DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Clustering Based Outlier Detection for Improved Situation Awareness within Air Trac Control

HANNA GUSTAVSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Clustering Based Outlier Detection for Improved Situation Awareness within Air Trac Control

HANNA GUSTAVSSON

Degree Projects Systems Engineering (30 ECTS credits)
Degree Programme in Aerospace Engineering (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at Saab: Daniel Jonasson
Supervisor at KTH: Johan Karlsson
Examiner at KTH: Johan Karlsson

TRITA-SCI-GRU 2019:396
MAT-E 2019:90

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

The aim of this thesis is to examine clustering based outlier detection algorithms on their ability to detect abnormal events in flight traffic. A nominal model is trained on a data-set containing only flights which are labeled as normal. A detection scoring function based on the nominal model is used to decide if a new and in forehand unseen data-point behaves like the nominal model or not. Due to the unknown structure of the data-set three different clustering algorithms are examined for training the nominal model, K-means, Gaussian Mixture Model and Spectral Clustering. Depending on the nominal model different methods to obtain a detection scoring is used, such as metric distance, probability and One-Class Support Vector Machine.

This thesis concludes that a clustering based outlier detection algorithm is feasible for detecting abnormal events in flight traffic. The best performance was obtained by using Spectral Clustering combined with a One-class Support Vector Machine. The accuracy on the test data-set was 95.8%. The algorithm managed to correctly classify 89.4% of the data-points labeled as abnormal and correctly classified 96.2% of the data-points labeled as normal.

Förbättrad översiktsbild inom flygtrafikledning med hjälp av klusterbaserad anomalidetektering

Sammanfattning

Syftet med detta arbete är att undersöka huruvida klusterbaserad anomalidetektering kan upptäcka onormala händelser inom flygtrafik. En normalmodell är anpassad till data som endast innehåller flygturer som är märkta som normala. Givet denna normalmodell så anpassas en anomalidetekteringsfunktion så att data-punkter som är lika normalmodellen klassificeras som normala och data-punkter som är avvikande som anomalier. På grund av att strukturen av nomraldatan är okänd så är tre olika klustermetoder testade, K-means, Gaussian Mixture Model och Spektralklustering. Beroende på hur normalmodellen är modellerad så har olika metoder för anpassa en detekteringsfunktion används, så som baserat på avstånd, sannolikhet och slutligen genom One-class Support Vector Machine.

Detta arbete kan dra slutsatsen att det är möjligt att detektera anomalier med hjälp av en klusterbaserad anomalidetektering. Den algoritm som presterade bäst var den som kombinerade spektralklustring med One-class Support Vector Machine. På test-datan så klassificerade algoritmen 95.8% av all data korrekt. Av alla data-punkter som var märka som anomalier så klassificerade denna algoritm 89.4% rätt, och på de data-punkter som var märka som normala så klassificerade algoritmen 96.2% rätt.

Contents

Glossary

VFR Visual Flight Rules. Set of rules that apply when whether is clear enough for pilot to navigate by Visual reference.

IFR Instrumental Flight Rules. Relay on instruments and may be assisted by air traffic control.

AGL Above Ground Level

MSL Mean Sea Level

1 Introduction

Every year, the Swedish airspace is trafficked by 700,000 flights. To keep the airspace safe it is of importance to have competent air traffic controllers. Their task is to organize and direct aircrafts safely and efficiently towards their final destination [16]. Within the military, the air traffic controllers main task is to get a correct and identified picture of the air traffic. They must be able to identify aircrafts and if they are allowed within the Swedish airspace, which can be a challenging task. They use several sources of information which demands the air traffic controllers to be alert, clear-headed and must be able to do the right priorities [8].

The monitoring of air traffic involves a multitude of sensors and sensor types, such as ground Secondary Surveillance Radars (SSRs) and aircraft bases radar transponders. Aircraft transponders sends a coded message that contains data such as aircraft identification and altitude. A ground station uses the radar data to determine direction of flight and distance, this information is then transmitted to the air traffic controller. In prior to a civil departure, a flight plan containing the planned route must be handed in to the local civil aviation authority. The air traffic controller monitors that the air traffic is following their planned routes, altitudes and speeds [31]. Due to the large quantity of data to monitor, a robust exploitation of the data can improve operations by identifying threats, abnormalities and reduce prosecution time. The task of a robust exploitation is challenging due to the large, and increasing amount of data that is collected [14].

Exploitation of data to find abnormalities is commonly called *anomaly detection* or *outlier detection*. The aim with anomaly detection is to identify rare items that differs from the larger part of the data. This type of analysis is not new and has been used to find bank frauds, medical problems, structural defect, to mention a few. A commonly used approach for anomaly detection is to impose a maximum and minimum acceptance value. If a value is outside this bound it will be classified as an outlier. Thus, when using complex data with numerous measurements, this approach tends to have a large number of false and missing alarms [7]. Typically a more complex model must be used, which can handle the correlation between different measurements and attributes. By using machine learning techniques it is possible to build such model given a training data-set [23]. For instance, in [14] they discussed a development of an *Automated Detection Processor* based on clustering and Bayesian analysis. When analysing simulated ship track data, containing mine laying behaviour amongst maritime non-combatants, their algorithm did successfully detected anomalies. Inspired by this, [26] used the same approach to detect abnormalities in road traffic data. By using clustering technique, they grouped normal behaviour. To classify if an event was normal or abnormal they used Bayesian statistics to decide the probability of a normal event given the observed data x.

The aim of this thesis is to examine clustering based outlier detection of air

traffic data. The purpose is to improve the situation awareness and reduce work load for air traffic controllers.

Clustering methods will be used to train a nominal model. The nominal model will represent the "normal" air traffic situation without any abnormal events. Due to the complexity of the data and the unknown behaviour of it, three different clustering algorithms will be examined, *K-means*, *Gaussian Mixture Modelling, GMM,* and *Spectral Clustering.* Given the nominal model, the outlier detection is based on if a point seems to belong to the nominal model or not.

In *section 2* there is a short introduction to Air Traffic Control/Surveillance in order give the reader an insight and motivation of problem formulation in this thesis. A concretization of the problem is presented in this section. *Section 3* contains the theoretical background needed for the used clustering methods in this thesis. *Section 4* defines outliers and introduce the different outlier measurements used. In *section 5* the method used to examine the different clustering and outlier detection algorithms is presented. *Section 6* presents the results from the outlier detection algorithms. The results are divided into two parts, firstly the tuning of the hyper-parameters. Secondly, the final results obtained. *Section 7* contains discussions of the outcome of the algorithms and conclusions that can be made. Future work and possible improvements are also discussed.

2 Background and Problem Formulation

In 2017 the air traffic within the European Civil Aviation Conference (ECAC) area[1] grew in average with 3.8%. The growth has been increasing for the last five years. This traffic increase has resulted in a decrease in the overall service quality in 2018. For instance, the average of flights arriving within 15 minutes of their scheduled time is 75.7%. This is the lowest level over the past 10 years.

The increased number of flight has led to a higher workload within the Air Traffic Control (ATC). In 2018 the delays caused by ATC Capacity grew with 76% compared to 2017, and the delays caused by ATC staffing grew with 186% [6].

The main task for a Flight Traffic controller is to ensure a safe and efficient air traffic. Every commercial flight fly an assigned route consisting of way points between departures and arriving airport. Under all phases of the flight, the aircraft are under traffic control/air traffic services (ATC/ATS), who provides communication, navigation and surveillance services. By providing location and identification of aircraft, surveillance services manage separation from other aircraft and obstacles. Surveillance can also serve as aircraft tracking. The flight tracking is mainly dependent on monitoring the air traffic by ground based Secondary Surveillance Radars (SSRs) and radar transponders located on the aircraft. The transponders sends a coded message to ATC consisting of the aircraft's identification, altitude and other information. Within the Swedish Flight Information Region (FIR) an aircraft should be equipped with a working transponder according to table (1) [15].

	Airspace	**IFR flight**	**VFR flight**
a)	Control Zone(CTR)	Mode A	Mode A
b)	Control Area (CTA)	Mode A and C	Mode A and C
c)	Traffic Information Region (TIZ)	Mode A	None
d)	Traffic Information Area (TIA)	Mode A and C	None
e)	Class G airspace*	Mode A and C	None
f)	Class G airspace**	None	None

Table 1: Transponder requirements within Swedish Airspace. Mode A: no altitude report, only identification. Mode C: Altitude report *Above the highest of 3000ft AGL or 5000ft MSL. **Except airspace specified in e) [15].

In table (1) the classifications of the airspace is introduced. In the controlled airspace zones CTR and CTA [2] there is an air navigation service which communicate with the air traffic. The CTR and CTA airspace is located around larger

[1] Consisting of 44 members, all 28 EU members, 31 of 32 European Aviation Safety Agency member states and all 41 EUROCONTROL member states.

[2] CTA: airspace extending upwards from specific limit above earth. CTR: airspace extending upwards from the surface of earth.

airports and is established when the air navigation service is established. Internationally the airspace is divided into class A-G. These classes decide which traffic is allowed and what rules apply. In Swedish FIR classes A and C are used for controlled airspace and class G for uncontrolled airspace. The area around airports from altitude 0 meters to 450 meters is called the control zone, CTR. Its purpose is to protect aircrafts during landing and take-off. If the airport is uncontrolled the corresponding airspace is called Traffic Information Zone, TIZ. Typically an aircraft during approach or departure phase to an airport is guided in a controlled airspace. This airspace is called Terminal Area, TMA, with corresponding uncontrolled Traffic Information area, TIA [17].

A new tracking technology that becomes more common in aircrafts is Automatic Dependent Surveillance-Broadcast (ADS-B). It is located on the aircraft and determines its position via satellite navigation. The information, such as the aircraft's identification, altitude, speed, velocity and projected path, is broadcasted every second to air traffic controllers and other ADS-B equipped aircrafts. Since the information is automated, in the sense that the pilot does not have to fill it in manually, and can be received by any other ADS-B receiver, it creates a improved situational awareness [31].

It is of importance to have an effective aircraft tracking. Therefore the aircraft tracking functionality must be active at take-off and remain operational during the flight. Every 15 minutes the aircraft's position should be reported. If there is any abnormal operational events, e.g. altitude deviations or changes to potential area of operation, the reporting rate can be increased. In those cases, it must be investigated if a alert phase is warranted, which can be a dialogue between ATC/ATS and the aircraft operator. If an alert phase is initiated, the ATS unit will contact the Rescue Coordination Center, RCC.

In summary, the aircraft tracking performance must:

- track aircraft within potential areas of operation and range.
- be available and operating while aircraft is airborne.
- consists of required information (latitude, longitude, altitude, time and aircraft identification).
- if transmitted by aircraft, position accuracy should be at least 1 NM.
- be able to increase reporting rate.
- include a protocol between the airline and ATS to aid coordination during alert phase [11].

In addition to a transponder on the aircraft, the pilot or a flight dispatcher from the local civil aviation authority must file a flight plan. It contains the planned route and must be filed in prior to departure. During the route, an air traffic

controller follow the tracks of the aircraft and make sure it follows the assigned route from the flight plan[31].

The amount of data and information processed in ATC is tremendous. Thus, the general ATC system consist of three main components:

- Flight Data Processing (FDP)
- Radar Data Processing (RDP)
- Display System or Human Machine Interface (HMI)

The system also have a lot of different sub-system, such as Environment Data Processing (ENV), Flight Plan Distributions (DIS) and Sectorisation (SEC). The diagram in fig. (1) shows the connection between main- and sub-systems [5].

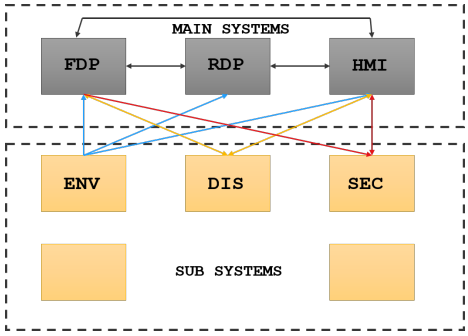


Figure 1: Example diagram of an ATC system

The ATC system is dynamic and generates an enormous amount of continuous information which ATC controllers must be able to analyse. The system has characteristics as rapidly escalating situations, severe time pressure, severe error consequences, complex and multi-component decisions and shifting goals. An air traffic controller is presumed to manage stressful situations, such as emergencies and abnormal situations from air misses. The controller must handle a lot of decision-making, which is an intensive, stressful and cognitively complex activity [19].

2.1 Problem Formulation

In air traffic control it is crucial to detect abnormal behaviour in an early stage. Due to high workload and a complex system, it is a challenging task to find those events. By using an assisting automated system that would highlight abnormal flight traffic, it could reduce the workload of the air traffic controllers. The system could also possible faster detect if something behaves out of ordinary. Therefore the aim of this thesis is to examine if it is possible to detect abnormal

events by an clustering based outlier detector.

Since there is an infinite number of abnormal events the approach in this paper is to find an nominal model, which would simulate "normal" air traffic, i.e., the case when everything work as it should. From the nominal model the objective is to classify if a new point belongs to the nominal model or not. If not, it should be classified as an anomaly. This results in two sub-problems:

I Is it possible to create an nominal model that simulates normal air traffic?

II Given the nominal model, is it possible to detect an anomaly?

3 Theoretical Background

Clustering is an unsupervised learning task with the objective to partition a data-set into clusters with data that has similar attributes. The partitioning is based on simultaneously maximizing the similarity within a cluster and minimizing the similarity to other clusters. A clustering method is said to perform well if the clusters is well defined with a high intra class similarity. Figure (2) illustrates an example of data that is clustered into two clusters.

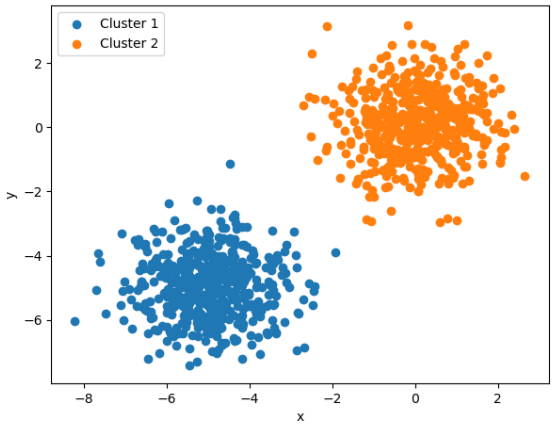


Figure 2: Example of clustering

Generally, clustering methods can be divided into three groups; *partitioning, hierarchical* and *density-based* clustering [29]. In this thesis, partitioning cluster methods as well as graph theory based clustering will be examined and compared.

This section introduce the different clustering algorithms used in this thesis. The relevant theory behind these algorithms will also be described. In the last section a method to validate the consistency of the clusters will be introduced.

