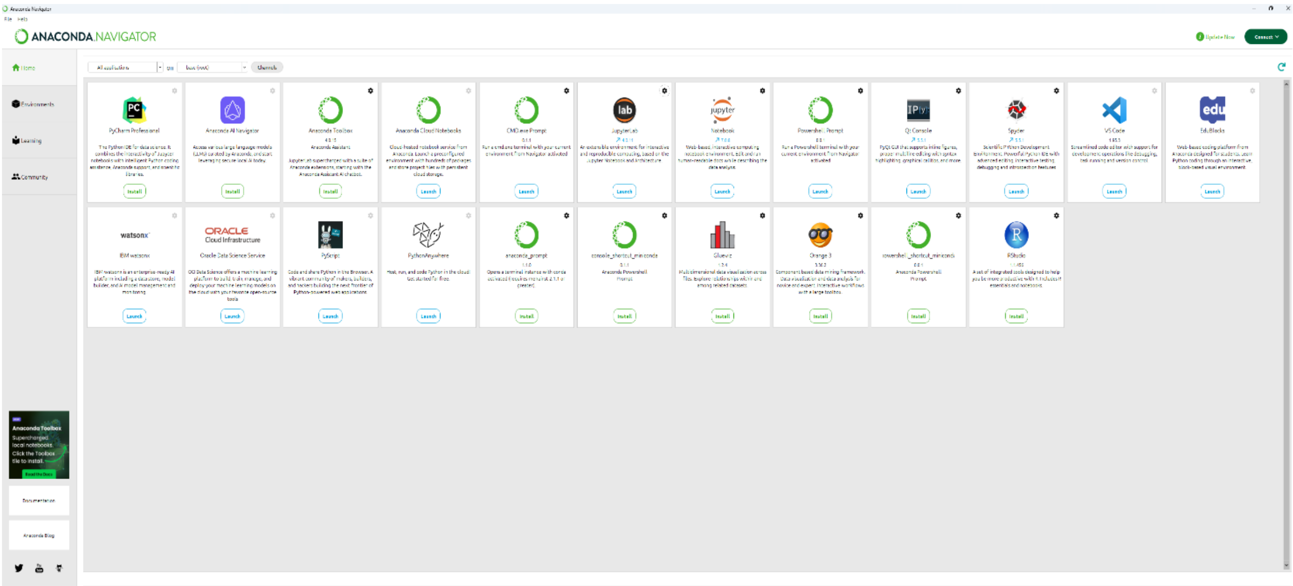
| Software/Tools | Version | Information |
|---|---|---|
| Python | | In this project, Python is utilized to create the model. |
| Anaconda | | The data science community favors Anaconda because it provides features for managing libraries, deploying models, and computational work in a Windows-friendly interface. |
| Pandas | | It works especially well with tabular data, like that found in databases or spreadsheets. |

| | | |
|---|---|---|
| NumPy | | Developed in 2023, NumPy is an open-source program used to solve complex mathematical problems in data. |
| Sci-kit Learn | | Data preprocessing, regression, and classification are among the activities that this package is used for. Python Machine Learning with Scikit-Learn – scikit-learn 0.24.2 docs, 2023. |
| TensorFlow | | This library is Ideal for building scalable hybrid deep learning models with extensive tools for deploying on various platforms. (2.17.1) |

