



ANOMALY DETECTION FOR DATABASE CONNECTIONS

June - August 2017

AUTHOR:

Swapneel Mehta
CERN Openlab, IT/DB

SUPERVISORS:

Prasanth Kothuri,
Daniel Lanza Garcia
IT/DB



PROJECT SPECIFICATION



The project entails an investigation of unsupervised learning approaches for analysing database connection logs across different instances of databases within the CERN network. The log data is stored using “notifications” sent via Apache Flume to Hadoop Distributed File System (HDFS) for long-term storage. These notifications are also sent to Elasticsearch and visualised using Kibana over a short-term period of upto two months.

The data collected is extracted from this data lake created in order to store the connection logs for varying durations of time as specified. We would filter and process the indicators within the data in order to derive some analysis from it. Specifically, we aim to provide a comparative evaluation of the performance of different machine learning models in predicting anomalies or outliers within this data and flag these for further investigation and to elucidate the rationale behind the approach implemented in order to perform this evaluation.





ABSTRACT



The project investigates the nature of database connection logs by analysing these logs for potential anomalies. We apply different models to the data in building an ensemble of classifiers that is able to flag potentially anomalous or malicious connections to the database instances within the CERN network. Further, we also utilise this research to shed light on usage patterns within the network in order to better understand the temporal dependencies and implement them in the decision-making process within the CERN system.

These models are trained on subsets of a data lake that comprises daily connection logs across all instances of databases on the network. The data lake comprises Javascript Object Notation (JSON) logs that may be visualised using short-term storage, Elasticsearch, Grafana and Kibana or pushed to long-term storage on Hadoop Distributed File Storage (HDFS). We extract subsets of this data and apply models that vary based on different parameters of the data, and enable us to classify the outliers among the dataset as anomalies. The approaches adopted vary based on distance, density, and classification policies. Further, we juxtapose the results of the models against each other in order to understand the distribution of anomalies and eliminate false positives occurring across different evaluative models.





TABLE OF CONTENTS

INTRODUCTION	01
<hr/>	
LOG STORAGE	02
<hr/>	
ANOMALY DETECTION	03
<hr/>	
MODELS	04
K NEAREST NEIGHBOURS	
K MEANS CLUSTERING	
ISOLATION FORESTS	
LOCAL OUTLIER FACTOR	
ONE-CLASS SUPPORT VECTOR MACHINE	
<hr/>	
EVALUATION	05
<hr/>	
FUTURE SCOPE	07
<hr/>	
CASE STUDY: ANOMALY DETECTION AT CERN	08



1. INTRODUCTION

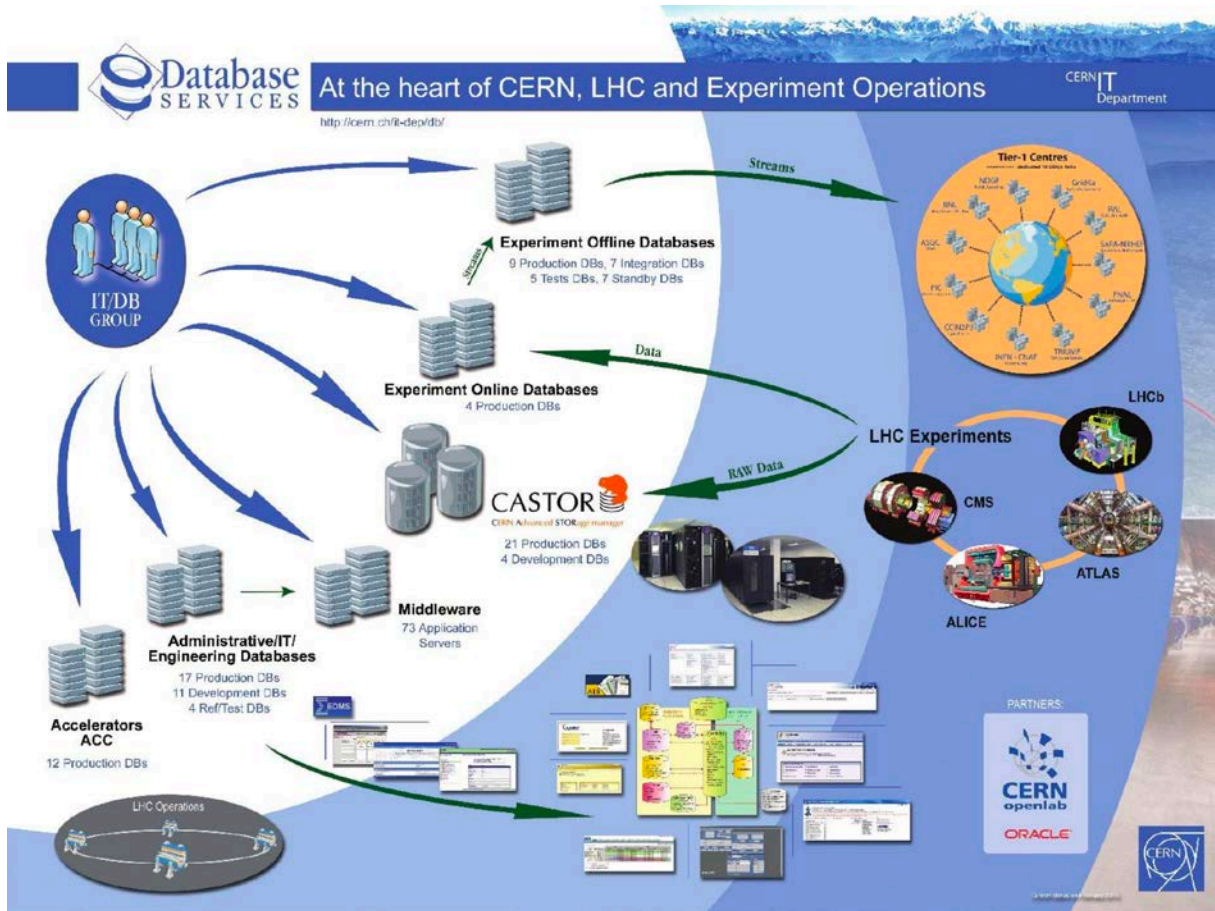
CERN follows an open access policy for most of its internal systems which extends to a significant percentage of database instances across the organisation. While this reduces the rule-based access control load on the system, it leads to a number of novel issues including but not limited to an unnecessarily large number of connection requests, connections accessing more resources than necessary, misused or abused credentials and more security-oriented issues such as malware causing repetitive connections from a host machine.