3.1 Partitioning Clustering

Given a data set with observed data $\mathcal{D} = \{\mathbf{x_1}, ..., \mathbf{x_N}\}$, where N is the number of observations and $\mathbf{x_i} \in \mathcal{R}^D$. The goal of this method is to cluster the data set into K different clusters. Note that K is a design parameter. Introduce a set $\mathcal{Z} = \{\mu_1, ..., \mu_K\}$ where $\mu_1, .., \mu_K \in \mathcal{R}^D$ are vectors representing the centre of each cluster within K. The aim is assign each data-point to the a cluster such that the distance between the data-point and the cluster centre is minimized.

By introducing a indicator variable such that

$$r_{nk} = \begin{cases} 1, & \text{if data point } \mathbf{x_n} \text{ is assigned to cluster } k \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

the clustering problem can be written as

$$\begin{aligned} \min_{\mu_k, r_{nk}} \quad & J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x_n} - \mu_\mathbf{k}||^2 \\ \text{s.t.} \quad & r_{nk} \in \{0, 1\} \\ & \sum_{k=1}^{K} r_{nk} = 1 \ \forall n. \end{aligned} \tag{2}$$

The optimization problem in eq. (2) is a mixed integer problem which is typically NP-hard to solve, however it can be approximately solved by using an iterative method [2].

3.1.1 K-means

Problem (2) can be approximately solved through a two step iterative optimization method called *K-means algorithm*. The method alternates between optimizing problem (2) with respect to μ_k and $r_{n,k}$, while the other parameter is held fixed. The method is described below.

□ **Input:** Data-set $X \in \mathbb{R}^{D \times N}$.

1. Randomly initialize μ_k.
2. Optimize (2) with respect to r_{nk} and μ_k held fixed. Since the objective function in eq. (2) is linear with respect to r_{nk} and also independent with respect to n we can optimize for each n in the following way:

$$r_{nk} = \begin{cases} 1 \text{ if } k = \arg\min_j ||\mathbf{x_n} - \boldsymbol{\mu_j}||^2 \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

3. Optimize (2) with respect to μ_k and keep r_{nk} fixed. Then the objective function in eq. (2) becomes quadratic and convex. Hence, the minimum can be found by find when the derivative is zero.

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0. \tag{4}$$

Solve eq. (4) for μ_k:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \tag{5}$$

4. Repeat step $2-3$ until convergence is reached.

□ **Output:** $r_{n,k},\ \forall n \in 1,...,N,\ k \in 1,...,K.$ and $\mathcal{Z} = \{\mu_1, ..., \mu_K\}$.

Since the objective function is reduced in each step the method guarantees convergence. Note that the objective function may have multiple local minima, and the K-means algorithm does not guarantee to converge to the global minima.

The K-means algorithm is relatively slow since it has to compute the euclidean distance for every data-point in every iteration. However, a faster convergence can be obtained by choosing the initial values in *step 1* from a random subset of K data points within the data-set [2].

3.1.2 Gaussian Mixture Model, GMM

Another approach to solve the partition problem is to use a probabilistic approach. By the assumption that each data-point is generated by sampling of a continuous function, the partitioning is based on the probability of a data-point being sampled by that function.

Typically a data distribution is normally distributed, also called the *Gaussian distribution*. Therefore, the Gaussian distribution is a natural choice of distribution for modelling a data-set. For a vector $\mathbf{x} \in \mathbb{R}^D$ the multivariate Gaussian distribution is given by:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\} \tag{6}$$

where $\boldsymbol{\mu} \in \mathbb{R}^D$ is the mean vector and $\Sigma \in \mathbb{R}^{D\times D}$ is the covariance matrix. Thus, a single Gaussian distribution is limited when it comes to modelling a real data-set. Especially if the data-set is divided into clusters, then a single Gaussian can not capture the structure. In such case, a superposition of Gaussian distributions is a suitable choice. This is commonly known as *Gaussian Mixture Model, GMM*. Assuming a sufficient number of Gaussian distributions with adjusted means and covariances, almost any data can be modelled using GMM.

By using a superposition of K distributions, the probability of a vector $\boldsymbol{x} \in \mathbb{R}^D$ is given by:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \tag{7}$$

In eq. (7) $\boldsymbol{\mu}_k$ is the mean, Σ_k is the covariance matrix and π_k is the mixing coefficient such that $0 \leq \pi_k \leq 1$. Assume a data-set $X \in \mathbb{R}^{N\times D}$, and that the data-points within the set are independent. To model the data-set by GMM, one need to fit the parameters μ, Σ and π. This is achieved by maximizing the

log-likelihood of eq. (7):

$$\begin{aligned} \underset{\pi,\mu,\Sigma}{\textbf{maximize}} \quad & \ln(p(X|\pi,\mu,\Sigma)) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)\} \\ \textbf{s.t.} \quad & 0 \leq \pi_k \leq 1 \\ & \sum_{k=1}^{K} \pi_K = 1. \end{aligned} \tag{8}$$

Due to the summation over K in problem (8), the problem becomes hard to solve. However, it is possible to solve the problem with gradient, or iterative based methods.

An example of GMM fitted to a data-set is shown in figure (3). In the figure a superposition of two Gaussian distributions are used.

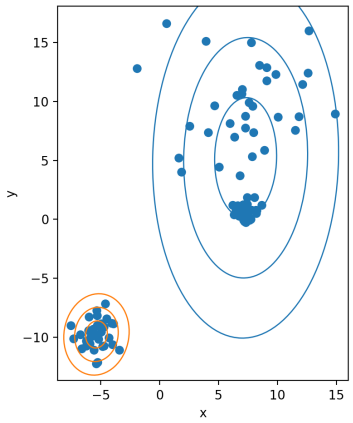


Figure 3: Example of two Gaussian distributions fitted to a data-set. Note that the solid orange and blue lines corresponds to one standard-deviation contour per line for each of the two Gaussian components.

3.1.3 Expectation Maximization of GMM

A powerful method for approximately solving problem (8) is called the *Expectation Maximization* (EM) algorithm. It is an iterative method that alternates between performing an expectation step, *E-step*, and a maximization step, *M-step*. In the E-step the current values of the parameters are used to calculate the posterior probabilities, or *responsibilities*. This probabilities are used in the M-step to re-estimate the means, covariances and the mixing coefficients. And finally, the log-likelihood is calculated. The algorithm is summarized below.

□ **Input:** Data-set $X \in \mathbb{R}^{D \times N}$.

1. Initialize μ_k, Σ_k and π_k
2. **E-step:** Calculate the responsibilities:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

3. **M-step:** Re-estimate the parameters:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$
$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1} N\gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$
$$\pi_k^{new} = \frac{N_k}{N}$$

   where $N_k = \sum_{n=1}^{N} \gamma(z_n k)$

4. Calculate the log-likelihood:

$$ln(p(X|\mu, \Sigma, \pi)) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)\}$$

5. Repeat step $2 - 4$ until convergence [2].

□ **Output:** π, μ, Σ.

Note that in *step 5*, one can check for convergence of either the parameters or the log-likelihood. Generally, there will be multiple local maxima of the log-likelihood, and the EM algorithm is not guaranteed to converge to the largest maxima.

Compared to the K-means algorithm, EM generally takes more iterations before reaching converge and also require more computations in each step. A faster convergence can be obtained by running K-means before and use the cluster means as a parameter initialization in the EM-algorithm.

3.2 Graph based clustering

In graph theory the data-set is represented by a graph which models the pairwise relations between each data-point. Given a data-set with observed data $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, where N is the number of observations and $\mathbf{x_i} \in \mathbb{R}^D$. Also assume that the data-points within the data-set is pairwise connected to each other.

Definition 3.1. Graph, $G(V, E)$. A graph, $G(V, E)$, is defined by two components:

- *vertex set*, $V = \{v_1, ..., v_N\}$ and
- *edge set*, $E = e_1, ..., e_C$, where C is the number of connected components.

An example of a graph representation is presented in figure (4).

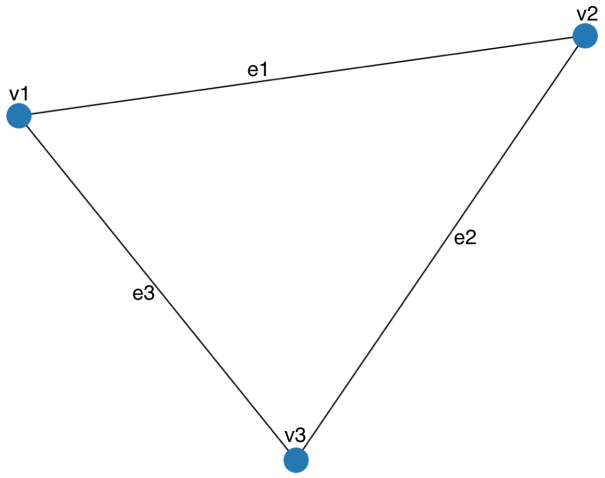


Figure 4: Example of a fully connected graph.

An *affinity matrix* describes the connection between the vertices. Assuming a weighted graph, with weighs $w_{ij} \geq 0$, then the weighted affinity matrix is given by [4]:

$$A_{ij} = \begin{cases} w_{ij}, \text{ if } i \neq j \text{ and } x_i, x_j \text{ is connected and } \mathrm{w}_{ij} \geq 0, \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

The *degree* of a vertex is defined as:

$$d_i = \sum_{j=1}^{n} w_{ij}. \tag{10}$$

A diagonal matrix containing the degrees is called the *degree matrix*, D [18].

When clustering data represented by a graph, the objective is still to find a partition such that the data within each group has similar attributes. In this thesis, the partitioning will be based on the objective to minimize the total weights of the edges connecting the different partitions. An example of such clustering is shown in figure (5).

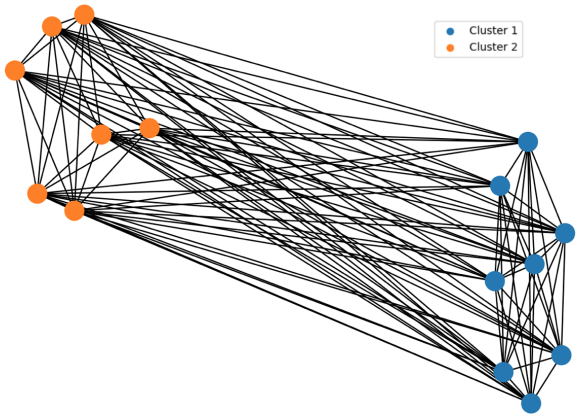


Figure 5: Example of graph based clustering.

This section will introduce relevant graph theory and the graph partition problem which gives rise to the *Spectral Clustering* algorithm.

3.2.1 Similarity Graph

A similarity graph, $G(V, E)$, represents a data-set as a graph with data-points as vertices and the edges are decided by the similarity between the data-points. Typically, two vertices are connected by an edge if $s_{ij} \geq 0$ or larger than a pre-defined threshold. The edge is then weighted by the similarity, s_{ij}, between the two data-points. There are several ways to measure the similarity, but with the common goal to model the local neighbourhood relationship. In table (2) some different ways to construct similarity graphs are presented.

ϵ-neighbourhood	Connect all points if pairwise distance is smaller than ϵ
k-nearest neighbour	Connect vertex v_i with v_j if v_j is within the k-nearest neighbours of v_i.
Fully connected	Connect all points with a positive similarity, and weight all edges with s_{ij}.

Table 2: Construction of similarity graphs

In the first approach, ϵ-neighbourhood, in table (2) every distance between connected data-points is approximately the same. This implies that weighting of edges will not contribute to more information to the graph and therefore this will construct an un-weighted graph. In contrast, k-nearest neighbour leads to a directed graph since the relationship between neighbourhoods is not symmetric. When using a fully connected construction of the similarity graph there

must exist a similarity function that itself encodes mainly local neighbourhoods [18]. The choice of an appropriate similarity function is important since it will affect the performance of many algorithms. Below, some similarity functions are presented:

I **Cosine similarity:** $s(x_i, x_j) = \frac{x_i^T \cdot x_j}{|x_i| \cdot |x_j|}$. Measures the cosine of the angle between two vectors.

II **Jaccard Distance:** $s(x_i, x_j) = \frac{x_i^T \cdot x_j}{||x_i||^2 + ||x_j||^2 - x_i^T \cdot x_j}$. Measures the similarity as intersection divided by the union of vectors [24].

III **Gaussian similarity function:** $s(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$. Measures the Gaussian distance, where the parameter σ is a scaling parameter. The larger value of σ, the faster the similarity $s(x_i, x_j)$ will decrease with distance between x_i and x_j [21].

3.2.2 Graph Laplacian

Assume a weighted and undirected graph G, with a weight matrix W. The graph laplacian is another way to represent the graph, G, and is defined as:

$$\begin{aligned} L &= D - W, \\ L_{sym} &= D^{-1/2} L D^{-1/2}, \\ L_{rw} &= D^{-1} L. \end{aligned} \tag{11}$$

In (11), L is the unnormalized Laplacian, L_{sym} is the normalized and symmetric Laplacian and L_{rw} is the random-walk normalized Laplacian. Below, some properties of the graph Laplacian will be presented.
Properties of the unnormalized Laplacian:

I $\forall\ x \in \mathbb{R}^N$:

$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} (x_i - x_j)^2$$

.

II L is positive semi-definite and symmetric.

III The smallest eigenvalue is 0 with corresponding eigenvector $\mathbf{1}$.

IV L has N real-valued eigenvalues such that $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$.

V Assume an undirected and weighted graph G. The number of connected components in the graph is equal to the multiplicity k of the Laplacian eigenvalue that is equal to zero.

Properties of the normalized Laplacian:

I $\forall\ x \in \mathbb{R}^N$:

$$x^T L_{sym} x = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2$$

II λ is an eigenvalue of L_{rw} with eigenvector v if and only if λ is an eigenvalue of L_{sym} and with the eigenvector, w, such that $w = D^{1/2}v$.

III λ and v must solve the generalized eigenproblem: $Lv = \lambda Dv$.

IV L_{rw} has an eigenvalue that is equal to zero with corresponding eigenvector $\mathbf{1}$.

V L_{sym} has an eigenvalue that is equal to zero with corresponding eigenvector $D^{1/2}\mathbf{1}$.

VI L_{sym} and L_{rw} are positive semi-definite and has N real-valued eigenvalues such that $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$

VII Assume a undirected and weighted graph G. The number of connected components in the graph is equal to the multiplicity k of the eigenvalue to L_{sym} and L_{rw} that is equal to zero [18].

3.2.3 Graph Cut

As mentioned above the goal with clustering is to separate data into groups with similar features. Assuming a similarity graph, the goal is to partition the vertices into non-empty sets such that the total weights between the sets is minimized.

