Stockholms universitet

Supervised Outlier Detection via Binary Classification: A Simulation Based Analysis

Arvand Jourabian

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2026:1
Matematisk statistik
Februari 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2026:1**
http://www.math.su.se

Supervised Outlier Detection via Binary Classification: A Simulation Based Analysis

Arvand Jourabian*

February 2026

Abstract

Detecting outliers is important in many applications and often requires problem specific solutions. A common framing is as binary classification. In this study, we compare Support Vector Machines (SVM), Random Forest (RF) and k-Nearest Neighbors (kNN) across four simulated scenarios, with known class labels. The scenarios are designed to vary class separability through cluster overlap, heavy-tailed outliers, outlier spread and boundary overlap. Models are trained in a supervised setting and evaluated using the Area Under the ROC Curve (AUC), F1-Score (F1) and Balanced Accuracy (BA). Overall, Random Forest is the most robust, kNN performs weakest in the most difficult settings and SVM is often competitive but shows high variability. As overlap increases or outliers become less extreme, the performance declines, suggesting that method choice and tuning are task dependent.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: arvand.j@hotmail.com. Supervisor: Ola Hössjer, Johannes Heiny, Daniel Ahlberg.

Acknowledgments

I would like to thank my supervisor, Daniel Ahlberg, for guidance and feedback, and for dedicating time to support the completion of this thesis. I also thank Johannes Heiny and Ola Hössjer for their support, feedback and initial guidance during the earlier stage of the thesis. I further acknowledge the use of ChatGPT for assistance with language proofreading, LaTeX formatting, and for troubleshooting and optimizing R code.

Contents

1 Introduction **4**

2 Background **4**

3 Theoretical Background **5**
3.1 Classification 5
3.2 Outlier Detection as Binary Classification 6
3.3 k-nearest neighbors (kNN) 6
3.4 Support Vector Machines (SVM) 7
3.5 Random Forest (RF) 9
3.6 Anomalies 10
3.7 Evaluation Metrics 11

4 Simulation Study Design **12**
4.1 Overview 12
4.2 Classification Methods 13
4.3 Evaluation Setup 14
4.4 Scenarios Overview 14

5 Simulation Scenarios **14**
5.1 Scenario 1: Two Clusters 15
5.2 Scenario 2: Heavy-Tailed Outliers 16
5.3 Scenario 3: Uniform Outlier Noise 16
5.4 Scenario 4: Nonlinear Boundary 17

6 Results **18**
6.1 Overview 18
6.2 Results Scenario 1 18
6.3 Results Scenario 2 20
6.4 Results Scenario 3 22
6.5 Results Scenario 4 24

7 Discussion **26**
7.1 Summary and overall conclusions 26
7.1.1 Scenario difficulty 26
7.1.2 Scenario 1: Effect of overlap δ 27
7.1.3 Scenario 2: Effect of degrees of freedom ν 27
7.1.4 Scenario 3: Effect of noise spread L 27
7.1.5 Scenario 4: Effect of boundary overlap τ 27
7.2 Limitations 28
7.3 Suggestions for further simulations 28

1 Introduction

The need and ability to detect outliers, also referred to as anomalies, arises across a wide range of applications including finance, industrial processes, scientific research and medical diagnostics. Because anomaly detection is a broad and loosely defined problem, there are many ways to approach it. For example, methods for monitoring continuous real time sensor streams in manufacturing differ from methods for identifying disease in periodic medical screening data. As a result, an extensive set of techniques is available depending on the problem scope, the attributes of data and computational resources. Some techniques are statistical models, whereas others are distance or density based. Classification based techniques and reconstruction based methods are also commonly used. The following chapter provides details on selected methods

In this paper the performance of three commonly used methods for anomaly detection is evaluated in the context of a binary classification task. For example, in credit card fraud detection, transactions confirmed as fraudulent are labeled to class $y = 1$ and legitimate transactions to class $y = 0$. Supervised classifiers are then trained and evaluated under large class imbalance, since fraud is rare [1]. Our simulations mirror this binary framing and enable controlled supervised comparisons. One class represents the *normal* data and the other class represents *anomalous* observations. Here, the "normal" class refers to a larger class of observations in the initial data generating process, and the "anomalous" comprises a smaller class generated under different parameters. A supervised setting using simulated data with known class labels allows for a controlled setting in which the primary focus is on comparison of performance. The three models selected are Support Vector Machines (SVM), Random Forests (RF), and k-Nearest Neighbors (kNN) chosen for their widespread usage in classification and anomaly detection but also for their differing learning paradigms and properties [2].