The CERN network comprises 10,000 active users and 1,500 different users^[1] each with devices, including personal and officially supplied equipment. While most of the software installed on the systems is regulated and updates are managed centrally, there is considerable privilege afforded to a user to manage device personally and consequently the loopholes for the installation of user-defined software or scripts. These can end up being not only inefficient but also malicious in their operation, raising multiple issues within the system. Specifically, these systems might be compromised and utilised to either initiate multiple (spam) connections or use stolen credentials with malicious intent. There are numerous security issues with such a setup if there is unmonitored access to database instances. It can be used in multiple manners to throw the network into disarray due to the load on servers.

We aim to use some concrete approaches in analysing these database connections by building models that can accurately fit this data and classify further connection requests by evaluating the likelihood of being anomalous. The data is analysed over several different parameters in order to understand disparate patterns that may not be present across all the sets of extracted data. We compare and contrast the results by viewing a common plot that sheds light on the nature of differing approaches.

The motivation underlying the project is twofold: there is no doubt a security angle but there are also the usage patterns that we seek to understand from this research. Usage patterns often impart crucial insights that may have been either too simple and thus overlooked or too complex to predict upon a cursory look at the data. We aim to investigate these anomalies in the patterns and analyse database connections at CERN.





CERN Databases Architecture (Eric Grancher, Luca Canali, IT-DB; 2013)



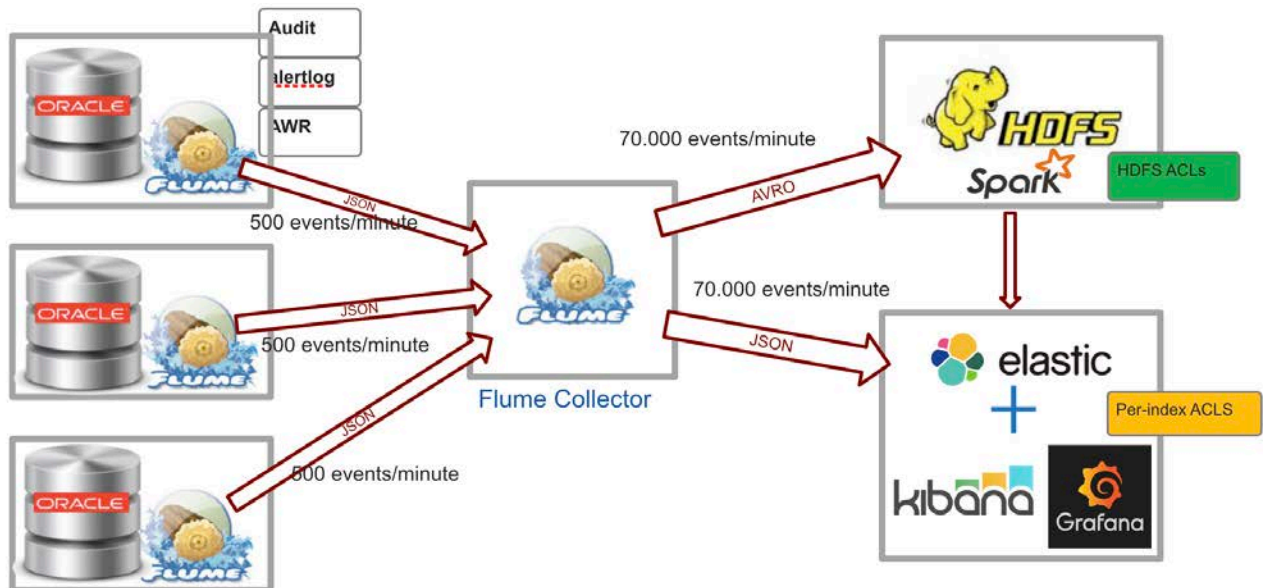
2. LOG STORAGE

Database connection logs comprise of multiple parameters per successfully established or failed connection which are utilised in building the feature vector for analysis. The architecture comprises of multiple instances of Oracle databases each of which have an Apache Flume agent running on it.

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tuneable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application^[2]. Unbeatable Database Performance, Scalability, and Reliability.

Oracle Real Application Clusters (Oracle RAC) is a clustered version of Oracle Database based on a comprehensive high-availability stack that can be used as the foundation of a database cloud system as well as a shared infrastructure, ensuring high availability, scalability, and agility for any application^[3].

The log data is transported via a central Flume Collector to Elasticsearch and Kibana for short-term storage primarily



meant for data visualisation and long-term storage on HDFS.



The log consists of a number of different fields each with a value corresponding to the details of the connection. Some of these connection details include:

- metadata: can be ignored
- data:
- database_type: type of database which currently is always oracle.
- listener_name: there are several listeners, this attribute not considered for analysis.
- service_name: name of the database that the client is connecting with.
- source_type: we collect different type of data; non-listener records can be filtered out.
- text: entire record extracted from database; already parsed to other attributes, ignored.
- event_timestamp: time when connection took place.
- type: listener event type; analysis considers "establish" type; describes the connections.
- flume_agent_version: static attribute, ignored.
- return_code: returns 0, representing the event was processed successfully.
- oracle_sid: client connect to that instance.
- client_program: software that the client uses to connect.
- client_host: host from where the client connects (not necessarily specified).
- client_ip: IP address of client.
- client_port: port user uses to connect.
- CONNECT... : parameters extracted from the second value of text; can vary.
- client_protocol: network protocol user uses to initiate connection.





- client_user: user name used to initiate

```

{
  "metadata": {
    "listener_name": "listener",
    "source_type": "listener",
    "type": "dbconnection",
    "database_type": "oracle",
    "timestamp_format": "yyyy-MM-dd",
    "hostname": "itrac1312.cern.ch",
    "type_prefix": "logs",
    "producer": "oracle",
    "json": "true",
    "topic": "oracle_logs",
    "flume_agent_version": "0.1.6-7.e16",
    "oracle_sid": "INT8R_RAC13",
    "timestamp": "1498034136989"
  },
  "data": {
    "client_program": "",
    "listener_name": "listener",
    "CONNECT_DATA_COMMAND": "status",
    "CONNECT_DATA_VERSION": "202375424",
    "source_type": "listener",
    "event_timestamp":
    "2017-06-21T10:35:24+0200",
    "type": "status",
    "database_type": "oracle",
    "CONNECT_DATA_SERVICE": "LISTENER",
    "client_user": "oracle",
    "CONNECT_DATA_ARGUMENTS": "64",
    "client_host": "itrac1312.cern.ch",
    "flume_agent_version": "0.1.6-7.e16",
    "return_code": "0",
    "oracle_sid": "INT8R_RAC13"
  }
}

```

connection.