Definition 3.2. Graph Cut
For two disjoint subsets $A, B \subset V$ the *value* of a cut is:

$$cut(A, B) := \sum_{i \in A, j \in B} w_{ij}.$$

The problem to find such partition is often refereed to as the *minimum cut problem* [13]:

$$\underset{A_1, A_2, .., A_k}{\textbf{minimize}}\ cut(A_1, ...A_k) := \sum_{i=1}^{k} cut(A_i, \bar{A}_i). \tag{12}$$

Note that in problem (12) $A_1, A_2, .., A_k$ is subsets such that $A_i \subset V$ and $\bar{A}_i$ is the complement of a subset $A_i \subset V$. For $k \leq 2$ the min-cut problem (12) can be solved efficiently. Thus, when solving a clustering problems there is typically several groups with a reasonable numbers of points. To achieve this one can use one of the balanced objective functions,

$$\text{RatioCut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|} \tag{13}$$

$$\text{Ncut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}. \tag{14}$$

The difference between RatioCut and Ncut is the measurements of the size of a subset A. In RatioCut the subset size is measured by its number of verices, $|A|$, in contrast to Ncut that measures the size by the weight of its edges, $vol(A)$.

When using the balanced objective functions the min-cut problem becomes NP-hard to solve. Thus it is possible to use spectral clustering to solve a relaxation of the problem.

3.2.4 Approximation of RatioCut

Assume a partition $A_1, ..., A_k \subset X$. Define k indicator vector $h_i = (h_{1,i}, ..., h_{n,i})^T$ such that

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_i|}} & \text{if } x_i \in A_j, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Let $H \in \mathbb{R}^{(n \times k)}$ be the matrix containing all indicator vectors as columns. Each column in H is orthonormal and hence, $H^T H = I$. By using property I of unnormalized Laplacian the following is obtained:

$$h_i^T L h_i = 2\frac{\text{cut}(|A_i|, |\bar{A}_i|)}{|A_i|} \tag{16}$$

$$h_i^T L h_i = (H^T L H)_{ii}. \tag{17}$$

Using eq. (16)-(17) *RatioCut* can be rewritten as:

$$RatioCut(A_1, .., A_k) = \frac{1}{2}\sum_{i=1}^{k} h_i^T L h_i = \frac{1}{2}\sum_{i=1}^{k}(H^T L H)_{ii} = \frac{1}{2}Tr(H^T L H). \tag{18}$$

Using this as objective function in the minimization problem (12) the following problem is obtained:

$$\begin{aligned} \underset{A_1, A_2, .., A_k}{\textbf{minimize}} \quad & Tr(H^T L H) \\ \textbf{s.t.} \quad & H^T H = I \\ & h_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_j|}} & \text{if } x_i \in A_j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{19}$$

By allowing H to have arbitrary real values, the following relaxed problem arises:

$$\begin{aligned} \underset{H \in \mathbb{R}^{n \times k}}{\textbf{minimize}} \quad & \text{Tr}(H^T L H) \\ \textbf{s.t.} \quad & H^T H = I. \end{aligned} \tag{20}$$

The problem above is a standard trace minimization problem where the optimal solution is given by H containing the first K eigenvectors of L as columns [18]. By using standard *K-means* (see section 3.1.1) one can convert the real valued solution matrix to a discrete partition. The solution of the relaxed *RatioCut* problem is called *unnormalized spectral clustering*.

3.2.5 Unnormalized Spectral Clustering

☐ **Input:** Similarity matrix $S \in \mathbb{R}^{N \times N}$, weighted affinity matrix W, number of clusters K.

1. Compute the unnormalized Laplacian.
2. Compute the first K eigenvectors $v_1, ..., v_K$ of L.
3. $V \in \mathbb{R}^{n \times K}$ contains the eigenvectors as columns.
4. For $i = 1, .., N$, let $y_i = V[i, :] \in \mathbb{R}^K$.
5. Let $Y = [y_1, ..., y_K] \in \mathbb{R}^{N \times K}$.
6. Treat each row of Y as a point and cluster into K clusters using *K-means*.
7. Assign original point (x_i) to cluster k if and only if row i of Y was assigned to cluster k.

☐ **Output:** A list containing which cluster $C_1, ..., C_K$ each data-point x_i belongs to.

3.2.6 Approximation of Ncut

The approximation of Ncut has a similar work of frame as for RatioCut. In this case the indicator vector $h_i = (h_{1,i}, ..., h_{n,i})^T$ is defined as:

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & \text{if } x_i \in A_j \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Let a matrix H contain K indicator vectors, and observe that $H^T H = I$, $h_i^T D h_i = 1$ and $h_i^T L h_i = 2cut(A_i, \bar{A}_i)/vol(A_i)$. Using this, the minimization of Ncut can be written as

$$\begin{aligned} \underset{A_1, A_2, .., A_k}{\textbf{minimize}} \quad & Tr(H^T L H) \\ \textbf{s.t.} \quad & H^T D H = I \\ & h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & \text{if } x_i \in A_j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{22}$$

By substituting $U = D^{1/2} H$ and allow it to have any value, the relaxed Ncut problem becomes:

$$\begin{aligned} \underset{U \in \mathbb{R}^{n \times k}}{\textbf{minimize}} \quad & Tr(U^T D^{-1/2} L D^{-1/2} U) \\ \textbf{s.t.} \quad & U^T U = I \end{aligned} \tag{23}$$

Similarly as for the relaxation of the *RatioCut*, a standard trace minimization problem arises. Hence, the optimal solution is given by U that contain the first K eigenvectors of L_{sym}. This results is known as the *normalized spectral clustering algorithm* [18].

3.2.7 Normalized Spectral Clustering

- ☐ **Input:** Similarity matrix $S \in \mathbb{R}^{N \times N}$, weighted affinity matrix W, degree matrix D and number of clusters K.

  1. Compute the symmetric normalized Laplacian.
  2. Compute the first K eigenvectors $v_1, ..., v_K$ of L.
  3. $V \in \mathbb{R}^{N \times K}$ contains the eigenvectors as columns.
  4. For $i = 1, .., N$, let $y_i = V[i, :] \in \mathbb{R}^K$.
  5. Let $Y = [y_1, ..., y_k] \in \mathbb{R}^{N \times K}$.
  6. Renormalize each row in Y to have unit length.
  7. Treat each row of Y as a point and cluster into K clusters using *K-means*.
  8. Assign original point (x_i) to cluster k if and only if row i of Y was assigned to cluster k [21].

- ☐ **Output:** A list containing which cluster $C_1, ..., C_K$ each data-point x_i belongs to.

3.3 Silhouette value

One challenging and important task in clustering is to find the right number of clusters. One way to measure the quality of clusters is to represent each cluster as a silhouette, which is based on comparison of its compactness and separation to other clusters. Let A be a cluster which data-point x_i has been assigned to, $x_i \in A$. The *internal dissimilarity* is defined as:

$$a(x_i) = \frac{1}{|A| - 1} \sum_{x_j \in A, i \neq j} d(x_i, x_j). \tag{24}$$

Note that $|A|$ is the number of data-points assigned to cluster A_i. Commonly the Euclidean distance is used, but it can be calculated with any distance metric.

Let C_k be the other clusters, such that $x_i \notin C_k$, $\forall k$. The *external dissimilarity* is defined as:

$$b(x_i) = \min_{x_i \notin C_k} \frac{1}{|C_k|} \sum_{x_j \in C_k} d(x_i, x_j). \tag{25}$$

Then, the Silhouette value for data-point x_i is defined as:

$$s(x_i) = \begin{cases} 1 - a(x_i)/b(x_i), & \text{if } a(x_i) < b(x_i), \\ 0, & \text{if } a(x_i) = b(x_i) \\ b(x_i)/a(x_i) - 1, & \text{if } a(x_i) > b(x_i). \end{cases} \tag{26}$$

As seen in eq. (26) $-1 \leq s(i) \leq 1$. When $s(i)$ is close to one, the internal dissimilarity is small and the external is large, and the cluster is well defined and compact. If the value is close to zero, then the external- and internal dissimilarity is similar and it is not well defined if point x_i should be assigned to cluster A or C_k. In the case when a point is closer to another cluster than it was assigned to $s(i)$ will be close to -1 and indicates that the point is misclassified.

The average of the silhouette value can be used as cluster evaluation and to select an appropriate number of clusters [27]. The aim is then to find the number of clusters, K, that solves:

$$\begin{aligned} \underset{K}{\textbf{maximize}} \quad & \frac{\sum_{i=1}^{N} s(x_i)}{N} \\ \textbf{s.t.} \quad & C_{1,\ldots,K} = \text{ClusteringMethod}(x_{1,\ldots,N}, K) \\ & K \geq 2. \end{aligned} \tag{27}$$

4 Outlier Detection

In this section relevant methods and theory about outlier detection algorithms will be introduced. First off all, an outlier must be defined. According to Hawkins an outlier is defined *"as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [9]."* Thus, an exact definition of an outlier depends on the data and the applied detection method. Therefore, this section contains three the different type of outlier detection methods which are relevant to this thesis.

4.1 Distance Based Method

In this thesis a distance based outlier detection algorithm will be used when the nominal model is obtained by *K-means*. Since K-means returns the center points, $\mu_k = 1, ..., K$, the detection measurement, d, will be the Euclidean distance between a new data-point, x_n, and its closest cluster center point,

$$d = \min_{\mu_k \in \{\mu_1, ..., \mu_k\}} ||\mu_k - x_n||^2. \tag{28}$$

A decision function is introduced, such that an outlier is assigned the value -1 and an inlier the value $+1$ according to

$$f(x_n) = \begin{cases} +1 & \text{if } d \leq th \\ -1 & \text{if } d > th, \end{cases} \tag{29}$$

where th is a pre-defined or trained threshold.

4.2 Probabilistic Based Method

Given a new data-point x_n and a probabilistic model with parameters θ. The probability $P(x_n|\mathcal{D}(\theta))$ is the probability of x_n being generated by a density function $\mathcal{D}(\theta)$. Then a point can be classified as an inlier, $+1$, or outlier -1 according to the decision function

$$f(x_n) = \begin{cases} +1 & \text{if } P(x_n|\mathcal{D}(\theta)) \geq th \\ -1 & \text{if } P(x_n|\mathcal{D}(\theta)) < th, \end{cases} \tag{30}$$

where th, as before, is a pre-defined or trained threshold.

4.3 One-Class Support Vector Machine

The one-class support vector machine, OCSVM, algorithm uses a kernel,

$$K(x, x') = \phi(x) \cdot \phi(x'), \tag{31}$$

to map input data, $\mathcal{X}$, into a high dimensional space, $\mathcal{F}$,

$$\phi : \mathcal{X} \to \mathcal{F},$$

and iteratively finds the maximal hyperplane which best separates data from the origin. To find this hyperplane the following quadratic program must be solved:

$$\begin{aligned} \min_{w\in\mathcal{F},\xi\in\mathbb{R}^N,\rho\in\mathbb{R},v\in(0,1]} \quad & \frac{1}{2}||w||^2 + \frac{1}{\nu N}\sum_{i=0}^{N}\xi_i - \rho \\ \text{subject to} \quad & (w\cdot\phi(x_i)) \geq \rho - \xi_i,\ \xi_i \geq 0, \end{aligned} \tag{32}$$

where ξ_i is a slack variable. Due to the penalization of the slack variable in the objective function, one can expect that if w and ρ solves problem (32) a decision function

$$f(x) = \text{sgn}((w\cdot\phi(x)) - \rho)$$

will be positive for the majority of the samples in the data set, and the regularization term $||w||$ will be small. The trade of between the decision function and the regularization term is restrained by v.

By introducing multipliers, $\alpha_i, \beta_i \geq 0$, the Lagrangian of problem (32) becomes:

$$L(w,\xi,\rho,\alpha,\beta) = \frac{1}{2}||w||^2 + \frac{1}{\nu N}\sum_i \xi - \rho - \sum \alpha_i((w\cdot\phi(x_i)) - \rho + \xi_i) - \sum_i \beta_i\xi_i. \tag{33}$$

Taking the derivative of the Lagrangian with respect to the primal variables,$w,\ \xi,\ \rho$, and finding when it is equal to zeros, yields

$$\begin{aligned} w &= \sum_i \alpha_i\phi(x_i), \\ \alpha_i &= \frac{1}{v^N} - \beta_i \leq \frac{1}{\nu N}, \\ \sum_i \alpha_i &= 1. \end{aligned} \tag{34}$$

By inserting (34) and (31) into (33), the dual problem of (32) is obtained:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{ij}\alpha_i\alpha_j K(x_i,x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu N} \\ & \sum_i \alpha_i = 1. \end{aligned} \tag{35}$$

By solving the dual program (35), and inserting α and equation (34) into the decision function, yields:

$$f(x) = sgn(\sum_i \alpha_i K(x_i,x) - \rho). \tag{36}$$

If α_i and β_i are non-zero at optimum, ρ can be recovered as

$$\rho = \sum_j \alpha_j K(x_j,x_i). \tag{37}$$

The decision function $f(x)$ is then used to label a data-point [28], x_n, if its normal or an outlier, according to if

$$\begin{aligned} f(x_n) &= -1, \quad \text{then } x_n \text{ is classified as an outlier,} \\ f(x_n) &= 1 \qquad \text{then } x_n \text{ is classified as normal.} \end{aligned} \tag{38}$$

5 Method

The methodology in this paper is divided into four parts, data pre-processing, training of a nominal model, fitting a decision boundary and outlier detection. The correlation between the different part is shown in figure (6).

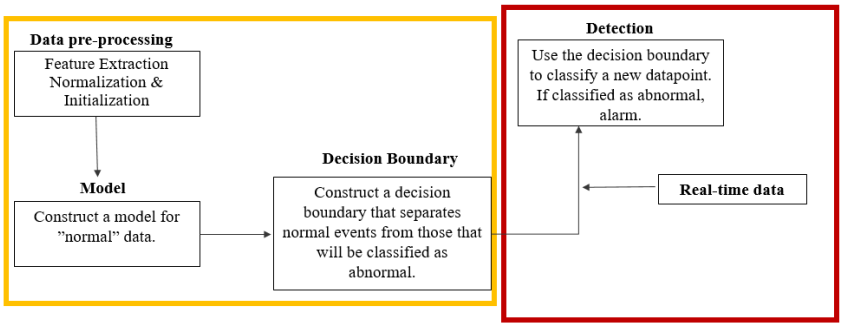


Figure 6: The workflow that is used in this thesis. Inside the yellow square off-line processes are shown. Inside the red square the online processes are shown.

5.1 Data-set

The data used in this paper contains simulated data from an air defence simulator system. It generates data with information from ground based radars, transponders and flight plans. In order to validate outlier detection methods, labelled data is needed. Therefore, each data-point is labelled "normal" or "abnormal".

The data-set is divided into three parts, training, validation and testing. The training data-set contains 242300 data-points. Each data-point represent properties of the plane at a given time and is represented by a vector, $x_n \in \mathbb{R}^D$, containing features such as altitude, speed, position and deviation from flight plan. This data-set is used to train the nominal model and hence is assumed to not consist of any outliers. The validation data-set and test data-set is of equal size contain approximately 10000 data-points labeled as normal and approximately 2500 labeled as outliers.