The paper is conducted as a simulation study which allows full control over the data, the nature of the outliers, and the dimensionality of the feature space. The models are evaluated using standard performance metrics such as the area under the ROC curve (AUC), F1-Score (F1), and Balanced Accuracy (BA), and are compared across several simulated scenarios.

2 Background

Outlier detection is a field firmly rooted in statistics. Early work is wide ranging with some approaching outliers with statistical tests, where an outlier is considered as either belonging to a heavy tailed distribution or a separate "contaminating distribution" [3]. Other approaches use leverage and residual analysis to find types of multivariate outliers [4]. Methods based on

distance and density such as assigning a measure to detect outliers, Local Outlier Factor (LOF) [5], are examples of unsupervised approaches that do not require labels but may struggle with high dimensional data. With access to labels the task is somewhat simplified, since there is some representation or ground truth to draw from. The approach to finding labels varies from synthetic generation to domain specific expertise to identify and compile class labels, which may be an arduous task in fields such as finance and fraud detection [6]. The idea to explore different framings and approaches to an outlier detection task is common. For example exploring synthetic data as labeled outliers to be able to use supervised learning in acoustic anomaly detection [7]. In outlier detection using classification, the framing of one class versus binary classes has been examined with one class showing better performance for high class imbalance [8].

Building on this prior work, the aim of this paper is to compare three widely used supervised algorithms: SVM, kNN and Random Forest, across four controlled scenarios by using a binary classification approach.

3 Theoretical Background

The theory in this section is taken from [9],[10] and [11].

3.1 Classification

Classification is a *supervised* learning task in which a model assigns a d-dimensional observation vector $x = (x_1, \ldots, x_d) \in R^d$ to one of K discrete categories, also called classes, $y \in \{1, \ldots, K\}$. Given a labeled training set $\{(x_i, y_i)\}_{i=1}^n$, the aim is to learn a decision rule $h(\cdot)$ which minimizes the probability of misclassification on future data. By contrast, *unsupervised* methods attempt the same goal without class labels in the training data and is often used in circumstances where labels are unavailable. Subsequent sections 3.4-3.5 describe the three supervised classification algorithms examined in this paper: k-Nearest Neighbors, Support Vector Machines and Random Forest.

Let (X, Y) be random variables, then the theoretical optimum of a classifier is the *Bayes classifier*:

$$h^*(x) \;=\; \arg \max_{k \in \{1,\ldots,K\}} P\left(Y = k \,\middle|\, X = x\right).$$

The Bayes classifier minimizes the probability of misclassification when all errors are penalized equally. The resulting minimum is called the *Bayes error rate*. Because the conditional probabilities are often unknown in practice, learning algorithms approximate h^* by modeling the probabilities or by estimating the decision boundary that separates the classes. Performance is typically summarised in a *confusion matrix* of true positives (TP), false

positives (FP), false negatives (FN), and true negatives (TN). With this, metrics such as Accuracy, Recall and the true-positive rate can be derived. The metrics are detailed in a subsequent section 3.7.

When the set of possible classes is restricted to two, $K = 2$, the task is referred to as *binary classification*.

3.2 Outlier Detection as Binary Classification

In many outlier detection tasks, also known as anomaly detection, the framing can be done as a binary classification problem in which one class represents rare or abnormal observation and the other class represents the prevalent behavior. In this paper we will refer to the classes as outlier and inlier classes.

Two characteristics distinguish this framing from ordinary balanced classification. First, the outlier class often constitutes a smaller subset of the data leading to a severe class imbalance. Second, the cost of failing to flag an outlier is therefore usually considered higher than a false positive. As a result, evaluations such as the F1-Score that account for imbalance is of more importance; see Section 3.7.