Log Storage Data

3. ANOMALY DETECTION

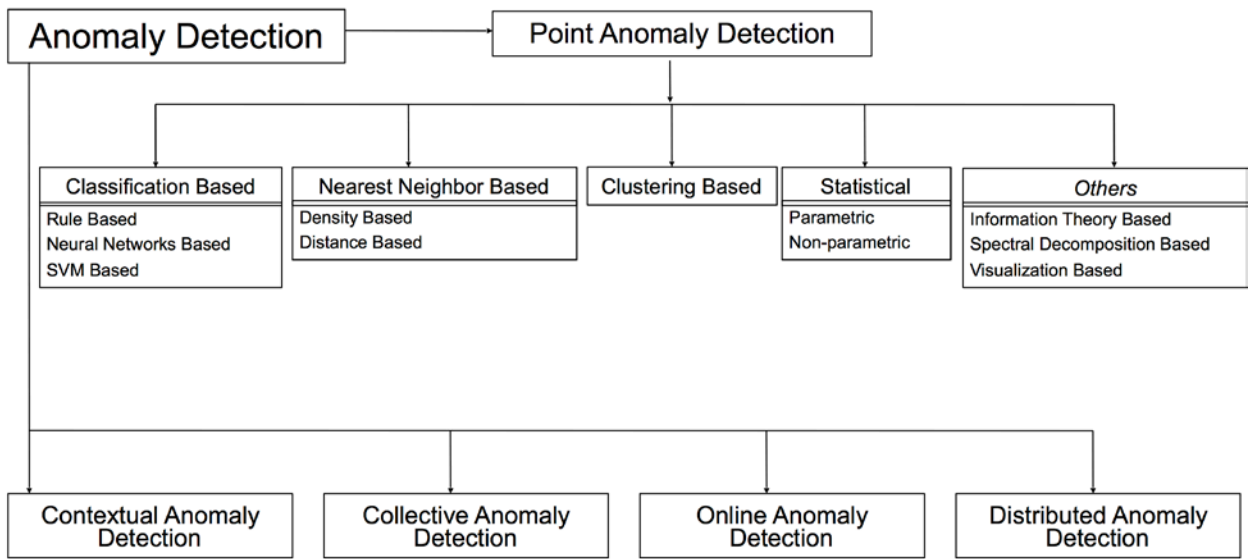
Anomaly Detection is the process of identifying unusual or outlying data from a given dataset. It has been a subject of interest as the amount of data available for analysis rises disproportionately as compared to the amount of available tagged data. Specifically, we utilise features extracted from established connection logs in order to determine the outliers for a given subset of temporal data obtained from a data lake.





There are many approaches adopted in order to solve the outlier detection problem; some widely applicable ones being unsupervised approaches that forego the need for labeled training data. These approaches involve clustering of data based on different metrics including distance, density and so on. Real-time anomaly detection is a heavily studied subject involving both supervised and unsupervised approaches.

Overview of Anomaly Detection Approaches



* Outlier Detection – A Survey, Varun Chandola, Arindam Banerjee, and Vipin Kumar, Technical Report TR07-17, University of Minnesota (Under Review)

4. MODELS

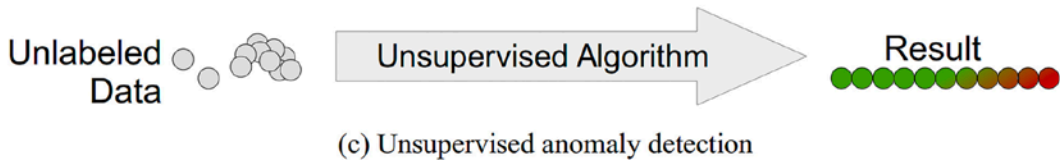
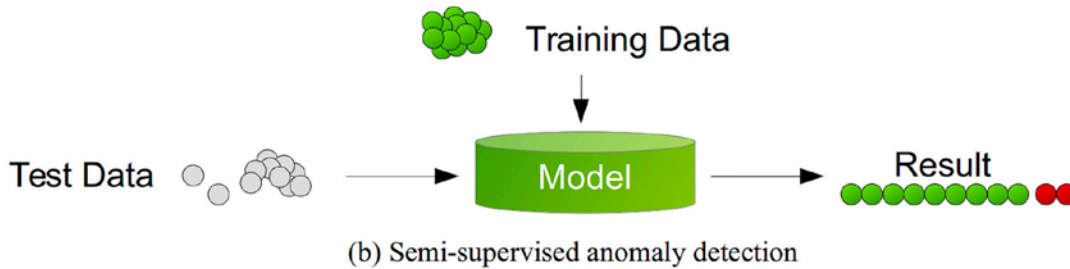
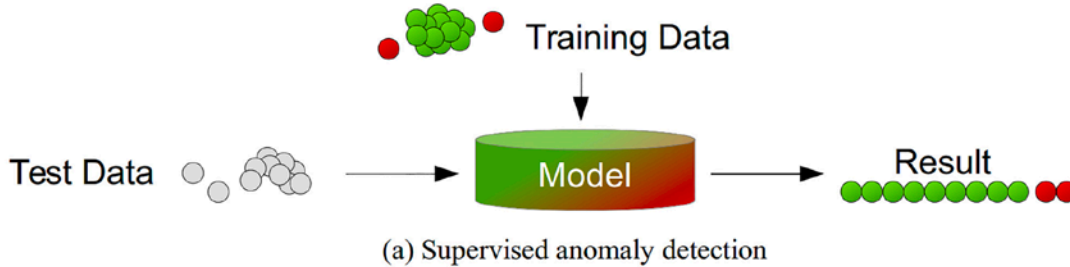
Supervised anomaly detection uses a fully labeled dataset for training. Semi-supervised anomaly detection uses an anomaly-free training dataset. Afterwards, deviations in the test data from that normal model are used to detect





anomalies. Unsupervised anomaly detection algorithms use only intrinsic information of the data in order to detect instances deviating from the majority of the data^[5].

We adopt an ensemble of these approaches to analyse the dataset.



In machine learning and statistics, dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration, via obtaining a set of principal variables. It typically involves feature selection and feature extraction.

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.





