University of Glasgow | School of Computing Science

Multivariate time-series anomaly detection applied to key business data

Patrick Menlove (2250066M)

April 7, 2020

ABSTRACT

Time-series data is incredibly valuable to businesses for making data-driven decisions, and it is therefore very important that it be of high quality and that irregularities or anomalies in the data are identified and dealt with. Anomaly detection techniques can be used to identify anomalous patterns in the data, allowing human operators to correct errors or respond to anomalous market activities or threats. In this work, the application of state-of-the-art anomaly detection methods to large-scale time-series business data is analysed, and a design for such an anomaly detection system is proposed. The performance of each anomaly detection method is independently evaluated on various sample datasets. The solution and recommendations draw upon existing research in the field of anomaly detection as well as from the organisational landscape of a large internet-economy company.

1. INTRODUCTION

Nowadays, organisations invest heavily in the gathering and processing of time-series data; to allow them to analyse their Key Performance Indicators (KPIs), as well as to monitor production systems to ensure quality of service. Business decisions are made based on this data, and thus the accuracy and quality of the data are of the utmost importance. Anomalies or irregularities present in the data, particularly those which are the result of a system or programming error, have the potential to skew the insights derived from the data, leading to sub-optimal, inefficient decision-making. In addition, non-erroneous, market anomalies may be present in the data and should be identified quickly to aid the organisation's response. To this end, there are several techniques from the field of anomaly detection that can be applied.

This work explores the application of anomaly detection techniques within the context of an internet economy company. The work was undertaken in partnership with Skyscanner Ltd, a travel search aggregation company with over 100 million monthly users. Skyscanner is a data-driven company that invests heavily in collection and analysis of key business data, for all of the reasons outlined and more.

The aim of this work is to determine the feasibility of building a large-scale anomaly detection system within the context of Skyscanner, and make recommendations on both the system's design and the optimal anomaly detection model(s) for this system to use. Specifically, the contributions of this work are:

- A design for a large-scale multivariate time-seires anomaly detection system is developed, containing recommendations for practical application, specifically aimed at application within Skyscanner, but also with the potential to be applied to other contexts. (Section 4)
- The Skyscanner redirects dataset is analysed, and representative sample datasets for evaluation are identified and chosen. (Section 3)
- The LSTM+STL model is recommended as the model to use, after evaluating various models for their applicability to the designed system. (Sections 5 and 8)

2. BACKGROUND

2.1 Anomaly Detection

2.1.1 Problem Definition

In anomaly detection, the aim is to classify some piece of data as being "normal" or anomalous. In this work, the focus is on time-series anomaly detection, which is simply anomaly detection applied to a time-series dataset - a collection of datapoints ordered by time, with the important property that the ordering must be maintained.

We can describe a time-series anomaly detection system as a function $A(x_t)$ which outputs whether the observed data $x_t \in \mathbb{R}^n$ is anomalous or not (where the n in $\mathbb{R}^n$ is the number of features or different series at each timestep t). The function can be described in terms of a function $P(x) \in [0,1]$ which outputs the probability of observation x being anomalous (the *anomaly score*), and a probability threshold $\epsilon \in [0,1]$.

$$A(x_t) = \begin{cases} P(x_t) \geq \epsilon & \rightarrow \text{anomalous} \\ P(x_t) < \epsilon & \rightarrow \text{normal} \end{cases}$$

If the anomaly score for a given timestep t with observations x_t is less than the threshold ϵ, then the timestep is classified as normal, otherwise, it is classified as anomalous.

Therefore, in order to accurately detect anomalies, we have to learn the function $P(x)$ and choose an appropriate value for the threshold ϵ.

Above, $x \in \mathbb{R}^n$ implies that all inputs are real-valued, but the data can also be categorical if a one-hot encoding is used, which would make the inputs in $\mathbb{R}^n$ as $0, 1 \in \mathbb{R}^n$.

2.1.2 Types of Anomalies

There are many types or categories of anomalies and they are defined in different ways in the literature.

Chalapathy and Chawla (2019) break anomalies down into three different types:

1. Point Anomalies - irregularities or deviations that happen randomly and may have no particular interpretation
2. Contextual Anomalies - points that could be considered anomalous in some specific contexts, but not in others.
3. Collective/Group Anomalies - where each of the individual data points in a group appear to be normal in isolation, but when considered as a group, they are anomalous

An alternative view of anomalies is given by Goldstein and Uchida (2016), who categorise anomalies into "global" and "local" anomalies. Global anomalies are those which can be identified as anomalous as they are examples whose features are significantly different from the dense areas of the dataset. Local anomalies only appear anomalous when they are compared with their close-by neighbourhood. These are analogous to contextual or collective anomalies in Chalapathy and Chawla (2019).

There is a consensus that there are very obvious (point/-global) anomalies that can be detected easily, and there are more subtle (contextual/group/local/collective) anomalies that require more advanced techniques to identify correctly.

Another way of categorising or describing anomalies is using autoencoders applied to anomaly detection. Autoencoders are neural networks which learn a reduced-dimensional representation of data and are able to re-construct higher-dimensional data given a lower-dimensional representation, along with some "residual error".

In relation to autoencoders, Zong et al. (2018) state that anomalies differ from normal samples in two aspects: (1) anomalies can be significantly deviated in the reduced dimensions where their features are correlated in a different way; and (2) anomalies are harder to reconstruct, compared with normal samples. This forms the basic intuition of reconstruction-based methods of anomaly detection, where the reconstruction error from an autoencoder is used as an anomaly score. An example of such a model can be seen in Malhotra et al. (2016).

2.1.3 Prediction Models

In this work, the majority of models considered are prediction models, that use a two-step process to classify points as anomalous or normal. Firstly, the model is trained on past observations, and the learned model parameters are used to output a predicted value for each training observation. The squared errors of the predicted values and actual values are calculated, and a normal distribution is fit over them.

It is important to note that, whilst the prediction step is multivariate, the normal distribution fitting is done separately for each time-series, that way separate anomaly scores can be generated on a per-time-series basis and aid in root cause identification.

Using the learned model and distribution parameters, the model can be evaluated on unseen data to output a prediction, and the squared errors between the predicted and real values obtained. The Cumulative Distribution Function (CDF) of the distribution is then calculated for each of the squared error values, and the output of the CDF (a probability, $p \in [0, 1]$) is used as the *anomaly score* $P(x)$ referred to in the problem definition (Section 2.1.1).