In this paper the outlier class is labeled in all simulations which enables the use of supervised learning algorithms while having full control of the prevalence, separation and distribution of the outlier class. This is used to highlight how different classifiers respond to rare but contextually important observations, under controlled conditions

3.3 k-nearest neighbors (kNN)

The k-Nearest Neighbors (kNN) classifier is an instance-based non-parametric method, meaning it is based on distances. The method assigns a test point x_0 the most common class label among its k closest training observations, where distance is usually measured in the Euclidean norm. More specifically; for a test point x_0, let $\mathcal{N}_k(x_0)$ denote the set of its k closest training observations using the Euclidean distance $d(x_0, x_i) = \|x_0 - x_i\|_2$. kNN assigns x_0 to the most common class among those neighbors,

$$\hat{G}(x_0) \;=\; \text{majority}\{\, y_i : i \in \mathcal{N}_k(x_0)\},$$

in other words, whichever label appears most frequently in $\mathcal{N}_k(x_0)$.

The neighborhood size k governs model complexity: small k gives a highly flexible (low-bias, high-variance) decision boundary, whereas large k produces a smoother (high-bias, low-variance) boundary

As the feature dimension increases, the size of a local neighborhood grows rapidly and all points become nearly equidistant. This devalues the information of $\mathcal{N}_k(x_0)$ and therefore leads to poor performance.

Prediction is done by calculating distances from the test point to all n training points. This can result in a high computational cost along with high memory usage. Although this can be mitigated with different techniques and methods such as tree based search structures or approximate nearest-neighbor algorithms, these are beyond the scope of this study.

In the context of outlier detection, kNN is expected to perform well when outliers are isolated from inliers in feature space but may falter under large class imbalance or in high dimensional settings.

3.4 Support Vector Machines (SVM)

Suppose we have a d-dimensional hyperplane

$$\beta_0 + \beta_1 X_1 + \ldots + \beta_d X_d = 0.$$

Where β_0 are constants and $X = (X_1, X_2, \ldots, X_d)^T$ is a point on the plane.

This hyperplane can be used to classify test data into two classes $y_1, \ldots, y_n \in \{-1, 1\}$ depending on what side of the hyperplane the data is on, by calculating its sign.

This an intuitive way of classifying, by drawing a line. However with perfectly separable classes an infinite amount of hyperplanes can be constructed. This is solved by choosing the hyperplane that has the farthest minimum perpendicular distance to training observations. This distance is called the margin of the hyperplane and is maximized by:

$$\begin{aligned} &\max_{\boldsymbol{\beta}} M \\ \text{subject to } &\sum_{j=0}^{d} \beta_j^2 = 1, \\ &y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_d x_{id}) \geq M, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

Where the margin M is maximized over the β constants while keeping the observation on the correct side of the hyperplane and also not on M. The resulting plane is called the *Maximal margin hyperplane* and again the test observation is classified by determining its sign.

While this hyperplane handles perfectly separated classes, this scenario is not always the case. When classes overlap a soft margin is needed since a strict margin cannot be maintained. Slack variables $\epsilon_1, \ldots, \epsilon_n$ cohering to a chosen budget C are added to allow flexibility in the constraints of the optimization problem (1). This results in some observations being allowed to be inside the margin or being assigned the wrong class. This is called a *soft margin* and is defined similarly to (1) as thus:

$$
\begin{aligned}
&\max_{\boldsymbol{\beta},\boldsymbol{\epsilon}} M \\
&\text{subject to } \sum_{j=0}^{d} \beta_j^2 = 1, \\
&\qquad y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_d x_{id}) \geq M(1-\epsilon_i), \quad i = 1, \ldots, n \\
&\qquad \epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C.
\end{aligned} \tag{2}
$$

The main difference in this soft margin classifier is the added slack variables. For instance if C is set to 0 then we have the previous shown Maximal margin classifier. If the ith observation is on the correct side of the hyperplane then its slack variable $\epsilon_i = 0$. If $\epsilon_i > 0$ or $\epsilon_i > 1$ then the observation is inside the margin or on the wrong side of the hyperplane, respectively. This impacts the budget set by C. In the soft margin classifier it is only these observations that make up the support vectors and therefore determine the hyperplane. This leads to C being the parameter that controls the bias-variance trade-off and is often tuned by cross validation. The soft margin classifier is also known as *support vector* classifier.