5.1.1 Data Pre-Processing

The raw-data contains a lot of information but only parts of the data is relevant for the aim in this thesis. Therefore, there must be a feature extraction where data of interest is extracted from the raw-data. A number of D features got extracted from the tracking data, such as altitude, position and velocity. Similar kind of data is filed in the flight plan and thus it is of interest to use deviations from the flight plan as a feature.

The flight plan contains predicted position and altitude at a given time and be able to measure the deviations between the tracking and flight plan, piece-wise linear interpolation is used between the points in the flight plan to obtain a function describing the predicted route. Then the euclidean distance between the interpolated function and the actual track is used as a feature, as shown in figure (7).

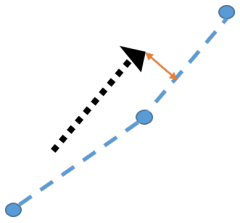


Figure 7: Mapping between flight tracking and flight plan. The black dashed line represents the tracking points and the triangle repserent the current state of a flight. The route filed in the flight plan is represented by the blue circles. The blue dashes is the interpolated path.

5.1.2 Normalization

Clustering methods are highly dependent of distances between data-points. This means that values in a higher range will be of more importance in the algorithm. Therefore the data must be normalized since the metrics within the data-set can vary from values of 10^{-1} to 10^{4}. Typically the normalization is done by removing the mean and scale with the variance. Thus, mean and covariance can be influenced by outliers, which might exist in the data-set. Therefore the scaling is done according to

$$x_{i,norm} = (x_i - \tilde{X})IQR^{-1},$$

where x_i is a data-point, $\tilde{X}$ is the median of the complete data-set and $IQR \in \mathbb{R}^{D\times D}$ is a diagonal matrix containing the interquartile range[3].

5.1.3 Data-set Representation

The data-set is represented by a matrix $X \in \mathbb{R}^{N\times D}$, where N is the number of data-points and D is the number of features. The data-set is divided into three parts, X_{train}, $X_{validate}$ and X_{test}.

5.2 Clustering Based Outlier Detection

This section describes four different clustering based outlier detection algorithms examined in this thesis. The first algorithm is based on K-means and will di-

[3]IQR is a measure of statistical dispersion and is equal to the difference between the $75th$ and $25th$ percentile.

vide the data into spherical clusters. The second algorithm is based on fitting a Mixture of Gaussian models, GMM by using the expectation maximization algorithm. The algorithms assign data-points into clusters based on the posterior probability of the data-points. The EM algorithm is similar to K-means, but instead of assuming a spherical cluster, the algorithm can form clusters of elliptical shapes. The last two algorithms are based on graph theory clustering, more specific spectral clustering. In contrast to K-means and EM-GMM this method form clusters of any geometrical form.

5.2.1 Algorithm I. K-means

The nominal model in algorithm I is based on K-means, which is described in section 3.1.1. The method returns K centre points which has the minimized distance to the nearest data-points. The nominal model is trained with X_{test} and will contain the centre points in each cluster, μ_k, the mean distance to centre within the cluster, $\hat{d_k}$ and the standard deviation of the distances, $\sigma_{d,k}$. The outlier detection will be distance based as described in section 4.1. The threshold, th, will determined by $\hat{d_k}$ and $\sigma_{d,k}$. See section 5.3.

5.2.2 Algorithm II. GMM

The second model is based on fitting a superposition of K Gaussian distributions to the training data-set. The nominal model is fitted such that the likelihood of the probability of each data-point belonging to a cluster is maximized. The model is learned by using expectation maximization, which is described in section 3.1.3. The trained parameters from the EM-algorithm will be denoted $\theta = (\pi, \Sigma, \mu)$. The outlier detection used is based on the posterior probability of a new and unseen data-point, and is described in section 4.2. Note that in this case the decision will be based on a superposition of K normal distributions. Therefore the decision will be based on the probability density function. The detection threshold, th, for each cluster will be based on the mean, $\hat{p_k}$, and standard deviation, $\sigma_{p,k}$ of $P(X_{train,k}|\theta)$. Here, $X_{train,k}$ are those point in the training data-set assigned to cluster k.

5.2.3 Algorithm III. Spectral Clustering 1

The third algorithm is based on spectral clustering which is described in section 3.2.7. In general, the nominal model is trained by representing the data as a graph and finding the graph partition that minimizes the total weights between the sub-graphs. The similarity graph is created from an appropriate choice of a semi-positive definite kernel (also called similarity measure)

$$\begin{aligned} &K(x,y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x) \cdot \phi(y) \\ &x, y \in \mathbb{R}^d,\ \phi(x) \in \mathbb{R}^r, \end{aligned} \tag{39}$$

where $\phi(x)$ is a function that maps input data x into a high dimensional space $\mathcal{F} \in \mathbb{R}^r$. The choice of such kernel function is a challenging task, due to the high dimension of the data-set. Therefore a flexible kernel that can handle possible complexity is an appropriate choice. In this paper the Gaussian kernel, *RBF* kernel,

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

is examined. Its a non-parametric kernel which makes it able to capture complex relations within the data. The kernel also subsumes polynomial and linear kernels since they are special cases of the Gaussian kernel. Advantages of using a Gaussian kernel versus a metric distance is that its values is within the range zero and one and the function value decreases with distance. A metric distance measure can easily reach large values which will results in an instability in the model. The Gaussian kernel has one hyper-paramter, σ, which specifies the bandwidth of the Gaussian kernel, this is shown in figure (8). The training

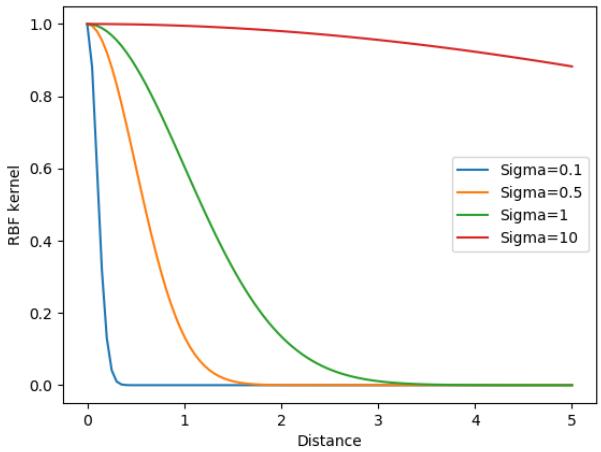


Figure 8: Gaussian kernel similarity bandwidth for different choices of σ.

of the nominal model follows the procedures described in section 3.2.7, *Normalized Spectral Clustering*. This method requires computations of eigenvalues and eigenvectors which requires a lot of computational power. Therefore the method will be limited by the maximum computational power available. Hence to use the method, another step in the data pre-processing has to be added. By using clusters created by the K-means algorithm, a number of data-points are randomly sampled from each cluster, based on the cluster size. The used training data-set for spectral clustering contains 10000 data-points which is a smaller set compared to the one used in K-means and GMM.

The outlier detection is based on the resulting clusters in the real space and OCSVM, see section 4.3. Thus, to keep the cluster structures obtained by spectral clustering, each cluster will be treated as a sub-data-set. This means that there will be K decision functions, $f(x)_k$. A data-point, x_n will be classified as an outlier if $f(x_n)_k = -1,\ \forall k \in \{1, ..., K\}$.

To solve the dual problem arising in OCSVM the LIBSVM library is used. It solves a scaled version of problem (35)

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1, i = 1, ..., N \\ & e^T\alpha = \nu N, \end{aligned} \tag{40}$$

by a Sequential Minimal Optimization decomposition method [3].

5.2.4 Algorithm IV. Spectral Clustering 2

In Algorithm IV the lower dimension embedding which arises in spectral clustering will be taken advantage of. The aim is to transform the embedding to a mapping by learning the leading eigenfunction.

In [1] they showed that when a data-set is empirically distributed

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{41}$$

the mapping can be learned by solving

$$\frac{1}{N}\sum_{j} \tilde{K}(x_i, x_j) f(x_j) = \lambda f(x_i), \tag{42}$$

where $\tilde{K}(x_i, x_j)$ is the normalized kernel used in spectral clustering, Note that x_i is a point from the training data-set and x_j is a new and unseen point. By substituting $u_j = f(x_j)$ and $\tilde{M}_{ij} = \tilde{K}(x_i, x_j)$, eq. (42) becomes:

$$\tilde{M}u = n\lambda u. \tag{43}$$

This is the same equation as solved in the normalized spectral clustering, scaled by a factor n. Therefore, by getting the K principal eigenvectors of normalized Laplacian, one can map a new point to the feature space by

$$f_k(x) = \sum_{i} \alpha_{ki}\tilde{K}(x_i, x), \tag{44}$$

where α_{ki} is the k-th principal eigenvector of L [1].

The resulting mapping in eq. (44) will be used in algorithm IV to perform the outlier detection in the feature space. The outlier detection will be based on OCSVM, see section 4.3.

5.3 Hyperparameters

Hyperparameters are those parameters that are not optimized and adjusted during training. Therefore, the choice of appropriate parameters are both challenging and central for the training outcome. A badly chosen hyperparameter might lead to a model that will poorly generalize to a new sample, also called overfitting, or vice versa, to be too general, also known as underfitted. For each algorithm (I-IV), their respective hyperparameters will be examined as described below.

Number of clusters, K***.*** A challenging part with clustering methods is to find the appropriate number of clusters. In this paper the Silhouette value is used as a quality measure of the clusters. Thus, when using large and complex data sets, the Silhouette value may not provide any useful information. Therefore the cluster content will be examined with measurements of mean and standard deviation of the different features. One other useful way to test if the number of clusters is appropriate is to test if the method classifies the data correctly.

For spectral clustering, eigengap heuristic will be used to find an appropriate number of clusters. The goal is to choose the number of clusters, K, such that all eigenvalues, $\lambda_1, ..., \lambda_K$ are small while λ_{K+1} is significantly larger. In the ideal case with K disconnected clusters, there will be K eigenvalues equal to zero. Therefore, the eigengap will be used as a indicator of an appropriate number of clusters [18].

Gaussian Width, σ***.*** Spectral clustering has a very important parameter, σ, in the Gaussian kernel. The impact of σ will be examined by looking at the Silhouette value and the general performance of the clustering algorithm. Its impact will also be examined by studying the affinity matrix.

The Gaussian Width also occurs in OCSVM, since a Gaussian kernel is chosen there as well. This value will be denoted σ^* and will be tuned based on performance measurements.

Detection Threshold, th***.*** This is the parameter that in the different algorithms decide if a sample will be classified as an inlier or an outlier. Different set-ups of the detection thresholds will be examined. The set-ups are mainly dependent on the nominal model, such as cluster means.

OCSVM, ν. Restrains the trade off between the decision function and the regularization term in the primal problem of OCSVM. This parameter will be

adjusted depending on performance measurements, desbribed below.

5.4 Performance Measurement

Each of Algorithm I-IV will be analysed with respect to performance. In this context the performance is based on the algorithms ability of correctly classify an inlier or outlier.

The performance measurement will be presented with a confusion matrix, which is shown in figure (9).

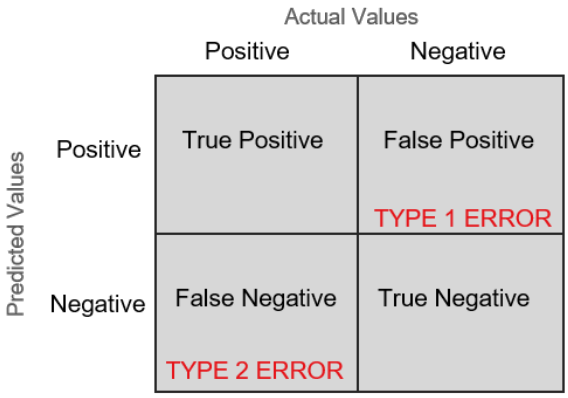


Figure 9: The layout of a confusion matrix.

The interpretations of the different values are:

True Positive, TP: The algorithm predicted a sample to be an outlier, when it is true.

True Negative, TN: The algorithm predicted a sample to be an inlier, when it is true.

False Positive, FP: Also called *Type 1 error*. The algorithm predicted a sample to be an outlier, when it is false.

False Negative, FN: Also called *Type 2 error*. The algorithm predicted a sample to be an inlier, when it is false.

These performance measurements will be presented both in number of samples in each category, and in percentage.

Other useful measurements based on the confusion matrix that will be used are:

Recall: the number of correctly predicted data-points within the positive classes. This value should be maximized.

$$Recall = \frac{TP}{TP + FN}$$

Precision: the quantity of correctly predicted positive classes, which is actually positive.

$$Precision = \frac{TP}{TP + FP}$$

Accuracy: the quantity of correctly predicted out of all classes. The accuracy should be maximized

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Comparing two models where the recall is high and the precision is low (or vice versa) is very challenging. In this case, the ***F-measure*** which measures *Recall* and *Precision* at the same time, in the sense of Harmonic Mean [20], can be used.

$$\text{F-measure} = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

5.5 Implementation

All implementation is done in PYTHON3 together with libraries from NUMPY [22] [30], SCIKIT-LEARN[25], SCIPY [12] and MATPLOTLIB [10].

6 Results

This section is divided into two parts. The first part shows result for tuning the different hyper-parameters based on their ability to cluster the data and the performance on the validation data-set. Tuning of hyperparameters are divided into three sub-parts, one for each method used. A proposed parameter set-up will be presented in the end of this part.

The second part presents the performance of each method on the testing data-set, i.e., the ability of each algorithm to classify inliers and outliers. Parameters obtained in the first part will be used there.

6.1 Tuning of Hyperparameters

In this section the results obtained when tuning hyperparameters are presented. Results will be presented for the three algorithms examined in this thesis. The section ends with a summary of hyperparameters that will be used in the performance measurements of the test data-set.

6.1.1 K-means

The K-means algorithm has one hyperparameter, which is the number of clusters, K. In figure (10) the average Silhouette values based on different cluster sizes are presented. The average Silhouette values are computed by solving equation (26) when using different number of clusters, K. If the clusters are well defined and with low internal dissimilarity the value should be $\hat{s} \approx 1$. As seen in the figure the values are relatively small and is not that effected by the number of clusters. This might indicate that the structure of the data-set is complex with overlapping clusters. If the clusters are overlapping the internal dissimilarity might be low, but the external dissimilarity will probably be relatively high. The number of clusters that maximizes the Silhouette values is $K = 25$, see figure (10).