Graph depicting subset of Data following Dimensionality Reduction using PCA

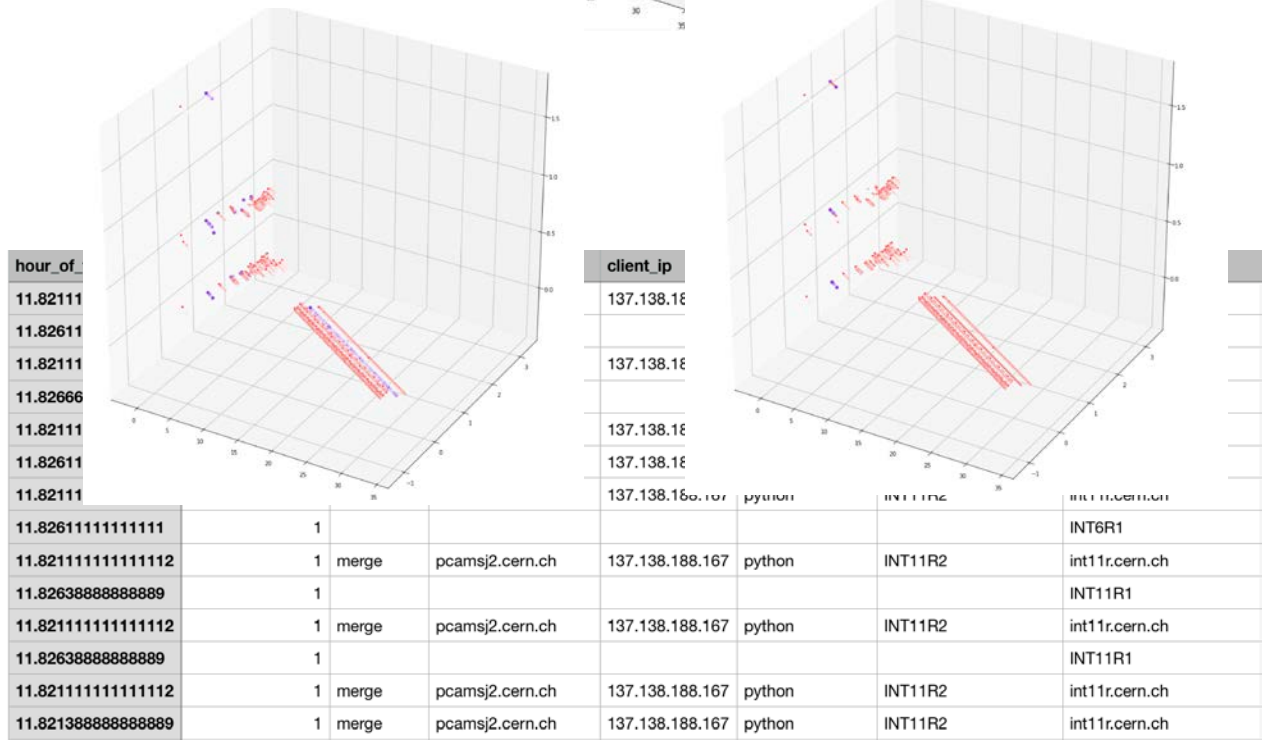
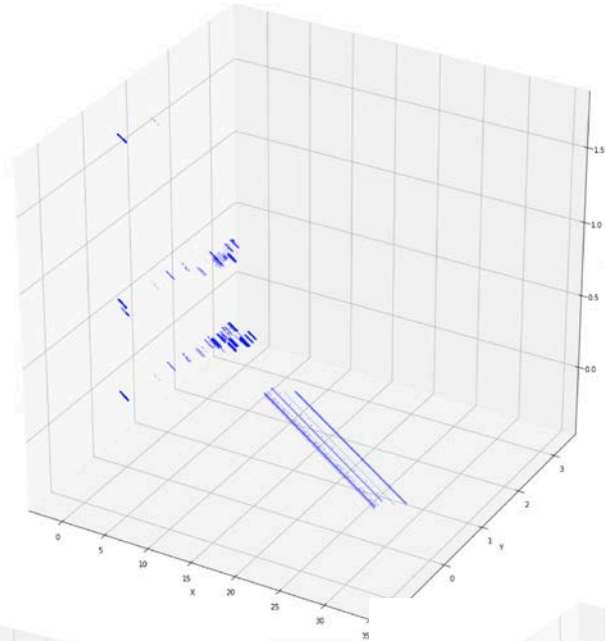
Building a Feature Vector from the Filtered Details among the Log Data





a. K-Nearest Neighbours

Unsupervised nearest neighbours is the foundation of many other learning methods, notably manifold learning and spectral clustering. Supervised neighbours-based learning comes in two flavours: classification for data with discrete labels, and regression for data with continuous labels. The principle behind nearest neighbour methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The user-defined constant (k- or vary based on the local density of points (radius-learning). The distance can, in general, be any metric: standard Euclidean distance is the most common choice.

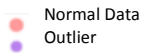


methods are known as *non-generalising* machine learning methods, since they simply “remember” all of its training data (possibly transformed into a fast indexing structure such as a Ball Tree or KD Tree.).





Despite its simplicity, nearest neighbours has been successful in a large number of classification and regression problems, including handwritten digits or satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular^[5].


 Normal Data
 Outlier

b. K-Means Clustering

The K-Means algorithm clusters data by trying to separate samples in n groups of equal variance, minimising a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

Left to Right: Filtering Number of Anomalies by Modifying
Clustering Parameters

The k-means algorithm divides a set of N samples M into K disjoint clusters C , each described by the mean of the samples in the cluster. The means are commonly called the cluster “centroids”; note that they are not, in general, points from X , although they live in the same space.

Inertia, or the within-cluster sum of squares criterion, can be recognised as a measure of how internally coherent clusters are. It suffers from various drawbacks:

- Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.





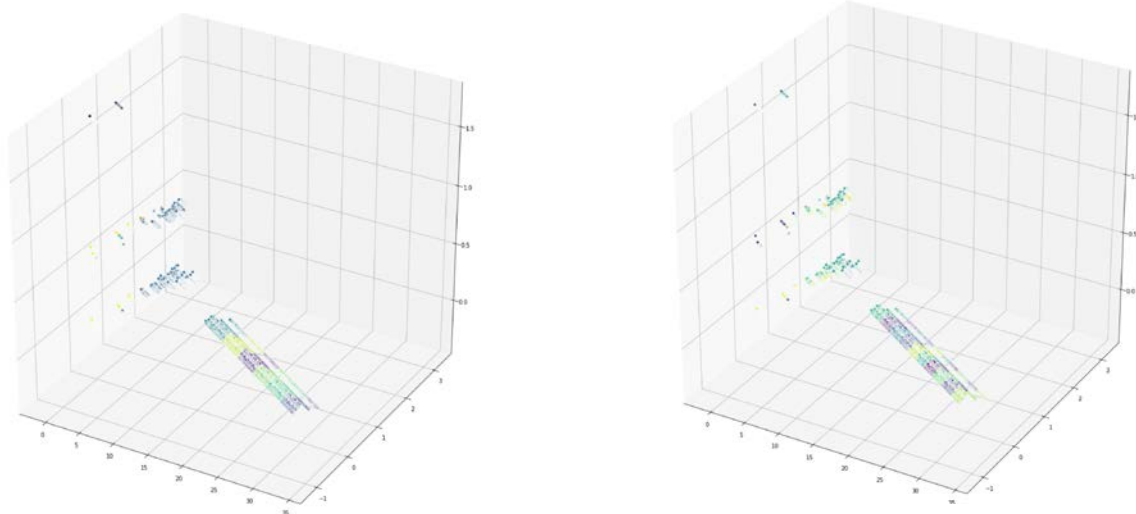
- Inertia is not a normalised metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called “curse of dimensionality”). Running a dimensionality reduction algorithm such as PCA prior to k-means clustering can alleviate this problem and speed up the computations.

