Orchestrating Urban Footfall Prediction: Leveraging AI and batch-oriented workflow for Smart City Application

Tom Komar [1], Philip James [1]

[1] Urban Observatory, School of Engineering, Newcastle University, Newcastle Upon Tyne, NE1 7RU – tom.komar@newcastle.ac.uk

Key Words: smart city, footfall, forecasting, machine learning, artificial intelligence, orchestration

Abstract

This paper explores development and deployment of a smart city prediction system, demonstrating this capability on data generated by footfall counting sensors. Presented approach integrates classical machine learning (ML) techniques with process orchestration framework Apache Airflow. The architecture is designed to handle datasets in periodic batches, ensuring updates are regularly integrated into the prediction system and new predictions are created at every increment. Our work demonstrates ease at which similar systems can be developed, given sufficient volume of data and availability of compute power. This approach highlights that increasing number of smart sensors, availability of proven ML techniques and modern processing frameworks create a critical mass for proliferation of real-time forecasting solutions. Our results indicate that the developed system is effective in predicting footfall patterns, a variable that can be instrumental in applications such as traffic control, resource allocation, public safety, and urban planning. Used methodology is not limited to footfall data, and can be applied to other timeseries datastreams, making it a versatile tool for smart city context. Showcasing practical implementation and benefits of the system, the paper contributes to the ongoing efforts in developing a class of digital urban infrastructure.

1. Introduction

With the term "smart cities" remaining high in popularity (Fig.1), the attempts to make urban spaces optimally designed and managed are important, and forecasting of conditions allows implementation of proactive measures (Kitchin, 2014). For instance, (X. Tao, 2024) demonstrate the potential of ML models in optimising road traffic. Their approach for prediction of urban activity trains on historical data and their research focuses on achieving improvement in predictive powers of applied modelling technique. Such objective leads to development of heavy-weight solutions with high demand for data, high computational complexity, and lack of interoperability (X. Yin, 2022). Thus, in line with advancements in quality of predictions, practitioners encounter growing demands for performant hardware or accessing expensive cloud services. Training cycles of large models are late in incorporating very recent datapoints, and the combination of demand for data with duration of training favours generation of thematic models, untailored to no specific data source. Gap in research of predictive systems with quick model turnaround time, low computational requirement and interoperability has been progressively making wide-scale implementation of such solutions more difficult. To address these issues, the primary objective of this study is to develop a real-time prediction system for footfall timeseries data, collected by Newcastle Urban Observatory (P. James, 2022). Our work presents an application of classical machine learning techniques, orchestrated using Apache Airflow, to aid in real-time prediction of footfall in a city. The system uses pedestrian movement data from AI-powered CCTV cameras installed on lamp posts and overlooking vehicular, cyclists and pedestrian paths. The data is processed as timeseries to train ML models for uncovering likely future patterns in people movement, which are important for effective planning and management of urban infrastructure. In this work, our focus is operationalising the forecasting solution by integrating machine learning with batch-oriented data processing framework. Unlike conventional methods and off-line experiments that rely on static, historical data, our system harnesses live, dynamic and accurate representation of urban movement. The system is online and the results are available publicly.

Utility of real-time predictions reach beyond themes of movement, they are highly relevant to accurate urban weather forecasting (M. Yu, 2021). A highly important area of implementing predictive systems is disease outbreak prevention (Ondrikova, 2021), where timely reaction is of essence.

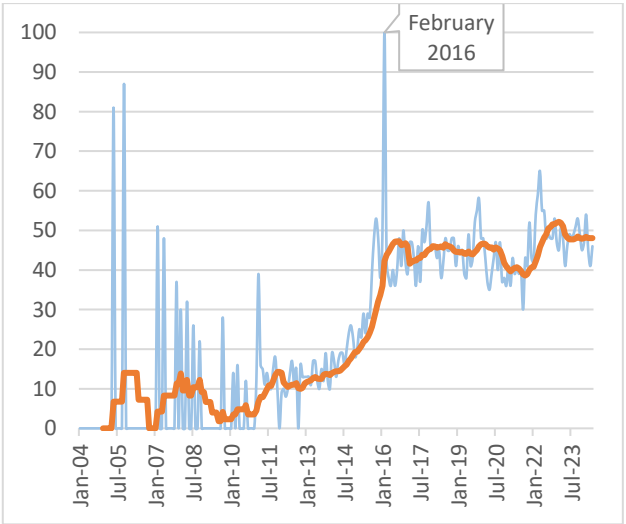

Figure 1. Normalised popularity score (blue) of term "smart city" in google scholar search. Peak popularity was in February 2016, 12 months running average (orange) is high to this day.

