National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Carlos Manuel Benitez Martinez
Student ID: 24104515

School of Computing
National College of Ireland

Supervisor: Mr. John Kelly

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Carlos Manuel Benitez Martinez
Student ID:	24104515
Programme:	Master in Data Analytics **Year:** 2025
Module:	MSc Research Project
Lecturer:	Mr. John Kelly
Submission Due Date:	11/08/2025
Project Title:	Leveraging Deep Learning for Pedestrian and Cycle Flow Prediction in Urban Environments
Word Count:	1756 **Page Count:** 14

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:	Manuel Benitez
Date:	11/08/2025

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

Carlos Manuel Benitez Martinez
Student ID: x24104515

1 Introduction

This document contains the detailed configuration for the project Leveraging Deep Learning for Pedestrian and Cycle Flow Prediction in Urban Environments. The current work aims to provide a stable platform for the implementation of pedestrian and cycle flow predictions.

2 System Configuration

2.1 System Configuration

The hardware requirements to successfully run this project are as follows:

- **Processor:** A CPU such as i7 of 10th generation or equivalent. For cloud-based environment, an Intel(R) Xeon(R) model 85 is needed.
- **RAM:** At least 12 GB for preprocessing and baseline modelling, and more to process all deep learning algorithms operations, which require sufficient memory.
- **Storage:** Fast storage, at least 256 GB SSD, for storing and retrieving data, and access to Microsoft Azure Blob Storage.
- **GPU:** NVIDIA GA100 for fast training of deep learning models with moderately complex architectures.

3 Execution Environment

This project employed Python 3 using two separate configurations in the Google Colab Pro+ environment (refer to **Table 1**), using Microsoft Azure to store and retrieve the datasets during the research process.

Table 1: Google Colab Pro+ Environment implemented.

Google Colab Pro+ Environment	Tasks	Accelerator	System RAM
Environment 1 – High memory CPU	• Exploratory Data Analysis • Data preprocessing	CPU	Up to 51 GB
	Baseline models: • Random Forest • Extreme Gradient Boosting	CPU	Up to 51 GB
Environment 2 – GPU NVIDIA GA100	Deep Learning • LSTM • CNN-LSTM	GPU NVIDIA GA100 (40 GB VRAM)	Up to 83.5 GB

4 Software Requirements

The software environment needed to implement this project includes:

- **Operating system:** A stable operating system such as Ubuntu Linux 6.1.123+ or Windows 10, able to support libraries and software.
- **Python 3.11 or higher:** The programming language used to develop and run the project.
- **Jupyter Notebook:** An interactive environment for development, can be accessed from Anaconda, Visual Studio, or Google Colab.
- **Tensor Flow 2.19:** Main framework dedicated for building and training deep learning models, including Keras for LSTM and CNN-LSTM modelling.
- **Libraries:** several libraries for different tasks along with the project (refer to **Table 2**). Some libraries related to TensorFlow or Azure need to be strictly installed when the Google Colab session starts to function properly (refer to **Figure 1**). Also, libraries are imported depending on the model in the notebook (refer to **Figure 2**).

Table 2: Libraries used in the project

Purpose	Library	Version
Data and text manipulation	pandas	2.2.2
	numpy	2.0.2
	re	N/A
Data visualisation	matplotlib	3.10.0
	seaborn	0.13.2
Traditional machine learning and data processing	Scikit-learn	1.6.1
	xgboost	3.0.3
Azure storage and access	adlfs	2024.12.0
	azure-storage-blob	12.26.0
Irish holidays	holidays:	0.78
Randomness control	random	N/A
Operating system functions	os	N/A
Progress bar	tqdm	N/A
Data flow management	io	N/A
Map creation	folium	0.20.0

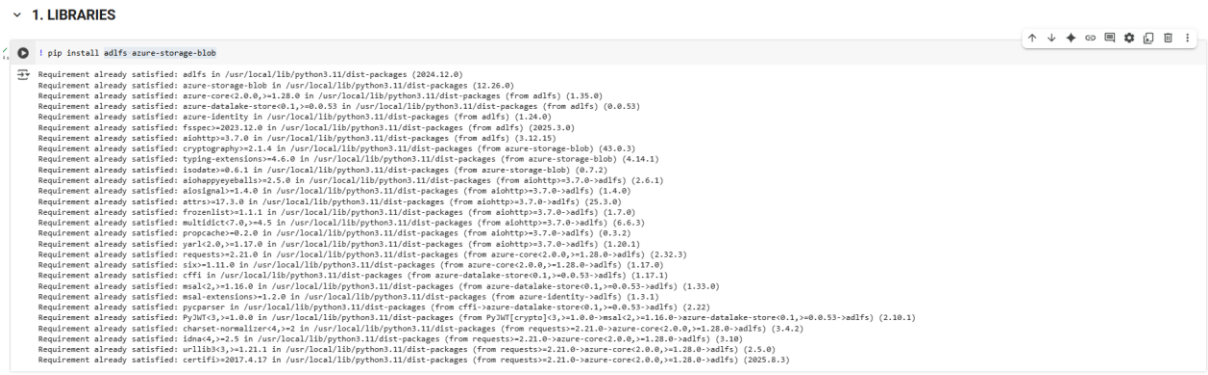
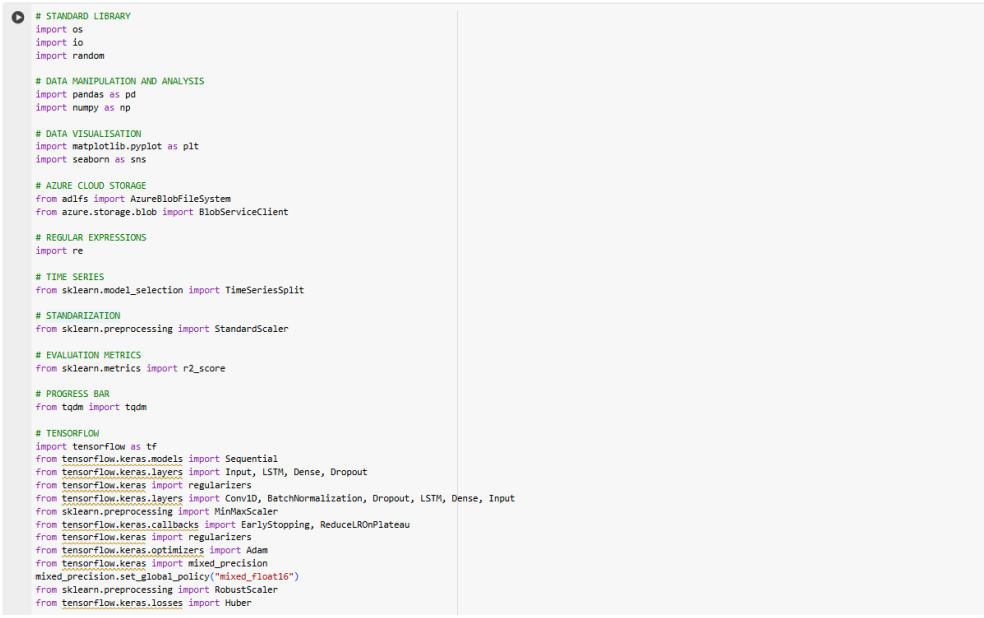