K-means is often referred to as Lloyd’s algorithm. In basic terms, the algorithm has three steps. The first step chooses the initial centroids, with the most basic method being to choose k samples from the dataset X . After initialisation, K-means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid. The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid. The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly^[5].

Left to Right: Increasing Number of Clusters for Optimal Grouping of Data

c. Isolation Forests

One efficient way of performing outlier detection in high-dimensional datasets is to use random forests. The IsolationForest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

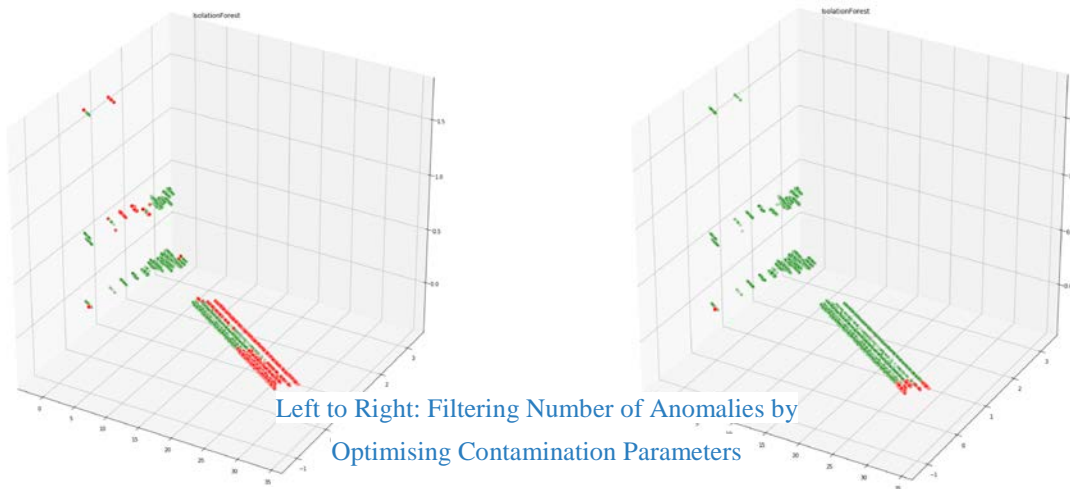




Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length, averaged over a forest of such random trees, is a measure of normality and our decision function. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies^[5].

● Normal Data
● Outlier



d. Local Outlier Factor

Another efficient way to perform outlier detection on moderately high dimensional datasets is to use the Local Outlier Factor (LOF) algorithm.



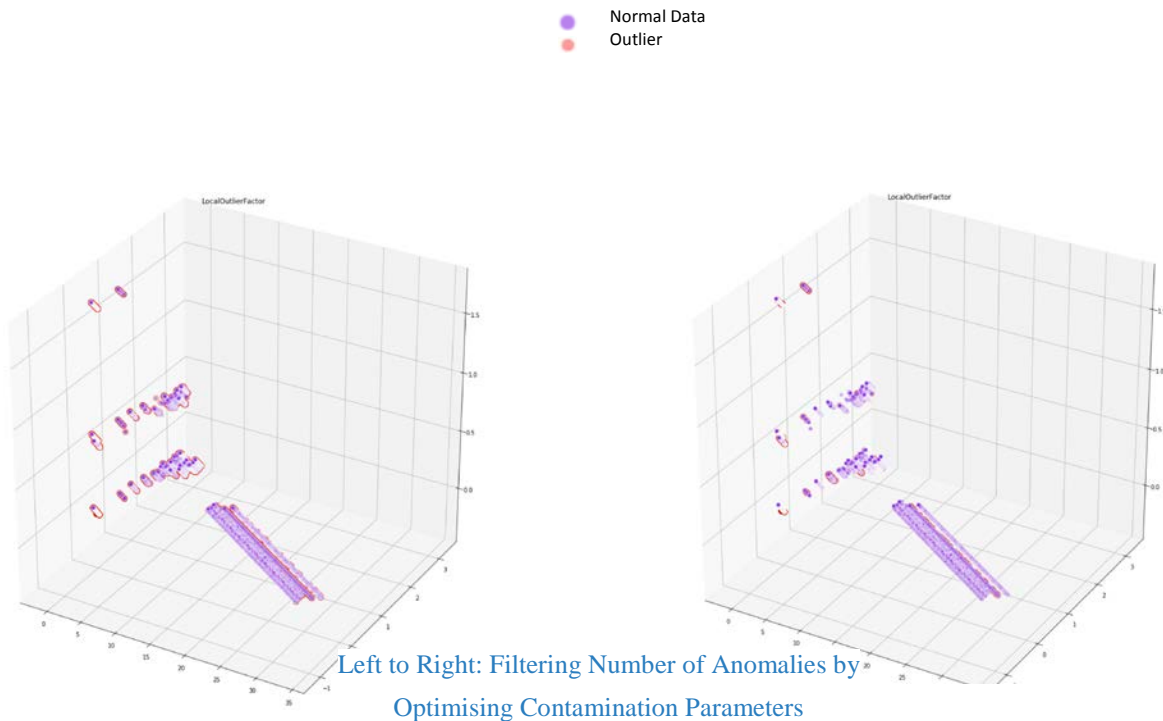


The LocalOutlierFactor (LOF) algorithm computes a score (called local outlier factor) reflecting the degree of abnormality of the observations. It measures the local density deviation of a given data point with respect to its neighbours. The idea is to detect the samples that have a substantially lower density than their neighbours.

In practice the local density is obtained from the k-nearest neighbours. The LOF score of an observation is equal to the ratio of the average local density of his k-nearest neighbours, and its own local density: a normal instance is expected to have a local density similar to that of its neighbours, while abnormal data are expected to have much smaller local density.

The number k of neighbours considered, greater than the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster, and 2) smaller than the maximum number of close by objects that can potentially be local outliers. When the proportion of outliers is high (i.e. greater than 10 %), number of neighbours should be set to a greater value.