This introduction, with included literature review, provides context and highlights importance of real-time prediction systems in smart city environment. The following methodology section explains details of the approach used in this study - covering orchestration of data collection, preprocessing and training of machine learning models. Results section, followed by discussion and recommendations for future work, presents challenges in implementation and operations of the system and an assessment of success criteria fulfilment.

2. Method

Our application postulates the need for the city to be able to forecast pedestrian footfall in a short time window (1 – 24 hours) for a variety of application use cases such as traffic control planning, crowd control, allocating municipal resources like

cleaning services and waste management, enhancing public safety and awareness of expected crowding levels.

This presents a challenge in terms of volume of data and the need for rapid, accurate processing. To address these challenges our application utilises an incremental learning approach, where the prediction models are continuously updated with new data, ensuring that the system remains relevant. Single run of the data preparation process takes a short moment, but with many (186 in our case) unique location-direction combinations being predicted, pre-processing data long in advance of when it is used offers significant time-savings compared with having to extract, transform and load raw data each time a new model is being trained. Training models and storing them for later use in handled prior to requests for their application, immediately after new batch of data is prepared ("pre-processed") and added to the training set. Once all models have been trained, new forecasts are made, saved, and uploaded to the user portal. This makes them available without re-running each time they are needed and they are cached close to where the application serving them runs.

To implement our solution, we rely on end-to-end orchestration framework that simplifies deployment, operations, and monitoring of processing pipelines. Airflow (Airbnb Development Team, 2015) and NiFi (Witt, J., 2006) are popular solutions for finite batch-oriented workloads and real-time event driven data flows, respectively. Airflow offers a more mature support for modern, custom processors written in Python, with little overhead and easier management of workflows. On the other hand, NiFi offered multithreaded and fault-resistant architecture. Evolution of frameworks has made implementing data processing pipelines using these products an attractive alternative to building bespoke solutions developed on building blocks of message queues, key-stores, databases, frontends and backend processes. Airflow's customisability and clearer monitoring (Fig.2) made it our framework of choice for our work at this stage.

Figure 2. Statistics of model training tasks in Airflow dashboard.

End-to-end orchestration frameworks are instrumental for handling interrelated processes involved in analysis of data generated by urban sensors. Application of an orchestration framework allows for coordination of various data processing steps, ensuring they are efficiently managed and, once put in motion, require minimum supervision.

Our methodology uses Directed Acyclic Graphs (DAGs) in Apache Airflow to streamline steps involved in delivery of all tasks. Airflow as a pipeline orchestration framework, assists in scheduling, triggering and monitoring of regular batch processes. Initially, aggregation of sensor data adds together high frequency measurements into larger temporal buckets with lower volatility and less noise. An important element of our methodology involves pre-processing where incomplete timeseries are discarded, dataset is transposed, and additional explanatory features are engineered from timestamps associated with every reading.

datetime	value
2024-05-01 00:00	4
2024-05-01 01:00	2
2024-05-01 02:00	1
2024-05-01 03:00	1
2024-05-01 04:00	1
2024-05-01 05:00	3
2024-05-01 06:00	4
2024-05-01 07:00	14
2024-05-01 08:00	38
2024-05-01 09:00	65
2024-05-01 10:00	109
2024-05-01 11:00	91
2024-05-01 12:00	128
2024-05-01 13:00	146
2024-05-01 14:00	123
2024-05-01 15:00	128
2024-05-01 16:00	112
2024-05-01 17:00	33
2024-05-01 18:00	23
2024-05-01 19:00	17
2024-05-01 20:00	25
2024-05-01 21:00	7
2024-05-01 22:00	15
2024-05-01 23:00	6

