Anomaly Detection in Time Series using Unsupervised Machine Learning Approach

Shruti Devlekar
AIDS
Vivekanand Education Society's Institute of Technology (VESIT)
Mumbai, India

Venkatesh Rallapalli
CMPN
Vivekanand Education Society's Institute of Technology (VESIT)
Mumbai, India

Sangeeta Oswal
AIDS
Vivekanand Education Society's Institute of Technology (VESIT)
Mumbai, India

ABSTRACT
Anomaly detection is the process of identifying data points, observations or events which deviate from normal behavior. To detect anomalies, we have used the PyCaret machine learning library. PyCaret is a low code, open-source library that automates our machine learning processes and helps us detect outliers/anomalies Its Anomaly detection and Regression modules contain various machine learning algorithms and frameworks such as XGBoost, CatBoost, Isolation Forest, DBSCAN, etc. It is a deployment-ready library that is easy to use and helps users to perform end-to-end experiments efficiently. In this paper, we applied clustering and regression-based methods on the NAB Twitter dataset for time series anomaly detection. In the regression method, we predicted the tweets, calculated the difference by comparing them with actual tweets, and used the thresholding technique for anomaly detection.

General Terms
Anomaly Detection in Time Series Data, Unsupervised Machine Learning Algorithms.

Keywords
Anomaly Detection, Time Series Data, Anomalies, PyCaret, Clustering, Regression, Machine Learning, Isolation Forest, XGBoost, One Class SVM, CatBoost, Extra Trees Regressor

1. INTRODUCTION
Everything in the world is a signal source. In order to adapt to the environment, humans have continuously measured and gathered natural signals like temperature and rainfall. Numerous industrial processes produce a large amount of data in many different industries, including finance, business (sales and marketing), biomedicine, manufacturing, logistics, health care, automotive, network log files, etc. Also with the advent of industry 4.0 various sensors to monitor systems for safety, security, and efficiency are generating data. In most cases, the collected data is streaming time series data. Time-series analysis refers to a range of tasks that aim to extract meaningful knowledge from time-ordered data. Different data analytics techniques to mine the time series data are available and anomaly detection is the backbone and the focus of many researchers because of its applicability to a wide range of domains.

2. LITERATURE
Anomaly :-
An Anomaly can be defined as an inconsistency in data. It is a deviation of a data point or set of data points from the established rule or norm. The majority of data follows a certain trend or pattern. Anomalies are the set of data points that deviate from this pattern.
There are different types of anomalies such as

1. Point Anomalies
2. Contextual Anomalies
3. Collective Anomalies

Anomaly Detection :-
Anomaly or outlier detection is the process of identifying anomalies or data which deviates from the rest of the dataset [3,4,5,6,8,9,11].

Time Series Data :-
Time Series Data (or time-Stamped Data) is a sequence of data points that are indexed using time. It helps us track changes in data over regular intervals. This may be a year, month, day or even minor change observed over an hour, minute, second, or millisecond.[2,11,12].

Time Series data are also of multiple types. They are broadly classified into

1. Univariate Time Series - Only one variable is measured or tracked over time. (Expenses of a certain household measured on a monthly basis)
2. Multivariate Time Series - Multiple variables vary over time. (Stock Market Data. Features like Closing Price, High, Low, and Volume Traded measured over a period of time)

- Characteristics of Time Series Data: Time series data have certain characteristics that help in analysis and modeling. [10]

1. Trend - This property refers to the increase or decrease of a value over time.
2. Seasonality - It is the recurrence of patterns found in the data over predictable time periods, like a season, quarter, month, or day of the week.
3. Cycle - This is similar to seasonality but the period is not fixed and the duration is above a decade.

4. Stationarity - Time Series data is said to be stationary if it has constant variance and no seasonality over a specific period. Its main characteristics remain the same.
5. Pulses and Steps - Data levels that abruptly change can be thought of as pulses or steps. A quick shift in data that lasts only a brief period of time is called a pulse, while one that lasts for a longer period is called a step.