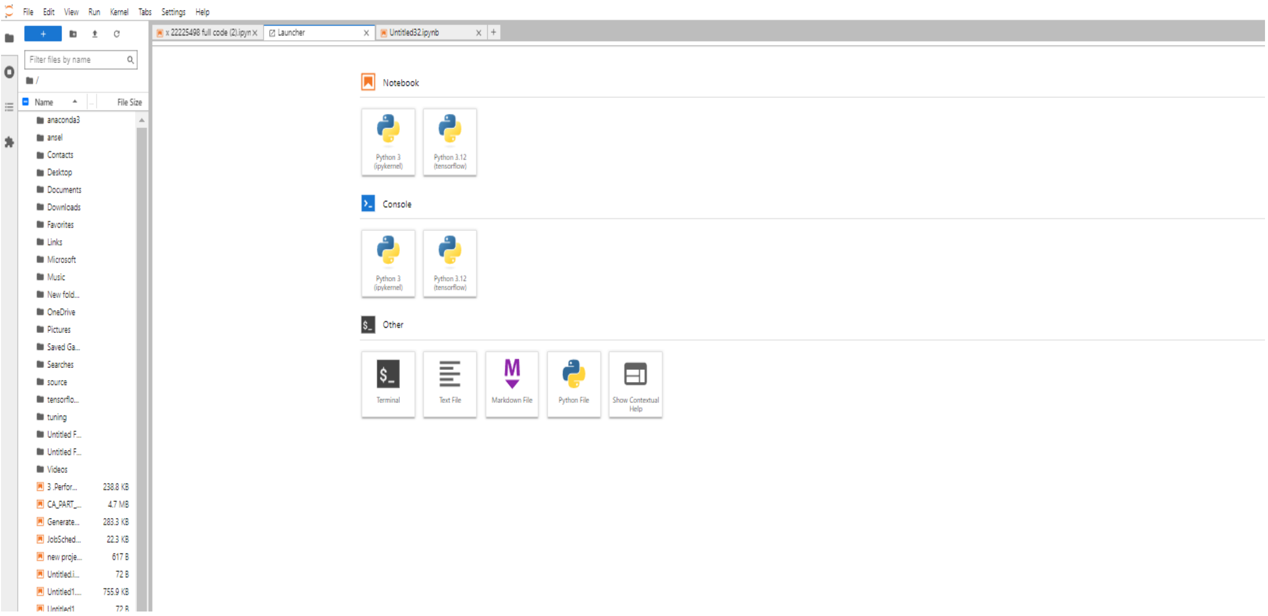
## 2. Implementation:

This part contains a comprehensive explanation on how to execute the project on any Windows system.

1. Install the Anaconda software on your Windows computer after downloading it. ([https://www.anaconda.com/products/individual](https://www.anaconda.com/products/individual))




2.Launch Anaconda's Jupyter Notebook Lab.




3. Click on the new notebook (python 3) after launching the Jupyter notebook Lab.

4. Import each of the necessary libraries into the notebook.

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
import networkx as nx
import time
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.cluster import DBSCAN
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Conv1D, Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.regularizers import l2
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from pykalman import KalmanFilter
import shap
import tensorflow as tf
from keras_tuner import Hyperband
import os
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, LSTM, Dense, Flatten, Dropout, Concatenate
from tensorflow.keras.optimizers import Adam
from pykalman import KalmanFilter
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, LSTM, Dense, Flatten, Dropout, Concatenate
from tensorflow.keras.optimizers import Adam
from pykalman import KalmanFilter
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_absolute_error, mean_squared_error
import math
import warnings
```

```python
#  Load and preprocess the dataset
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import train_test_split
from pykalman import KalmanFilter
import math
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, Dense, Dropout, Concatenate, Conv1D, Flatten
from tensorflow.keras.optimizers import Adam
import plotly.graph_objects as go
import warnings
```

5. Bring in the supplied dataset.

```python
df = pd.read_csv('Flight_delay.csv')
print( df.head())
```

6. The next action will be To complete the pre-processing step, the following code will be used.

```
   DayOfWeek         Date  DepTime  ArrTime  CRSArrTime UniqueCarrier  \
0          4  03-01-2019     1829     1959        1925            WN
1          4  03-01-2019     1937     2037        1940            WN
2          4  03-01-2019     1644     1845        1725            WN
3          4  03-01-2019     1452     1640        1625            WN
4          4  03-01-2019     1323     1526        1510            WN

                  Airline  FlightNum TailNum  ActualElapsedTime  ...  TaxiIn  \
0  Southwest Airlines Co.       3920  N464WN                 90  ...       3
1  Southwest Airlines Co.        509  N763SW                240  ...       3
2  Southwest Airlines Co.       1333  N334SW                121  ...       6
3  Southwest Airlines Co.        675  N286WN                228  ...       7
4  Southwest Airlines Co.          4  N674AA                123  ...       4

   TaxiOut  Cancelled  CancellationCode Diverted CarrierDelay WeatherDelay  \
0       10          0                 N        0            2            0
1        7          0                 N        0           10            0
2        8          0                 N        0            8            0
3        8          0                 N        0            3            0
4        9          0                 N        0            0            0