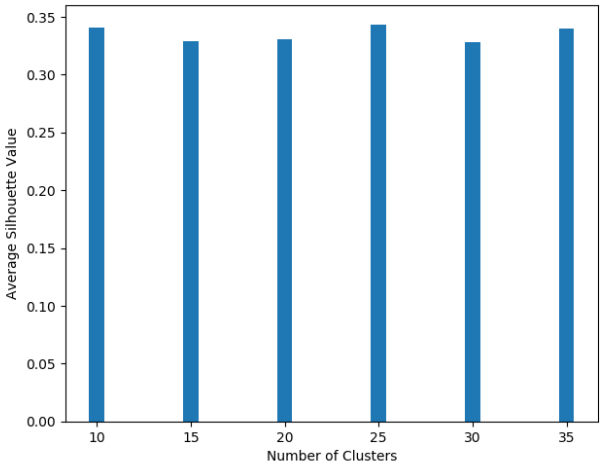
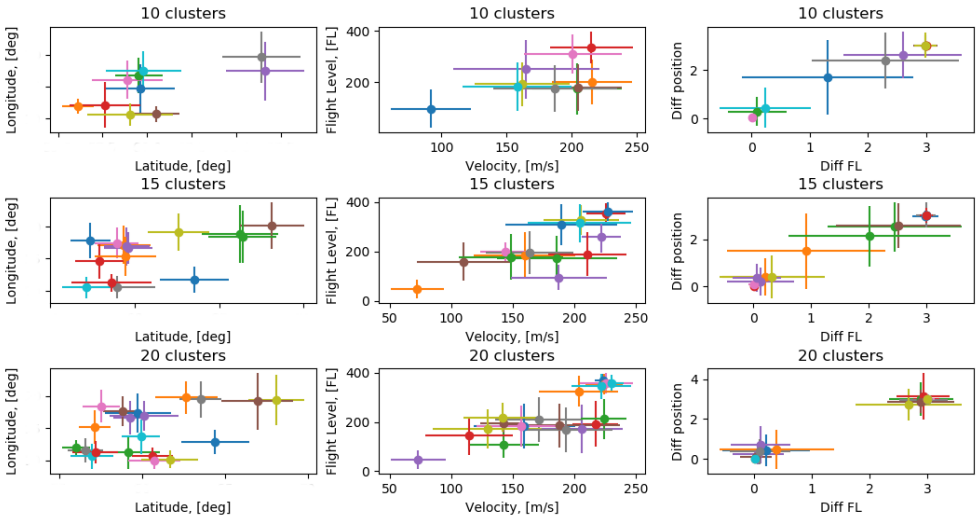


Figure 10: Average Silhouette values when using different number of clusters.

In figure (11) the means and standard deviation of different features within each cluster is shown. Due to the high dimension the data-set the partitioning is examined by looking at a few features separately. If the features within each cluster is well defined with low spread, it will indicate a well defined clusters. As seen in the figure, the more clusters the less variance of each feature.



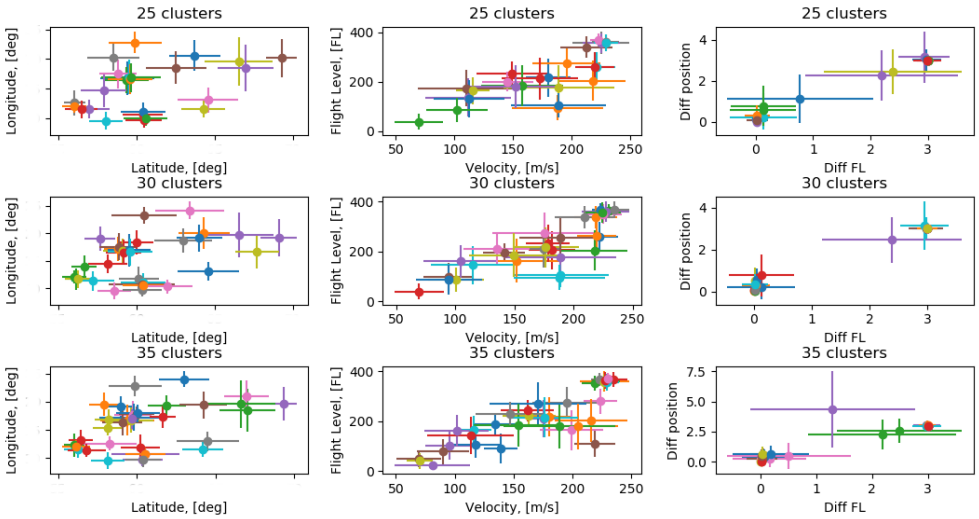


Figure 11: Feature means for different number of clusters. The dot represent the mean and the lines the variance. The metrics used are: latitude, longitude [deg], Flight Level [Fl], Velocity [m/s] and Diff FL, Diff position in relative difference from flight path. Note. A relative difference of 3 indicates that there was no flight plan.

When using more clusters, the data-set gets more evenly distributed within the clusters, see Appendix A.

The outlier measurement used together with the K-means algorithm is the Euclidean distance between a new data-point and its closets cluster mean. Figure (12) shows the outlier measurement for each data-point in $X_{validate}$. As seen in the figure, the data-points labelled as outliers are generally further away from the closest cluster mean than the data-points labelled as inliers. Thus, only based on this figure, an appropriate detection threshold is not obvious.

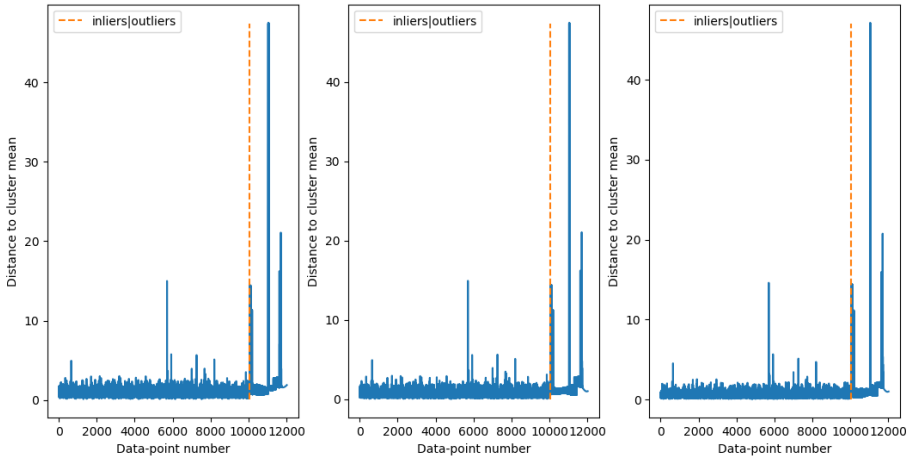


Figure 12: Outlier measurement. The point on the left hand side of the dashed lines are the normal data, on the right hand side the outliers. From left to right shows the scoring function for $K = 10$, $K = 20$ and $K = 30$.

Both for inliers and outliers it seems to be a fluctuation in the distance to the cluster means. This might be to that some clusters has data with a higher spread, which also can been seen in figure (11). This would mean that the mean distance to each cluster mean within the nominal model should as well be fluctuating.

Figure (13) presents the performance results for different detection thresholds based on the mean and standard deviation distance to cluster means in the nominal model. Recall that $\hat{d}_k$ is the mean Euclidean distance from each data-point in cluster k to the cluster mean, $\hat{d}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} ||\mathbf{x}_{i,k} - \boldsymbol{\mu}_k||^2$. Here, N_k is the number of points in the training data-set assigned to cluster k, $x_{i,k}$ is each data-point assigned to cluster k and $\boldsymbol{\mu}_k$ is the cluster centre. The variance of the distance within cluster k is denoted $\sigma_{d,k}$.

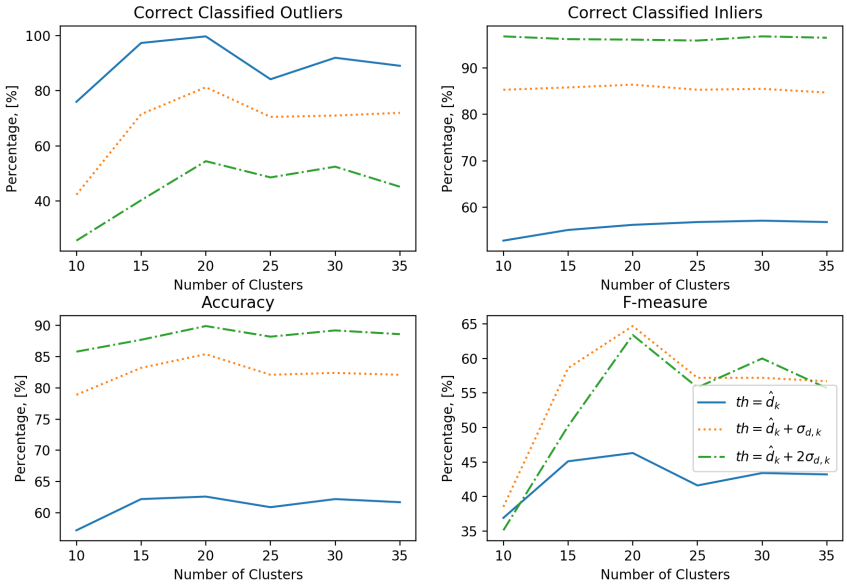


Figure 13: Performance measurements for different set-ups for detection threshold.

The maximum accuracy is 89.2%. Thus, that set-up could only correctly classify approximately 50% of the outliers. Therefore, to increase the algorithms ability to classify outliers, a more appropriate choice of threshold is $th = \hat{d}_k + \sigma_{d,k}$. This threshold performe best when using 20 clusters. The confusion matrix corresponding to this set-up is presented in figure (14).

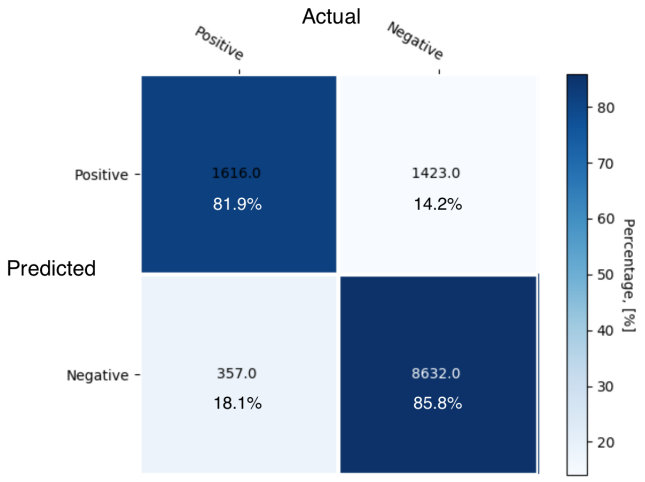


Figure 14: Confusion matrix when $K = 20$ and $th = \hat{d}_k + \sigma_{d,k}$. The numbers presented in the confusion matrix are the number of points which is classified according to each category.

6.1.2 Gaussian Mixture Modelling, GMM

When using the expectation maximization algorithm to learn the superposition of Gaussian, there is one hyperparameter to tune, which is the number of distributions to use. In figure (15) the average silhouette values for different number of distributions are presented. The average silhouette values indicates that the most appropriate number of Gaussian distributions are $K = 10$. As discussed above, the low Silhouette values indicates that the clusters are not well defined, which could be due to overlapping structures.

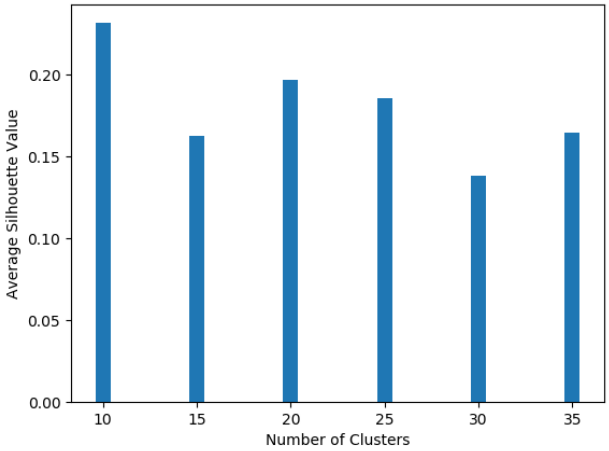
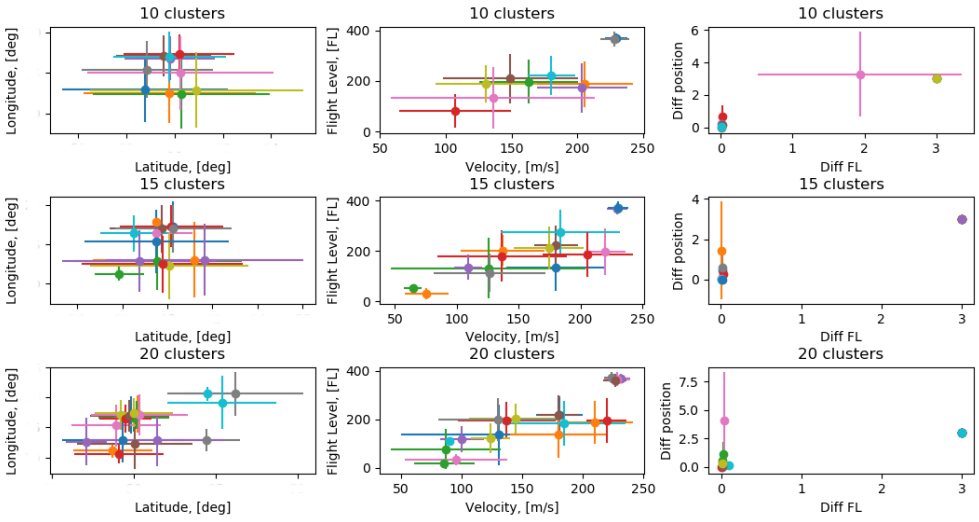


Figure 15: Average Silhouette values for different number of Gaussian distributions.

In figure (16) the means and standard deviation of features within each cluster are presented. As seen, when using more than, or equal to 20 clusters the spread is getting lower. When looking in the feature mean for the diff position and diff FL, GMM seems to separate flight, with and without a flight plan. This partitioning was not seen when using K-means.



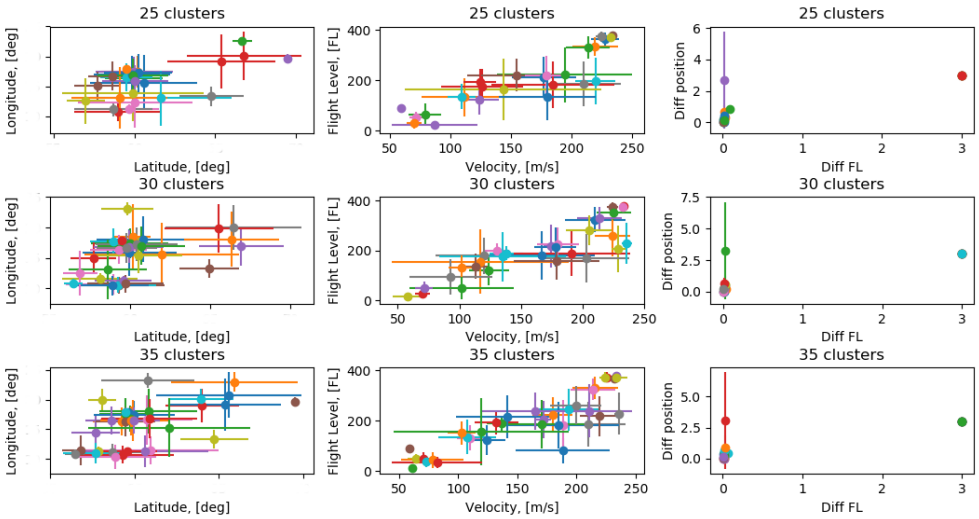


Figure 16: Feature means for different number of Gaussian distributions. The dot represents the mean and the lines the variance. The metrics used are: latitude, longitude [deg], Flight Level [Fl], Velocity [m/s] and Diff FL, Diff position in relative difference from flight path. Note. A relative difference of 3 indicates that there was no flight plan.