Approaches for Anomaly Detection: There are various methods to detect anomalies. They can be classified as

1. Statistical Methods
2. Machine Learning Methods
3. Deep Learning Methods

The application of these methods varies for univariate and multivariate data. The approaches for both sets of data are as follows:

2.1 Statistical Approach for Anomaly Detection

1. For Univariate Data:

Here we make use of regressive models like AR (Auto Regression), MA(Moving Averages) or ARIMA(AR + I + MA) where I stands for Integrated.

→ ARIMA[7]:
AR (Auto Regression) - It is the Linear Regression of the current value of a series against one or more prior values.
I (Integrated) - Used to make the time series stationary.
Moving Average (MA) - Future predictions are based on errors in past predictions.
A combination of these techniques is used to create the ARIMA model. It takes the parameters p, d, and q, and the model is initialized ARIMA(p,d,q) using these components. Here, p is the order of the Auto-Regressive Component, q is the order of the Moving Average Component and d is the number of differentiation steps performed to make the time series data stationary. The best values for p can be found using PACF (Partial AutoCorrelation Function) plot whereas for q we use ACF(AutoCorrelation Function) plot.

2. For Multivariate Data:

Here we can use common approaches like taking the average of individual univariate time series anomaly scores to calculate the anomaly score for the entire multivariate series data. More sophisticated techniques include :-

a. VAR (Vector-based AutoRegressive) model. It can be described as an AR model for multivariate time series.
b. VARMA (Vector AutoRegressive Moving Average). Combination of AR model of order p and Moving Average model of order q for multivariate time series data.

2.2 Deep Learning Approach for Anomaly Detection

Neural Networks are used to identify anomalies in univariate as well as multivariate data.

Some of the techniques used in deep learning-based anomaly detection are:

a. Convolutional Neural Networks (CNN)
b. Residual Neural Networks (Resnet)
c. Long Short-Term Memory (LSTM) Networks

2.3 Machine Learning Approach for Anomaly Detection

Machine Learning approaches automate anomaly detection by identifying values that don't follow the normal pattern. It does not consider the underlying process of the data and operates without assuming a specific model. The theoretical or statistical foundation of the model is ignored and predictions are made on the time series data to identify anomalies.
This approach includes methods like :

1. Clustering Based: (Isolation Forest, DBSCAN - Density-Based Spatial Clustering of Applications with Noise, One Class SVM - One-Class Support Vector Machine)
2. Regression-Based: (XGBoost - Extreme Gradient Boosting, CatBoost Regressor, ET - Extra Trees Regressor)

2.3.1 Machine Learning Approach

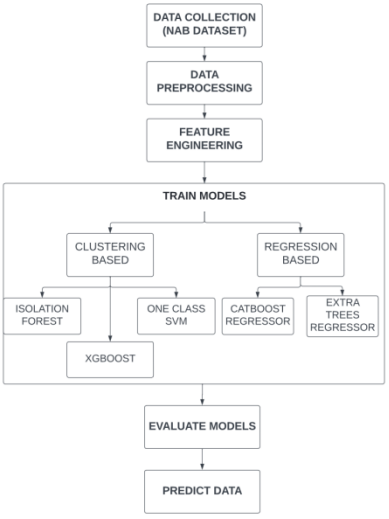

Fig 1: Block Diagram of Machine Learning Approach

This paper focuses on Machine Learning approaches specifically clustering and regression-based methods for anomaly detection.

- Clustering-Based Methods:

1. Isolation Forest:

Isolation forest using a sliding window is one of the machine learning approaches for detecting anomalies in time series. It builds an ensemble of Isolation Trees (iTrees), which are binary trees isolating data points. Anomalies are more likely to be isolated as they are closer to the root of an iTree [3,6]. Model training is completed once an ensemble of iTrees (Isolation Forest) is created.