To handle nonlinear boundaries between classes we can enlarge the feature space while still maintain linearity. For example the hyperplane $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ with expanded features (X_1, X_2, X_2^2) gives us $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_2^2 = 0$.

This concept, along with the fact that the solution to (2) only involves inner products of observations, leads to the linear soft margin classifier being written as

$$
f(x) = \beta_0 + \sum_{i \in S}^{n} \alpha_i \langle x, x_i \rangle. \tag{3}
$$

Here S is the set of indices for the support vectors of the training observations. This is because the parameter α_i, for which there is one for every observation, is only non-zero for the support vectors. Since this is the form for a linear classifier we can replace the inner product with a general form K called kernel:

$$
f(x) = \beta_0 + \sum_{i \in S}^{n} \alpha_i K(x, x_i). \tag{4}
$$

When the kernel K is replaced we perform the change in feature space as discussed earlier and allows for many different mappings. This extension of (3) is a Support Vector Machine (SVM). Commonly used kernel choices include linear, polynomial, and Gaussian radial basis function (RBF) kernels. The kernel we use in this paper is the RBF kernel and is defined as

$$K(x_i, x_k) = \exp\ (-\sigma \sum_{j=1}^{d} (x_{ij} - x_{kj})^2), \tag{5}$$

where σ sets the kernel width. A smaller σ gives smoother large-margin boundaries, and larger σ yields more flexible local ones. The euclidean distance between x_i and x_k is what causes the impact on the predicted class label to highly depend on local behavior.

Both hyper-parameters mentioned, C and σ, therefore influence the performance and suitable values can be selected by evaluating the models with different values. The method of selecting parameters for the simulation is outlined in section4.2.

SVMs often perform well in high dimensional spaces because predictions depend only on the support vectors, a subset of training points. However, when classes overlap heavily or when C and σ are poorly selected the margin may grow to include too many points, resulting in degraded accuracy and performance.

3.5 Random Forest (RF)

Random Forest is an ensemble method that combines the predictions of many classification trees, each grown on a bootstrap sample of training data. It leverages randomness by using random subsets of the data and in the feature selection when splitting individual trees. This makes the individual trees less correlated and such averaging their votes greatly reduces the variance. The final class prediction is obtained by majority vote across the ensemble of trees.

A decision tree is a model that uses a tree structure and is based on dividing the predictor space $X = (X_1, \ldots, X_d)$ into regions by *Binary splits*. At the starting region, the root *node*, a variable X_j is split according to some threshold t into left *child* node $\{x : x_j < t\}$ and right child node $\{x : x_j \geq t\}$. Repeated splitting results in regions R_m which are distinct and non-overlapping and belong to the mth node. Decision trees can be used for regression and classification tasks, for regression the split is chosen by minimizing the mean squared error.

However when constructing a classification tree, the most simple metric is the *classification error rate*, the fraction of training observations in the region that do not belong to the most common class. A more commonly used alternative, and the one used in this study, is the *Gini index*

$$\mathrm{G} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{6}$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region from the kth class. Nodes deeper in the tree, large m, hold fewer points and

therefore G tends to be smaller since $\hat{p}_{mk}$ is larger. The Gini index provides a measure of total variance across the K classes.

A bootstrap dataset is obtained by randomly sampling the training data with replacement to the same size. This is done to produce B bootstrap datasets. The averaging over some estimate from these sets is known as *bootstrap aggregation* (bagging) and reduces variance.

In Random Forest a bootstrap sample is drawn from the training data and a tree T_b is recursively grown, until a minimum node size n_{min} is reached, according to the following procedure:

i. Select m_{try} variables randomly from p total features.

ii. Pick the best split-point among the m_{try}.

iii. Split the node into two child nodes.

The split point in (*ii*) is chosen to minimize the weighted average Gini index (6) of the two child nodes. This is the same as maximizing the Gini decrease. If $m_{\text{try}} = p$ the algorithm reduces to bagging, since every split considers all features. This procedure is done for all bootstrap samples $b = 1, \ldots, B$ and produces an ensemble of trees $\{T_b\}_1^B$. When classifying an observation x, the prediction of the ensemble is summarized by a majority vote,

$$\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B \tag{7}$$

using the individual class predictions $\hat{C}_b(x)$ of the bth random forest tree.