Similarly as for K-means, the average number of data-points within each cluster gets more evenly distributed when increasing the number of clusters. A figure showing this can be seen in Appendix A.

The scoring function values for each data-point in the validation data-set is presented in figure (17). As seen in the figure, the scoring value generally differs between inliers and outliers, such that the value for an outlier is generally lower. When increasing the number of cluster, the scoring function gets values are smaller in magnitude.

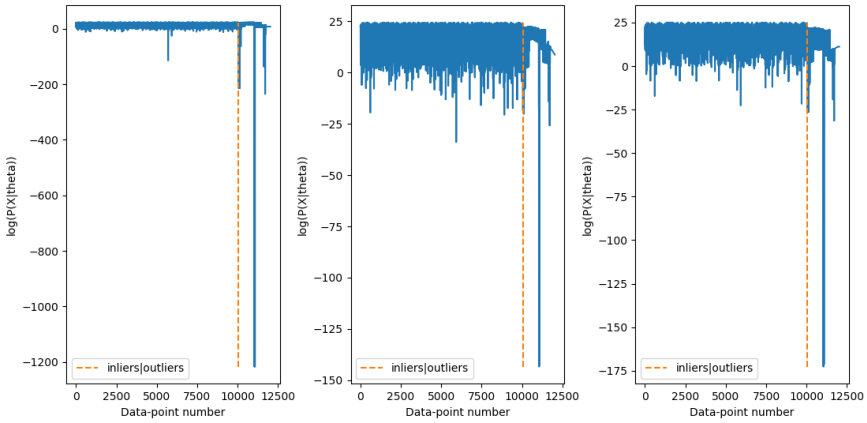


Figure 17: Score function, $\ln P(X_n|\theta)$. The points on the left hand side are normal, the one on the right, the outliers. From left to right shows the scoring function for $K = 10$, $K = 20$ and $K = 30$.

The threshold for determining if a point is an outlier is based on the mean and standard deviation of the scoring function within the training data-set. The performance of the outlier detection for different set-ups of the threshold is shown in figure (18). Recall that $\hat{p}_k$ is the mean of the score function $\ln P(X_{train,k}|\theta)$ where $X_{train,k}$ is the data-points in the training data-set assigned to cluster k, i.e., $\hat{p}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \ln(P(X_{n,k}|\theta_k))$. Here, N_k is the number of data-points assigned to cluster k, $X_{n,k}$ is the data-points assigned to cluster x and θ_k is the trained parameters of the Gaussian density function which corresponds to cluster k. The variance of the scoring function within cluster k is denoted $\sigma_{p,k}$.

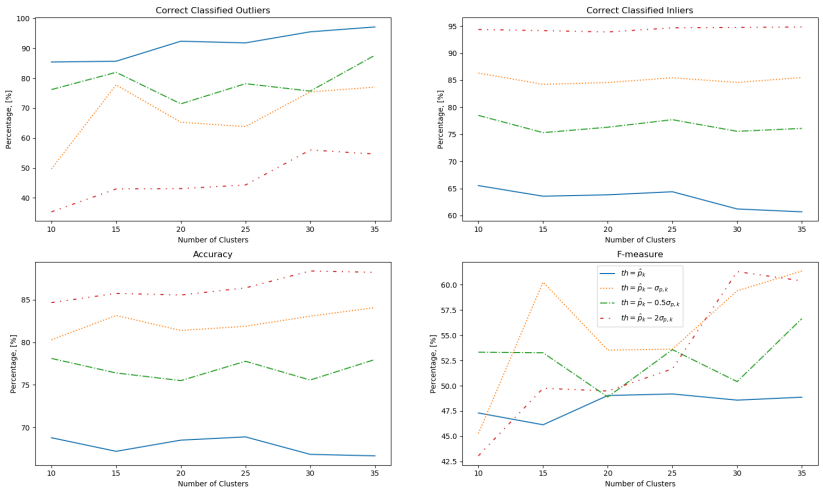


Figure 18: General outlier detection performance on validation data-set for different set-ups of parameters K and th.

The maximum accuracy of 89.2% and a F-measure of 59.8% is obtained when using 30 clusters and $th = \hat{p_k} - 2\sigma_{p,k}$. Thus, it lacks in performance of detecting outliers. Therefore, an more appropriate choice is using 25 clusters with $th = \hat{p_k} - \sigma_{p,k}$. Then the accuracy is 83.3% and the F-measure is 52.2%. The corresponding confusion matrix is presented in figure (19).

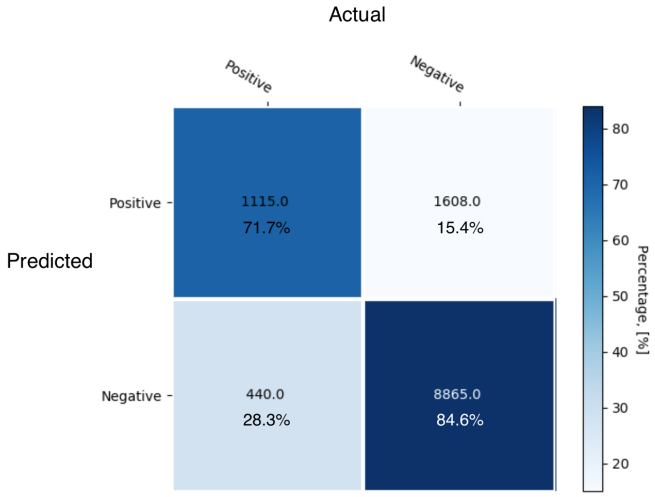


Figure 19: Confusion matrix when using $K = 25$ and $th = \hat{p_k} - \sigma_{p,k}$.

6.1.3 Spectral Clustering

In spectral clustering the hyperparameters are the width of the Gaussian kernel (from here on redefined as $\sigma = \frac{1}{2\tilde{\sigma}^2}$) and the number of clusters, K. In figure (20) and (21) the affinity matrix of the training data-set and the corresponding Laplacian's eigenvalues and first two eigenvectors are presented. The affinity matrix shows the similarity between each of the data-points. In figure (20) the first, second and third eigengap are presented. They are sorted in decreasing order, where the first are the largest eigengap. When $\sigma = 1$ and $\sigma = 10$ the first eigengap is between the $10th$ and the $11th$ eigenvalues. By using eigengap heuristics an appropriate choice of number of clusters should be $K = 10$. When $\sigma = 50$ the largest eigengap is shifted such that it is between the $16th$ and the $17th$ eigenvalue. Then, by using eigengap heuristics an appropriate choice of number of clusters would be $K = 10$. However, due to to complexity of the data, both $K = 10$, $K = 14$ as well as $K = 16$ will be investigated.

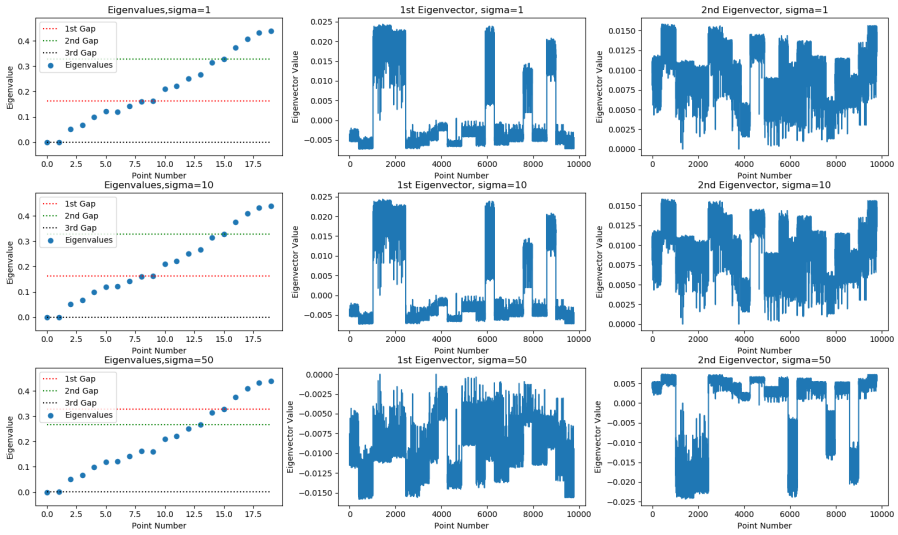


Figure 20: The first 20 eigenvalues and first two eigenvectors for different choices of σ.

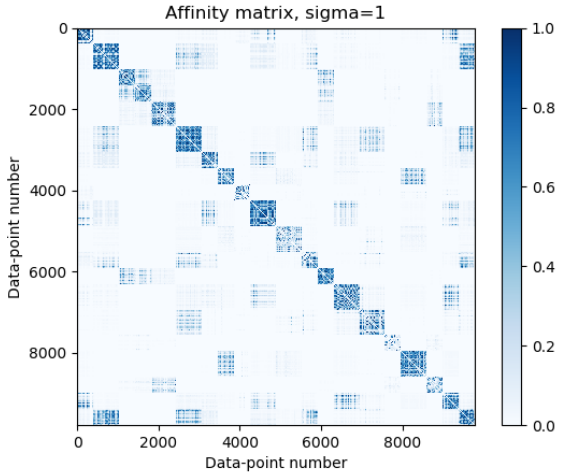


Figure 21: The affinity matrix of training data.

The corresponding affinity matrix in the feature space, when using the first 10 eigenvectors of the Laplacian is shown in figure (22). The similarities becomes

a bit more well defined, however it still has a complex character.

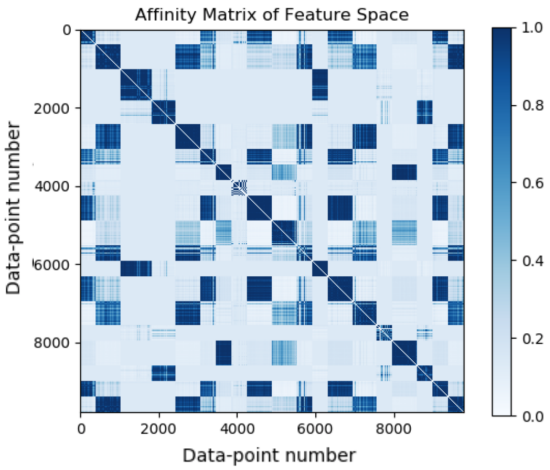


Figure 22: The affinity matrix of the training data-set in the feature space.

Figure (23) shows the average Silhouette values for $\sigma = 0.5, 1, 10, 50$ and for $K \in \{10, 14, 16\}$. The average Silhouette values is the same for $\sigma = 0.5, 1, 10$ and the value decreases if $\sigma = 50$. The maximum Silhouette value is obtained for $K = 14$ for all values of σ.

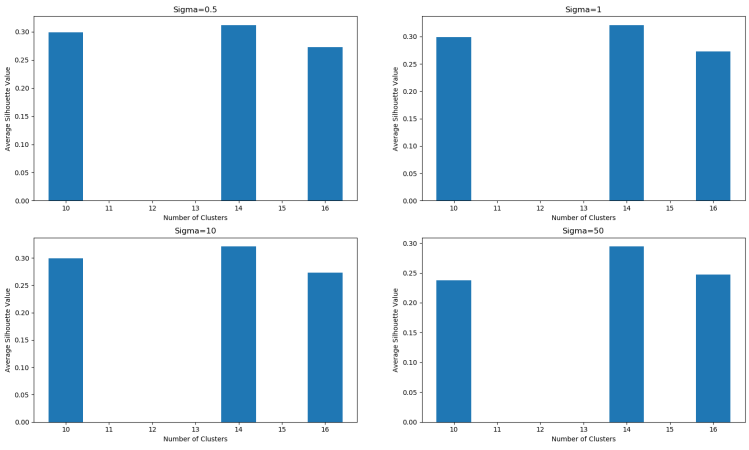


Figure 23: Average silhouette values for different choices of Gaussian widths and number of cluster. Note that sigma= $0.5, 1, 10$ obtained the same values.

In figure (24) the feature mean within each cluster when $\sigma = 1$ is presented. In this case, the difference between the different number of clusters is not as obvious as for K-means and GMM. However, there is a slightly decrease in variance within each feature when increasing the number of clusters,

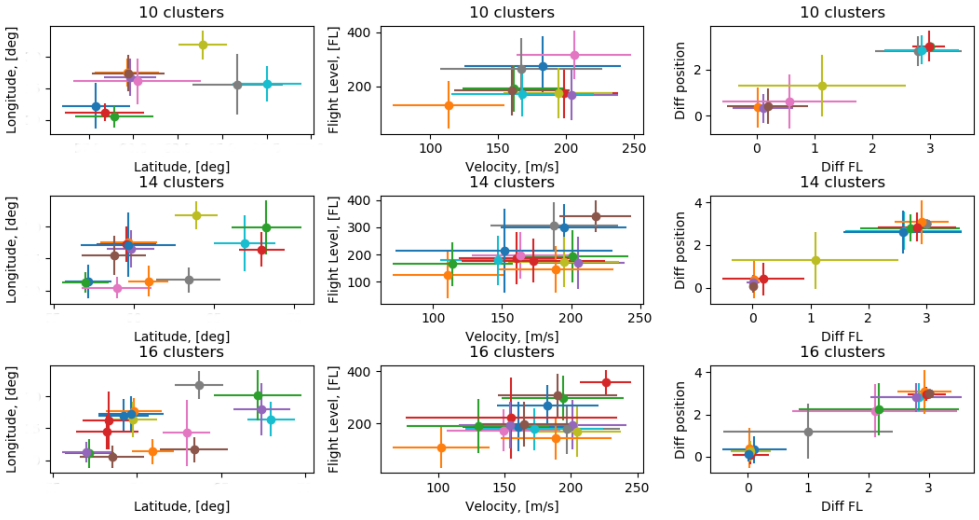
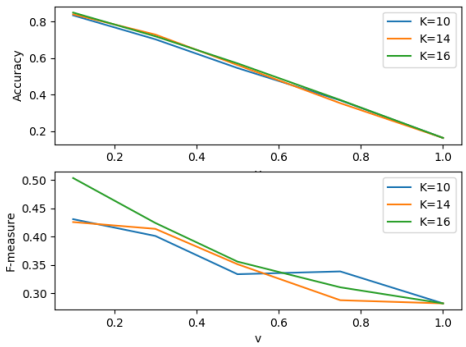


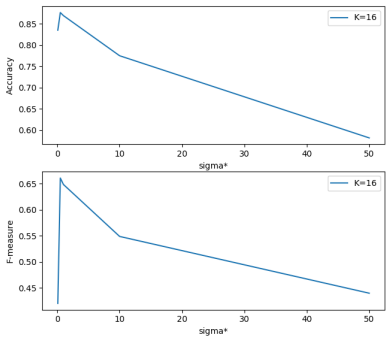
Figure 24: Feature means for different number of clusters. The dot represents the mean and the lines the variance. Metrics used are: latitude, longitude [deg], Flight Level [Fl], Velocity [m/s] and Diff FL, Diff position in relative difference from flight path. Note. A relative difference of 3 in the indicates that there was no flight plan.