This method relies upon using squared or absolute errors, due to the nature of the cumulative distribution function being defined as the probability of an observation taking on a value less than or equal to a particular threshold.

$$CDF(x) = \int_{-\infty}^{x} PDF(t)\, dt$$

One drawback of using absolute or squared errors is that directionality in anomaly detection is lost - for example, if errors skewed more positively or negatively this would not be captured in the fitted distribution, which could lead to misclassification of data points.

2.2 Challenges in time-series anomaly detection

2.2.1 Temporal dependency between timesteps

Time-series fundamentally have a temporal dependency between each timestep. A time-series cannot be re-ordered or shuffled. As such, this presents a problem for many machine learning algorithms, which rely on datapoints being independent and identically distributed (i.i.d.), or being able to "shuffle" examples in order to improve their training.

This nature of time series' makes distance or clustering based methods of anomaly detection less effective on time-series data (Zhang et al., 2019). It is more effective to use recurrent neural networks (RNNs) or Long-Short-Term Memory (LSTM) networks which are designed to work on time-series data (Zhang et al., 2019).

2.2.2 Unlabelled Datasets

In most real-world time-series datasets, labels depicting whether a given time-step is anomalous or not are not typically available. It may also be impractical to create these labels by manual inspection due to the size or complexity of the dataset. Therefore, to classify a timestep as anomalous or not, simple supervised learning more difficult to achieve.

As seen in Section 2.1.1, we can model a function to give the probability of a given timestep's observations being anomalous. This function can be learned via unsupervised or semi-supervised techniques, as described by Goldstein and Uchida (2016).

Unsupervised anomaly detection techniques focus mainly on clustering or distance measurement from "normal" values, for example, k-Nearest-Neighbours. However, as explained in Section 2.2.1, these techniques fall short due to their assumption of the data being i.i.d.

The ideal solution may lie in semi-supervised anomaly detection, where a model is trained on training data that only contains "normal" data without any anomalies, and anomalies can be detected as observations that deviate from that model (Goldstein and Uchida, 2016).

Ren et al. (2019) faced this issue in building Microsoft's anomaly detection system, and forced the solution to adopt an unsupervised (but, upon further inspection, fitting the definition of semi-supervised) approach.

Regardless of which method is chosen, there remains a fundamental problem of evaluation of the learned model. In order to compute the confusion matrix to determine the model's performance, a labelled dataset is required. Several approaches can be taken here:

1. **Creation of labels by manual inspection**

This involves visualising the data and manually labelling timesteps as anomalous or not. This technique is error prone and it implicitly assumes that all anomalies are easily detectable by a human, which may not be the case. This technique may also be impractical for very large datasets.

2. **Creation of labels from other sources**
   There may be other time-series data available which can give information about a subset of the possible anomalies. For example, historical recorded incidents would have start and end timestamps, which could be used to mark timesteps as anomalous or not. The drawback of this method is that there may not be a full coverage if only the most significant anomalies that had major impact are considered. This method would also be unable to pinpoint which time-series is anomalous (root cause identification) as it would consider the entire timestep anomalous.

3. **Synthetic anomaly generation**
   Synthetic anomalies can be introduced to the dataset and the performance of the model evaluated on the modified dataset. However, the synthetic anomalies may not be representative of the actual anomalies in the dataset. Furthermore, there may already exist anomalies in the dataset which are not labelled as such, which the model would learn as normal values, reducing the predictive ability of the model.

4. **Model evaluation on sample datasets**
   In this approach, rather than trying to make labels for the intended application's dataset, labelled open datasets are used to evaluate the model, in order to have a predictive indication of its performance on the intended dataset. The performance of the model can then be evaluated and constantly improved "online" after being deployed for production use on the target dataset, and false-positives can be flagged as such by collecting this data from operators when alerts are triggered from the system.

   Perhaps the most important consideration with this approach is that the sample datasets must be representative enough of the intended target dataset in order to provide a fair comparison of performance. If the sample dataset is nothing like the intended target dataset, we can infer very little about a model's performance on the target dataset from the model's performance on the sample dataset.

2.2.3 Incorrectly labelled datasets

Given the nature of real-world time series discussed in Section 2.2.2, it is to be expected that, in datasets where the labels are manually created or inferred, there will be some wrongly-labelled examples. Thus, one of the criterion for selecting a good model to use is its robustness to such contaminations of its training data.

This can be incorporated into the evaluation of models on sample datasets, by repeating the evaluation criterion for varying levels of contamination. At each level, a given proportion of the dataset can be contaminated with synthetic anomalies, which are labelled as normal datapoints, in order to mimic incorrectly labelled anomalies in the target dataset(s). This approach is taken in Zong et al. (2018), with contaminations ranging from 0% to 5% of training data.

2.2.4 Creating meaningful alerts

Whilst anomalies can be easily identified, in order for this to be useful, an anomaly detection system must be able to create meaningful alerts about the anomalies it identifies to notify the human operators that would be best informed of them.

This would make the usability of the system very dependant on its false-positive rate. If the number of false-positive anomalies that the system detects is too high, humans will simply begin to ignore the alerts and the model will be deemed as "crying wolf" too often.

This requirement needs to be traded off against the will to identify more subtle anomalies, more difficult for humans to detect, at the risk of the anomaly potentially being a false-positive.

This can be expressed more formally as the tradeoff between precision and recall. Precision and recall can be expressed as follows:

$$Precision = \frac{true\,positives}{true\,positives + false\,positives},$$

$$Recall = \frac{true\,positives}{true\,positives + false\,negatives}.$$

A model biased towards precision would result in a greater number of the anomalies it identified actually being anomalous, at the expense of missing some anomalies which the classifier deems to have less probability of being anomalous.

On the other hand, a model biased towards recall would ensure that more potential anomalies are flagged at the risk of "crying wolf" by raising alerts when there is no anomaly present in the data.

Given the need to balance between precision and recall, it is sensible to benchmark detected anomalies based on the F_1 score (or, more generally, the F_β score or F-measure).

The F_1 score is the harmonic mean of precision and recall. However, more generally it can be written as F_β, where recall is weighted more than precision by a factor of β. With $\beta = 1$, precision and recall are weighted equally. $\beta < 1$ would give greater importance to precision and $\beta > 1$ would give greater importance to recall.

The full definition of F_β is as follows (Sasaki, 2007):

$$F_\beta = \frac{(\beta^2 + 1) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}.$$

It is important to note that in order to evaluate models using precision, recall, or F_β score, one must compute the confusion matrix (a matrix of true positives, false positives, true negatives, false negatives). In order to do this, labelled data is required, and so the caveats and issues identified in Section 2.2.2 apply.