Figure 1: Mandatory pip installation for Azure retrieval and storage libraries



```
# STANDARD LIBRARY
import os
import io
import random

# DATA MANIPULATION AND ANALYSIS
import pandas as pd
import numpy as np

# DATA VISUALISATION
import matplotlib.pyplot as plt
import seaborn as sns

# AZURE CLOUD STORAGE
from adlfs import AzureBlobFileSystem
from azure.storage.blob import BlobServiceClient

# REGULAR EXPRESSIONS
import re

# TIME SERIES
from sklearn.model_selection import TimeSeriesSplit

# STANDARIZATION
from sklearn.preprocessing import StandardScaler

# EVALUATION METRICS
from sklearn.metrics import r2_score

# PROGRESS BAR
from tqdm import tqdm

# TENSORFLOW
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, LSTM, Dense, Dropout
from tensorflow.keras import regularizers
from tensorflow.keras.layers import Conv1D, BatchNormalization, Dropout, LSTM, Dense, Input
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras import regularizers
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import mixed_precision
mixed_precision.set_global_policy("mixed_float16")
from sklearn.preprocessing import RobustScaler
from tensorflow.keras.losses import Huber
```

Figure 2: Libraries implemented for CNN-LSTM

5 Project Implementation

The project has been developed in different Google Colab notebooks, providing a unique view of each dataset and stage of the research.

5.1 Data Collection

All datasets used in the project are publicly accessible and published under the Creative Commons Attribution 4.0 International (CC BY 4.0). Four different datasets that contain Dublin city information have been collected from the Dublin City Council (DCC) Transport City Centre Projects (2020, 2025), Smart Dublin (2025) and © Met Éireann (n.d.). Some datasets were divided into different subsets that needed integration (refer to **Figure 3**). It is important to note that the files have been stored in Microsoft Azure Blob Storage for retrieval and access to reduce loading and processing time.

The datasets can be downloaded at:

- https://data.gov.ie/dataset/dublin-city-centre-cycle-counts
- https://data.smartdublin.ie/dataset/dublin-city-centre-footfall-counters
- https://www.met.ie/climate/available-data/historical-data
- https://data.smartdublin.ie/dataset/pedestrian-and-cycle-counter-api-for-dublin-region/resource/f80d1ec3-e915-4ef0-988f-e3304b97dd37?view_id=71a929c0-b6f8-48ac-86bf-a2d58bc0eee7

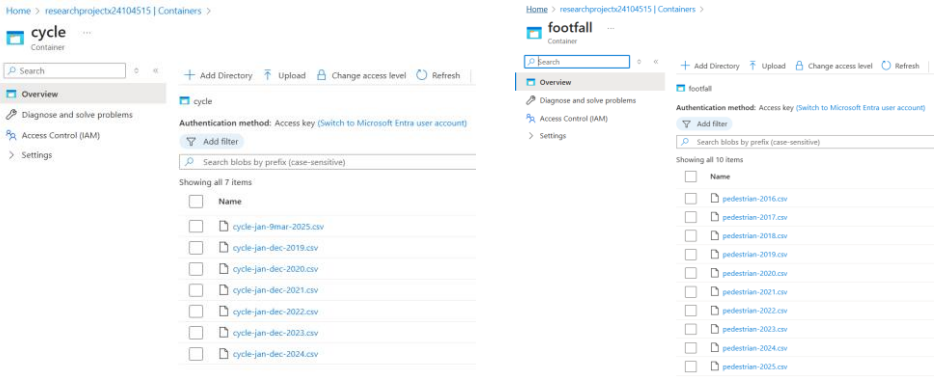


Figure 3: Cycle and footfall subsets per year in Azure Blob Storage.

5.2 Feature Selection

Prior the modelling stage, feature selection was completed, along with pre-processing for hourly footfall, cyclists, and weather data (refer to **Figure 4**), and static counter locations data from multiple sources as mentioned in Section 5.1

Footfall:
(79700, 21)

	date and time	FTF_Aston Quay Merged	FTF_Bachelors Walk Merged	FTF_Capel St Mary St Merged	FTF_College St Dame St Merged	FTF_Dame St Merged	FTF_Dolier St Burgh Quay Merged	FTF_Grafton St Monsoon Merged	FTF_Grafton St Nassau St Suffolk St	FTF_Henry St Merged	FTF_Mary St Merged	FTF_Oconnell St Princes St North Merged	FTF_Oconnell St Pennys Merged	FTF_Talbot St Guineys Merged	FTF_Talbot St Murrays Pharmacy Merged	FTF_Westmoreland St East Fleet St Merged	FTF_Westmoreland St West Merged
0	2019-01-01 00:00:00	0.0	0.0	16.0	0.0	0.0	0.0	140.0	0.0	95.0	20.0	1914.0	244.0	0.0	0.0	1670.0	1988.0
1	2019-01-01 00:15:00	0.0	0.0	62.0	0.0	0.0	0.0	0.0	0.0	265.0	24.0	0.0	454.0	0.0	0.0	0.0	0.0
2	2019-01-01 00:30:00	0.0	0.0	99.0	0.0	0.0	0.0	0.0	0.0	117.0	69.0	0.0	391.0	0.0	0.0	0.0	0.0