  NASDelay  SecurityDelay  LateAircraftDelay
0        0              0                 32
1        0              0                 47
2        0              0                 72
3        0              0                 12
4        0              0                 16

[5 rows x 29 columns]
```

```
[4]: df.shape
```

```
[4]: (484551, 29)
```

```
[5]: df.isnull().sum()
```

```
[5]: DayOfWeek               0
     Date                    0
     DepTime                 0
     ArrTime                 0
     CRSArrTime              0
     UniqueCarrier           0
     Airline                 0
     FlightNum               0
     TailNum                 0
     ActualElapsedTime       0
     CRSElapsedTime          0
     AirTime                 0
     ArrDelay                0
     DepDelay                0
     Origin                  0
     Org_Airport          1177
     Dest                    0
     Dest_Airport         1479
     Distance                0
     TaxiIn                  0
     TaxiOut                 0
     Cancelled               0
     CancellationCode        0
     Diverted                0
     CarrierDelay            0
     WeatherDelay            0
     NASDelay                0
     SecurityDelay           0
     LateAircraftDelay       0
     dtype: int64
```

```
[6]: missing_percentage= df.isnull().sum().sort_values(ascending=False)/ len(df)
     missing_percentage
```

```
[6]: Dest_Airport          0.003052
     Org_Airport           0.002429
     DayOfWeek             0.000000
     SecurityDelay         0.000000
     NASDelay              0.000000
     WeatherDelay          0.000000
     CarrierDelay          0.000000
     Diverted              0.000000
     CancellationCode      0.000000
     Cancelled             0.000000
     TaxiOut               0.000000
     TaxiIn                0.000000
     Distance              0.000000
     Dest                  0.000000
     Origin                0.000000
     Date                  0.000000
     DepDelay              0.000000
     ArrDelay              0.000000
     AirTime               0.000000
     CRSElapsedTime        0.000000
     ActualElapsedTime     0.000000
     TailNum               0.000000
     FlightNum             0.000000
     Airline               0.000000
     UniqueCarrier         0.000000
     CRSArrTime            0.000000
     ArrTime               0.000000
     DepTime               0.000000
     LateAircraftDelay     0.000000
     dtype: float64
```

7. We have used the following code to perform exploratory data analysis and visualization.

8. Data splitting is done following data preprocessing before a model is constructed.

```python
# Step 1: Load and preprocess the dataset
file_path = 'Flight_delay.csv'  # Change this to the path of your file
data = pd.read_csv(file_path)
# Data Overview
print("Dataset Overview:")
# Parse date and calculate delays
data['Date'] = pd.to_datetime(data['Date'], dayfirst=True)
data['ScheduledDelay'] = data['ArrTime'] - data['CRSArrTime']
data['ScheduledDelay'] = data['ScheduledDelay'].apply(lambda x: x if x >= 0 else x + 2400)

# Select relevant columns
columns = [
    'DayOfWeek', 'DepTime', 'ArrTime', 'ScheduledDelay', 'CarrierDelay',
    'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay',
    'TaxiIn', 'TaxiOut'
]
data = data[columns]

# Normalize data
scaler = MinMaxScaler()
data_normalized = pd.DataFrame(scaler.fit_transform(data), columns=columns)

# Apply smoothing
def moving_average(series, window_size=5):
    return series.rolling(window=window_size, min_periods=1).mean()

smoothed_data = data_normalized.copy()
for col in ['ScheduledDelay', 'CarrierDelay', 'WeatherDelay', 'NASDelay', 'LateAircraftDelay']:
    smoothed_data[f'{col}_MA'] = moving_average(smoothed_data[col])

kf = KalmanFilter(initial_state_mean=0, n_dim_obs=1)
for col in ['ScheduledDelay', 'CarrierDelay', 'WeatherDelay', 'NASDelay', 'LateAircraftDelay']:
    state_means, _ = kf.smooth(smoothed_data[col].values.reshape(-1, 1))
    smoothed_data[f'{col}_KF'] = state_means.flatten()

# Statistical Models for Forecasting
train_size = int(0.8 * len(smoothed_data))
train_data = smoothed_data['ScheduledDelay'][:train_size]
test_data = smoothed_data['ScheduledDelay'][train_size:]

# Exponential Smoothing
exp_model = ExponentialSmoothing(
    train_data, seasonal='add', seasonal_periods=12
).fit()
exp_forecast = exp_model.forecast(len(test_data))