datetime	transposed timeseries of preceding values	target value
2024-05-01 09:00	{4, 2, 1, 1, 1, 3, 4, 14, 38}	65
2024-05-01 10:00	{2, 1, 1, 1, 3, 4, 14, 38, 65}	109
2024-05-01 11:00	{1, 1, 1, 3, 4, 14, 38, 65, 109}	91
2024-05-01 12:00	{1, 1, 3, 4, 14, 38, 65, 109, 91}	128
2024-05-01 13:00	{1, 3, 4, 14, 38, 65, 109, 91, 128}	146
2024-05-01 14:00	{3, 4, 14, 38, 65, 109, 91, 128, 146}	123
2024-05-01 15:00	{4, 14, 38, 65, 109, 91, 128, 146, 123}	128
2024-05-01 16:00	{14, 38, 65, 109, 91, 128, 146, 123, 128}	112
2024-05-01 17:00	{38, 65, 109, 91, 128, 146, 123, 128, 112}	33
2024-05-01 18:00	{65, 109, 91, 128, 146, 123, 128, 112, 33}	23
2024-05-01 19:00	{109, 91, 128, 146, 123, 128, 112, 33, 23}	17
2024-05-01 20:00	{91, 128, 146, 123, 128, 112, 33, 23, 17}	25
2024-05-01 21:00	{128, 146, 123, 128, 112, 33, 23, 17, 25}	7
2024-05-01 22:00	{146, 123, 128, 112, 33, 23, 17, 25, 7}	15
2024-05-01 23:00	{123, 128, 112, 33, 23, 17, 25, 7, 15}	6

Figure 3. Transformation of timeseries data into supervised learning format

In the timeseries transformation the dataset is restructured to fit into a supervised learning workflow (Fig.3). To achieve that we apply lag features where target feature at timestep T is accompanied by explanatory features from periods T-1 ... T-8. Lag features are important for timeseries forecasting as they allow learning from multiple past data points to predict future values. By incorporating past readings, the model can learn temporal dependencies and patterns that are inherent in the data. It helps the model understand sequence of events and their development over time, leading to more accurate and reliable predictions. In the case of urban footfall, knowing counts at previous time steps enhances model's ability to predict future footfall.

Lag features are generated by shifting target variable's values backward by an increasing number of steps until reaching specified length lag length (in our case we used 8 past readings).

datetime	to/from12AM	to/from6AM	to/from12PM	to/from6PM
2024-05-01 09:00	9	3	3	9
2024-05-01 10:00	10	4	2	8
2024-05-01 11:00	11	5	1	7
2024-05-01 12:00	12	6	0	6
2024-05-01 13:00	11	7	1	5
2024-05-01 14:00	10	8	2	4
2024-05-01 15:00	9	9	3	3
2024-05-01 16:00	8	10	4	2
2024-05-01 17:00	7	11	5	1
2024-05-01 18:00	6	12	6	0
2024-05-01 19:00	5	11	7	1
2024-05-01 20:00	4	10	8	2
2024-05-01 21:00	3	9	9	3
2024-05-01 22:00	2	8	10	4
2024-05-01 23:00	1	7	11	5

Figure 4. Features engineered from timestamps.

Using raw hour values (ranging from 0 to 23) as explanatory features in machine learning model training can introduce an issue related to cyclical nature of time being represented as a linear value growing from 0 to 23 and then resetting back to 0. This reset creates discontinuity in what in fact is a gradually changing variable. For example, hours 23 (11PM) and 2 (2AM) are only 3 hours apart, however, if used as raw values, the algorithm reads them as being 21 units apart. To address the issue of discontinuity in representing time, we engineer the raw hour value into several "distance from set hour" features, so that the values used for time are transitioning in a smooth and cyclical manner (Fig.4).

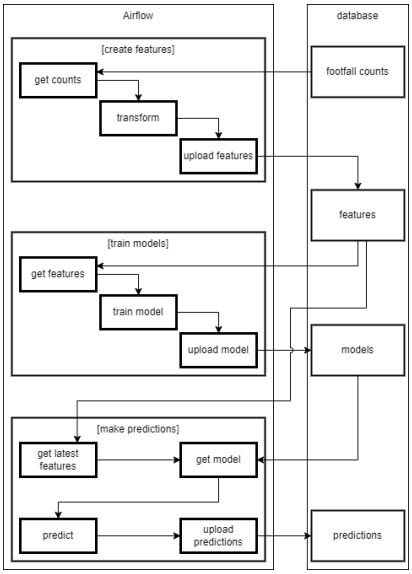

Figure 5. Architecture of the prediction system.

Following these pre-processing steps, three ML regressors (Support Vector, Random Forest, Gradient Boosting) are trained using cross-validation (Sci-kit Development Team, 2007). Evaluation is done using Mean Average Percentage Error. In order to deliver predictions beyond the next hour, predictions are made for the following hours, by adding the prediction to the predictors at each step. The process is delivered every hour, with new training features, new models and new predictions added to a database at the same frequency.