2.2.5 Scale and the curse of dimensionality

There is a significant problem with large scale, heterogeneous data, which is that it is usually (and definitely in the case of Skyscanner's redirects - see Section 3.1) highly-dimensional. Highly dimensional data allows many features to be captured and therefore can allow more accurate modelling of normal and anomalous patterns. However, with

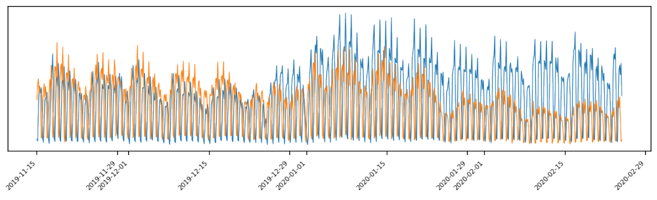

FIGURE 1: Skyscanner - Early effects of COVID-19 on travel in Asia

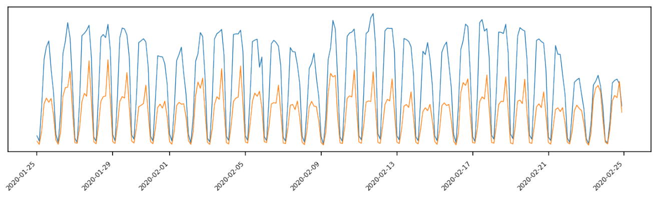

FIGURE 2: Skyscanner - Early effects of COVID-19 on travel to Italy

higher dimensional data, each input sample has a higher probability of being considered anomalous (Zong et al., 2018). The types of models that suffer most from this are density estimation methods, where probability distributions are learned from the raw data. The "curse of dimensionality" motivates the use of dimensionality reduction techniques like Principal Component Analysis (PCA) or autoencoders (Zong et al., 2018).

2.2.6 Heteroskedasticity

In time-series business data, especially data of financial markets or consumer behaviour, the variance in a given subset of the dataset can be very different to that in other subsets of the dataset. The variance of a subset is highly indicative of how anomalous points within this subset are. Hamilton (1994) states that there may be an interest in forecasting not only the level or value of the series, but also its variance. Hamilton (1994) also states that a variance that changes over time has implications for the validity and efficiency of a model.

3. DATASETS

3.1 Skyscanner Dataset(s)

This research focused on the application of anomaly detection to Skyscanner's "redirects" dataset, which records when users redirect to partner websites, and is a way of gauging demand in the market.

The raw dataset is a large, heterogeneous dataset of individual events. In order to extract meaningful information from it, it has to be integrated and summarised over time windows. Given the redirects dataset's scale, this transformation was applied using an ETL job developed with Apache Spark.

The redirects dataset is highly periodic, and has several trends that appear. The types of anomalies that can be found in the dataset are mainly contextual anomalies, that are subtle deviations from normal.

It should be noted that more obvious group or point anomalies may occur in the data, but these are mostly due to errors and are usually "backfilled" to correct them, so it is difficult to get an example of these kinds of anomalies in hindsight.

In Figures 1 and 2, the y-axis values have been redacted so as to conceal commercially sensitive information, however they are both on the same scale and represent the same metric for users in different markets. These particular series were chosen as one series is nominal and one series is anomalous in each figure, and they provide a fair comparison to identify an anomaly in context. In Figure 1, one can clearly detect that the orange line is anomalous as it falls significantly below the trend in the right half of the figure. In Figure 2, the blue line experiences a significant anomalous drop at the end. Both Figure 1 and 2 are examples of contextual anomalies. It could be argued that there is a point which is anomalous in Figure 2 and therefore it is a point anomaly, but the subsequent points are also anomalous given the context of the previous values of the series.

Figure 1 is also an example of *heteroskedasticity*, where in approximately the last third of the graph, the variance in the data is reduced, and this becomes the "new normal". There is a danger of misclassifying this "new normal" as normal despite it being anomalous. The graphed anomaly is long in duration, and so this is a potentially contentious decision for manual labelling - it is not clear if the entire second half of the graph should be labelled as anomalous, or just the two main points where the anomaly manifests itself in a change in peak values and variance.

3.2 Sample Datasets

In order to reason about any model's performance, this work uses publicly-available sample datasets with labelled anomalies in order to benchmark models. Several datasets were considered, with the goal of being as representative of the Skyscanner dataset as possible.

3.2.1 Yahoo A1

The Yahoo (Yahoo) A1 benchmark dataset contains extracts from real Yahoo production data (obfuscated to conceal its true representation) which, after some transformation, can be categorised as a multivariate time-series dataset, consisting of 65 series and 1400 datapoints. The data is a real-world dataset, therefore it is representative of the Skyscanner data with intended application.

Its anomaly labels are also very accurate and tightly-fit to the anomalous points. It contains examples of all the types of anomalies described in Section 2.1.2 - point (Figure 3), group (Figure 4) and contextual (Figure 5). The dataset also has examples of different patterns, such as periodic, static, heteroskedastic and noisy. An example of a periodic pattern can be seen in Figure 6, which is similar to the shape of the Skyscanner redirects data. All of these criteria make the dataset very suitable to the problem at hand.

In Figures 3, 4, 5 and 6, it is not known precisely what the y-axis represents, all that is known is that these are extracts of real Yahoo production data. The y-axis values are simply denoted as "value" in the dataset. Presumably the true units have been concealed to not reveal commercially sensitive data.

The x-axis of Figures 3, 4, 5 and 6 is given as an integer

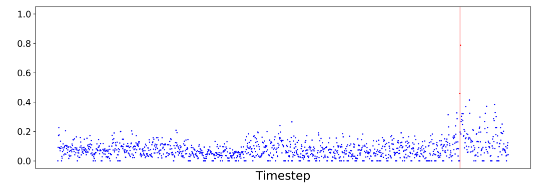

FIGURE 3: Yahoo A1 - Series 1 - Point anomaly

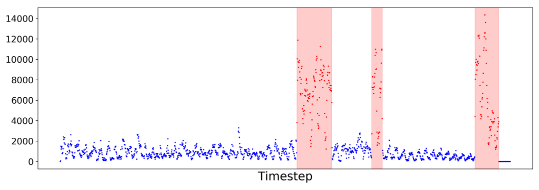

FIGURE 4: Yahoo A1 - Series 17 - Group anomalies

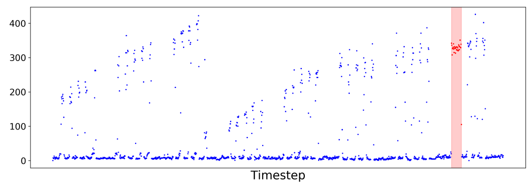

FIGURE 5: Yahoo A1 - Series 37 - Contextual anomaly

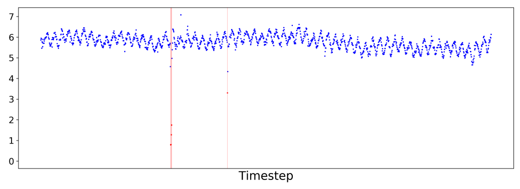

FIGURE 6: Yahoo A1 - Series 36 - Periodic Pattern

representing the time-step, starting from 1. Given the lack of clarity on the units, some labels and ticks were omitted as they did not add useful information to the figures.

