National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Prajwal Joshi
Student ID: x22111034

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Prajwal Joshi
Student ID:	x22111034
Programme:	MSc in Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Abid Yaqoob
Submission Due Date:	31/01/2024
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	

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:	
Date:	31st January 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 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

Prajwal Joshi

x22111034

1 Introduction

The techniques and various software and hardware specifications utilized in the research project ”Traffic flow forecasting using DeepAR” is described in this configuration manual.

This guide’s parts are as follows: Section 3 provides further information on the environment setup’s characteristics. Section 4 discusses the libraries required to finish this project. The whole dataset is described in Section 5. Section 6 offers information about the code repository and the models’ implementation.

2 Overview

This research compares several time series models for traffic flow prediction in various boroughs of New York City.

3 System Specifications

The prerequisites are as follows: the hardware and software infrastructures needed to train a model on such a large amount of data.

3.1 Hardware Requirements

Processor: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz; Installed RAM:16.0 GB (15.8 GB usable);

System type: 64-bit operating system, x64-based processor;

OS: Microsoft Windows 11

3.2 Software Requirements

Programming Language: Python version 3.10 Integrated Development Environment (IDE): Google Colab

4 Python Libraries Required

The image below indicates all the python libraries required for the project.

```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.metrics import mean_squared_error
import gluonts
from gluonts.dataset.common import ListDataset
from gluonts.mx.model.deepar import DeepAREstimator
from gluonts.mx.trainer import Trainer
from gluonts.evaluation.backtest import make_evaluation_predictions
from optuna import create_study
from gluonts.evaluation import Evaluator
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
```

Figure 1: Required Python Libraries

5. Data Sources

5.1 Dataset

The data set was obtained from Opendata NYC website. The top 5 entries of the data set are shown in Figure 2.

	RequestID	Boro	Yr	M	D	HH	MM	Vol	SegmentID	WktGeom	street	fromSt	toSt	Direction	Datetime
0	32110	Bronx	2020	3	3	0	0	6	88955	POINT (1022455.0901136672 260806.54150942338)	EAST 218 STREET	Barnes Avenue	White Plains Road	WB	2020-03-03 00:00:00
1	32110	Bronx	2020	3	3	0	15	15	88955	POINT (1022455.0901136672 260806.54150942338)	EAST 218 STREET	Barnes Avenue	White Plains Road	WB	2020-03-03 00:15:00
2	32110	Bronx	2020	3	3	0	30	11	88955	POINT (1022455.0901136672 260806.54150942338)	EAST 218 STREET	Barnes Avenue	White Plains Road	WB	2020-03-03 00:30:00
3	32110	Bronx	2020	3	3	0	45	3	88955	POINT (1022455.0901136672 260806.54150942338)	EAST 218 STREET	Barnes Avenue	White Plains Road	WB	2020-03-03 00:45:00
4	32110	Bronx	2020	3	3	1	0	5	88955	POINT (1022455.0901136672 260806.54150942338)	EAST 218 STREET	Barnes Avenue	White Plains Road	WB	2020-03-03 01:00:00

Figure 2: output of df.head() function

6. Modelling

LSTM networks, Autoregressive Integrated Moving Weight Average(ARIMA) models and Deep Autoregressive networks(DeepAR) were used for the task.

6.1 ARIMA MODELLING

1. A grid search is ran in order to determine the best parameters for the model before it's fitted.

```
for p in range(3):  # Maximum p value to test
    for d in range(2):  # Maximum d value to test
        for q in range(3):  # Maximum q value to test
            with warnings.catch_warnings():
                warnings.simplefilter("ignore")
                try:
                    model = ARIMA(train_data, order=(p, d, q))
                    fitted_model = model.fit()
                    forecast = fitted_model.forecast(steps=test_size)
```

Fig 3: Function to run grid search

2. The model is fitted to training data with optimal parameters.

```
# Fit ARIMA model with best order using all the available data
model = ARIMA(data, order=best_order)
fitted_model = model.fit()
forecast = fitted_model.forecast(steps=test_size)
```

Fig 4: ARIMA Model Fitting

6.2 Long Short-Term Memory Networks(LSTM)

The LSTM model is fitted with training data and implemented. The figures below indicate the code and sample output.