Presented approach (Fig.5) leverages machine learning models and continuous learning technique to ensure quick turnaround and adaptability. Use of cross-validation ensures selection of the best performing model. By automatically updating all the individual models with new data, the system remains responsive to changing patterns and trends in the dynamic urban environment they operate it.

3. Results

Application of classical ML models provided us with computational efficiency, interoperability, and interpretability of the model. Scaling the solution across multiple sites, reliance on small amounts of data from some newly deployed sensors and operating with limited computational resources pointed towards utilisation of these classical ML methods. The solution does not require access to GPU and has been tested on hardware ranging from desktop grade machine, high-end PC, and server machines.

Leveraging Airflow for orchestration provided an out-of-the-box solution for version control, scheduling, monitoring, and alerting. The use of a database for storing code, source data, restructured features, models and predictions provides a unified, structured and open data-engine, available for integration with other systems. Airflow can fetch new versions of jobs code (DAG) and automatically deploy it. This approach replaces local files most up to date version from the database. We also tested python library "importlib" which allows dynamically importing new functions during application runtime. This way the process, as defined by DAG, remains the same and it is the tools (functions) that are downloaded from the database. We have settled for the all-in-one solution (formerly explained fetching of most up-to-date DAGs). This way we limit frequency of database connections only to the times when new version become available, unlike with importing functions which requires instantiating database connection on every run of each job.

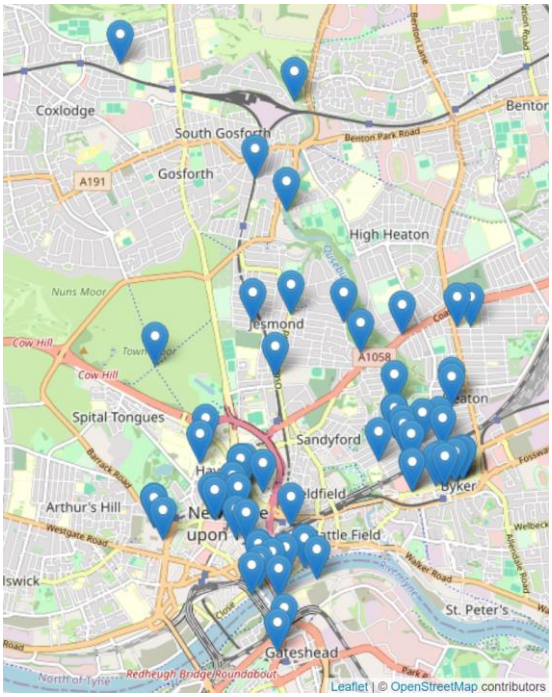

Figure 6. Locations of Newcastle Urban Observatory footfall sensors.

In the study, the pedestrian movement data comes from computer vision traffic sensors distributed across a city in 74 locations (Fig.6). These devices are detecting vehicles, bicycles, and pedestrians, tracking their trajectories, and verifying if they cross a virtual line placed within cameras field of view (Fig.7). Hardware used in the sensors is a combination of a microcomputer (Raspberry Pi) and an AI accelerator (Google Coral TPU, offering 4 trillion operations per second) (K. Seshadri, 2022). The object detection model is Mobilenet V2 (M. Sandler, 2018) and tracking is done using SORT method (A. Bewley, 2016). The devices communicate with an online API to submit all individual datapoints, representing line crossing events, collected over previous 5 minutes. The devices are centrally managed using Balena IoT platform (R. Botez, 2020). This platform provides a unified mechanism for commissioning of new devices, deployment of applications, management of environmental variables and many other operations on individual devices and on whole fleet level.

Figure 7. Example camera view of a sensor with a virtual count-line displayed in orange.

If considering the count to be a feature of a pedestrian movement network, it can be treated as a point measurement. Contrary to movement of vehicular traffic, the wide range of pedestrian activity at micro level with multiple input/output points on adjacent buildings and dwell areas on the street, the footfall count gathered in this manner is not equal, but related to the flow over a street segment. At the time of writing, in 26 of these locations the flow of people was too low to adequately train models and were excluded, but that number can vary every time the process runs (i.e. every hour). The sensors provide data at event-in-time granularity that is aggregated into 5-minute bins, then 15-minute, 1-hour and daily periods (Fig.8). Most of these sensors are in operation for over 12 months, offering clarity on daily and weekly flows, and a glimpse of seasonal dynamics. Others are fresh to the field, with limited amounts of data available for training.