The main challenges of the isolation forest algorithm are the following parameters:

- Window length (w)
- Number of iTrees in the iForest
- Contamination

2. One-Class SVM:

One-Class Support Vector Machine (SVM) is an unsupervised model for detecting anomalies. Unlike the regular supervised SVM, OC-SVM does not have target labels for the model training process. Instead, it learns the normal data point boundary and identifies data outside the border as anomalies.[8]

- Regression-Based Methods:

1. XGboost:

XGBoost is a distributed gradient boosting library that has been optimized to be highly efficient, flexible, and portable. XGBoost offers parallel tree boosting to solve a wide range of data science problems quickly and accurately.[4]
This is accomplished by combining decision trees to form a combined strong learner. When forecasting a time series, the model employs a lookback period to forecast a number of steps ahead.

2. Extra Trees Regressor:

It employs the perturb-and-combine technique made especially for trees. As with other classifiers, forest classifiers require the fitting of two arrays: an array Y of shape (n samples) storing the target values (class labels) for the training samples, and an array X of the form (n samples, n features) holding the training samples. Forests of trees, like decision trees, may solve problems with multiple outputs if Y is an array of shapes (n samples, n outputs). A random subset of candidate features is employed, just like in random forests, but instead of searching for the thresholds that are the most discriminative, the best of these randomly generated thresholds is chosen as the splitting criteria. This typically makes it possible to lower the model's variance.

3. Catboost Regressor:

CatBoost is an implementation of gradient boosting, [1] which uses binary decision trees as base predictors. A decision tree is a model built by a partition of the feature space Rm into several disjoint regions (tree nodes). Each final region (leaf of the tree) is assigned a value, which is an estimate of the response y in the region for the regression task or the predicted class label.

3. METHODOLOGY

- Components of Our Project:

Dataset Used :- NAB Dataset

	timestamp	value
0	2015-02-26 21:42:53	53
1	2015-02-26 21:47:53	64
2	2015-02-26 21:52:53	49
3	2015-02-26 21:57:53	48
4	2015-02-26 22:02:53	22
...	...	...
15828	2015-04-22 20:42:53	93
15829	2015-04-22 20:47:53	101

Fig 2: NAB Dataset

The Numenta Anomaly Benchmark (NAB) is an open-source framework for evaluating real-time anomaly detection algorithms. It consists of 50 labeled real-world and artificial time series data files plus a novel scoring mechanism. Here we have used the real tweets data (Twitter_volume_FB.csv) available in the NAB Dataset.

Feature Engineering: -

The raw data cannot be used for creating models and making predictions. So we apply feature engineering over this dataset to smoothen the data and extract features that can be used in the learning process.

timestamp	value	day	day_name	day_of_year	week_of_year	hour	is_weekday
2015-02-26 21:00:00	214	26	Thursday	57	9	21	4
2015-02-26 22:00:00	382	26	Thursday	57	9	22	4
2015-02-26 23:00:00	328	26	Thursday	57	9	23	4
2015-02-27 00:00:00	224	27	Friday	58	9	0	5
2015-02-27 01:00:00	182	27	Friday	58	9	1	5

Fig 3: Feature Engineering-Extracting Features from data

We have used two methods to detect Anomalies:

A. Clustering Based

B. Regression Based

A. Clustering Based:

1. Setting Up Environment:

In order to prepare the data for analysis we use the setup() function in Pycaret. This function takes a pandas DataFrame for training and other optional parameters which can be used to customize the pre-processing pipeline.

	Description	Value
0	session_id	123
1	Original Data	(1321, 7)
2	Missing Values	False
3	Numeric Features	4
4	Categorical Features	3
5	Ordinal Features	False
6	High Cardinality Features	False
7	High Cardinality Method	None
8	Transformed Data	(1321, 27)
9	CPU Jobs	-1
10	Use GPU	False

Fig 4: Initialising Setup

1. Create a Model:

To create models for anomaly detection we use the function create_model(). It allows us to create a model with only one mandatory parameter i.e the name of the model. We use the function models() to check the 12 ready-to-use models in Pycaret.anomaly module.