The hyper-parameters of the algorithm control the structure of the individual trees (depth, node size) or the ensemble (the number of trees B). Commonly tuned parameters are randomly selected variables m_{try}, node size of the trees and total number of trees B.

3.6 Anomalies

Anomalies (outliers) are observations that deviate from the behavior and distribution of the majority (inliers). Although it can be challenging to provide a strict universal definition generally, an anomaly can be described as a pattern that does not follow expected normal behavior [2].

A conceptual way to formalize this idea is to define a *Normal region* in the feature space of our working domain and consider everything outside this region as an outlier. In practice this approach can be difficult to apply. A true normal region is hard to specify because of what counts as normal can be diverse. The boundary of the normal region may be a challenge to define since observations may belong to either region when overlap or extreme outliers exist. In many cases the concept of normality can also be seasonal or evolve to not be of representative behavior in the future. Applications of anomaly detection that are focused on fraud or intrusion detection also face

difficult hurdles in defining normal behavior. In these cases malicious actors may try to mimic normal data to deliberately avoid being outside a normal region. Many similar challenges, unique to each application and industry, exist which require definitions of normality to depend on their domain. For example small deviations in medical data may be of highly anomalous while the same magnitude of fluctuations in stock market data may be considered normal. A large and always present obstacle with anomalies are the data challenges. Often availability of labeled data for training and validation of models is a concern which may require a solution in itself. Class imbalance, since anomalies are rare, is also an important aspect which differentiates it from other classification tasks.
Because of these issues, most anomaly detection methods employ solutions tailored to the data type, label availability and anomaly definition of their problem domain. The simulation scenarios introduced in Chapters 4 and 5 are designed to reflect some of these challenges.

3.7 Evaluation Metrics

In supervised classification performance is often summarized in a confusion matrix that contains counts of outcomes. These counts are detailed in Table 1 and are the basis of classifier evaluation metrics. For a classifier that outputs a probability score $p(x) = P(Y = 1 \mid X = x)$, varying the decision threshold c, for which the prediction is compared against, impacts our metrics. A Bayes classifier, as described in Chapter 3.1, assumes a threshold of 0.5 and this is also usually the default value.

Table 1: Confusion matrix terminology.

Outcome	**Description**
True Positive (TP)	Outlier correctly classified as outlier
False Negative (FN)	Outlier incorrectly classified as normal
False Positive (FP)	Normal observation incorrectly classified as outlier
True Negative (TN)	Normal observation correctly classified as normal

The most intuitive is *Accuracy* which is simply the proportion of correctly classified observations from the whole

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

However in our case the Negative outcomes will be by far the highest counts due to the inherent class imbalance. As a consequence this metric will be inflated and the differences between Accuracy may only provide us with some insight. An alternative metric that addresses this issue is *Balanced*

Accuracy which averages the True-Positive Rate (TPR) and True-Negative Rate (TNR) as

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\text{TPR} + \text{TNR}\right) = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right).$$

This makes Balanced Accuracy less dependent on the class prevalence and a more informative choice for imbalanced classes. Two other metrics; *Precision* (positive predictive value) and *Recall* (sensitivity) are the basis of many other metrics and are defined as

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

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

These two metrics can be used to define the *F1-score*

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

which is a harmonic mean of Precision and Recall and provides a more weighted measure. The F1-Score is widely used when assessing classifier performance with imbalanced datasets but may still suffer in large imbalances for example with 1% positives[12]. A measure that has shown to perform well under these circumstances[13] is the *Receiver Operating Charactersitic* (ROC) Curve which uses True-Positive Rate and

$$\text{False-Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

When plotting TPR against FPR we get the ROC curve which for random guessing follows the diagonal and good performance shows the curve hugging the left upper corner. To summarize the information from the ROC curve, the *Area Under the (ROC) Curve* (AUC) is used and gives a single number. An AUC of 1 is a perfect ranking and 0.5 means no better performance than guessing.

The metrics used to determine performance in this paper are AUC, F1-Score and Balanced Accuracy.

