Project Group 14: Time Series Anomaly Detection for ECONet Dataset

Moksh Jain
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
mrjain@ncsu.edu

Pratik Devnani
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
prdevnan@ncsu.edu

Rahil Sarvaiya
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
rhsarvai@ncsu.edu

, ,

ACM Reference Format:
Moksh Jain, Pratik Devnani, and Rahil Sarvaiya Project Group 14: Time Series Anomaly Detection for ECONet Dataset

1 PROBLEM STATEMENT

The ECONet Dataset [1] contains 23 measurements at 45 weather stations at each minute of each day for 2021.

This means that we have 23 (measurements) * 45 (weather stations) * 60 (number of minutes per hour) * 24 (number of hours per day) * 365 (number of days per year) = 544M data points!

This data is run through automated quality control (QC) and the flagged data is further reviewed manually by experts. This results in a total of 8M+ reviewed readings each year which is not only tedious but also requires a huge amount of manual effort.

The goal of this project is to accurately predict which measurements are erroneous and which are not using Machine Learning Techniques to automate the reviewing process with a high level of confidence in the model.

2 RELATED WORK

Various unsupervised learning approaches have been created to treat anomalies in data. The simplest of these are out-of-limit methods, where we tag any data beyond a certain threshold. However, due to their simplicity they fail to detect contextual anomalies. More advanced techniques such as proximity-based, prediction-based, and reconstruction-based anomaly detection have been proposed.

In some studies, Convolution Neural Networks [2] have been used to detect abnormal data. This study uses an attention-based Convolutional Neural Network LSTM model to detect anomalies. This attention-based model uses CNN units to capture important features which prevent memory loss and gradient dispersion problems. Further, the use of the memory feature in LSTMs [3] improves the prediction of the anomalous data.

Another approach uses Support Vector Regression [4] to detect anomalies in meteorological data mentioned in detail in a paper published in 2018. The SVRs are used to predict the observational values from a spatial perspective and the different between this estimated value and the original value determines whether the data is abnormal or not.

These papers present novel ideas for finding anomalies in the data, but the data that they work on are geo-spatial, temporal and highly preprocessed data. We propose a simple method for improving the dataset and implementing a simple classifier for the anomaly detection problem. Pre-processing and cleaning our dataset will help our approach produce better results. In a paper from January 2016 [5], a data reduction method is used to treat the imbalance in the data followed by few sampling strategies. They combine the T-link [6] algorithm with few sampling methods including SMOTE [7] to maintain the balanced class distribution.

3 APPROACH

We implemented various Neural Network models to find the anomalous entries in the weather data. These approaches were fed with data that was sampled using multiple sampling techniques that can balance the dataset. The different sampling techniques that we implemented are Random Under Sampling, Random Over Sampling, and SMOTE.

We implemented the models on non-processed data as well as on data after feature selection using PCA [8]. The eigenvectors capture 95-99 percent variance in the dataset which reduces the attributes to 7-8 respectively. The data were scaled using sklearn's inbuilt StandardScalar module which was redacted by removing the mean and scaling to unit variance. The PCA reduced dataset did very well on the training set but on testing it against the test CSV file, we could notice that it under-performed by a heavy margin.

We used a random state in all the applicable sections of our code to maintain reproducible results. We then compare the various results and justify our implementations with various charts that show the sampled data.

We tried two main approaches to classify anomalous data - a Multi-Layer Perceptron (MLP) [9] with 2 hidden layers that try to learn and model the complex relationships from the data and a Long Short-Term Memory (LSTM) approach that can leverage the "memory" capabilities of the network to predict future values based on the previous sequential data.

MLPs are feed-forward neural networks with multiple hidden layers. They are known to be able to learn complex time correlations and dependencies within the data that are not linearly separable. MLPs are highly effective in learning configurations that will lead to a certain output and we can use this attribute to our use.

LSTMs are capable of capturing both the long-term and short-term patterns in the model, which would help us in finding the anomalies. The different gates inside the LSTM augment its capacity to extract the non-linear relationships in time series data.