ID	Name	Reference
abod	Angle-base Outlier Detection	pyod.models.abod.ABOD
cluster	Clustering-Based Local Outlier	pyod.models.cblof.CBLOF
cof	Connectivity-Based Local Outlier	pyod.models.cof.COF
iforest	Isolation Forest	pyod.models.iforest.IForest
histogram	Histogram-based Outlier Detection	pyod.models.hbos.HBOS
knn	K-Nearest Neighbors Detector	pyod.models.knn.KNN
lof	Local Outlier Factor	pyod.models.lof.LOF
svm	One-class SVM detector	pyod.models.ocsvm.OCSVM

Fig 5: Checking list of available models

To create an Iforest model we used:
create_model("iforest').
The same syntax was followed for the other models.

2. Assign Model:

To analyze the results obtained using the created model and to assign anomaly labels to the data we use the assign_model() function. The output is an iForest model with 2.5% of outliers and Anomaly Scores assigned to the data in the dataset.

timestamp	value	day	day_name	day_of_year	week_of_year	hour	is_weekday	Anomaly	Anomaly_Score
2015-02-26 21:00:00	214	26	Thursday	57	9	21	4	1	0.002959
2015-02-26 22:00:00	382	26	Thursday	57	9	22	4	1	0.006903
2015-02-26 23:00:00	328	26	Thursday	57	9	23	4	1	0.007372
2015-02-27 00:00:00	224	27	Friday	58	9	0	5	0	-0.012385
2015-02-27 01:00:00	182	27	Friday	58	9	1	5	0	-0.015391

Fig 6: Trained Isolation Forest model with 2.5% outliers

3. Predictions using Model:

The model created can be used to predict unseen data to check model performance. Or can be used on the training data itself (not a good practice to do so)

```
iforest_results[iforest_results['Anomaly'] == 1].head()
```

timestamp	value	day	day_name	day_of_year	week_of_year	hour	is_weekday	Anomaly	Anomaly_Score
2015-02-26 21:00:00	214	26	Thursday	57	9	21	4	1	0.002959
2015-02-26 22:00:00	382	26	Thursday	57	9	22	4	1	0.006903
2015-02-26 23:00:00	328	26	Thursday	57	9	23	4	1	0.007372
2015-02-27 03:00:00	838	27	Friday	58	9	3	5	1	0.007157
2015-02-27 18:00:00	1098	27	Friday	58	9	18	5	1	0.013618

Fig 7: Displaying the anomalous points

The pre-processing steps remain the same. Now we create regression-based models to detect anomalies.

B. Regression Based:

1. Compare Models:

The Compare_models() function trains all models in the model library and scores them using k-fold cross-validation for metric evaluation. The output is a scoring grid that shows the average MAE, MSE, RMSE, R2, RMSLE, and MAPE across the folds of all the available models.

```
best = compare_models(sort = "MAE")
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	57.7239	8393.6260	91.0494	0.4780	0.4053	0.2942	0.4633
et	Extra Trees Regressor	58.9732	9040.9112	94.8169	0.4366	0.4177	0.3002	0.5233
gbr	Gradient Boosting Regressor	60.2876	9151.6719	95.3016	0.4399	0.4234	0.2973	0.0733
rf	Random Forest Regressor	60.4128	9209.9576	94.8195	0.4393	0.4108	0.3061	0.5600
xgboost	Extreme Gradient Boosting	60.4219	9643.7917	97.2959	0.4164	0.4235	0.3097	0.4200
lightgbm	Light Gradient Boosting Machine	68.8383	10777.5883	102.6116	0.3604	0.4504	0.3480	0.2433
dt	Decision Tree Regressor	69.1871	12453.2516	110.2540	0.2684	0.4480	0.3415	0.0167
knn	K Neighbors Regressor	71.9789	13683.2378	114.7781	0.1625	0.4590	0.3843	0.0800
huber	Huber Regressor	76.2290	13613.5243	115.6578	0.2218	0.6108	0.3839	0.0467
llar	Lasso Least Angle Regression	84.6140	13488.7738	115.8290	0.2078	0.5425	0.4786	0.0167

Fig 8: Metrics with average MAE, MSE, RMSE, R2, RMSLE, and MAPE showing the highest performing metric

In order to improve the training time, we have changed the fold parameter from the default value (10) to 3 using the fold parameters. For comparison purposes, the score grid above shows the highest-performing metric. It is sorted by changing the sort parameter to MAE (highest to lowest).

2. Create a Model:

create_model() function creates a model and scores it using stratified cross-validation.