Cycle:
(54239, 7)

	date and time	CYC_Clontarf James Larkin Rd	CYC_Clontarf Pebble Beach Carpark	CYC_Grove Road Totem	CYC_North Strand Rd N B	CYC_Richmond Street Cyclists 1 Merged	CYC_Richmond Street Cyclists 2 Merged
0	2025-01-01 00:00:00	3.0	7.0	12.0	0.0	9.0	5.0
1	2025-01-01 01:00:00	2.0	2.0	17.0	0.0	1.0	1.0
2	2025-01-01 02:00:00	0.0	1.0	19.0	0.0	3.0	15.0
3	2025-01-01 03:00:00	0.0	1.0	7.0	0.0	3.0	12.0
4	2025-01-01 04:00:00	0.0	1.0	5.0	0.0	1.0	6.0

Weather:
(54025, 8)

	date and time	WTH_rain	WTH_temp	WTH_wetb	WTH_dewpt	WTH_vappr	WTH_rhum	WTH_msl
0	2019-01-01 00:00:00	0.0	9.6	7.8	5.6	9.1	76.0	1035.1
1	2019-01-01 01:00:00	0.0	8.6	7.1	5.3	8.9	79.0	1035.1
2	2019-01-01 02:00:00	0.0	8.3	6.9	5.3	8.9	81.0	1034.9
3	2019-01-01 03:00:00	0.0	9.1	7.6	5.8	9.2	79.0	1035.5
4	2019-01-01 04:00:00	0.0	9.2	7.7	5.9	9.3	79.0	1035.7

```
[ ] counters = counters.sort_values(by=["User Type", "Bike Counter Locations"]).reset_index(drop=True)
    counters
```

	Bike Counter Locations	Latitude	Longitude	User Type
0	castleknock totem	53.370000	-6.349436	CYC
1	charleville mall	53.356950	-6.245480	CYC
2	clonskeagh totem	53.306376	-6.237057	CYC
3	clontarf james larkin rd	53.368540	-6.170320	CYC
4	clontarf pebble beach carpark	53.358250	-6.191070	CYC

Figure 4: Footfall and Cyclist counts, Weather conditions and counter locations datasets

5.3 Data Pre-processing

A detailed inspection for data quality was conducted due to inconsistencies in the name of the cycle and footfall counter locations. For pedestrian dataset for 2019, the granularity is different than the rest, so it needed to be adjusted (refer to **Figure 5**). Some locations contained all null or zero values, and the remaining variables had repeated names and typos (refer to **Figure 6**). Cyclist counter did not face many challenges rather than different names for a single locations, meanwhile footfall encountered difficulties that were solved through another dataset containing the Eco-Visio Output of the counter that helped to identify the right spot of data integration (refer to **Figure 7**).

```
[8] p2019.head()
```

	Time	O'Connell St Outside Pennys	O'Connell St Outside Clerys	Mary Street	Capel Street	Aston Quay	Grafton Street @ CompuB	Talbot Street North	Doilier Street, Burgh Quay	Dawson Street Replacement	Dame Street	Talbot Street South	O'Connell St, Parnell St @ AIB	Grafton Street / Nassau Street / Suffolk Street	College Green, Bank Of Ireland	Henry Street	Westmoreland Street East	Dawson Street	Liffey Street	Westmoreland Street West	Grafton Street	Bachelors Walk	College Green @ Church Lane	College Green - Dame St Side
0	01-01-2019 00:00:00	244	1914.0	20	16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	95.0	1670.0	255.0	881.0	1988.0	140.0	NaN	NaN	NaN
1	01-01-2019 00:15:00	454	NaN	24	62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	265.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	01-01-2019 00:30:00	391	NaN	69	99	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	117.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	01-01-2019 00:45:00	415	NaN	50	61	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	01-01-2019 01:00:00	319	885.0	24	47	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	90.0	767.0	209.0	723.0	1270.0	215.0	NaN	NaN	NaN

Figure 5: Footfall 2019 datasets original timestamp

```
Dropping columns with all null or zero values: ['dawson street old', 'drumcondra cyclists inbound not on site roadworks pedestrian', 'drumcondra cyclists inbound not on site roadworks pedestrian east', 'drumcondra cyclists inbound not on site roadworks pedestrian west', 'liffey stre
(186004, 60)
```

	aston quay	aston quay fitzgeralds	bachelors walk	bachelors walk bachelors way	bachelors walk bachelors way nonresponsive counter removed to return to supplier	baggot st lower wilton tce inbound	baggot st lower wilton tce inbound missing from site	baggot st upper mespil rd bank	capel street	capel street removed from site 20 10	capel st mary street	college green dame st side	college green church lane	college green bank of ireland	college st westmoreland st	dolier st burgh quay	dame street	dame street counter missing	dame street londis	date and time	dawson street	dawson street replacement	dawson street molesworth	dawson street molesworth pedestrian	dolier street burgh quay	grafton street	grafton street nassau street suffolk street	grafton street compub	grafton st monsoon	graf st pede
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2016-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	139.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2016-01-01 01:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	125.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2016-01-01 02:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	196.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2016-01-01 03:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	130.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2016-01-01 04:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Figure 6: Footfall datasets after dropping all null values columns

_id	Counter Locations	Eco-Visio Oupput	Latitude	Longitude	Setup Date	User Type
1	Henry Street	Henry Street/Coles Lane/Dunnes	53.34973	-6.2609	2013-07-16T00:00:00	Pedestrians
2	O'Connell St Outside Clerys	O'Connell st/Princes st North	53.34902	-6.26005	2011-07-06T00:00:00	Pedestrians
3	Mary Street	Mary st/Jervis st	53.34877	-6.26674	2011-09-06T00:00:00	Pedestrians
4	Capel Street	Capel st/Mary street	53.34842	-6.26918	2011-01-08T00:00:00	Pedestrians
5	Aston Quay	Aston Quay/Fitzgeralds	53.34662	-6.25982	2019-02-27T00:00:00	Pedestrians
6	Grafton Street @ CompuB	Grafton Street/CompuB	53.34337	-6.25898	2019-02-27T00:00:00	Pedestrians
7	Talbot Street North	Talbot st/Guineys	53.35054	-6.25528	2019-02-27T00:00:00	Pedestrians
8	Doilier Street, Burgh Quay	D'olier st/Burgh Quay	53.3469	-6.25872	2019-02-27T00:00:00	Pedestrians
9	Dame Street	Dame Street/Londis	53.34424	-6.26116	2019-02-27T00:00:00	Pedestrians
10	Talbot Street South	Talbot st/Murrays Pharmacy	53.35012	-6.2577	2019-02-27T00:00:00	Pedestrians
11	O'Connell St, Parnell St @ AIB	O'Connell St/Parnell St/AIB	53.3523172	-6.2616753	2019-03-19T00:00:00	Pedestrians
12	Grafton Street / Nassau Street / Suffolk Street	Grafton Street / Nassau Street / Suffolk Street	53.34352	-6.25912	2019-02-27T00:00:00	Pedestrians
13	College Green, Bank Of Ireland	College Green/Bank Of Ireland	53.34505	-6.25939	2019-02-27T00:00:00	Pedestrians

Figure 7: Footfall locations and name variants.