We decided against using GANs for our project due to multiple reasons, some of them being GANs being prone to non-convergence, the possibility of the discriminator getting too successful that the generator gradient vanishes and learns nothing, being highly sensitive to the hyper-parameter selection, etc.

4 RATIONALE

Due to the highly imbalanced nature of the dataset, there is a high possibility of a model not learning any aspect of the data that has less representation in the dataset. Even though the accuracy of such a model might be high, the prediction accuracy and recall of the class that matters the most in anomaly detection problems will be very low. To overcome this issue, we have to balance the dataset using various techniques that under-sample the majority class or up-sample the minority class. There are pros and cons to both methods but by performing sampling, we ensure that equal weightage is being given to all the possible outcomes and that the model won't overfit to one class.

5 DATASET

The dataset that we have decided to work on is the ECONet Dataset. This dataset is provided by The NC Climate Office. The reduced dataset has 4 QC flags along with location, timestamp, and 2 other properties. The dataset is highly imbalanced so we will have to perform some transformations for pre-processing the data before we can train a model. This dataset is tagged as normal or anomalous using the boolean values and has been split into 2 different CSV files.

6 HYPOTHESIS

We make the following hypothesis:

- Over Sampling the data will get better results on the test dataset as compared to Under Sampling.
- An LSTM model will outperform an MLP model for predicting anomalous data.

7 EXPERIMENTAL DESIGN

7.1 Data Pre-processing

Before working on our algorithm and model, we performed few data pre-processing steps. The time attribute in the dataset had to be converted to a datetime object. Moreover, a lot of the string values in the class had to be converted to numerical values using LabelEncoder.

Once we converted all the values in our dataset to a numerical form (so that it could be processed by our model), we standardized the data. We needed to convert it to a format that enables our model to easily process and analyze it. We scale the data and make it internally consistent to better improve our results.

7.2 Sampling the data

On studying the dataset, we observed certain imbalances in it. This could result in our algorithm learning more information from the majority class compared to the minority class. To deal with this data imbalance in our dataset, we performed resampling techniques.

We aimed at balancing the class distribution so that our MLP or LSTM can be trained directly on this transformed dataset which would help predict better results.

We mainly experimented with the following sampling techniques to remove any kind of bias that would exist with a particular class.

- Random Over Sampling
  We randomly duplicated examples from the minority class in the training dataset. This would make the count of the minority class equivalent to the majority class, thus ignoring any possible bias that could arise due to the discrepancy. SMOTE is a more novel approach to up-sampling and for that reason this approach was just experimental.
- Random Under Sampling
  Undersampling methods remove a subset of our data from the majority class. This technique randomly deletes examples from the training dataset of the majority class.
- Synthetic Minority Oversampling Technique (SMOTE)
  SMOTE selects examples that are close in the feature space, draws a line between the examples in this feature space and generates a new sample point along that line.

In our many trial runs of sampling, we noticed that a higher accuracy was obtained with the SMOTE method. Hence, we went ahead with this method while implementing our Neural Network. Since LSTM needs data in a windowed form, Oversampling was creating issues for the model and hence we decided to cap our oversampling at a fixed number of representative samples in each class.

We only perform sampling on the training dataset and not on the holdout test or validation dataset to prevent our model from overfitting.

7.3 Implementation

We used Sequential Keras API for creating our models and experimented with the following hyperparameters:

- Epochs: Since the model training was extensive in nature due to the sheer size of the training dataset, we limited our models to a maximum of 10 epochs. Three different epoch values were tested (3, 5, 10) and the best AUC-PR was obtained for epoch of size 5.
- Batch Size: Batch sizes of 32 and 64 were used for model training
- Number of hidden layers: We experimented using different number of hidden layers and found that 2 hidden layers worked best for our problem.
- Number of hidden nodes in hidden layer: We used different amounts of hidden nodes in hidden layers and found that larger width would overfit the model to the training data.