```
et = create_model('et')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	64.7221	9148.7742	95.6492	0.5437	0.5280	0.3545
1	56.6171	7346.5967	85.7123	0.6210	0.3326	0.2582
2	55.5804	10627.3628	103.0891	0.1451	0.3925	0.2878
Mean	58.9732	9040.9112	94.8169	0.4366	0.4177	0.3002
Std	4.0871	1341.5370	7.1184	0.2085	0.0818	0.0403

Fig 9: Creating Extra Trees Regressor model

Similar to compare_models(), the output is a scoring grid that shows MAE, MSE, RMSE, R2, RMSLE, and MAPE by fold.

3. Tune Model:

tune_model() function is used to automatically tune the hyperparameters of a model on a predefined search space and to score it using k-fold cross-validation. The output is a scoring grid that shows MAE, MSE, RMSE, R2, RMSLE, and MAPE by fold

```
tuned_et = tune_model(et)
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	70.1928	11374.0536	106.6492	0.4327	0.5615	0.3710
1	65.7141	9356.0650	96.7268	0.5173	0.3811	0.3210
2	65.2407	9024.8268	94.9991	0.2740	0.4055	0.3770
Mean	67.0492	9918.3151	99.4584	0.4080	0.4494	0.3564
Std	2.2312	1038.2070	5.1334	0.1009	0.0799	0.0251

Fig 10: Tuning Extra Trees Regressor

MAE(et) < MAE(tuned_et) so we use the et model. The mean MAE for 'et' is 58.97 compared to 67.05 for 'tuned_et', which means 'et' is a better model. The same process was carried out for Catboost Regressor.

4. Predict on Test Data:

We separated 430 samples from the data as a test sample. All the evaluation metrics above are cross-validated results based on the training set only. We performed one final check before finalizing the model by predicting on test data.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Extra Trees Regressor	126.6618	37038.4131	192.4537	-0.937	0.6583	0.7512

timestamp	value	day	day_name	day_of_year	week_of_year	hour	is_weekday	Label
2015-04-05 00:00:00	83	5	Sunday	95	14	0	7	133.71
2015-04-05 01:00:00	113	5	Sunday	95	14	1	7	110.38
2015-04-05 02:00:00	70	5	Sunday	95	14	2	7	123.32
2015-04-05 03:00:00	112	5	Sunday	95	14	3	7	93.29
2015-04-05 04:00:00	157	5	Sunday	95	14	4	7	76.31