In Appendix A a figure showing the data-point distribution over each cluster is presented. Similar as for K-means and GMM, an increased number of cluster does distribute the data more evenly.

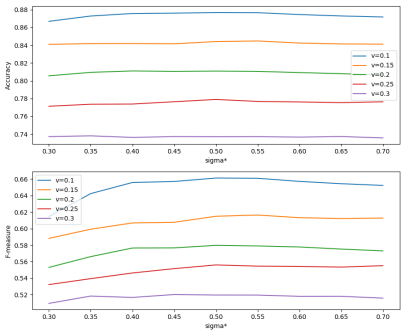
When testing the performance of the outlier detection, two new hyperparameters must be tuned, the detection threshold, ν and the Gaussian width of the kernel, $\sigma*$ used in OCSVM. In figure (25a), $\sigma* = 1$ is kept constant. The performance is measured when changing ν and number of clusters, K. Note that, as long as nothing else is mentioned, the performance is based on the outlier detection in the real space.



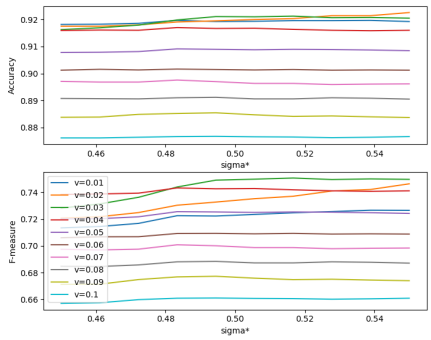
(a) $K \in \{10, 14, 16\}$



(b) $\sigma^* \in (0, 50],\ K = 16$



(c) $\nu \in \{0.1, 0.15, 0.2, 0.25, 0.3\}$, $\sigma^* \in [0.3, 0.7],\ K = 16$



(d) $\nu \in [0.01, 0.1],\ \sigma^* \in [0.44, 0.55]$, $K = 16$

Figure 25: Accuracy and F-measure for different set-ups on ν, σ^* and K.

Based on the results presented in figure (25a), the detection threshold should be around $\nu \approx 0.1$. The F-measure indicates that $K = 16$ is the best performed setting and will be used from here.

Next, the impact of changing the kernel width σ^* is investigated. The result is presented in figure (25b). From the figure it can be seen that $\sigma^* \approx 0.5$. In figure (25c) both σ^* and ν are changed in an interval close to the values mentioned above. The maximum accuracy and F-measure are obtained for $\sigma^* = 0.5$ and $\nu = 0.1$. In figure (25d) a more narrow range of σ^* and ν are presented. The maximum accuracy of 92.1% and maximum F-measure of 75.1% is obtained with $\sigma^* = 0.517$ and $\nu = 0.03$. The corresponding confusion matrix when using these parameters is presented in figure (26).

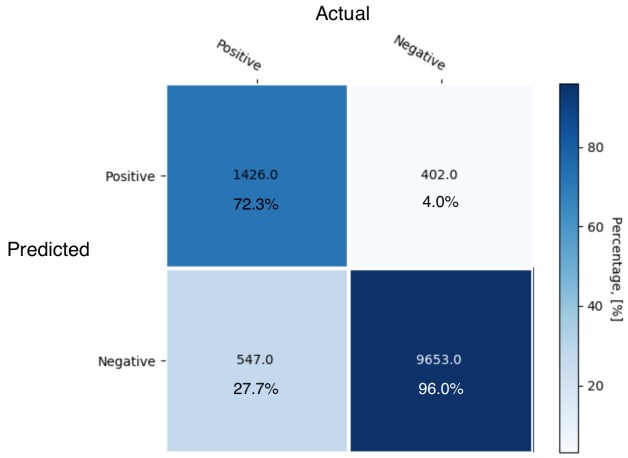


Figure 26: Resulting confusion matrix of outlier detection of the validation data-set with $\nu = 0.03$ and $\sigma^* = 0.517$.

The first three detection-functions obtained when solving problem (40) is shown in (27). A summary of the solutions to problem (40) is presented in table (3). The number of support vectors is equal to the number of non-zero elements in the dual variable α.

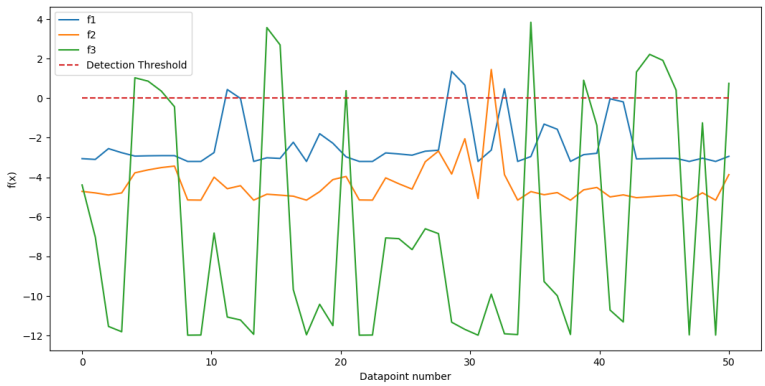


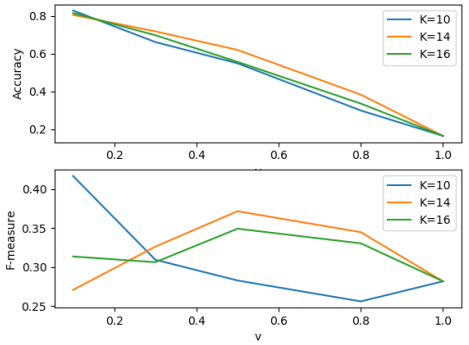
Figure 27: The first three detection functions. If any of the functions classifies a point as an inlier, it will be classified as an inlier.

Function number	**# iter**	**Objective Function**	ρ	**# Support Vectors**
1	64	25.30	3.20	25
2	67	73.56	5.16	22
3	58	209.30	11.98	39
4	42	70.54	5.80	31
5	84	53.55	5.28	29
6	46	57.88	6.11	23
7	64	2.95	0.97	24
8	65	18.73	2.95	23
9	46	38.29	3.81	27
10	30	7.44	1.84	12
11	64	45.22	5.05	25
12	38	19.76	3.05	19
13	94	4.66	0.87	34
14	66	23.04	2.67	26
15	51	3.65	1.22	19
16	64	77.37	6.15	30

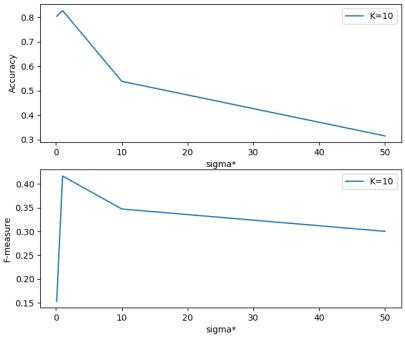
Table 3: The solution obtained by solving problem (40).

In total, 408 support vectors must be used to detect if a new data-point is an inlier or outlier.

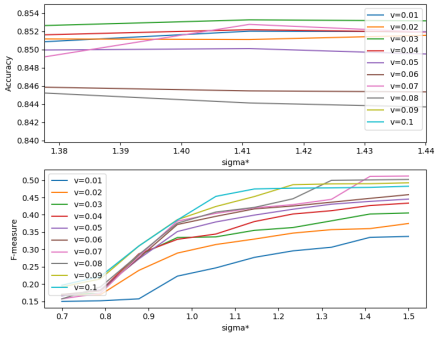
When using outlier detection in the feature space the same hyper-parameters for OCSVM must be re-tuned. In figure (28a) the performance when changing ν and number of clusters K is presented.



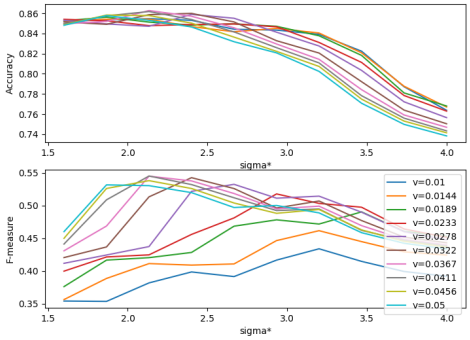
(a) $K \in \{10, 14, 16\}$



(b) $K = 16,\ \sigma^* \in (0, 50]$



(c) $\nu \in [0.01, 0.1],\ \sigma^* \in [0.7, 1.5],$
$K = 16$



(d) $\nu \in [0.05, 0.01],\ \sigma^* \in [1.5, 4],$
K=16

Figure 28: Accuracy and F-measure for different set-ups on ν, σ^* and K when using outlier detection in the mapped feature space.

Results presented in figure (28a) proposes to use $K = 10$ clusters and detection threshold, $\nu \approx 0.1$. From here on $K = 10$ will be used.

In figure (28b) $\nu = 0.1$ is kept constant. This result proposes a kernel width of $\sigma^* \approx 1$. Figure (28c) shows the performance when changing both σ^* and ν according to the results above. From this result the best choice is $\sigma^* \approx 1.41$ and $\nu \approx 0.07$. A more precise tuning of the hyperparameters are presented in figure (28d). The maximum accuracy of 86.3% and maximum F-measure are obtained with $\sigma^* = 2.134$ and $\nu = 0.0367$. The corresponding confusion matrix for this parameter setting is presented in figure (29).

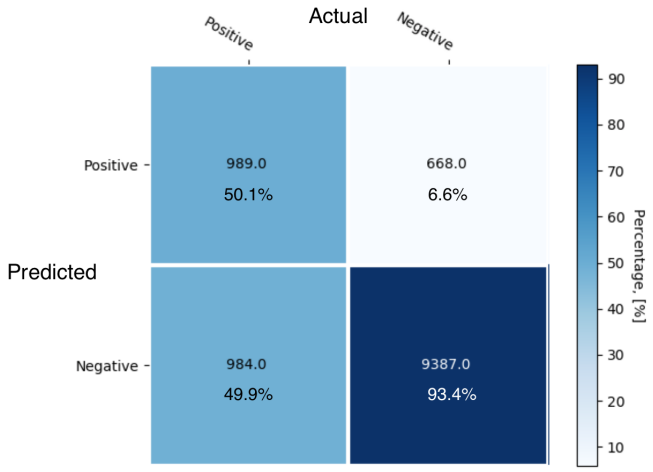


Figure 29: Resulting confusion matrix of outlier detection of the validation data-set with $\nu = 0.0367$ and $\sigma^* = 2.134$.

The corresponding optimal solutions of problem (40) for each of the decision functions is presented in table (4).

Function number	**# iter**	**Objective Function**	ρ	**# Support Vectors**
1	117	1089.79	31.35	75
2	70	476.98	21.06	51
3	48	261.85	15.66	37
4	50	593.5	30.03	43
5	36	274.40	20.40	29
6	60	101.58	9.24	26
7	128	448.22	15.85	64
8	66	85.85	6.93	32
9	36	50.97	6.97	17
10	19	40.15	7.49	13

Table 4: The solution after solving problem (40).

In total, 387 support vectors must be used to detect if a new data-point is an inlier or an outlier. This is a slightly decrease from the number of support vectors that must be used in the real space.

6.1.4 Suggested Hyperparameters and Performance

The best accuracy and F-measure obtained by each method is presented in table (5) and with corresponding parameters in table (6).

	Accuracy	F-measure
K-means	85.2%	64.5%
GMM	83.3%	52.2%
Spectral + OCSVM	**92.1**%	**75.1**%
Spectral* + OCSVM	86.3%	54.5%

Table 5: Best Accuracy and F-measure obtained on the validation data-set. Note that Spectral* means that the outlier detection is done in the feature space.

	K	σ	σ^*	ν	th
K-means	20	-	-	-	$> \bar{d}_k + \sigma_{d,k}$
GMM	25	-	-	-	$< \log(P(X\|\theta)) - \sigma(\log(P(X\|\theta)))$
Spectral + OCSVM	16	1	0.517	0.03	$== -1$
Spectral* + OCSVM	10	1	2.134	0.0367	$== -1$

Table 6: Proposed parameter settings for each method used. Note that Spectral* means that the outlier detection is done in the feature space.

The average detection time, i.e., the time it takes to classify a new data-point as an inlier or an outlier is presented for the different algorithms in table (7). The fastest algorithm was outlier detection based on K-means.

	K-means	**GMM**	**Spectral+OCSVM**	**Spectral*+OCSVM**
Time	**0.0002** s	0.0005 s	0.0014 s	0.0017 s

Table 7: The average detection time for a single data-point over 100 runs.

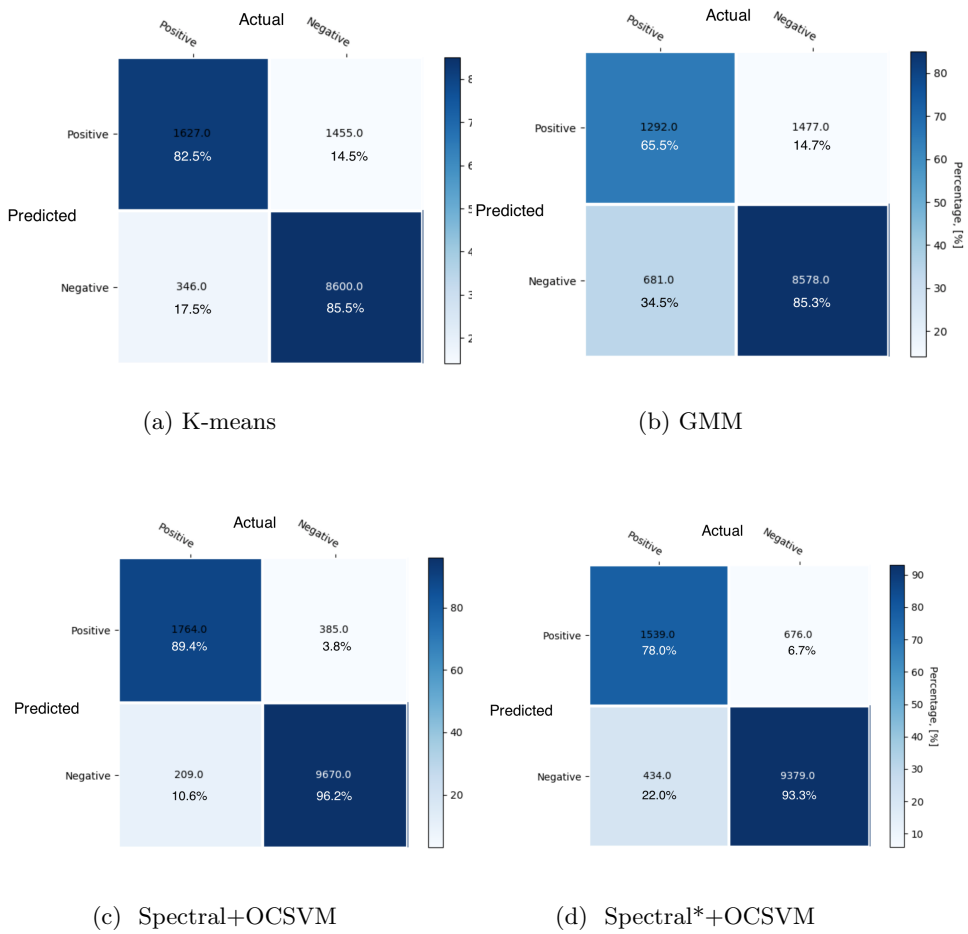
6.2 Performance

The outlier detection performance of each method using the suggested parameter set-up from section 6.1.4 is presented in table (8) and figure (30).