5.4 Exploratory Data Analysis

Employing libraries such as pandas, matplotlib and seaborn, different temporal patterns were plotted, and analysed, such as hourly, daily and monthly patterns (refer to **Figure 8, Figure 9, Figure 10** and **Figure 11**), for all datasets, except the counter locations which only a map was included which locates all counter available in Dublin city. Plot and lines charts shows the information about variables included in this study.

HEAT MAP: AVERAGE TEMPERATURE BY HOUR AND MONTH

```
[ ] # Create matrix for heatmap (hour x month)
    pivot_temp = data.groupby(["month", "hour"])["temp"].mean().unstack()

    plt.figure(figsize=(20, 8))
    sns.heatmap(pivot_temp, cmap="viridis", annot=False, fmt=".1f")
    plt.title("Heat Map: Average Temperature by Hour and Month")
    plt.xlabel("Hour of Day")
    plt.ylabel("Month")
    plt.tight_layout()
    plt.show()
```

Figure 8: Weather Exploratory Data Analysis

```python
def plot_patterns(data, exclude_cols, x_label, title, x_ticks, x_tick_labels=None, peak_label_prefix="Peak", figsize=(10, 8), color_palette=None, save_path=None):
    # Set the plot style
    set_plot_style()

    # Filter columns by location
    location_columns = [col for col in data.columns if col not in exclude_cols]

    # Calculate average counts for sorting legend
    avg_counts = {col: data[col].mean() for col in location_columns}

    # Sort location columns by average count descending (highest to lowest)
    sorted_columns = sorted(avg_counts, key=avg_counts.get, reverse=True)

    # Set color palette
    if color_palette:
        sns.set_palette(color_palette)

    # Plot with sorted columns
    fig, ax = plt.subplots(figsize=figsize)
    data[sorted_columns].plot(kind="line", marker="o", ax=ax)

    plt.title(title, fontsize=20)
    plt.xlabel(x_label, fontsize=14)
    plt.ylabel("Average Count", fontsize=14)
    plt.xticks(x_ticks, x_tick_labels)
    plt.grid(True, alpha=0.3)

    # Legend is already sorted by traffic volume since we plotted sorted_columns
    ax.legend(title="Location", loc="upper right", frameon=True)

    # Highlight peak
    peak_index = data[location_columns].mean(axis=1).idxmax()
    if x_tick_labels and peak_index <= len(x_tick_labels):
        peak_label = f"{peak_label_prefix}: {x_tick_labels[peak_index-1]}"
    else:
        peak_label = f"{peak_label_prefix}: {peak_index}"

    plt.axvline(x=peak_index, color="red", linestyle="--", alpha=0.7, label=peak_label)

    plt.tight_layout()
    plt.show()

    return fig, ax
```

Figure 9: Cyclist flow Exploratory Data Analysis

```python
[ ] def map_user_type(x):
        if "Pedestrians" in x and "Cyclists" in x:
            return "FTF_CYC"
        elif "Pedestrians" in x:
            return "FTF"
        elif "Cyclists" in x:
            return "CYC"
        else:
            return x

    counters["User Type"] = counters["User Type"].apply(map_user_type)
```

```python
# Dublin city centre map
dublin_center = [53.34, -6.26]
dmap = folium.Map(location=dublin_center, zoom_start=12, width=1800, height=800)

# Add markers to map
for _, row in counters.iterrows():
    icon_color = (
        "blue" if row["User Type"] == "CYC"
        else "purple" if row["User Type"] == "FTF_CYC"
        else "green"
    )

    popup = folium.Popup(row["Bike Counter Locations"], max_width=200)

    folium.Marker(
        location=[row["Latitude"], row["Longitude"]],
        popup=popup,
        tooltip=row["User Type"],
        icon=folium.Icon(color=icon_color)
    ).add_to(dmap)

dmap
```

Figure 10: Exploratory Data Analysis

```python
plt.figure(figsize=(10, 7))

# Days of the week
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

# Data per weekday
daily_pattern = footfall.groupby("weekday")[location_cols].sum()

# Main locations
for location in top_locations:
    plt.plot(daily_pattern.index, daily_pattern[location], marker="o", linewidth=2, label=location)
plt.title("Footfall patterns daily during the week", fontsize=16)
plt.xlabel("Day of the week", fontsize=12)
plt.ylabel("Pedestrian count", fontsize=12)
plt.xticks(range(7), days)
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```

Figure 11: Weather Exploratory Data Analysis

5.5 Data Integration

All datasets were merged with a pipeline that preserves the individuality of the features from each dataset, identifying date and time as index for the alignment of heterogeneous times series ensuring the hourly timestamps of all data are respected.

5.6 Data Transformation

The work by different authors and institutions provided an understanding and explanation for the feature engineering (see **Table 3**) to improve the predictive power of the model, and resulted in the addition of more explanatory features such as dynamic and time-varying features (refer to **Figure 12**). Each model includes normalising all data suitable for their implementation, whether scaling the data, or creating LSTM sequences that help the model to detect different patterns and forecast pedestrian and cyclist counts.
After pre-processing the data, the resulting DataFrame is stored in Microsoft Azure Blob Storage, combined with Power BI, it resulted key for data visualization, to understand patterns in data in clean file for further modelling or analysis (refer to **Figure 13**).

Table 3: Feature Engineering references

Feature	Description	Author
Rolling averages	Average of temperatures within the last three hours	ACCA Global (n.d.),
Condensation probability through dew point spread	The closer the temperature and dew point temperature are the probability of condensation or fog increases	National Weather Service (n.d.) and Fiveable (2025),
Weather severity thresholds	Considering rough weather conditions as low temperature and super heavy rain,	National Center for Hydrology and Meteorology (NCHM) (n.d.)
Rainfall intensity classification	The rain intensity classification	U.S. Geological Survey,
Time components	Cyclical codification of time for hour, day and month through sine and cosine	Rodrigo & Escobar Ortiz (n.d.)