...	...	...	...	...	...	...	...	...
2015-04-22 17:00:00	432	22	Wednesday	112	17	17	3	277.26
2015-04-22 18:00:00	439	22	Wednesday	112	17	18	3	436.53

Fig 11: Prediction on test data using ExtraTreesRegressor

5. Identifying anomalies based on predictions:

timestamp	day	day_name	day_of_year	week_of_year	hour	is_weekday	value	Label	difference
2015-04-05 00:00:00	5	Sunday	95	14	0	7	83	133.71	-50.71
2015-04-05 01:00:00	5	Sunday	95	14	1	7	113	110.38	2.62
2015-04-05 02:00:00	5	Sunday	95	14	2	7	70	123.32	-53.32
2015-04-05 03:00:00	5	Sunday	95	14	3	7	112	93.29	18.71
2015-04-05 04:00:00	5	Sunday	95	14	4	7	157	76.31	80.69
...	...	...	...	...	...	...	...	...	...
2015-04-22 17:00:00	22	Wednesday	112	17	17	3	432	277.26	154.74
2015-04-22 18:00:00	22	Wednesday	112	17	18	3	439	436.53	2.47
2015-04-22 19:00:00	22	Wednesday	112	17	19	3	648	375.98	272.02
2015-04-22 20:00:00	22	Wednesday	112	17	20	3	1806	393.19	1412.81
2015-04-22 21:00:00	22	Wednesday	112	17	21	3	117	292.73	-175.73

430 rows × 9 columns

Fig 12: Calculating the Difference between Value and Label i.e. Original Value and Predicted Value

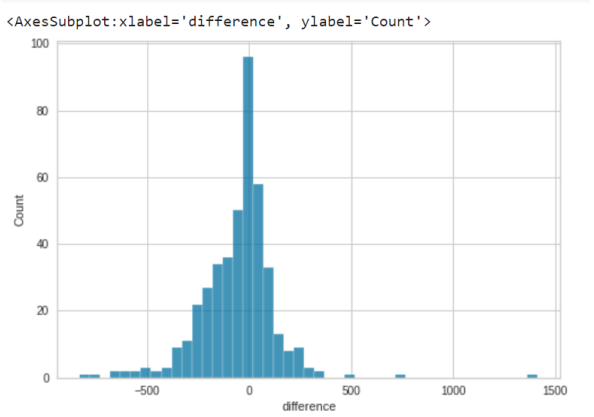

Fig 13: Plotting Histogram of Difference

- Identify threshold values above or below which a datapoint is considered as an outlier. We use the 1.5*IQR rule to identify anomalies. Q1 - Quartile 1 (25%), Q2 - Quartile 2 (50%), Q3 - Quartile 3 (75%).

  IQR - Interquartile Range = Q3 - Q1. To Calculate Acceptable Range we follow the following steps:

  1. Calculate IQR. Multiply 1.5
  2. Lower Bound = Q1 - 1.5IQR. Values Below Lower Bound are Outliers/Anomalies.
  3. Upper Bound = Q3 + 1.5IQR. Values Above Upper Bound are Outliers/Anomalies.

Acceptable Range = [Lower Bound, Upper Bound]

```
Lower_bound= -437.00
Upper_bound = 315.00
```

```
predictions["Anomaly_Score"]= predictions["difference"].apply(lambda x:1 if x<Lower_bound or x>Upper_bound else 0)
```

Fig 14: Setting upper and lower bound for finding anomalous points

6. Predict Unseen Data: Finally, we check which data points are anomalies and which are not.