	Accuracy	F-measure
K-means	85.0%	64.4%
GMM	82.1%	54.9%
Spectral + OCSVM	**95.1**%	**85.4**%
Spectral* + OCSVM	90.8%	73.2%

Table 8: Accuracy and F-measure obtained on the testing data-set. Note that Spectral* means that the outlier detection is done in the feature space.

The confusion matrices for each of the algorithms is presented in figure (30a)-(30d). The algorithm that performed best, in the context of correctly classifying outliers was Spectral+OCSVM, see figure (30c). It should also be noted that the percentage of correctly classified outliers increased when using Spectral+OCSVM and Spectral*+OCSVM on the test data-set. However, the percentage of correctly classified inliers remained almost the same. The algorithm based on K-means had a similar performance on validation data-set and test data-set. The outlier detection algorithm based on GMM had a slightly decrease in performance when used on the test data-set compared to the validation data-set.



(a) K-means (b) GMM

(c) Spectral+OCSVM (d) Spectral*+OCSVM

Figure 30: Confusion matrix when using the suggested parameter set-up and test data-set.

7 Discussion and Conclusion

This section concludes the results of this thesis. It will also discuss future extents of this work and possible improvements.

7.1 Hyper-parameters

The choice of appropriate hyper-parameters is a challenging. Due to the high dimensional data-set it is hard to visualize its structure to get any information about cluster shapes and sizes. The Silhouette values was used as an metric on the cluster quality. The values was generally low which indicates complex structures and possibly overlapping clusters. However, it was still possible to find indications on an appropriate number of clusters. For k-means and GMM the choice of clusters was also depending on the data averages in each cluster and the general performance. One could find correlations between well separated clusters with lower spread and the performance of the outlier detection.

In spectral clustering the eigengap was used to decide the number of clusters. Due to the complexity of the data, the first eigengap might not be the optimal number of clusters. Therefore the first two largest eigengaps determined the number of clusters to test. The Silhouette values did indicate that the second eigengap would be the best choice. Thus, when the number of clusters was determined, it depended on the the eigengap, the Silhouette value and the performance of the outlier detection.

The Gaussian width, σ, as similarity measure was examined through the eigenvalue decomposition and the Silhouette values. One could find that $\sigma \in [0.75, 10]$ had similar behaviour, and therefore $\sigma = 1$ was chosen, since a higher σ tend to decrease the Silhouette value.

The scoring metrics for K-means and GMM was studied. There was a general distinction between the scoring of the inliers versus outliers. The scoring was generally noisy and therefore an appropriate threshold was challenging to find. The scoring could tell interesting behaviour of the data, however, it might not be an appropriate choice when classifying a single point as an inlier or outlier.

An interesting result that did arise was that GMM did partition those flights without flight plan into separate cluster, which neither K-means or spectral clustering did. This was the major difference in the cluster features between the methods. For all three methods, an increased number of cluster resulted in clsuters with lower spread and more evenly distributed data-point.

The hyper-parameters of OCSVM could easily be found by measuring the performance of the outlier detection. The trade-off between the true positive and the false positive rate was generally more even in this case than in both K-means and GMM. This might be due to the fact that spectral clustering better capture

the structure of the nominal model, or the fact that the algorithm uses a better outlier detection algorithm.

7.2 Performance

According to data needed, detection computational time and performance, there was one algorithm that outperformed the others. Using spectral clustering with OCSVM in the real space performed very well on both validation and test data-set. The outlier detection only depends upon the decision functions, which makes it relatively fast to compute when classifying a new data-point. However, as seen in table (7) it was the second slowest algorithm, but the diffrence between this algorithm and the fastest is not that big. As seen in the result the outlier detection performed better on the test data-set than on the validation data-set, see e.g., figure (30) and table (5). The increased performance was mainly at the true positive - false negative rate. This indicates that the outliers in the test data-set might be slightly easier to classify. However, it should be noted that the outliers were sampled from a data-set containing different suspicious flights with similar behaviour, and that the test and validation set does only have outliers generated from unique flights.

Outlier detection by using the spectral mapping is not an appropriate choice for the aim in this thesis. Its false positive rate was high and therefore missed many of the outliers. As discussed above, the performance increased when using the test data set. But similar as above, the true positive rate increased, which indicates that the outliers might have been "easier". Another downside of the spectral mapping is that it takes a lot more computational power and storage. The training data-set must be stored and used every time a new data point will be classified. For a small data-set it may be feasible. However, in reality a tremendous amount of data must be processed every second which makes this method a bad choice.

Both K-means and GMM performed surprisingly well considering its simplicity. These algorithm did have the fastest detection time, see table (7). However, they where outperformed by spectral clustering. The performance could possibly be improved by a more complex outlier detection measurement.

As mentioned in section 5 spectral clustering had one more step in the data pre-processing due to the memory constraint when computing eigenvalues. The data-set was sampled based on the resulting clusters obtained by K-means. This might be a factor to the performance success of the spectral clustering based outlier detection. Hence, K-means or GMM is proposed to be used in the data pre-processing to reduce the size of the training data-set.

7.3 Conclusion and Further Work

The aim of this thesis was to examine the possibility of using clustering based outlier detection algorithms to improve the situation awareness within ATC. In this thesis three different algorithms has been investigated, K-means clustering with a distance based outlier detection, GMM clustering with probability based outlier detection and Spectral Clustering with a One-Class Supoort Vector Machine for outlier detection. This thesis has shown that by the use of spectral clustering and OCSVM the outlier detection accuracy is over 90%. This concludes that its possible to train a nominal model that simulates the normal air traffic by the use of spectral clustering. Given the nominal model, OCSVM can detect outliers, as well as separating them from inliers. Therefore, an outlier detection algorithm based on spectral clustering and OCSVM could possibly be used to improve the situation awareness within ATC. However, its necessary with further testing of different cases of outliers.

In this thesis only simulated data has been used, which is generally smoother and without any disturbances, compared to real sampled data. Therefore, the algorithm will probably be less accurate on a real data-set. A further extent to this work would hence be to train, validate and test the algorithm on a real data-set and evaluate its performance.

The data-set used contains discrete time series of tracking points. These point are then partitioned into clusters. By calculating the occurrence of data within each cluster and the transition rate to next cluster, a natural further extent of this work would be to model a Markov Chain. Given a data-point, belonging to any cluster, the Markov chain can then calculate the most likely transition. This could then be used as a measurement in the outlier detection, i.e. does the flight follow the most likely pattern.

Appendix A

Data-point distributions for each cluster, K-means.

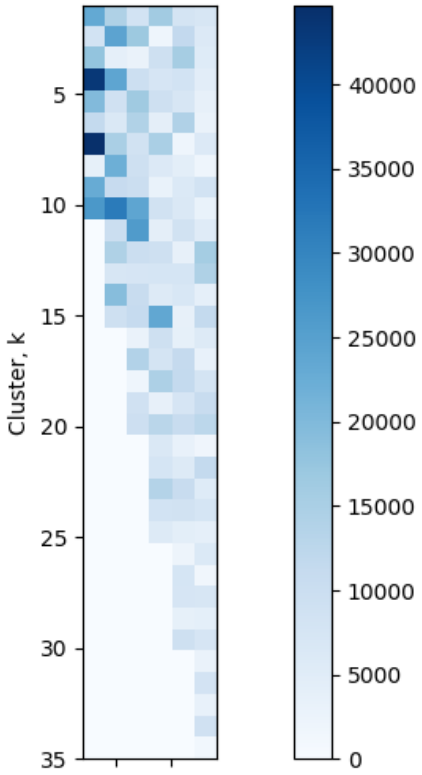


Figure 31: Distribution of data-points within in each cluster, K-means.

Data-point distribution for each cluster, GMM.

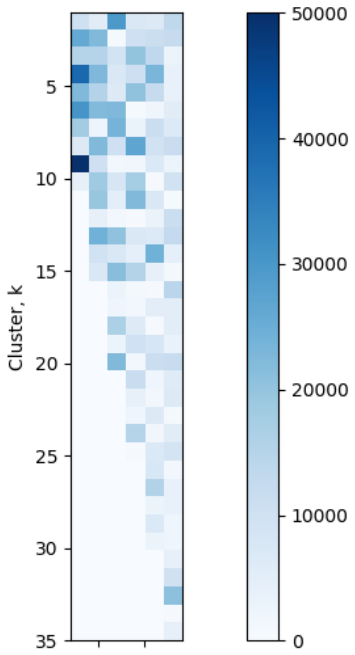


Figure 32: Distribution of data-points in each cluster, GMM

Data-point distribution for each cluster, Spectral Clustering.

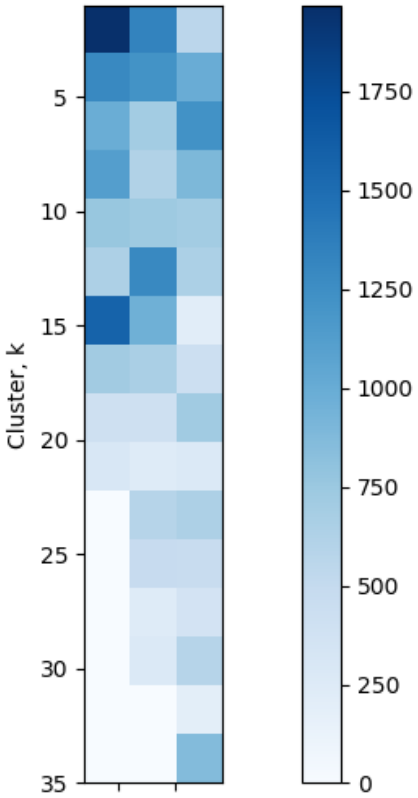


Figure 33: Distribution of data-points within each cluster.

References

[1] Y. Bengio, P. Vincent, and J-F. Paiement. Spectral clustering and kernel pca are learning eigenfunctions. *CIRANO*, 2003.

[2] CM. Bishop. *Pattern Recognition and Machine Learning*. Springer, Cambridge, 2006.

[3] C. Chang and C. Lin. Libsvm: A library for support vector machines. 2001.

[4] F. Chung. *Lectures on Spectral Graph Theory*. University of Pennsylvania, Philadelphia, Pennsylvania 19104, 1997.

[5] Eurocontrol. Eurocontrol manual for airspace planning. -common guidlines'. 2, 2003.

[6] Eurocontrol and Performance Review Comission. Performance review report: An assessment of air traffic management in europe during the calaender year 2018. 2018.

[7] V. Flovik. How to use machine learning for anomaly detection and condition monitoring. `https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection\-and-condition-monitoring-6742f82900d7`, December 2018.

[8] Försvarsmakten. Luftbevakningsoperatör officier. `https://jobb.forsvarsmakten.se/sv/utbildning/befattningsguiden/luftbevakningsoperator-officer/`, July 2019.

[9] D. Hawkins. Identification of outliers. *Chapman and Hall*, 1980.

[10] J. D. Hunter. Matplotlib: A 2d graphichs environment. `https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.537`, 2007.

[11] International Civil Aviation Organization. Aircraft tracking task force: Report and recommendations. `https://www.icao.int/safety/globaltracking/Documents/Aircraft%20Tracking%20Task%20Force%20Report%20and%20Recommendations%20-%20Final.pdf#search=tracking`, November 2014.

[12] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`, 2001–.

[13] D. Karger and C. Stein. A new approach to the minimum cut problem. Technical report, Laboratory for Computer Science Massachusetts Institute of Technology, Department of Computer Science Dartmouth College, December 1996.

[14] J. Kraiman, S. Arough, and M. Webb. Automated anomaly detection processor. *Proc. SPIE 4716*, 2002.

[15] Luftfartsverket. Aip sverige/sweden: 1.5 luftfartygs instrument. utrustning och dokument. `https://aro.lfv.se/Editorial/View/4605/ES_GEN_1_5_en`, August 2017.

[16] Luftfartsverket. Air traffic control. `https://www.lfv.se/en/services/airport-services/air-traffic-control`, July 2019.

[17] Luftfartsverket. Airspace - classification, ctr and tma. `https://www.lfv.se/en/services/airspace/obstacle-analysis/airspace`, September 2019.

[18] U. Luxburg. A tutorial on spectral clustering. Technical report, Max Planck Institute for Biological Cybernetics, 2006.

[19] S. Malakis, T. Kontogiannis, and B. Kirwan. Managing emergencies and abnormal situations in air traffic control (part i): Taskwork strategies. *Applied Ergonomics*, 41:620–627, 2010.

[20] S. Narkhede. Understanding confusion matrix. `https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62`, May 2018.

[21] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Proceedings of the 2001 Neural Information Processing Systems*, pages 849–856, 2001.

[22] T. Oliphant. A guide to numpy, 2006.

[23] S. Omar, A. Ngadi, and H. Jebur. Machine learning techniques for anomaly detection: An overview. *International Journal of Computer applications (0975-8887)*, 79(2), 2013.

[24] A. Patidar, J. Agrawal, and N. Mishra. Analysis of different similarity measure functions and their impacts on shared nearest neighbor clustering approach. *International Journal of Computer applications (0975-8887)*, 40(16), 2012.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] M. Riveiro, M. Lebram, and M. Elmer. Anomaly detection for road traffic: A visual analytics framework. *IEEE Transactions on intelligent transportation systems*, 18(8):2260–2270, 2017.

[27] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 1987.

[28] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 2001.

[29] D. Sisodia, L. Singh, S. Sisodia, and K. Saxena. Clustering techniques: A brief survey of different clustering algorithms. *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, 1(3), 2012.

[30] S. Walt, C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. `https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37`, 2007.

[31] M. Yasar. Flight anomaly tracking for improved situation awarness: Case study of germanwings flight 9525. Technical report, United Technoligies Research Center, East Hartford, CT, 06118, USA, 2016.

TRITA -SCI-GRU 2019:396