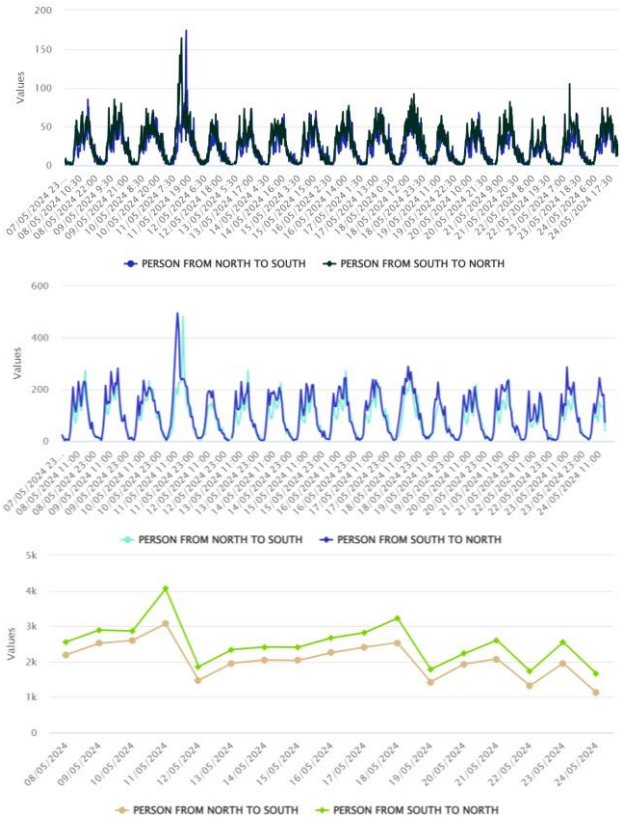

Figure 8. 15-minute (top), hourly (middle) and daily (bottom) ebbs and flows of footfall counts at one busy location.

In the pre-processing step involving reshaping of the timeseries dataset into supervised learning structure, value from row T was used as the target value (prediction - y) and 8 preceding values were used as explanatory features (predictors - x). If a preceding value was missing, generation of this training item was skipped. Additionally, feature engineering added more explanatory features including one-hot-encoded day of week (single categorical column is divided into 7 numerical columns, one for each day), binary feature for weekends, distance-from-set-time values for time of day, and binary feature for time adjustment ("daylight saving time"). Such preprocessing gives the model understanding not only of how the timeseries developed in short time-period prior to the prediction, but also extra information that lets the model embed this knowledge in a temporal context.

Use of 8 preceding values as explanatory features might have put too little emphasis on most recent readings, potentially causing the models to unnecessarily account for less relevant values from the past.

4. Discussion

Implementation and use of such system does not come without challenges. Relying solely on a relational database for storage, came with its performance limitations meaning that large model files were slow to upload and download, occasionally leading to application of not the most recent models. To address this limitation, we have migrated the database to a server machine on the same physical network as the Airflow instance.

Locations with pedestrian volumes regularly near the minimum threshold of 100 people in the previous 24 hours, resulted in benefits from training on values where regularity can be observed, but suffered from lack of exposure to relatively low footfall periods, leading to their predictions being a combination of values weakly supported by the training data and, effectively, inferencing on values not represented in the training set. We are yet to develop solutions addressing this issue.