The only major issue with the Yahoo dataset is that the number of datapoints (1400) is relatively low for a machine learning model to be effective - this motivated the dataset being upsampled and interpolated in order to create more training examples.

3.2.2 *Numenta Anomaly Benchmark (NAB)*

The Numenta Anomaly Benchmark (Numenta) or NAB provides several time-series datasets, and also provides a framework for evaluating models on these datasets.

The "tweets" dataset is comprised of the volume of tweets relating to particular stock market stocks. This is a multivariate time-series with roughly 15,000 datapoints and 10 individual time-series. It is also seasonal and periodic, and as such is a suitable dataset for benchmarking on, as it matches the criteria of the Skyscanner data.

The NAB Tweets dataset mainly has examples of point

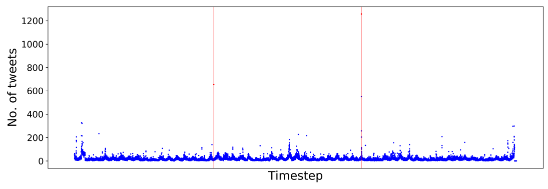

FIGURE 7: NAB Tweets - FB - Point anomalies

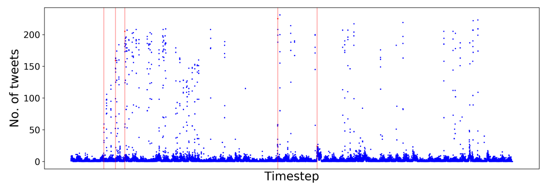

FIGURE 8: NAB Tweets - UPS - Contextual anomalies

anomalies (like in Figure 7), but it also does not label seemingly anomalous periods occurring directly after a first "spike" as anomalous, given that the anomalous spike has set a new "normal", and so the dataset considers a contextual "normal" rather than contextual anomalies directly. An example of this can be seen in Figure 8.

Similar to the *Yahoo A1* dataset, the NAB Tweets dataset also benefited from upsampling and interpolation, though with 15000 datapoints it was not as significant that this be done.

The other datasets in the NAB collection are either univariate or not representative of the Skyscanner dataset for envisaged application.

4. THE SYSTEM

4.1 Overview

Though the work described in this paper did not include the creation of a production-ready large-scale anomaly detection system, its aim is to determine the feasibility of building such a system and to make recommendations as to how such a system could be implemented, specifically in the context of Skyscanner. This section details these recommendations.

4.2 Constraints

4.2.1 *Scale*

Given the title "large-scale anomaly detection system", it is clear that the scale of the datasets the system must be able to run on is significant.

The Skyscanner redirects dataset (one of the intended target datasets of the system) is well within the territory of "big data", with millions of records being produced per day. The sheer volume of data warrants the use of distributed big data technologies such as Spark (expanded on below in Section 4.4.2).

Given the volume of data, and the need to label it (see Section 2.2.2) the decision was taken to use sample datasets to approximate the redirects dataset. This allowed an ex-

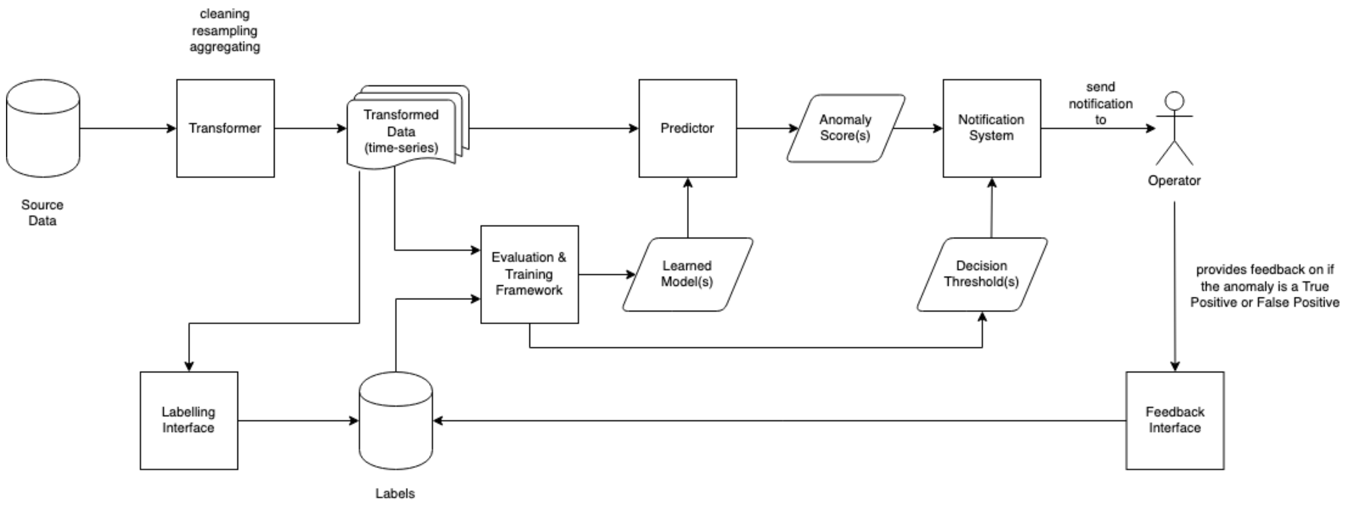

Figure 9: Proposed System Design Diagram

perimental implementation to be created with Pandas [1] and allowed for faster prototyping of models and anomaly detection techniques.

4.2.2 *Alerting Norms*

The need for meaningful alerts is discussed extensively in Section 2.2.4, however it is important that it be called out as a key constraint on the system being acceptable in practice in Skyscanner. Alerts which do not serve their purpose are quickly silenced, so the *precision* of trained models is especially important.

4.2.3 *Generalisation to other datasets*

In order for the anomaly detection system to have impact in a context like that of Skyscanner, it must have some ability to generalise to different datasets. It is important to note that the Transformer component (see Figure 9) could be different for each dataset to transform the data into a representation which is optimal for the anomaly detection system. Models may be trained per-dataset so that each dataset may be modelled correctly. It would also be valuable for multiple datasets to be joined and collated into a multivariate time-series, that allows them to be modelled and predicted jointly, increasing the predictive potential of the system.