```python
def time_components(data):
  # Irish Holiday checked added to the dataframe
    ie_holidays = holidays.Ireland()
    datetime_index = data.index
    dates_series = pd.Series(datetime_index.date, index=data.index)

    # Time Components
    time_components = pd.DataFrame({
        "TEMP_hour": datetime_index.hour,
        "TEMP_weekday": datetime_index.dayofweek,
        "TEMP_month": datetime_index.month,
        "TEMP_year": datetime_index.year,
        "TEMP_holiday": dates_series.isin(ie_holidays).astype(int)
    }, index=data.index)

    data = pd.concat([data, time_components], axis=1)

    # Reset index
    data = data.reset_index()
    data.columns.name = None

    print(f"Shape: {data.shape}")
    print(f"Columns: {data.columns.tolist()}")

    return data
```

```python
# DYNAMIC WEATHER CONDITIONS

def weather_fts(data):

    # WEATHER INTERACTION

    # Mean of temperature in the last three hours recorded
    data["WTH_temp_3h"] = data["WTH_temp"].rolling(window=3).mean()

    # Rain intensity bins (from no rain to very heavy)
    data["WTH_rain_int"] = pd.cut(
        data["WTH_rain"],
        bins=[-0.01, 0.0, 10.0, 30.0, 70.0, float("inf")],
        labels=[ 0, 1, 2, 3, 4 ]

        # the labels for the numbers are as follows ["no_rain", "slight", "moderate", "heavy", "very_heavy"]
    )

    # Probability of condensation
    # Dew point spread
    data["WTH_dew_spread"] = data["WTH_temp"] - data["WTH_dewpt"]
    data["WTH_prob_cond"] = np.where(
        (data["WTH_rhum"] >= 90) & (data["WTH_dew_spread"] <= 2), 1, 0)

    # WEATHER SEVERITY

    # Weather severity thresholds
    data["WTH_severe"] = 0
    data.loc[data["WTH_temp"] < 5, "WTH_severe"] += 1
    data.loc[data["WTH_rain"] > 4, "WTH_severe"] += 1

    return data
```

Figure 12: Time components and feature engineering.

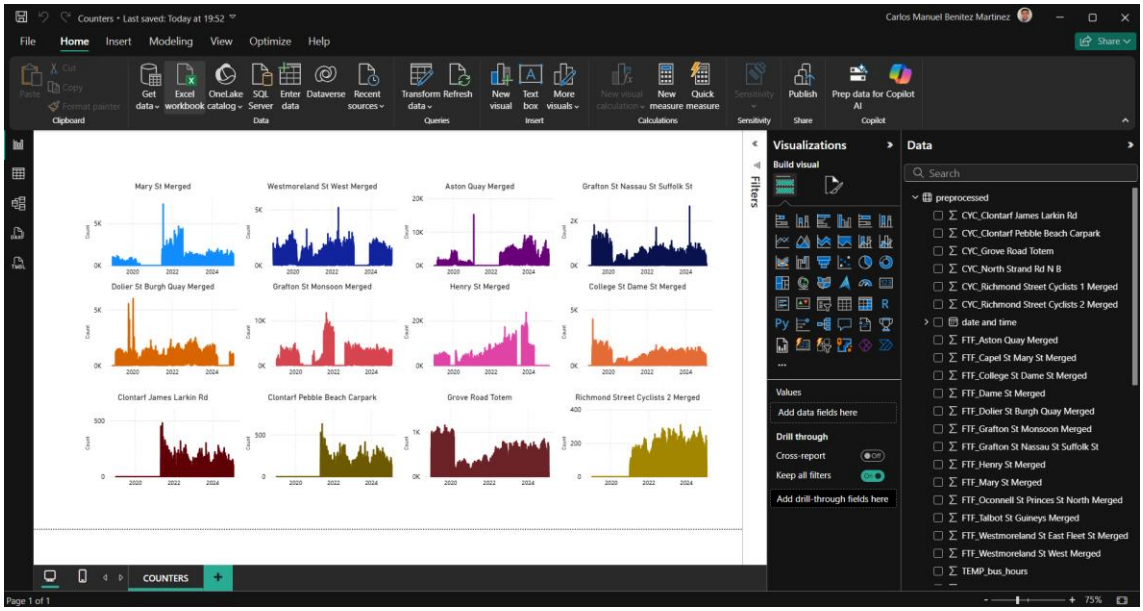


Figure 13: Integration of Pre-processed dataset with Power BI.

5.7 Data Modelling

There are four models implemented in this study, categorised as baselines (2) and deep learning (2), deep learning is implemented with a variant to understand the influence that

geographical coordinates have on deep learning models, which results in a total of six models being implemented to the data. The models implemented are shown in **Table 4.**

Table 4: Models implemented

Category	Model	Description
Baseline	**Random Forest**	An ensemble of multiple tree-based model (refer to **Figure 14**).
	Extreme Gradient Boosting (XGBoost)	Boosting algorithm that builds decision trees sequentially, focuses on error reduction by learning from previous trees (refer to **Figure 15**).
Deep Learning	**LSTM**	Signal processed into Long-term Short-term (LSTM) network to capture temporal dependencies within data (refer to **Figure 16**).
	CNN-LSTM	Convolutional Neural Networks for feature extraction and LSTM for data forecasting on pedestrian and cycle counts (refer to **Figure 17**).