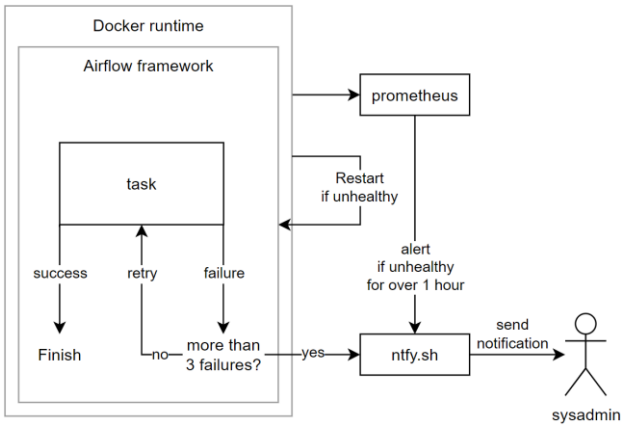

Figure 9. Process of generating alerts

Alerting built into Apache Airflow generates notifications delivered by ntfy.sh online service (Heckel P., 2021) which prompts the system administrator if any tasks of the pipelines cannot be completed, despite delayed retries. However, the framework itself is made of several interdependent micro-services (schedule, trigger, worker, tasks database, web interface) which may also fail. In such case the system is yielding no alerts and a higher-level watchdog service becomes necessary. To this end we took advantage of Airflow containers health check feature reported on host system, which restarts the service if "Unhealthy" status is listed. Also, all services running at Newcastle Urban Observatory are monitored using Prometheus event monitoring (SoundCloud Development Team, 2012) with separate alerting mechanism (Fig.9). Since deployment in late October 2023 the system remained largely uninterrupted, continuously training, and generating forecasts.

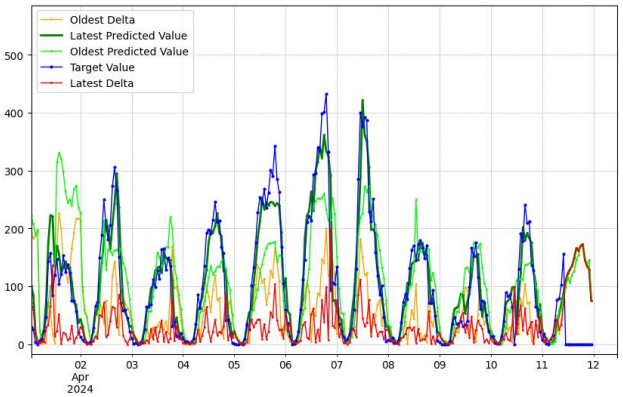

Figure 10. Real values compared with predictions from earliest and latest models show decrease in error (from orange to red line) and part of the value that remains unpredicted (divergence of blue line from dark green).

Although our work was concerned with operationalizing a forecasting solution, quality of the results has been observed throughout the whole time the system is live (Fig.10). Some locations had footfall fluctuations in the range of the threshold set for sufficient volume to be included in the training set. Such instances made these locations drop in and out from the training regime, effectively making their models only useful for the above threshold periods, as they have never been trained on below threshold data. Another implication of using locations with low footfall is the susceptibility of evaluation metrics to smallest inaccuracies in predictions – with a minimum volume of only 100 people in the previous 24 required to include a timeseries in the process, it meant that there are times when the hourly count can be zero or not much higher. Due to these not so rare cases, the MAPE score appears evidently higher in low footfall locations and during low footfall periods. Potential impact of the issue is that new models with better predictive power for the more important high footfall periods but worse for the less important low footfall periods, could have been discarded by the system if the overall MAPE score was lower than currently deployed models. A solution for the problem that we are considering is by applying a weight to individual precision errors prior to calculating the overall mean value.

The system performs well both in terms of predicting normally expected values and serving as an alert system when actual values divert from forecasts. Since the predictions are made at least one hour into the future, and new set of predictions is available every hour, that effectively provides multiple reference values and trends for user's consideration.

5. Conclusions

The deployment of footfall predictions aims to enhance the real-time, data-driven governance of the city, contributing to creation of smart digital infrastructure and more responsive city management. Presented work has demonstrated that creating a prediction system by orchestration of classical machine learning methods using off the shelve components is straight forward to implement. These capabilities are particularly relevant for traffic operations and retail, where predicting footfall can improve safety, comfort, and enhance economic performance. Applied approach enabled continuous refinement of the system and delivery of up-to-date forecasts, ensuring that users have access to the latest estimates of future values.

The usefulness of presented approach extend beyond traffic operations and retail. Applying similar solution to other timeseries datastreams can support a wider range of urban management activities, from emergency response to public health monitoring. Creation of a digital-twin-like para-simulation environment could revolutionise how cities are managed.

6. Future work

Areas requiring improvement include issues highlighted in the results sections include verifying how short a period of preceding values can be without detrimental effect on quality of predictions, prototyping solutions addressing issue with locations where counts oscillate around the threshold value, introducing a model quality metric that puts emphasis on prediction of counts for busy periods.

Avenues for future research that we are considering involve deployment on edge devices performing data collection. That would solve the issue of scaling the application across severs, and accelerate prediction delivery time from doing them at regular interval to generating them at every new data point in real-time.

Another direction to explore is integrating timeseries representing conditions related to the predicted parameter, similarly to solution developed by (Makkar G., 2019) for the retail context. This way the system could gain accuracy by considering signals in other timeseries. This can involve e.g. weather data (different theme) as well as coming from other sensors collecting similar data in proximity, expecting geographical auto-correlation.