4 Simulation Study Design

4.1 Overview

In the simulations, data is generated from known probability distributions to generate multivariate numerical observations belonging to two classes: a normal class and an outlier class. Both classes are labeled which allows for

a supervised learning setting. This structure enables a controlled evaluation of model performance across different scenarios and parameters.

The outlier class is designed to have attributes similar to real-anomalies, it is therefore a smaller subset of the whole data set. This class imbalance is intentional and an important aspect since this affects model behavior and mirrors realistic conditions.

Simulation also allows control over class ratios, dimensionality, data spread and noise. This provides the tools to design scenarios where the classification methods may respond differently. A key motivation for this approach is reproducibility and interpretability.

4.2 Classification Methods

The three classification methods kNN, SVM and RF were chosen due to their wide usage in classification and anomaly detection. They represent different learning paradigms which will allow for comparison of how these three methods handle different types of models with outliers and unbalanced data.

For the Support Vector Machine (SVM) a Radial Basis Function (RBF) kernel, a commonly used Gaussian kernel, is used due to its ability to model non-linear decision boundaries and requiring few hyperparameters. The parameters can be chosen to get a behaviour similar to linear and polynomial kernels. This allows for flexibility across data structures. The RBF kernel can perform well in high dimensional spaces but may be sensitive to noise or class overlap. The two main tunable parameters are C (regularization) and σ (width of kernel).

For Random Forest (RF) the tunable parameters are the number of trees **ntree** and the number of variables randomly selected at each split **mtry**. Since RF is an ensemble of decision trees that uses bagging and random feature selection it is robust to noise and nonlinearities and it is generally less sensitive to parameter tuning than the other models.

The non-parametric method k-Nearest Neighbors (kNN) is arguably the simplest and most interpretable model. It assigns class labels based on majority class among the k closest training points in feature space. This makes it sensitive to class imbalance and curse of high dimensionality. It is therefore highly dependent on appropriate choice of the parameter k.

Models are implemented and trained in R with the **caret** package for model training and tuning. SVM, kNN and Random Forest were implemented using the **e1071**, **class** and **randomForest** packages, respectively. AUC was computed using **pROC** and data generated with **MASS** (**mvtnorm** in Scenario 2). Cross-validation (5-fold) is used during the parameter tuning and the tuning grid is chosen manually. The aim is to provide some optimization for each scenario but not to the point of an exhaustive parameter search for individual scenarios.

4.3 Evaluation Setup

In each simulation labeled data are generated ($N = 500$ observations) and split into training and test sets according to a 70/30 ratio, so that the number of training data points is $n = 350$. To preserve class proportions of normal and outlier (10%) observations, stratified sampling is used, with 35 (15) outliers among training (test) data. Each simulation scenario is repeated multiple times (30 times) to reduce variability and obtain an averaged performance estimate. The performance of each model is evaluated using Balanced Accuracy, F1-Score and Area Under the ROC Curve as metrics.

4.4 Scenarios Overview

Four different scenarios, with simulated data in d dimensions, are investigated to reflect common challenges in anomaly detection. The first three scenarios have data from the normal (or inlier) class drawn from a standard multivariate (d-dimensional) Gaussian distribution, whereas the structure and distribution of the outlier class varies. For the fourth scenario, a combined data set is drawn from a standard multivariate (d-dimensional) Gaussian distribution, and outlier labels are assigned probabilistically using a soft radial boundary where overlap is controlled by parameter. The goal is to evaluate performance under varying types of distance, noise structure and overlap.

- **Scenario 1: Separated Clusters** Outliers form a class that is (partially) separated from the normal class, simulating separated anomalies.
- **Scenario 2: Heavy-Tailed Outliers** Outliers are drawn from a heavy-tailed distribution (t-distribution with varying degrees of freedom ν, which corresponds to a Cauchy distribution for $\nu = 1$) , making them harder to isolate.
- **Scenario 3: Uniform Outlier Noise** Outliers are spread across a uniform region, similar to dispersed and noise-like anomalies.
- **Scenario 4: Soft radial boundary (overlap)** Outlier labels are given based on their distance from origin, with overlap controlled by a parameter, representing uncertainty near borders.