# Correlation Heatmap
plt.figure(figsize=(12, 8))
correlation_matrix = smoothed_data.corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title('Correlation Heatmap of Variables')
plt.show()
print(smoothed_data.describe())

# Highlight any high correlations for further analysis
high_corr_pairs = correlation_matrix.unstack().sort_values(ascending=False)
high_corr_pairs = high_corr_pairs[high_corr_pairs < 1.0]  # Exclude self-correlation
high_corr_threshold = 0.8
print("\nHighly Correlated Pairs (Threshold > 0.8):")
print(high_corr_pairs[high_corr_pairs > high_corr_threshold])

# ARIMA
arima_model = ARIMA(train_data, order=(5, 1, 0)).fit()
arima_forecast = arima_model.forecast(steps=len(test_data))

# Prepare sequences for the hybrid model
def create_sequences(data, target, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(target[i + seq_length])
    return np.array(X), np.array(y)

sequence_length = 10
features = [col for col in smoothed_data.columns if 'MA' in col or 'KF' in col]
X, y = create_sequences(smoothed_data[features].values, smoothed_data['ScheduledDelay'].values, sequence_length)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the hybrid deep learning model
input_layer = Input(shape=(X_train.shape[1], X_train.shape[2]))

# CNN component
cnn_layer = Conv1D(filters=64, kernel_size=3, activation='relu')(input_layer)
cnn_layer = Dropout(0.2)(cnn_layer)
cnn_layer = Flatten()(cnn_layer)

# LSTM component
lstm_layer = LSTM(64, return_sequences=False)(input_layer)
lstm_layer = Dropout(0.2)(lstm_layer)

# Combine CNN and LSTM
combined = Concatenate()([cnn_layer, lstm_layer])
output_layer = Dense(1, activation='linear')(combined)

# Compile the model
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])

# Train the hybrid model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=5,
    batch_size=32,
    verbose=1
)

# Evaluate the hybrid model
evaluation = model.evaluate(X_test, y_test, verbose=1)
print(f"Test Loss: {evaluation[0]}, Test MAE: {evaluation[1]}")
```

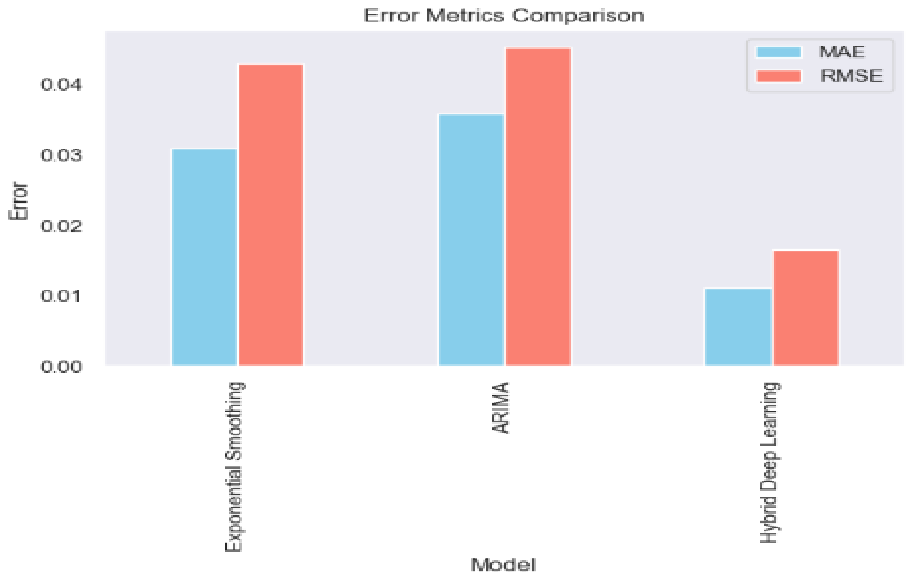
9. The implementation of hybrid models has been carried out using the following Code

```
# ARIMA
arima_model = ARIMA(train_data, order=(5, 1, 0)).fit()
arima_forecast = arima_model.forecast(steps=len(test_data))