4.3 System Design

The recommended system architecture for the anomaly detection system can be seen in Figure 9. The architecture diagram is generic and is intended to generalise the system's function, whether it were to use the streaming or batch paradigm (discussed in Section 4.4.1).

The data is assumed to already be ingested into a Data Source in a raw format, where the data is then transformed by the Transformer component. The transformation applied could be resampling, cleaning, aggregating or filtering, to name a few. The resulting transformed data should be in the format of a time-series. This time-series can be stored in an intermediate location (be that persistent storage or a separate data stream depending on the paradigm).

[1]`https://pandas.pydata.org/`

Initially, labels must be created for the data, using any of the methods discussed in Section 2.2.2. This could be accomplished via crowdsourcing, or simply manual inspection via a graphical interface. This is generalised by the "Labelling Interface" on the diagram. These labels, combined with the transformed time-series, are used to train and evaluate models using the Evaluation & Training Framework. The outputs of this stage are the learned models and decision thresholds that can be used to classify anomalies.

In live evaluation mode, the transformed time-seires data is given as input to the Predictor module which evaluates trained models and outputs anomaly score(s). The anomaly scores, combined with the learned decision threshold(s), are combined by the Notification System, to decide whether or not to alert an operator.

In order to employ online learning and continuously improve the anomaly detection system, any alerts sent to an operator should include instructions on providing feedback on the alert. This occurs via the Feedback Interface, and can, in this way, add labels for more up-to-date data, augmenting the training set and allowing future trained models to be more accurate.

The design is heavily inspired by Microsoft's (Ren et al., 2019) time series anomaly detection system, but is intended to be more generic to facilitate application to other tech stacks and paradigms.

4.4 Design Choices

4.4.1 *Real-time vs Batch*

There are several convincing arguments for using both real-time and batch processing paradigms to implement the anomaly detection system, and this could have significant implications for the applicability and usage of the system.

Using the real-time (or streaming) paradigm, anomaly detection could occur on data at a much finer granularity, allowing for a prompt response to new, developing anomalies (say, within 5 minutes). This would be useful in detecting production incidents and live software issues, but could be limited in its ability to identify long-term, systemic or trend anomalies happening over a larger timeframe. The real-time paradigm would also add complexity to the system, in the form of handling data replays, and persisting the streaming data for use in training - though realistically, it is likely that it would be available in archives, as would be the case in Skyscanner.

Using the batch paradigm, extremely quick responses (e.g. sub 5 minutes) would not be possible, however, in practical terms this may not matter. Traditional monitoring systems with thresholds can detect fast-occurring issues. The added value of having an anomaly detection system would be to identify larger, or more subtle medium-to-long-term anomalies. Using the batch paradigm reduces complexity as evaluation is done in a cron-based or offline manner, and there are many ETL platforms able to facilitate this. It also allows more complex or computationally-expensive models to be used as they can take much longer to complete, and less focus can be given to the optimisation of the training code.

4.4.2 Distributed vs single-process

Given that the system should be able to cope with large-scale datasets, this would motivate the use of distributed big data technologies like Apache Spark, which is extensively used in Skyscanner. Spark offers PySpark which has a Dataframe interface similar to that of the Pandas library. Though there is some similarity, Pandas is much more featured and allows for faster prototyping, and therefore Pandas was used in the creation of the evaluation framework for this work, to derive the results from different models.

When implementing the system, the experimental code created would have to be migrated to use Spark dataframes rather than Pandas, and this would allow large datasets that cannot fit on one node to be used as inputs.

4.4.3 Root Cause Identification

Once an anomaly alert is created and sent to an operator, the operator must respond, by analysing visualisations and/or debugging the system state, to determine the nature of the anomaly. Ultimately the operator's goal is to identify the root cause of the anomaly and address the problem or respond appropriately.

Though it is unlikely that the system could fully identify the root cause, one way in which the system could assist with this is by identifying the specific time-series that has caused the anomaly alert to be created. However, by reducing the problem to individual series, we remove the multivariate element of the data and therefore predictions and scores may be less accurate overall.

In order to overcome this for our prediction models (Section 2.1.3), a mixed approach of multivariate prediction with univariate scoring may be used, where the prediction model makes predictions taking all time-series into account, but the computed errors are used to model individual univariate normal distributions for each time-series, allowing each series to have an anomaly score and alerts to be created for individual series.

Zhang et al. (2019) identify root cause identification as one of their contributions, and that their MSCRED framework jointly addresses the issues of anomaly detection, root cause identification and anomaly severity. Implementing MSCRED may be an area for future work.

4.4.4 Ensemble Methods

Given that there are several models evaluated in this work, it may be possible to create better anomaly scores by combining the output of multiple models, also known as using *ensemble methods*. This approach is also recommended by Ren et al. (2019) as future work. The merging of results could be a mean average, maximum, or percentile. It should be noted that this would require the decision threshold to be treated separately from that of the individual models.

Different models may output anomaly scores with different variances, and so the anomaly scores may not lend themselves to combination with a single decision threshold in this way. Alternatively, each model may have its own decision threshold and "vote" for normal (0) or anomalous (1). With an odd number of models, a consensus may always be reached by the ensemble.

To further improve predictions, a weighting may be added to each of the ensemble models, to favour more accurate models and reduce the overall impact of noisy model outputs.

5. EVALUATION

5.1 Metrics & Evaluation Framework

As discussed at length in Section 2.2.4, it is important that an anomaly detection system trade-off between precision and recall, and this is achieved using the F_β score.

Given the importance of producing *meaningful* alerts, the system should balance between precision and recall. Malhotra et al. (2015) use $\beta \ll 1$ to give more weight to precision, however, in the end, a value of $\beta = 1$ was chosen for this work as it allowed for better optimisation of the decision threshold.

In order to evaluate different models, an evaluation framework was constructed, using mainly Pandas and Numpy [2]. The framework evaluates models by splitting the sample datasets into training and cross-validation sets. K-fold methods were not used due to the nature of time-series having to remain ordered. It is not intuitive to use datapoints earlier than the training data as a cross-validation set, since the problem consists of forecasting future anomalies based on past data.

The evaluation framework also evaluates the models performance on cross-validation sets at varying degrees of synthetic contamination in the training set, ranging from 0% to 5%, inspired by Zong et al. (2018)'s approach. As discussed in Section 2.2.3, this provides an indication into the model's robustness to *some* incorrect labelling in the training data.