The strength of the LOF algorithm is that it takes both local and global properties of datasets into consideration: it can perform well even in datasets where abnormal samples have different underlying densities. The question is not, how isolated the sample is, but how isolated it is with respect to the surrounding neighbourhood^[5].





e. One-Class Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

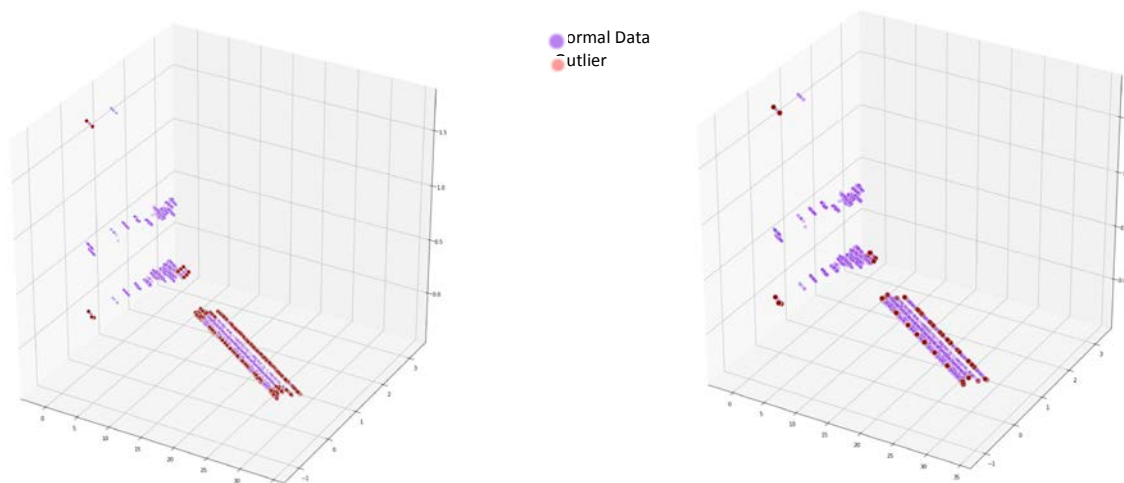
The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularisation term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

The support vector machines in support both dense and sparse sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data.

One-class SVM is used for novelty detection, that is, given a set of samples, it will detect the soft boundary of that set so as to classify new points as belonging to that set or not. The class that implements this is called OneClassSVM.

In this case, as it is a type of unsupervised learning, the fit method will only take as input an array X, as there are no class labels^[6].

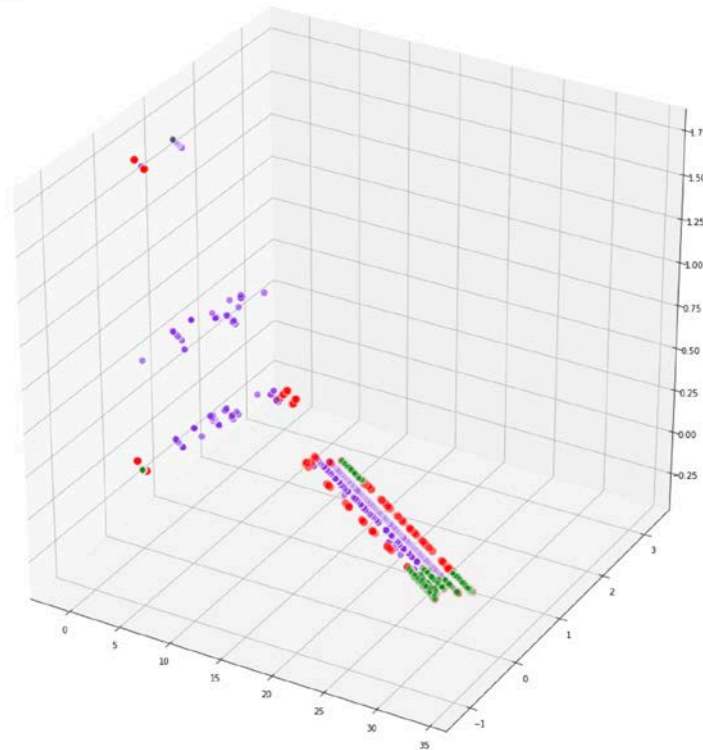




Left to Right: Filtering Number of Anomalies by Optimising Contamination Parameters

5. EVALUATION

We
by
order
the
model
how it
no
merit,
to
of a
since



would
investigation into whether the flagged data points are anomalies or false positives, we simply cross-verify the anomalies obtained by one model with the anomalies generated by another model and use an ensemble of these in order to flag the most common anomalies across multiple models.

evaluate the models
juxtaposing the plots
against each other in
to better understand
parameters each
attaches most
importance to and
affects the
distribution of
anomalies across the
dataset. In other
words, since there is
absolute figure of
or confusion matrix
evaluate these
models on the basis
pre-determined
benchmark, and
this is a proof-of-
concept for flagging
anomalous database
connections that
require further





Juxtaposition of anomalies of multiple models in order to
filter the common ones

6. FUTURE SCOPE

Future work in this regard could focus on implementing temporal approaches to this analysis, introducing a concrete dependency on time as compared with the timestamp parameter that has been incorporated into the feature vector in the previous models. Further, there is a possibility to build richer features by leveraging available data regarding failed connection requests from the log data at the point where the original filter for “established” connections was applied. A great amount of current research focusses on the application of neural networks to unsupervised learning, such as replicator neural networks, recurrent neural networks and self-growing neural networks, to name a few. Thus, the incorporation of such approaches could be interesting to note when contrasted with the results of the current models.





7. CASE STUDY: ANOMALY DETECTION AT CERN

While a number of projects undertaken on a personal as well as organisational level involve similar high-level practices involving anomaly detection on different data, there exists no common system to compare and contrast approaches applied on these projects. There have been cases of common practices applied on different datasets providing similar results however due to the lack of a common platform to showcase these approaches, there is a difficulty in understanding the pre-existing research undertaken in the subject area. We aim to resolve this issue by proposing a system that can be utilised by multiple entities, researchers and developers alike, in showcasing their datasets, approach adopted, and results. This is undertaken in order to make the subject of anomaly detection much easier to interpret and provide a solid foundation for further research in this area.

The application is built in Django utilising the Gentelella and comprises of a system that can extract parameters from within datasets and offer a visual interface to navigate through the process of analysis. It displays various statistics for the users or administrators to view on a dashboard, displaying the most used models, latest updates and messages within the system. It offers the functionality to choose the parameters including time period to use within different models and accordingly analyse the given data. Finally, it allows the plotting of two dimensional and three dimensional graphs and charts in order to allow users to use these interactive plots and understand the analysis intuitively.

Primarily meant for unsupervised approaches involving clustering, the system can be extended to supervised learning as well. A few screen-captures have been attached for a better visualisation of the proposed system titled - Anomaly Detection Project Exposition and Extension.