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, batch_size=64, verbose=1)
```

Fig 5: LSTM Modelling

```
Epoch 1/50
20/20 [==============================] - 5s 54ms/step - loss: 0.0083 - val_loss: 0.0435
Epoch 2/50
20/20 [==============================] - 0s 11ms/step - loss: 0.0025 - val_loss: 0.0254
Epoch 3/50
20/20 [==============================] - 0s 10ms/step - loss: 0.0011 - val_loss: 0.0060
Epoch 4/50
20/20 [==============================] - 0s 10ms/step - loss: 4.0113e-04 - val_loss: 0.0100
Epoch 5/50
20/20 [==============================] - 0s 9ms/step - loss: 3.8186e-04 - val_loss: 0.0063
Epoch 6/50
20/20 [==============================] - 0s 9ms/step - loss: 3.6638e-04 - val_loss: 0.0067
Epoch 7/50
20/20 [==============================] - 0s 9ms/step - loss: 3.7123e-04 - val_loss: 0.0067
Epoch 8/50
20/20 [==============================] - 0s 9ms/step - loss: 3.7469e-04 - val_loss: 0.0062
Epoch 9/50
20/20 [==============================] - 0s 9ms/step - loss: 3.6091e-04 - val_loss: 0.0069
Epoch 10/50
20/20 [==============================] - 0s 10ms/step - loss: 3.8596e-04 - val_loss: 0.0058
Epoch 11/50
20/20 [==============================] - 0s 9ms/step - loss: 3.7575e-04 - val_loss: 0.0070
Epoch 12/50
20/20 [==============================] - 0s 9ms/step - loss: 3.6072e-04 - val_loss: 0.0065
Epoch 13/50
20/20 [==============================] - 0s 10ms/step - loss: 3.5437e-04 - val_loss: 0.0064
```

Fig 6: Sample output

6.3 DeepAR Modelling

1. Before model fitting, hyperparameter optimization is implemented using modules from Optuna library.

2. Trainer function is defined and is used for model training with most optimal hyperparameters. The dataset is also converted into list dataset format.

```
trainer = Trainer(
    epochs=10,
    learning_rate=learning_rate,
    num_batches_per_epoch=50
)

estimator = DeepAREstimator(
    freq='15min',
    prediction_length=4,
    trainer=trainer,
    context_length=context_length,
    num_layers=num_layers,
    cell_type=cell_type
)


train_ds = ListDataset(
    [{"start": data.index[0], "target": data['Vol'].values}],
    freq="15min"
)
```

Fig.7: Trainer and estimator function definition

3. The model is then fitted to the training data using the optimal hyperparameters.

```
[1536 rows x 1 columns]] [gluonts.model.forecast.SampleForecast(info=None, item_id=None, samples=array([[437.9876 , 415.86917, 410.39432, 413.9658
       [443.0591 , 399.44125, 415.88443, 448.5428 ],
       [428.95602, 448.7329 , 414.70282, 419.31085],
       [447.2245 , 411.14673, 414.68796, 448.83813],
       [396.55368, 421.1451 , 425.6858 , 422.3639 ],
       [346.73294, 418.6611 , 379.9171 , 397.97632],
       [416.40085, 419.99637, 427.3099 , 376.09482],
       [447.77286, 383.85318, 407.94373, 405.07596],
       [416.9696 , 418.57346, 452.13882, 409.9089 ],
       [440.92743, 429.60785, 442.85068, 411.96576]], dtype=float32), start_date=Period('2020-11-03 23:00', '15T'))]
Running evaluation: 100%|██████████| 1/1 [00:00<00:00, 17.77it/s]RMSE for Manhattan Data: 54.602818365332574
```

Fig 8: Sample output