# Prepare sequences for the hybrid model
def create_sequences(data, target, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(target[i + seq_length])
    return np.array(X), np.array(y)

sequence_length = 10
features = [col for col in smoothed_data.columns if 'MA' in col or 'KF' in col]
X, y = create_sequences(smoothed_data[features].values, smoothed_data['ScheduledDelay'].values, sequence_length)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the hybrid deep learning model
input_layer = Input(shape=(X_train.shape[1], X_train.shape[2]))

# CNN component
cnn_layer = Conv1D(filters=64, kernel_size=3, activation='relu')(input_layer)
cnn_layer = Dropout(0.2)(cnn_layer)
cnn_layer = Flatten()(cnn_layer)

# LSTM component
lstm_layer = LSTM(64, return_sequences=False)(input_layer)
lstm_layer = Dropout(0.2)(lstm_layer)

# Combine CNN and LSTM
combined = Concatenate()([cnn_layer, lstm_layer])
output_layer = Dense(1, activation='linear')(combined)

# Compile the model
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])

# Train the hybrid model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=5,
    batch_size=32,
    verbose=1
)

# Evaluate the hybrid model
evaluation = model.evaluate(X_test, y_test, verbose=1)
print(f"Test Loss: {evaluation[0]}, Test MAE: {evaluation[1]}")
```

10. Accuracy is regarded as an evaluation factor following the implementation of the model.

```
# Advanced Visualizations
# Training Metrics
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()
plt.show()

# Forecast Comparison
plt.figure(figsize=(12, 6))
plt.plot(test_data.values[:50], label='True Values', marker='o')
plt.plot(exp_forecast.values[:50], label='Exponential Smoothing', marker='x')
plt.plot(arima_forecast.values[:50], label='ARIMA', marker='s')
plt.plot(y_test[:50], label='Hybrid Model Predictions', marker='d')
plt.title('Forecast Comparison: Statistical Models vs Hybrid Model')
plt.xlabel('Time Steps')
plt.ylabel('Normalized Delay')
plt.legend()
plt.grid()
plt.show()

# Residual Analysis
residuals = y_test - model.predict(X_test).flatten()
plt.figure(figsize=(12, 6))
plt.plot(residuals, marker='o', linestyle='', label='Residuals')
plt.axhline(y=0, color='r', linestyle='--', label='Zero Line')
plt.title('Residual Analysis')
plt.xlabel('Time Steps')
plt.ylabel('Residuals')
plt.legend()
plt.grid()
plt.show()

# Error Metrics Comparison
exp_mae = mean_absolute_error(test_data, exp_forecast)
arima_mae = mean_absolute_error(test_data, arima_forecast)
hybrid_mae = mean_absolute_error(y_test, model.predict(X_test))

exp_rmse = math.sqrt(mean_squared_error(test_data, exp_forecast))
arima_rmse = math.sqrt(mean_squared_error(test_data, arima_forecast))
hybrid_rmse = math.sqrt(mean_squared_error(y_test, model.predict(X_test)))

metrics_df = pd.DataFrame({
    'Model': ['Exponential Smoothing', 'ARIMA', 'Hybrid Deep Learning'],
    'MAE': [exp_mae, arima_mae, hybrid_mae],
    'RMSE': [exp_rmse, arima_rmse, hybrid_rmse]
})

print(metrics_df)
```

```
3029/3029 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step
3029/3029 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step
                   Model       MAE      RMSE
0  Exponential Smoothing  0.031063  0.043047
1                  ARIMA  0.035914  0.045387
2   Hybrid Deep Learning  0.011159  0.016549
<Figure size 1200x600 with 0 Axes>
```




Total Execution Time:The dynamic model took 968.395 seconds to train overall, and 2.875 milliseconds to test the data using the trained model.

The final code files can be found under the heading "Enhancing Time Series Forecasting Accuracy and Resilience in High-Frequency Data Environments through Hybrid Deep Learning Smoothing Models" in the "full code on thesis ipynb" files. These files contain all of the experiments and implementation details.

## References

Anaconda. 2021. Anaconda | The World's Most Popular Data Science Platform. [online] Available at: .<https://www.anaconda.com/>.

Numpy.org. 2021. NumPy. [online] Available at: <https://numpy.org/>.

Scikit-learn.org. 2021.

scikit-learn machine learning in Python — scikit-learn 0.24.2

documentation. [online] Available at: <https://scikit-learn.org/stable/>.