```
[ ]  def rforest_setup(n_estimators=200, max_depth=6, random_state=42):
         config = {
             "n_estimators": n_estimators,
             "max_depth": max_depth,
             "random_state": random_state,
             "n_jobs": -1,
             "verbose": 0
         }
         return {
             "config": config,
             "models": {},
             "results": {}
         }
```

Figure 14: Random Forest model

```
[ ]  def xgboost_setup(n_estimators=200, max_depth=6, learning_rate=0.1, random_state=42):
         config = {
             "n_estimators": n_estimators,
             "max_depth": max_depth,
             "learning_rate": learning_rate,
             "random_state": random_state,
             "n_jobs": -1,
             "verbosity": 0
         }
         return {
             "config": config,
             "models": {},
             "results": {}
         }
```

Figure 15: XGBoost model

```python
# LSTM model

def build_lstm(input_shape):
    reg_strength = 1e-4  # Level 2 regularisation

    model = Sequential()
    model.add(Input(shape=input_shape))

    # First  LSTM layer
    model.add(LSTM(64, return_sequences=True,
                   kernel_regularizer=regularizers.l2(reg_strength)))
    model.add(Dropout(0.3))

    # Second LSTM layer
    model.add(LSTM(32, return_sequences=False,
                   kernel_regularizer=regularizers.l2(reg_strength)))
    model.add(Dropout(0.3))

    # Middle dense layer
    model.add(Dense(64, activation="relu",
                    kernel_regularizer=regularizers.l2(reg_strength)))
    model.add(Dropout(0.2))

    # Output
    model.add(Dense(1))

    model.compile(optimizer=Adam(learning_rate=0.001),
                  loss="mse",
                  metrics=["mae"])

    return model
```

Figure 16: LSTM model

```python
def build_cnnlstm(input_shape, conv_filters=64, lstm_units=64, dense_units=64, dropout_rate=0.2):
    # Level 2 regularisation
    reg = 1e-4

    # Batch Normalization for Training stabilisation and Dropout to avoid overfitting

    # Model
    model = Sequential()
    model.add(Input(shape=input_shape))

    # FEATURE EXTRACTION
    # First convolutional layer for feature extraction from short-term sequential data
    model.add(Conv1D(conv_filters, 3, activation="relu", padding="same",
                     kernel_regularizer=regularizers.l2(reg)))
    model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    # Second convolutional layer for more abstract features and long-term temporal patterns
    model.add(Conv1D(conv_filters//2, 5, activation="relu", padding="same",
                     kernel_regularizer=regularizers.l2(reg)))
    model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    # LSTM FOR LONG TERM DEPENDENCIES
    # First LSTM layer, returns to a sequence that keeps temporal information
    model.add(LSTM(lstm_units, return_sequences=True, dropout=dropout_rate, recurrent_dropout=dropout_rate,
                   kernel_regularizer=regularizers.l2(reg)))
    model.add(Dropout(dropout_rate))

    # Second LSTM layer, returns to a single output
    model.add(LSTM(lstm_units//2, return_sequences=False, dropout=dropout_rate, recurrent_dropout=dropout_rate,
                   kernel_regularizer=regularizers.l2(reg)))
    model.add(Dropout(dropout_rate))

    # Dense layer for learned features before final prediction
    model.add(Dense(dense_units, activation="relu", kernel_regularizer=regularizers.l2(reg)))
    model.add(Dropout(dropout_rate))

    # Single output neuron with no negative predictions
    model.add(Dense(1, activation="relu"))

    # Model compilation
    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss=Huber(delta=1.0),
        metrics=["mae"]
    )
    return model
```

Figure 17: Hybrid CNN-LSTM model

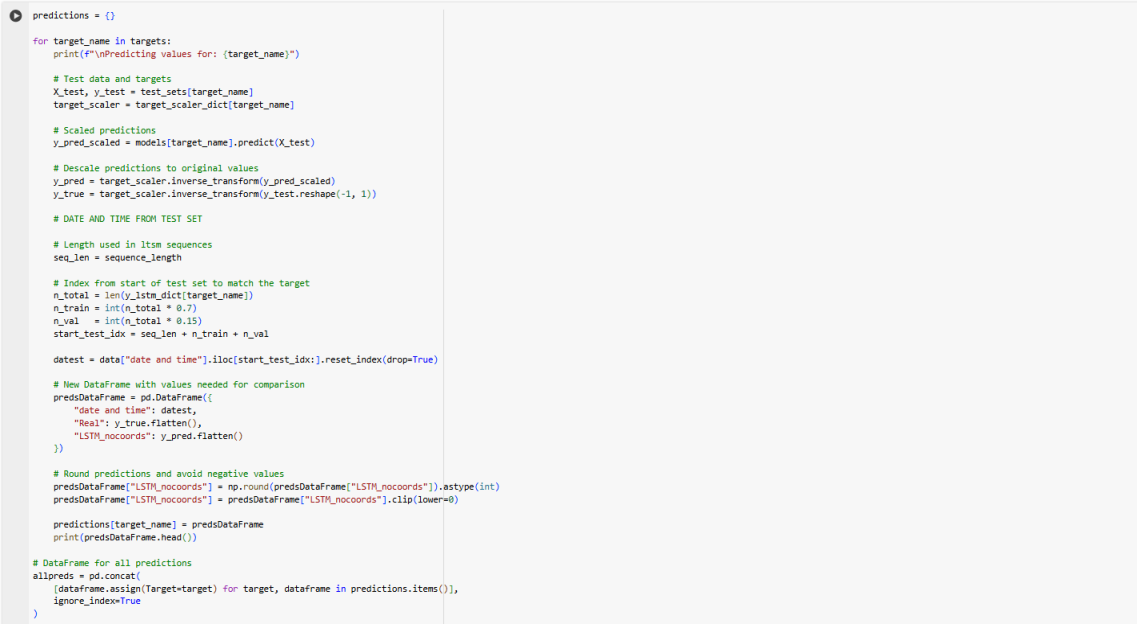
6 Evaluation

Each model was evaluated on four metrics, such as root mean square error, mean absolute error, coefficient of determination, and mean absolute percentage error. The development of each model in a separate notebook raised the need to store the results in Microsoft Azure Blob Storage, which provided two different outputs:

- **Evaluation metrics:** MAE, RMSE, R^2 and MAPE (refer to **Figure 18**).
- **Predictions for validation sets:** index, location, predicted value and real value for each model, for further integration and comparison (refer to **Figure 19**).