The framework runs training on each of the models, then uses the outputs of the trained model on the training set to optimise a threshold parameter, using the F_β score as the function to maximise (in reality, minimise $1 - F_\beta$) with the choice of threshold parameter. This is similar to the approach of the Numenta Anomaly Benchmark.

5.2 Models

In order to perform anomaly detection, the system must utilise the best model for the problem. As such, several models were evaluated, and they are described here.

5.2.1 Last Value

In order to benchmark more complicated models, this work uses a simple prediction model using the previous value

[2]`https://numpy.org/`

for every timestep as the prediction. During training, the model fits a normal distribution over the squared errors using this prediction. This normal distribution's CDF is evaluated for the squared prediction error at each point, to produce the probability of the observation being anomalous. This model is intended purely as a naive benchmark and would not be used in practice.

5.2.2 *ADSaS*

ADSaS is a time-series analysis model developed by Lee and Kim (2019) which makes the claim that it is more performant than machine-learning-based models. In particular, the work is critical of LSTM. Therefor it is a good model for comparison, given this work also implements LSTM.

ADSaS is a combination of two main time-series analysis methods - Seasonal Auto-Regressive Integrated Moving Average (SARIMA) and Seasonal Trend Decomposition using LOESS (STL). The former is a prediction model which has a seasonality parameter and is adapted to non-stationary data with a seasonality. This prediction is then differenced with the real values to compute the error, and this error is decomposed with STL. The *residuals* of the decomposition are then collected and modelled as a normal distribution, and the CDF used as the anomaly score, as described in Section 2.1.3.

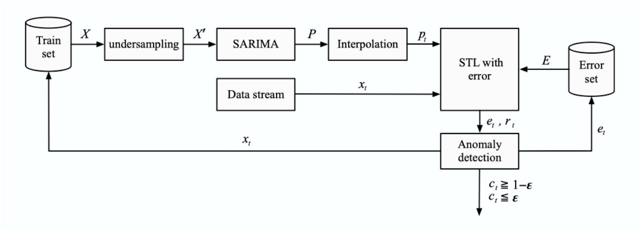

FIGURE 10: ADSaS diagram from Lee and Kim (2019)

In the original implementation, under-sampling and interpolation steps are also used around the SARIMA model (as can be seen in Figure 10), but these were ommited as the data was not - when evaluating on the sample datasets - on a scale that warranted under-sampling.

5.2.3 *LSTM*

The LSTM model is an LSTM recurrent neural network that outputs a prediction given n previous timesteps of observations. The full specification of LSTM and the training algorithm Backpropagation Through Time (BPTT) can be found in Hochreiter and Schmidhuber (1997). For the purposes of this evaluation, the model was implemented using the Keras [3] library.

LSTM networks function using multiplicative gates that enforce constant error flow through the internal states of special time units called "memory cells" (Malhotra et al., 2015). This allows them to learn long-term correlations in data and model complex multivariate sequences.

In the simple LSTM model, normal distributions were fit around the squared prediction errors, individually for each series in the input, and the CDF used to generate anomaly scores in the evaluation step of the model lifecycle.

[3] `https://keras.io/`

5.2.4 *LSTM + STL*

In addition to LSTM, a combination of the ADSaS and LSTM approaches is achieved with the LSTM + STL model. The intuition is similar to that of ADSaS, in the way of using the STL-decomposed residuals of prediction errors to produce anomaly scores, but the prediction itself is done by an LSTM neural network.

5.3 Results

5.3.1 *Model Comparison Results*

Figure 11 shows the comparison of the F_β score, precision and recall obtained on each dataset (with no contamination) for each evaluated model, and shows which model obtained the best score for each metric.

As can be seen from the results, the LSTM + STL model was the highest scoring model for precision and F_β score on both datasets. Given the importance of creating meaningful alerts, this implies that the best model of all the models compared was LSTM + STL.

The plain LSTM model achieved the highest recall on the NAB dataset by quite some margin. ADSaS achieved higher recall on the Yahoo A1 dataset, but not by as much comparatively to LSTM which had a lower but similar score.

5.3.2 *Contamination Results*

Figure 12 shows the precision, recall and F_β score of each model, split by dataset, at varying degrees of contamination (as discussed in Section 5.1). The contamination % refers to the % of the *training* set that was contaminated, whereas the metric results are the results of the same model on the *cross-validation* set.

Though the results are not as clean as would intuitively be expected, they do show that overall recall tends to drop as contamination increases, suggesting the model is less able to accurately predict anomalies. However, precision seems to show an increasing trend, which is counter-intuitive.

This could be because the total positive predictions (true positives + false positives) is reduced due to the model learning less precise decision boundaries due to the contamination, and therefore gives less overall positive guesses. By reducing TP+FP comparatively more than TP alone, precision improves, given that TP+FP is the denominator of precision. Further investigation is required to understand why this occurs and if the model(s) are overfitting at 0% contamination.

The contamination results do not conclusively say which model is most robust to incorrectly labelled training sets. On the Yahoo A1 results, if considering recall as the metric to evaluate on, the reduction in recall from 0% to 5% contamination in the ADSaS results was the most significant and in LSTM+STL was the least significant. This provides *some* indication that LSTM+STL may cope with contaminations better, but further experimentation would be required, perhaps with different datasets, to prove this conclusively.

5.3.3 *Experimental Critique / Improvements*

The reported precision, recall and F_β scores were computed on the flattened predictions and actual values across all series. However, by inspecting the values of the metrics on each series and taking the maximum or average of these, better results could be reported. However, the single

	NAB Tweets			Yahoo A1		
Model	F_β score	Precision	Recall	F_β score	Precision	Recall
ADSaS	0.24697	0.1622	0.51732	0.17589	0.12844	0.27895
LSTM	0.22898	0.13966	0.6351	0.22511	0.19972	0.25789
LSTM + STL	0.29207	0.25125	0.34873	0.24063	0.3249	0.19108
LastValue	0.0038938	0.0020373	0.04388	0.047892	0.034459	0.07849
Best	LSTM+STL	LSTM+STL	LSTM	LSTM+STL	LSTM+STL	ADSaS

FIGURE 11: Comparison of F_β score, precision and recall across datasets by model

(A) Yahoo A1

Contamination (%)	**ADSaS**			**LSTM**			**LSTM + STL**		
	Precision	Recall	F_β score	Precision	Recall	F_β score	Precision	Recall	F_β score
0	0.12844	0.27895	0.17589	0.19972	0.25789	0.22511	0.3249	0.19108	0.24063
1	0.19644	0.21739	0.20639	0.24689	0.23638	0.24152	0.37265	0.19519	0.25619
2	0.2055	0.1984	0.20189	0.27489	0.24325	0.2581	0.39376	0.18192	0.24887
3	0.21649	0.19108	0.20299	0.2796	0.23021	0.25251	0.35118	0.17414	0.23283
4	0.21289	0.17986	0.19499	0.29536	0.22723	0.25685	0.42687	0.16362	0.23656
5	0.28159	0.18307	0.22188	0.24779	0.17986	0.20843	0.40784	0.1762	0.24609