```
predictions[predictions['Anomaly_Score']==1]
```

timestamp	day	day_name	day_of_year	week_of_year	hour	is_weekday	value	Label	difference	Anomaly_Score
2015-04-07 15:00:00	7	Tuesday	97	15	15	2	594	259.33	334.67	1
2015-04-08 14:00:00	8	Wednesday	98	15	14	3	718	246.08	471.92	1
2015-04-09 17:00:00	9	Thursday	99	15	17	4	599	248.57	350.43	1
2015-04-10 03:00:00	10	Friday	100	15	3	5	134	576.32	-442.32	1
2015-04-10 18:00:00	10	Friday	100	15	18	5	263	840.58	-577.58	1
2015-04-10 19:00:00	10	Friday	100	15	19	5	226	883.40	-657.40	1
2015-04-10 20:00:00	10	Friday	100	15	20	5	244	993.04	-749.04	1
2015-04-10 21:00:00	10	Friday	100	15	21	5	135	655.66	-520.66	1

Fig 15: Predictions on unseen data

4. RESULTS AND DISCUSSION

4.1 Plot Model

Then we analyzed our model to see how it performs using different aspects of the dataset. To do this, Pycaret provides us with the plot_model() function. We used clustering and regression-based models to test the performance.

4.1.1 For Clustering Based:

In Pycaret.anomaly module we have 2 plots:

- T-distributed Stochastic Neighbour Embedding (t-SNE)
- Uniform Manifold Approximation and Projection (UMAP)

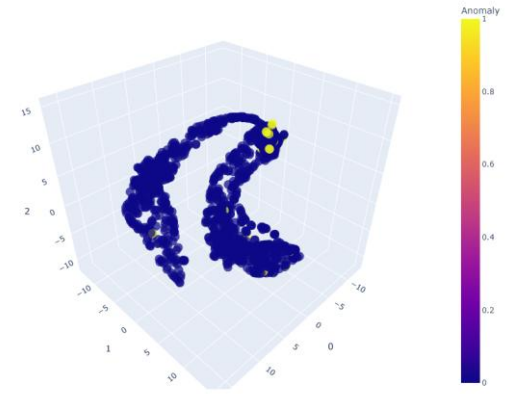

Fig 16: 3D TSNE Plot for Outliers (iforest model)

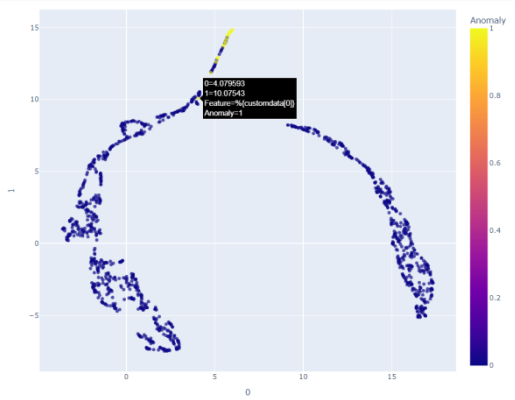

Fig 17: uMAP for outliers (iforest model)

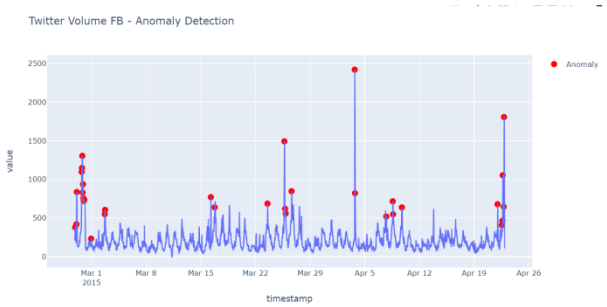

Fig 18: Line plot showing point anomalies(in red) using SVM model

4.1.2 For Regression Based:

We have used the plot_model() function to analyze the performance before model finalization across different aspects such as Validation Curve, Residuals Plot, and Prediction Error. It takes a trained model object and returns a plot based on the test / hold-out set.

Fig 19: Validation Curve for ExtraTreesRegressor

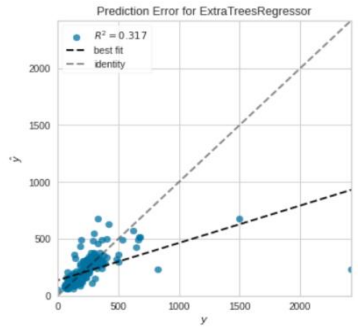

Fig 20: Prediction Error for ExtraTreesRegressor

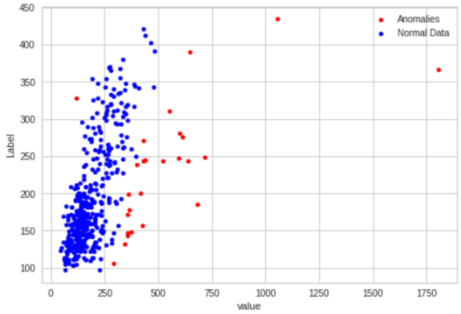

Fig 21: Scatter Plot showing Normal Data (in blue) vs. Anomalies (in red) using CatBoost

5. CONCLUSION AND FUTURE WORK

The primary objective of this project is to use PyCaret for anomaly detection in time series data. We made use of some of the most significant features and modules the library has to offer. Secondly, we explored the NAB twitter dataset and understood its key features. Both the clustering and regression-based approaches yielded different results. The difference in performance for clustering-based algorithms like Isolation forest and One-Class SVM is minor whereas, in the case of Regression-Based algorithms, Catboost performs better than Extra Trees Regressor when evaluated on the basis of metrics like MAE, MSE, and RMSE, In the future, we would like to work with more datasets of similar type and also wish to compare PyCaret with other tools like Facebook Prophet for anomaly detection in time series. We also plan to deploy the model on cloud platforms such as Heroku, AWS, or Azure.