```python
def evaluate_targets(model, X_test, y_test, target_scaler, target_name, target_idx, n_targets):
    print(f"\n----- Evaluating model for: {target_name} ({target_idx + 1}/{n_targets})")

    # Predictions
    y_pred = model.predict(X_test)

    # Verifying if y_test is in normalised scale
    if np.max(y_test) <= 1.0:
        y_true = target_scaler.inverse_transform(y_test.reshape(-1, 1))
    else:
        print("Warning: y_test seems to be scaled. Evaluation metrics could be inconsistent.")
        y_true = y_test.reshape(-1, 1)

    # Descale predictions
    y_pred = target_scaler.inverse_transform(y_pred)
    y_pred = np.round(y_pred).astype(int)

    # Evaluation metrics
    rmse = np.sqrt(np.mean((y_true - y_pred) ** 2))
    mae = np.mean(np.abs(y_true - y_pred))

    # Avoid division by zero in MAPE
    mask = y_true != 0
    if np.any(mask):
        mape = np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100
    else:
        mape = np.nan  # Unable to calculate MAPE

    r2 = r2_score(y_true, y_pred)

    # Results
    print("Evaluation Metrics:")
    print(f" -RMSE : {rmse:.2f}")
    print(f" -MAE  : {mae:.2f}")
    print(f" -MAPE : {mape:.2f}%")
    print(f" -R²   : {r2:.4f}")

    return {
        "RMSE": rmse,
        "MAE": mae,
        "MAPE": mape,
        "R2": r2
    }
```

Figure 18: Evaluation of LSTM time series through metrics



```python
predictions = {}

for target_name in targets:
    print(f"\nPredicting values for: {target_name}")

    # Test data and targets
    X_test, y_test = test_sets[target_name]
    target_scaler = target_scaler_dict[target_name]

    # Scaled predictions
    y_pred_scaled = models[target_name].predict(X_test)

    # Descale predictions to original values
    y_pred = target_scaler.inverse_transform(y_pred_scaled)
    y_true = target_scaler.inverse_transform(y_test.reshape(-1, 1))

    # DATE AND TIME FROM TEST SET

    # Length used in ltsm sequences
    seq_len = sequence_length

    # Index from start of test set to match the target
    n_total = len(y_lstm_dict[target_name])
    n_train = int(n_total * 0.7)
    n_val   = int(n_total * 0.15)
    start_test_idx = seq_len + n_train + n_val

    datest = data["date and time"].iloc[start_test_idx:].reset_index(drop=True)

    # New DataFrame with values needed for comparison
    predsDataFrame = pd.DataFrame({
        "date and time": datest,
        "Real": y_true.flatten(),
        "LSTM_nocoords": y_pred.flatten()
    })

    # Round predictions and avoid negative values
    predsDataFrame["LSTM_nocoords"] = np.round(predsDataFrame["LSTM_nocoords"]).astype(int)
    predsDataFrame["LSTM_nocoords"] = predsDataFrame["LSTM_nocoords"].clip(lower=0)

    predictions[target_name] = predsDataFrame
    print(predsDataFrame.head())

# DataFrame for all predictions
allpreds = pd.concat(
    [dataframe.assign(Target=target) for target, dataframe in predictions.items()],
    ignore_index=True
)
```

Figure 19: Predictions for each target and concatenation into a single DataFrame

Once the evaluation metric values of each model were obtained and stored, these were arranged into a single DataFrame (refer to **Figure 20**). Azure Blob Storage was accessed through Power BI to visualise the performance comparison from the Deep Learning techniques and baselines, and the predicted values were compared against each prediction per model to evaluate predictive power (refer to **Figure 22** and **Figure 22**). All of which assessed the research question: **How can deep learning techniques be effectively leveraged to predict pedestrian and cycle flows on an hourly basis in urban environments for Dublin City by integrating data from multiple sources such as footfall counts, cycle counts, weather conditions, temporal and spatial features?**

```
# Consistency in datatypes
for dataframe in [pred_rf, pred_xgb, pred_lstm_nocoords, pred_lstm_coords, pred_cnnlstm_nocoords, pred_cnnlstm_coords]:
    dataframe["date and time"] = pd.to_datetime(dataframe["date and time"], errors="coerce")

# Merge all data using date and time and Target as join
allpreds = (
    pred_rf
    .merge(pred_xgb, on=["date and time", "Target", "Real"], how="inner")
    .merge(pred_lstm_nocoords, on=["date and time", "Target", "Real"], how="inner")
    .merge(pred_lstm_coords, on=["date and time", "Target", "Real"], how="inner")
    .merge(pred_cnnlstm_nocoords, on=["date and time", "Target", "Real"], how="inner")
    .merge(pred_cnnlstm_coords, on=["date and time", "Target", "Real"], how="inner")
)
```

```
/tmp/ipython-input-3634343238.py:9: UserWarning: You are merging on int and float columns where the float values are not equal to their int representation.
  .merge(pred_lstm_nocoords, on=["date and time", "Target", "Real"], how="inner")
/tmp/ipython-input-3634343238.py:10: UserWarning: You are merging on int and float columns where the float values are not equal to their int representation.
  .merge(pred_lstm_coords, on=["date and time", "Target", "Real"], how="inner")
```