(B) NAB Tweets

Contamination (%)	**ADSaS**			**LSTM**			**LSTM + STL**		
	Precision	Recall	F_β score	Precision	Recall	F_β score	Precision	Recall	F_β score
0	0.1622	0.51732	0.24697	0.13966	0.6351	0.22898	0.25125	0.34873	0.29207
1	0.12375	0.51501	0.19955	0.18066	0.65589	0.28329	0.26412	0.33487	0.29532
2	0.10181	0.50577	0.1695	0.16755	0.7321	0.27269	0.18783	0.41339	0.2583
3	0.12188	0.47344	0.19385	0.17761	0.63741	0.27781	0.16779	0.46189	0.24615
4	0.17466	0.47113	0.25484	0.2223	0.72286	0.34003	0.24242	0.35104	0.28679
5	0.098175	0.52194	0.16527	0.1521	0.71132	0.25061	0.23714	0.38337	0.29303

FIGURE 12: Model robustness to contamination

computation on flattened results gives more robust results.

6. RELATED WORK

The field of applied time-series anomaly detection has seen several works in recent years, focusing on different techniques such as temporal prediction methods, Convolutional Neural Networks (CNNs), LSTM and density estimation methods. This section explores some alternative anomaly detection methods.

6.1 Convolutional Neural Networks (CNNs)

Zhang et al. (2019) and Ren et al. (2019) are examples of works where CNNs are applied to time-series anomaly detection. Both works have similarities and differences in their approach.

Both works do not apply Convolutional Neural Networks directly to the input data - they instead transform the data into a spatially-significant representation. In the MSCRED framework described by Zhang et al. (2019), the multivariate time-series are collated into "system signature matrices" by computing the inner product of two time-series, for every pair of time-series. This allows for two-dimensional convolution operations to be applied to these matrices and makes the model more robust to noise. The signature matrices allow the correlations between different pairs of time-series to be captured. Zhang et al. (2019) reference previous studies stating these are critical to characterise system status.

In Ren et al. (2019), the *Spectral Residual* transform, a simple yet powerful approach based on *Fast Fourrier Transform (FFT)* (Ren et al., 2019), is applied to the input data to produce a "saliency map". This saliency map is then given as input to one-dimensional convolutional layers of a CNN. Ren et al. (2019) state that this is because of the lack of labelled data - by transforming the data into a saliency map, the CNN can be trained on saliency maps, which is a much more constrained domain and yield better results when the labels are lacking.

Both Zhang et al. (2019) and Ren et al. (2019) also use a combination of real and synthetic data for training. Ren et al. (2019) use real production time-series augmented with synthetic anomalies as training data. This is done because of the problem of lack of labelled data mentioned as the main challenge in their paper. By utilising production data as the basis, they aid their model to learn "normal" behaviours. Zhang et al. (2019) evaluate their model on both synthetic data and a real-world power plant dataset. Their synthetic data is completely synthetic, generated based on a sinusoidal pattern with noise.

The works differ in the case of their expected output - Ren et al. (2019) only wish to be able to identify if a given window of points is anomalous, however Zhang et al. (2019) concern themselves also with *anomaly diagnosis*, pointing out which time-series are to blame for the anomaly and the precise point at which the anomaly occurs.

There is generally a consensus that convolutional neural networks require some transformation of the input data, such that convolution operations are effective on the representation.

6.2 Temporal prediction methods

Temporal prediction methods operate on the intuition of detecting deviation-based outliers of specific time-instants with the use of regression-based forecasting models - outliers are declared on the basis of deviations from expected (or forecasted) values (Aggarwal, 2016).

Temporal prediction methods rely on the principle of *temporal continuity*, which assumes that patterns in the data are not expected to change abruptly, unless there are abnormal processes at work (Aggarwal, 2016).

Aggarwal (2016) states that the regression models used in temporal methods can utilise correlations both across time (referring to temporal continuity) or across series, as many applications output multivariate time series that are often closely correlated with one another.

Conventional time-series analysis methods often do not perform well in anomaly detection, or find anomalies in limited conditions (Lee and Kim, 2019). For example, SARIMA and STL, are used only for stationary and periodic time-series respectively (Lee and Kim, 2019), but in the ADSaS system (Lee and Kim, 2019), combining STL and SARIMA is shown to detect anomalies with high accuracy for data that is even noisy and non-periodic.

ARIMA (Auto-Regressive Integrated Moving Average) models generalise the simplest time-series forecasting models. Autoregressive (AR) models predict the next point in a time-series based on a weighted combination of a given number of previous terms in the time-series. Moving Average (MA) models predict the next point based on a given number of previous *white noise* terms.

$$AR(p) : X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t$$

(Aggarwal, 2016)

$$MA(q) : X_t = \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \mu + \epsilon_t$$

(Aggarwal, 2016)

$$ARMA(p,q) : X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$$

(Aggarwal, 2016)

The ARMA model can be enhanced to capture persistent trends with the ARIMA model (Auto Regressive Integrated Moving Average), which works better for such *non-stationary* time series (Aggarwal, 2016).

STL is a versatile and robust method for time-series decomposition (Lee and Kim, 2019). It is an algorithm developed to decompose a time-series into three components. Namely, the trend, seasonality and residuals. The trend shows a persisting direction in the data, seasonality shows seasonal factors and residuals show the noise of the time-series (Lee and Kim, 2019).

6.3 Density Estimation

Density estimation is a popular technique that has been utilised in anomaly detection. Anomaly detection itself ***is*** density estimation - given many input samples, anomalies are samples that reside in low probability density areas (Zong et al., 2018).

One of the main problems with density estimation is the *curse of dimensionality* (see Section 2.2.5) which motivates the use of dimensionality reduction techniques, like Principal Component Analysis or Autoencoders, coupled with density estimation. Most approaches focus on dimensionality reduction followed by density estimation, however they suffer from decoupled model learning with inconsistent optimisation goals and incapability of preserving essential information in the low-dimensional space (Zong et al., 2018). The DAGMM (Deep Autoencoding Gaussian Mixture Model) model proposed by Zong et al. (2018) provides an approach which uses an autoencoder and Gaussian Mixture Model, jointly optimising the parameters of the autoencoder and mixture model simultaneously in an end-to-end fashion.