Acknowledgements

We are grateful for the support this research has received from UK Research and Innovation in partnership with Natural Environment Research Council under "Digital Solutions" programme initiative.

This research utilises datastreams captured and provided by Newcastle University Urban Observatory. Their data collection efforts and access to real-time data have been instrumental in development of our prediction system.

References

Beauchemin M., Airbnb Development Team, Apache Software Foundation, 2015: Airflow Software

Cournapeau D., Scikit-learn Development Team, 2007: Scikit-learn Library for Machine Learning

Haines, S., 2022: Workflow Orchestration with Apache Airflow. *Modern Data Engineering with Apache Spark*. Apress, Berkeley, CA. doi.org/10.1007/978-1-4842-7452-1_8

Asher, M., Oswald, Y., and Malleson, N., 2023: Predicting Pedestrian Counts using Machine Learning. *AGILE GIScience Ser.*, 4, 18, doi.org/10.5194/agile-giss-4-18-2023

Pedregosa et al., 2011: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, p.2825-2830, doi.org/10.48550/arXiv.1201.0490

Cohen A., Dalyot S., Natapov A., 2021: Machine Learning for Predicting Pedestrian Activity Levels in Cities, *Proceedings of the 16th International Conference on Location Based Services*, doi.org/10.34726/1758

Kitchin, R., 2014: The real-time city? Big data and smart urbanism. *GeoJournal* 79, 1–14 (2014). https://doi.org/10.1007/s10708-013-9516-8

Tao X, Cheng L, Zhang R, Chan WK, Chao H, Qin J., 2024: Towards Green Innovation in Smart Cities: Leveraging Traffic Flow Prediction with Machine Learning Algorithms for Sustainable Transportation Systems. *Sustainability*. 2024; 16(1):251. https://doi.org/10.3390/su16010251

P James, J Jonczyk, L Smith, N Harris, T Komar, D Bell, 2022: Realizing smart city infrastructure at scale, in the wild: A case study. *Frontiers in Sustainable Cities*, 2022

X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi and B. Yin, 2022: Deep Learning on Traffic Prediction: Methods, Analysis, and Future Directions, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4927-4943, June 2022, doi: 10.1109/TITS.2021.3054840

M. Yu, F. Xu, W. Hu, J. Sun and G. Cervone, 2021: Using Long Short-Term Memory (LSTM) and Internet of Things (IoT) for Localized Surface Temperature Forecasting in an Urban Environment, *IEEE Access*, vol. 9, pp. 137406-137418, 2021, doi: 10.1109/ACCESS.2021.3116809

Ondrikova N, Harris J, Douglas A, Hughes H, Iturriza-Gomara M, Vivancos R, Elliot A, Cunliffe N, Clough H, 2023: Predicting Norovirus in England Using Existing and Emerging Syndromic Data: Infodemiology Study, *J Med Internet Res* 2023;25:e37540

Witt J., US National Security Agency, Apache Software Foundation, 2006: NiFi Software

A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, 2016: Simple online and realtime tracking, *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA, pp. 3464-3468, doi: 10.1109/ICIP.2016.7533003

K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami and A. Yazdanbakhsh, 2022: An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks, *2022 IEEE International Symposium on Workload Characterization (IISWC)*, Austin, TX, USA, pp. 79-91, doi: 10.1109/IISWC55918.2022.00017

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, 2018: MobileNetV2: Inverted Residuals and Linear Bottlenecks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, pp. 4510-4520. doi: 10.1109/CVPR.2018.0047

R. Botez, V. Strautiu, I. -A. Ivanciu and V. Dobrota, 2020: Containerized Application for IoT Devices: Comparison between balenaCloud and Amazon Web Services Approaches, *2020 International Symposium on Electronics and Telecommunications (ISETC)*, Timisoara, Romania, pp. 1-4, doi: 10.1109/ISETC50328.2020.9301070

Heckel P., 2021: Ntfy.sh Software

SoundCloud Development Team, 2021: Prometheus Software

Makkar, G., 2020: Real-Time Footfall Prediction Using Weather Data: A Case on Retail Analytics. *Sharma, N., Chakrabarti, A., Balas, V. (eds) Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing*, vol 1042. Springer, Singapore. doi:10.1007/978-981-32-9949-8_37