```
print(f"Merged shape: {allpreds.shape}")
allpreds.head()
```

```
Merged shape: (235279, 9)
```

	date and time	Real	RF	Target	XGB	LSTM_nocoords	LSTM_coords	CNNLSTM_nocoords	CNNLSTM_coords
0	2024-03-07 06:00:00	20	15	CYC_Clontarf James Larkin Rd	20	23	27	26	76
1	2024-03-07 07:00:00	49	55	CYC_Clontarf James Larkin Rd	55	42	40	54	77
2	2024-03-07 08:00:00	70	55	CYC_Clontarf James Larkin Rd	75	52	46	73	77
3	2024-03-07 09:00:00	51	55	CYC_Clontarf James Larkin Rd	43	52	44	52	77
4	2024-03-07 10:00:00	31	55	CYC_Clontarf James Larkin Rd	40	45	39	52	76

Figure 20: Real values and predicted values from each model implementation.

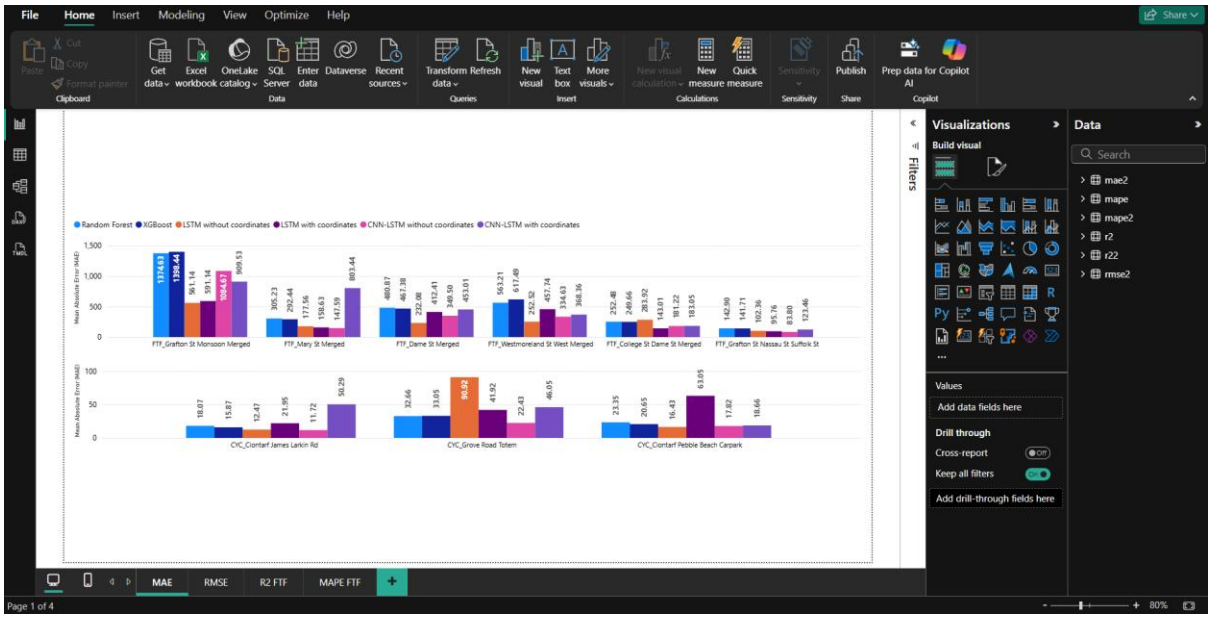


Figure 21: results visualisation using Power BI.

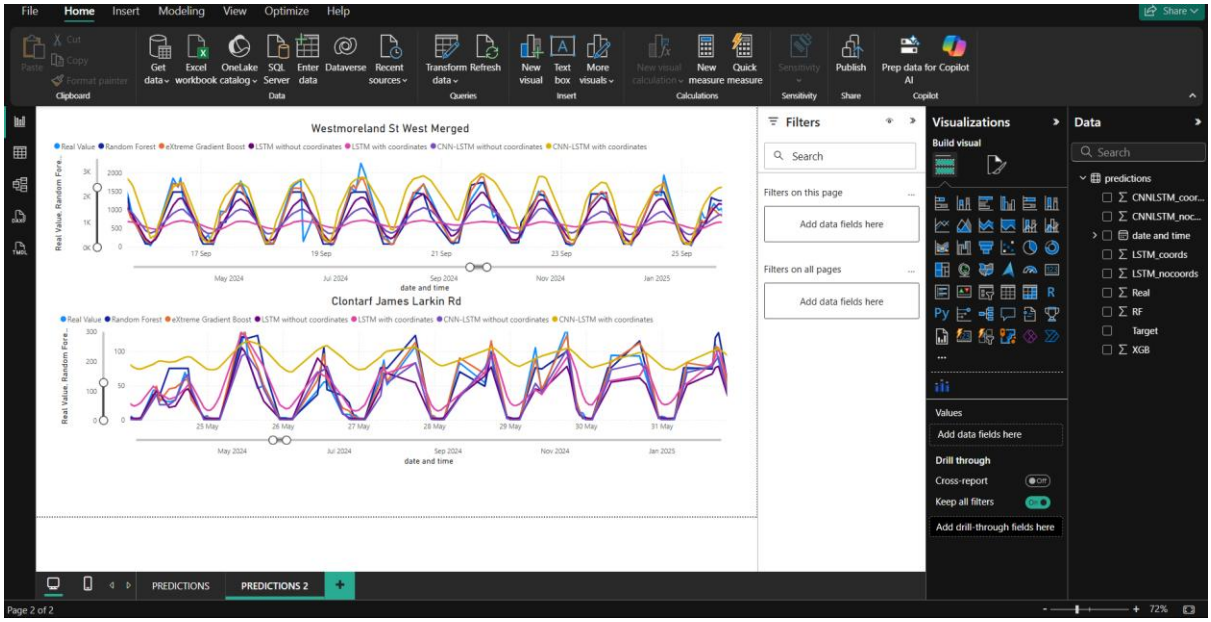


Figure 22: predictions visualisation using Power BI

7 Conclusions

This is a comprehensive and well-structured guide for the configuration and deployment of the project "Leveraging Deep Learning for Pedestrian and Cycle Flow Prediction in Urban Environments" that guarantees the reproducibility and scalability of the prediction pipeline. Data collected from open-source and cloud-based technology, such as Google Colab and Microsoft Azure, combined with a reasonable selection of machine learning (Random Forest, XGBoost) and deep learning models (LSTM, CNN-LSTM) provide reasonable results across diverse scenarios for comparison purposes in the field. Multi-source data integration, pre-processing pipeline and feature engineering allows to capture the temporal, spatial, and meteorological complexity of the research problem.

This report shows how results can be evaluated and presented using metrics typically employed in regression analysis, and visualised in Power BI, promoting critical analysis and decision making. In conclusion, the project implementation, challenges heterogeneous data integration to effectively advance deep learning techniques in answering the research question, providing a novel contribution to previous literature in the field of pedestrian and cyclist flow prediction.

8 References

ACCA Global. (n.d.). *Time series and moving averages*. Retrieved June 8, 2025, from https://www.accaglobal.com/gb/en/student/exam-support-resources/fundamentals-exams-study-resources/f5/technical-articles/time-series.html

Amat Rodrigo, J., & Escobar Ortiz, J. (n.d.). *Cyclical features in time series forecasting*. Retrieved June 2, 2025, from https://skforecast.org/0.9.0/faq/cyclical-features-time-series

Fiveable; Bahr, Becky. (2024). *5.4 Dew point and relative humidity – Meteorology.* Retrieved June 2, 2025, from https://library.fiveable.me/meteorology/unit-5/dew-point-relative-humidity/study-guide/RfxicYTAmshEsWlH

National Center for Hydrology and Meteorology (NCHM). (n.d.). *Rainfall intensity classification.* Retrieved June 2, 2025, from https://www.nchm.gov.bt/attachment/ckfinder/userfiles/files/Rainfall%20intensity%20classification.pdf

National Weather Service. (n.d.). *What is Relative Humidity?* Retrieved June 2, 2025, from https://www.weather.gov/lmk/humidity#:~:text=Relative%20humidity%20(RH)%20