The dataset was split in a 70:30 ratio for training and testing our model. The training set was further split for validating our model. For pre-processing the data to make it usable for the model, we used one hot encoding to convert our categorical output classes into a binary vector. We also reshaped the output class for fitting it to our binary classification model using the Numpy module. We tried various runs using SMOTE and Random Undersampler which yielded a variety of results using these 2 strategies. We faced memory issues while implementing LSTM with SMOTE and for that reason we decided to cap off the maximum representation of each class.

The following classification techniques with variations in sampling methods were performed before and after Principal Component Analysis:

- 2 hidden layer MLP with Random Under Sampler
- 2 hidden layer MLP with Random Under Sampler + SMOTE
- 2 hidden layer MLP with SMOTE
- 1 hidden layer LSTM with Random Under Sampler
- 1 hidden layer LSTM with Random Under Sampler + SMOTE
- 1 hidden layer LSTM with SMOTE

LSTM and MLP were used as a part of a sequential model which is a part of the Keras library. We used an Adam optimizer for weight updates for our network and considered 'binary cross entropy' loss as our loss metric while fitting the model. We also used Dropout regularization to prevent our model from overfitting. We used Relu activation function for all the layers except the output layer, in which, a Sigmoid activation function was used.

7.4 Model Summary

The following are the summaries of the variations of the models that gave us the highest results on the test dataset:

7.4.1 ANN model summary. Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense_13 (Dense)	(None, 8)	72
dropout_6 (Dropout)	(None, 8)	0
dense_14 (Dense)	(None, 16)	144
dense_15 (Dense)	(None, 8)	136
dropout_7 (Dropout)	(None, 8)	0
dense_16 (Dense)	(None, 2)	18
Total params: 370		
Trainable params: 370		
Non-trainable params: 0		

The MLP model that performed the best has 2 hidden layers with 16 and 8 hidden nodes in each. The input layer is of size 8 which is the size of the training columns of the dataset and the output layer has 2 nodes.

7.4.2 LSTM model summary. Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 40, 4)	208
dropout (Dropout)	(None, 40, 4)	0
lstm_1 (LSTM)	(None, 4)	144
dropout_1 (Dropout)	(None, 4)	0
dense (Dense)	(None, 2)	10
activation (Activation)	(None, 2)	0
Total params: 362		
Trainable params: 362		
Non-trainable params: 0		

The LSTM model has 1 hidden layer of width 4 and the input to the model is windowed with our window size being 40. We experimented with different window sizes and found that 40 gave the best results. The output layer is of width 2 which corresponds to our binary classification problem

8 PLAN OF ACTIVITY

8.1 Work division

Task	Description	Team Member
Research	Exploring strategies to deal with imbalanced data	Pratik
	Exploring strategies to detect anomalies in Time Series data	Pratik
Data Pre-processing	Analysis of the data	Moksh
	Oversampling of the data	Rahil
	Undersampling of the data	Rahil
	Principal Component Analysis on the data	Moksh
Modeling with various Sampling Strategies	Performing Random Forest [10] classification with Random Undersampling	Pratik
	Performing Random Forest with SMOTE and Random Undersampling	Moksh
	Performing Random Forest classification with Random Undersampling	Moksh
	Performing Random Forest classification with ADASYN [11]	Rahil
	Performing Balanced Random Forest classifier. [12]	Rahil
	Performing Logistic Regression	Rahil
Model Exploration	Exploring K-Means Clustering on the data (treating the task as Unsupervised Learning)	Moksh
	Exploring GANs for anomaly detection using Time Series data	Pratik, Rahil
Documentation	Preparing the Mid-Term Report	Moksh, Pratik, Rahil
Model Implementation and Evaluation	Implementing Multi-layer Perceptron	Moksh, Rahil
	Implementing LSTMs	Pratik
	Evaluating the model and comparing results	Moksh, Pratik, Rahil
	Final Project Documentation	Moksh, Pratik, Rahil