6.4 Long-Short Term Memory (LSTM)

Anomaly detection techniques involving LSTM deep neural networks are becoming popular in the field.

Malhotra et al. (2016) present an "LSTM Encoder-Decorder" (autoencoder) model for detecting anomalies in sensor data from machines, engines, vehicles etc. It is an example of an application of modern anomaly detection to a more traditional, established application of anomaly detection (mechanical engineering). The article states that unmonitored environmental conditions or load may lead to inherently unpredictable time-series. Detecting anomalies in such scenarios becomes challenging using standard approaches based on mathematical models that rely on stationarity, or prediction models that utilise prediction errors to detect anomalies. This motivates the use of a more advanced model like an LSTM autoencoder.

The model presented in Malhotra et al. (2016) consists of two neural networks, an Encoder and Decoder, which are chained together to form an autoencoder that is trained to reconstruct instances of normal time-series. As such, this can be considered a semi-supervised technique. Malhotra et al. (2016) explicitly mention that this is useful in scenarios when anomalous data is not available or is sparse. This is the case in their application as "the relevant information pertaining to whether a machine is laden or unladen may not be available". This is similar to the problem the research intends to solve wherein the anomalies are largely unknown and labels for the anomalies are not immediately available.

The work by Malhotra et al. (2016) is based on previous work by some of the same authors. Malhotra et al. (2015) introduce LSTM neural networks to time-series anomaly detection, but rather than using them as an autoencoder, it uses a "stacked LSTM based prediction model", and then uses prediction errors on a training set to build a probability distribution of errors, which is then used to output a probability representing the anomaly score for a given timestep, and, combined with a decision threshold, can output binary anomaly labels. Malhotra et al. (2015) also mention that by giving the network features that are "control variables" (features describing the settings chosen on manual operator controls) the network learned normal usage patterns, and was able to recognise anomalous control input as well as anomalous outcomes.

Unfortunately, in the ADSaS system, Lee and Kim (2019) point out several criticisms of LSTM models. The ADSaS system is designed to be real-time and is memory constrained to work on IoT devices, therefore the large training time and memory complexity of a deep learning model like LSTM is listed as a disadvantage when compared to conventional time-series forecasting methods. They also observe a better overall performance in the ADSaS system based on SARIMA and STL (discussed in Section 6.2). However, the main critique of this finding is that the evaluation of the algorithm was conducted on univariate time-series, and it is acknowledged in the conclusion that future work is required to apply the algorithm to multivariate time series, whereas it is known that LSTM performs well on multivariate data. Lee and Kim (2019) also recognise that LSTM had the lowest prediction error compared to other algorithms - due to the fact that they adjust quickly to the norm after an abrupt change in pattern.

6.5 Large-scale application of anomaly detection

There are not many examples of research detailing the application of anomaly detection to large-scale datasets in a real-world context.

Ren et al. (2019) describe Microsoft's implementation of time-series anomaly detection, which operates on the scale of 10,000 to 100,000 datapoints per second. The system is split into three major components: data ingestion, experimentation platform and online compute. The system is based around streaming real-time data in Apache Kafka processed by Apache Flink. Data is first ingested from a source and pushed into Kafka. The data is then evaluated by the "online compute" component. Finally, the experimentation platform allows researchers to train models and deploy them, validating their efficacy using evaluation metrics.

Ren et al. (2019)'s system monitors minute-level time series, and is used by more than 200 product teams within Microsoft. This work therefore also is subject to the challenges discussed in Section 2.2.4 on creating meaningful alerts. There is also the problem of determining which is the correct team or person to alert for a given anomaly. There are many parallels between Microsoft's use case and the use case of this work in Skyscanner.

Zhang et al. (2019)'s MSCRED framework also details an application of anomaly detection, as it was intended for use on data from a power plant. Though this is a real-world application of a significant scale, it is not comparable in terms of scale to Microsoft's anomaly detection system, or to the target dataset for this work.

7. FUTURE WORK

There are many areas for future research on anomaly detection techniques that could be applied.

First of all, the Convolutional Neural Network models discussed in Section 6.1 have not been evaluated and compared with the prediction models, which could be beneficial in determining if CNNs could provide better performance for a large-scale anomaly detection system.

The LSTM-based model's performance could potentially be improved by jointly learning the correct prediction values as well as the decision threshold, letting it be dymanic and better cope with *heteroskedasticity* (Section 2.2.6).

Another area of future research could be in root cause identification and anomaly severity scoring. By capturing qualitative information or market sentiments, anomalies could be classified as system errors or genuine observed anomalies.

8. CONCLUSIONS

In this paper, various models are evaluated for application to a large-scale time-series anomaly detection system, whose design is proposed (Section 4).

The LSTM+STL model was chosen as the recommended model for the system to use, based on the fact that its precision and F_β score results were the best of all the evaluated models (Section 5.3.1; Figure 11) and there was some indication that it is more robust to incorrect labelling (Section 5.3.2). In addition, due to LSTM+STL being a machine-learning based model, it would be expected to become more accurate the larger the training set, and so it would be expected that its performance improve when applied to the full dataset, relative to that on the sample datasets.

This paper concludes that anomaly detection is feasible within the context of Skyscanner, and details how this could be accomplished, with a system design and model recommendation, serving as a blueprint for its implementation, that could be used as a reference point for similar use cases.

References

Charu C. Aggarwal. *Outlier Analysis*. 2016. ISBN 978-1461463955.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.

Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4):1–31, 04 2016. doi: 10.1371/journal.pone.0152173. URL `https://doi.org/10.1371/journal.pone.0152173`.

J. D. Hamilton. *Time series analysis*. Princeton University., 1994. ISBN 0-691-04289-6.

Sepp Hochreiter and JÃijrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Sooyeon Lee and Huy Kang Kim. Adsas: Comprehensive real-time anomaly detection system. *Lecture Notes in Computer Science*, page 29âĂŞ41, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-17982-3_3. URL `http://dx.doi.org/10.1007/978-3-030-17982-3_3`.

Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. 04 2015.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection, 2016.

Numenta. Numenta anomaly benchmark (NAB). `https://github.com/numenta/NAB`.

Hansheng Ren, Qi Zhang, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, and Jie Tong. Time-series anomaly detection service at microsoft. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD âĂŹ19*, 2019. doi: 10.1145/3292500.3330680. URL `http://dx.doi.org/10.1145/3292500.3330680`.

Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.

Yahoo. A benchmark dataset for time series anomaly detection. `https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly`.

Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1409âĂŞ1416, Jul 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33011409. URL `http://dx.doi.org/10.1609/aaai.v33i01.33011409`.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Dae ki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *ICLR*, 2018.