6. REFERENCES

[1] Prokhorenkova L, Gusev G, Vorobev A, Dorogush A, Gulin A. CatBoost: unbiased boosting with categorical features.

[2] Astha G, Wenyu, Jules S, Ramasamy S, Chuan-Sheng F. An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. IEEE Transactions on Neural Networks and Learning Systems

[3] Yu Qin, YuanSheng Lou. Hydrological Time Series Anomaly Pattern Detection based on Isolation Forest. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019)

[4] Yiyang Niu. Walmart Sales Forecasting using XGBoost algorithm and Feature engineering. 2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)

[5] Subutai Ahmad, Alexander Lavin, Scott Purdy, Zuha Agha, “Unsupervised real-time anomaly detection for streaming data”, Neurocomputing, Volume 262, 1 November 2017, Pages 134-147, ISSN 0925-2312.

[6] S. Zhong, S. Fu, L. Lin, X. Fu, Z. Cui, and R. Wang, "A novel unsupervised anomaly detection for gas turbine using Isolation Forest," 2019 IEEE International

[7] Conference on Prognostics and Health Management (ICPHM), 2019, pp. 1-6, DOI: 10.1109/ICPHM.2019.8819409.

[8] Fattah, Jamal & Ezzine, Latifa & Aman, Zineb & Moussami, Haj & Lachhab, Abdeslam. (2018). Forecasting of demand using ARIMA model. International Journal of Engineering Business Management. 10. 184797901880867. 10.1177/1847979018808673.

[9] Y. Wang, J. Wong and A. Miner, "Anomaly intrusion detection using one-class SVM," Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., 2004, pp. 358-364, DOI: 10.1109/IAW.2004.1437839.

[10] Chun-Hsiang Lee, Xu Lu, Xiunao Lin, Hongfeng Tao, Yaolei Xue, and Chao Wu. 2020. Anomaly Detection of Storage Battery Based on Isolation Forest and Hyperparameter Tuning. In Proceedings of the 2020 5th International Conference on Mathematics and Artificial Intelligence (ICMAI 2020). Association for Computing Machinery, New York, NY, USA, 229–233. https://doi.org/10.1145/3395260.3395271

[11] Jebb, A. T., Tay, L., Wang, W., & Huang, Q. (2015). Time series analysis for psychological research: examining and forecasting change. Frontiers in Psychology. https://doi.org/10.3389/fpsyg.2015.00727

[12] Y. Jin, C. Qiu, L. Sun, X. Peng and J. Zhou, "Anomaly detection in time series via robust PCA," 2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE), 2017, pp. 352-355, DOI: 10.1109/ICITE.2017.8056937.

[13] Wentai Wu, Ligang He, Weiwei Lin, Yi Su, Yuhua Cui, Carsten Maple, Stephen A. Jarvis, "Developing an Unsupervised Real-time Anomaly Detection Scheme for Time Series with Multi-seasonality," in IEEE Transactions on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2020.3035685.