8.2 Meeting Minutes

Date	Time	Attendees	Agenda
16 Mar	1pm to 2pm	Moksh, Pratik, Rahil	Finalize project problem statement, brainstorm ideas and estimate work division amongst the team.
18 Mar	1pm to 3pm	Moksh, Pratik, Rahil	Information collation, experimental approach discussion and creating the proposal document.
29 Mar	12pm to 1:30pm	Moksh, Pratik, Rahil	Discussion about Data exploration techniques, viable options for dimensionality reduction and data pre-processing.
3 Apr	4pm to 5pm	Moksh, Pratik, Rahil	Plan for midway report, discuss data imbalance handling methodologies and finalize options for classification approaches.
9 Apr	3pm to 6pm	Moksh, Pratik, Rahil	Comparing results of finalized data pre-processing and classification methodologies and collating our findings in the report.
16 Apr	1pm to 2pm	Moksh, Pratik, Rahil	Discuss about our final approach, platform/software to train models. Final work distribution for model training.
20 Apr	12pm to 2pm	Moksh, Pratik, Rahil	Update on model training and development. Tackle issues faced regarding model training or preprocessing for model training.
25 Apr	1pm to 2pm	Moksh, Pratik, Rahil	Compare results and upload predicted results on codalab. Fine tuning models to get better results. Discuss and start final documentation.

9 RESULTS

We implemented multiple iterations of our Multi Layer Perceptron and our LSTM. Our 6 top performing models are documented below:

9.1 Multi-Layer Perceptron

9.1.1 Result 1 - MLP with 2 hidden layers.

Parameter	Value
AUC_PR	0.653709
RECALL	0.807339
POSITIVE_F1	0.557783
AUC_ROC	0.926206
ACC	0.948483

We trained our Neural Network on 2 hidden layers. The first hidden layer had 16 units and the second had 8 units. In addition to this we used SMOTE for sampling our data before passing it into our model to train.

9.1.2 Result 2 - MLP with 2 hidden layers and dropout.

Parameter	Value
AUC_PR	0.624175
RECALL	0.754619
POSITIVE_F1	0.534036
AUC_ROC	0.956273
ACC	0.947005

Before training our model, we sampled the data using SMOTE. Our model consisted of 2 hidden layers of 16 units each. This model was trained for 5 epochs and with a dropout of 0.1.

9.1.3 Result 3 - MLP with 2 hidden layers and no dropout.

Parameter	Value
AUC_PR	0.389097
RECALL	0.506078
POSITIVE_F1	0.385534
AUC_ROC	0.662878
ACC	0.935080

Before training our model, we sampled the data using SMOTE. Our model consisted of 2 hidden layers of 16 units and 8 units respectively. We ran this model for 10 epochs and without any dropout.

9.2 Long Short Term Memory (LSTM)

We used a window size of 40 units. We tested our model with different window sizes and found 40 to be the most optimal size on which our model could learn the dependencies.

9.2.1 Result 4 - LSTM with Random Under Sampling.

Parameter	Value
AUC_PR	0.522833
RECALL	0.983935
POSITIVE_F1	0.089798
AUC_ROC	0.760023
ACC	0.197283

We added a hidden layer of 4 units to our model.

9.2.2 Result 5 - LSTM with Random Under Sampling.

Parameter	Value
AUC_PR	0.499193
RECALL	0.941978
POSITIVE_F1	0.102466
AUC_ROC	0.640824
ACC	0.335893

We also added a hidden layer with 8 nodes to this model.

9.2.3 Result 6 - LSTM with hidden nodes and PCA.

Parameter	Value
AUC_PR	0.519045
RECALL	0.999585
POSITIVE_F1	0.086285
AUC_ROC	0.580635
ACC	0.148035

We added a hidden layer of 32 nodes to our model. We used dimensionality reduction with a 99% variance capture.