8. ACKNOWLEDGEMENTS

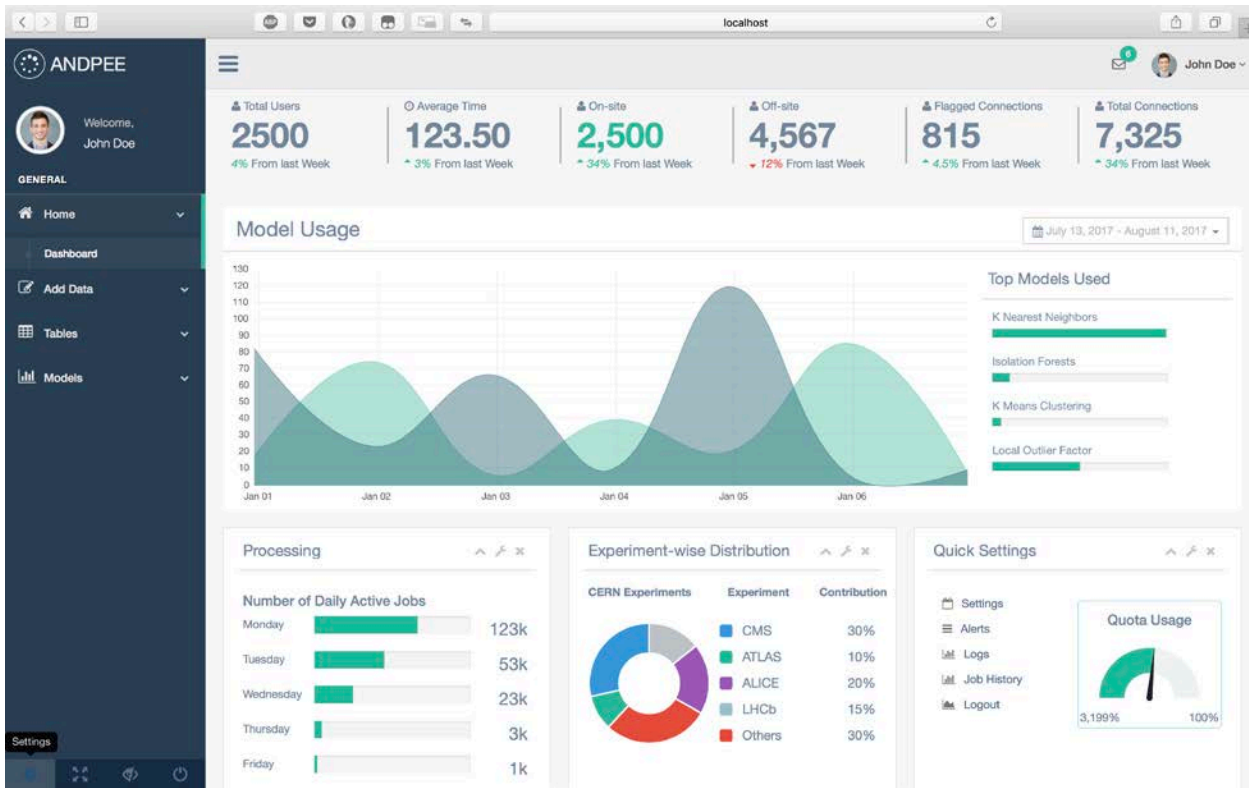
I would like to thank my guides, Mr. Prasanth Kothuri and Mr. Daniel Lanza Garcia, for their encouragement and support. They have been not only proactive in resolving my issues but also in directing me to find others who could better enable me to move ahead with my research. I would like to extend my gratitude to Dr. Jean-Roch Vlimant, Mr.



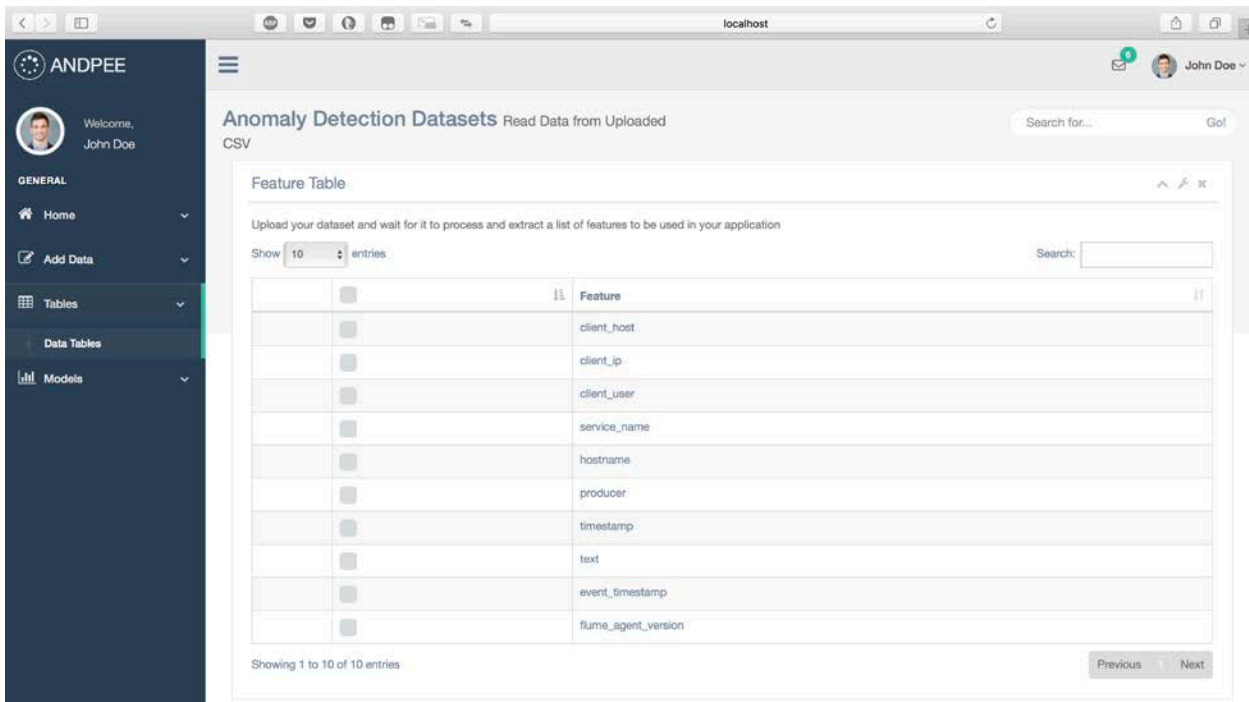


Adrian Alan Pol, and Mr. Michael Davis. Finally I would like to thank the IT-DB-SAS Team including and especially