The data that was sampled using SMOTE and the count of representation for each class was 4450846 instances. The total training instances were 8901692 and each epoch had an average run time of 1500 seconds for MLPs. The data that was sampled using RandomUnderSampler had 235172 instances representing each class and the count of total instances was 6593274. The running time of an epoch for an LSTM model varied depending on the number of hidden nodes in the hidden layer. We standardized our training dataset and testing dataset to ensure similar inputs for our model while learning as well as predicting. As we can see from the results, our models performed fairly with many different variations implemented. The best performing model was an MLP with 2 hidden layers with 16 hidden units in the first layer and 8 hidden units in the second layer.

10 DISCUSSION

After analyzing the results, we can conclude that we did get some interesting results but there is a lot of scope for improvement. Handling data pre-processing differently especially in terms of time tsries location tagged data, can produce better results than what we achieved using our method. Our first hypothesis was proven by the MLP model that performed the best after being sampled using SMOTE. In general, we can conclude that for our case oversampling the data was better than undersampling.

Our second hypothesis was disproved by our results since the MLP performed better than the LSTM in every iteration of the network that we tested. One of the reasons for this could be the correlation between time and the anomaly was not high which could explain the low performance. Our model does not perform better than the models cited in previous related works and there can be a few reasons for the same. One being the lack of computational power limits the training time and this could result in a low-performing model. The second could be the pre-processing methodology that would yield better results for others and comparatively lower results for us.

11 CONCLUSION

While working with this dataset, we performed various sampling methods to get a substantial representation of the minority data. This helped us realize the importance of sampling in the data preprocessing pipeline, which is often overlooked when the data is equally represented.

For our dataset, SMOTE sampling technique provided the best augmentation of the data.

Working with imbalanced data is very difficult as there is fewer data available to synthesize new data. Moreover, the synthesized data does not always represent real-world data. Thus, we cannot be sure that the model is accurately identifying anomalous data.

While building the models, we find that even though basic implementation seems easy, fine-tuning the parameters to get the best possible accuracy is very time consuming. Training the model over a large dataset takes a lot of time.

After evaluating the results, we found that MLPs when paired with the right sampling strategies, performed better than their LSTM counterparts.

Additionally, we believe that given more training time and resources, our model could have been much better, providing more accurate results. This project gave us a complete overview of the Machine Learning Pipeline, from data extraction to evaluation of results. We were able to apply the concepts learned throughout the course and bolster our understanding of the domain by applying our knowledge to build a model for a real-world problem.

12 REFERENCES

[1] North Carolina State Climate Office, NC State University. Cardinal [data retrieval interface] Available at https://products.climate.ncsu.edu/cardinal/request. Accessed March 18, 2022.

[2] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

[3] Yin, C., Zhang, S., Wang, J., Xiong, N. N. (2020). Anomaly Detection Based on Convolutional Recurrent Autoencoder for IoT Time Series. IEEE Transactions on Systems, Man, Cybernetics: Systems.

[4] Min-Ki Lee, Seung-Hyun Moon, Yourim Yoon, Yong-Hyuk Kim, Byung-Ro Moon. (2018). Detecting Anomalies in Meteorological Data Using Support Vector Regression. Advances in Meteorology.

[5] Elhassan, Tusneem M, Aljourf F, Al-Mohanna Shoukri, Mohamed. (2016). Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method. Global Journal of Technology and Optimization. 01. 10.4172/2229-8711.S1111.

[6] Tomek Ivan (1976) An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems. Man, and Cybernetics 6: 448-452.

[7] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

[8] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2, no. 1-3 (1987): 37-52.

[9] Minsky, Marvin, and Seymour Papert. "An introduction to computational geometry." Cambridge tiass., HIT 479 (1969): 480.

[10] Breiman, Leo. "Random forests." Machine learning 45, no. 1 (2001): 5-32.

[11] He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning." IEEE world congress on computational intelligence, pp. 1322-1328. IEEE, 2008.

[12] Chen, Chao, Andy Liaw, and Leo Breiman. "Using random forest to learn imbalanced data." University of California, Berkeley 110, no. 1-12 (2004): 24.