User/Administrator Dashboard with System Statistics



Automated Feature Extraction from Dataset and Feature Selection for Model

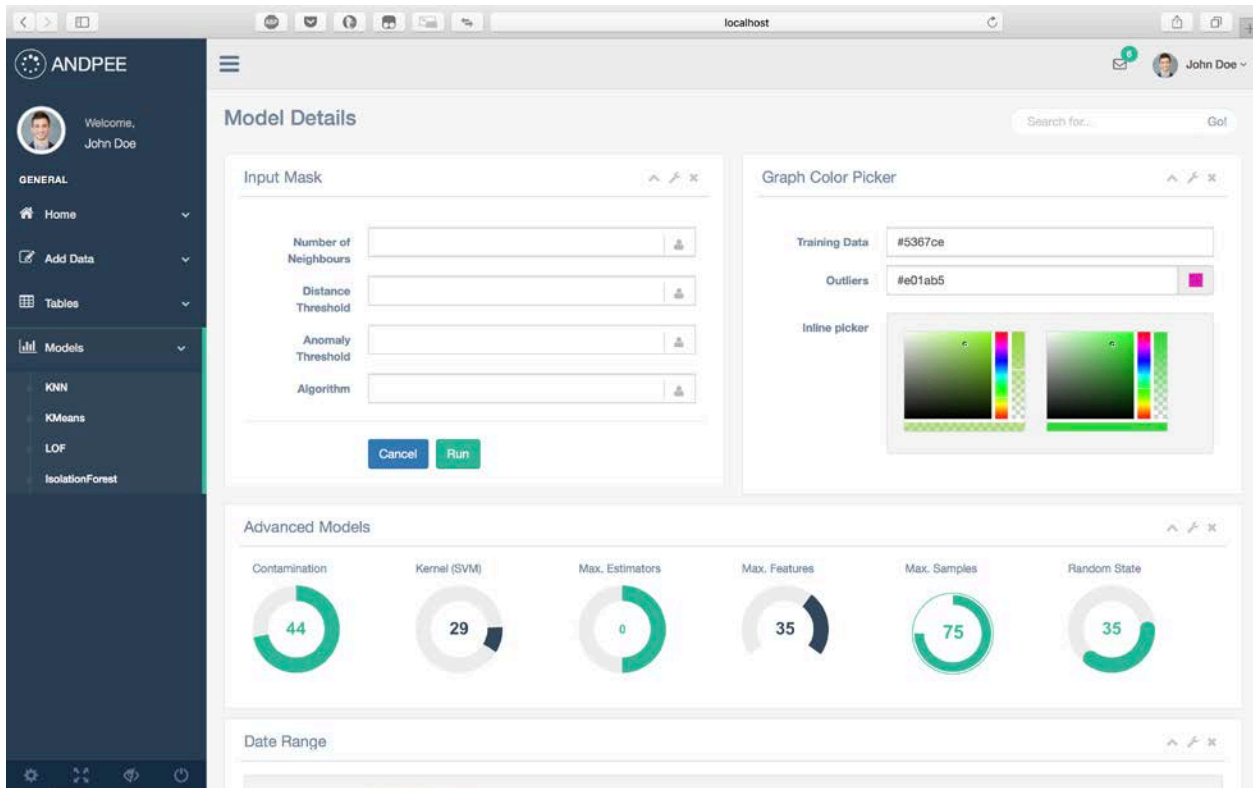
Mr. Eric Grancher and Mr. Luca Canali for the warm welcome extended to Summer Students within the Group.



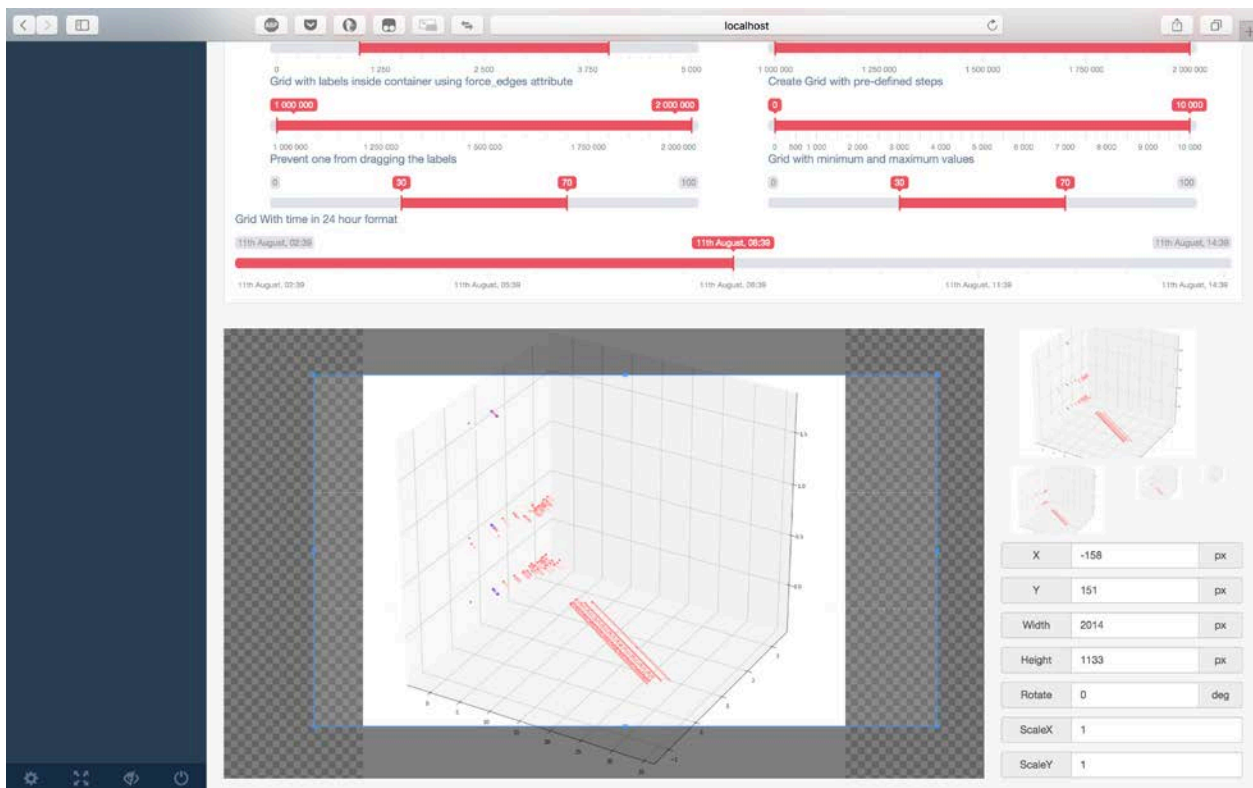
9. REFERENCES

- [1] Oracle, “Oracle Developer Suite 9”, Online. Available: <http://www.oracle.com/technetwork/developer-tools/developer-suite/cern-084110.html>
- [2] Apache, “Apache Flume 1.7.0 Documentation”, User Guide. Available: <https://flume.apache.org/>
- [3] Oracle, “Oracle Real Application Clusters”, Whitepaper. Available: <http://www.oracle.com/technetwork/database/options/clustering/rac-wp-12c-1896129.pdf>
- [4] M. Goldstein, S. Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms, 2016.
- [5] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011
- [6] Estimating the support of a high-dimensional distribution Schölkopf, Bernhard, et al. Neural computation 13.7 (2001): 1443-1471.





Setting Parameters for the Selected Model



Visualising the Interactive Plots