5 Simulation Scenarios

In this section we give a more mathematical description of all four simulation scenarios. A data point from the normal and outlier classes are observations of random variables X_{normal} and $X_{outlier}$ respectively. The distributions of

these random variables for each scenario are described in the subsequent subchapter. Each scenario varies a specific parameter between four values and the data generation is done with a new seed. Since for each parameter the repetition is done 30 times we end up with a total of 120 repetitions overall for each scenario.

5.1 Scenario 1: Two Clusters

For Scenario 1, data from the normal and outlier classes will be generated from two different multivariate normal distributions, with the same covariance matrix but different expected values. More specifically, we have that

$$X_{\text{normal}} \sim \mathcal{N}(\mu_1, \Sigma), \quad X_{\text{outlier}} \sim \mathcal{N}(\mu_2, \Sigma)$$

$$\mu_1 = (0, \ldots, 0), \quad \mu_2 = (\delta, \ldots, \delta), \quad \Sigma = I_d.$$

Varying the mean δ (denoted μ_{shift}/μ in figures) of the outlier class to values $\delta \in [0.5, 1, 1.5, 2]$ for 30 repetitions while all other parameters stay fixed. Resulting in 30 repetitions for $\delta = 0.5$ and 30 for $\delta = 1$ and so on. The aim of this scenario is examine how the methods perform depending on how close and overlapping two differently sized classes are. With a larger separation of means the performance of the methods are expected to trend toward the trivial case of two clearly separated classes.

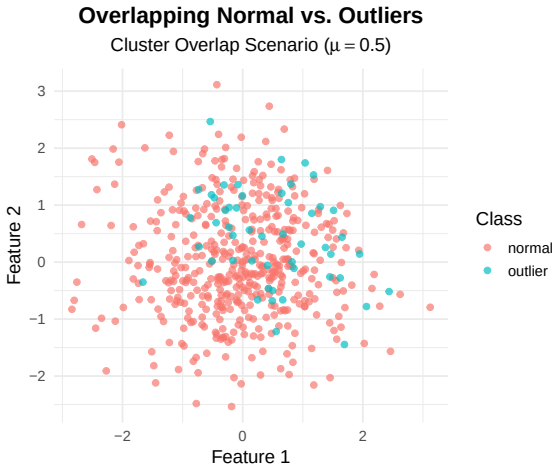

Figure 1: Visualization for $d = 2$ of Scenario 1, $\mu = 0.5$

In Figure 1 we see a visual representation of the 2D case with $\delta = 0.5$ where the outlier and normal class are clearly overlapping.

5.2 Scenario 2: Heavy-Tailed Outliers

For Scenario 2, data from the normal class are drawn from a standard normal distribution, whereas data from the outlier class are drawn from a radially symmetric, heavy-tailed distribution (Multivariate t). More specifically, we have that

$$X_{\text{normal}} \sim \mathcal{N}(\mu, \Sigma), \quad X_{\text{outlier}} \sim t_\nu(\mu, \Sigma)$$

$$\mu = (0, \ldots, 0), \quad \Sigma = I_d, \quad \nu = \text{degrees of freedom.}$$

Varying the degrees of freedom of the outlier class to be $\nu \in [1, 2, 5, 8]$. Following the same procedure as all other scenarios, this parameter is the only one that changes value. The expected behavior of the performance when increasing ν is for the methods to struggle more. This is due to outliers becoming less extreme and the distribution tending to the inlier class as ν increases and overlap becomes high.

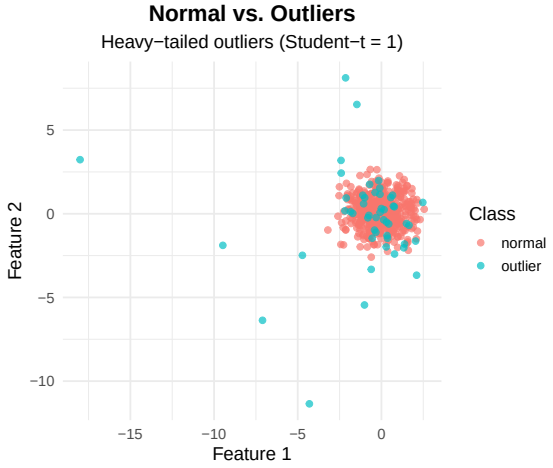

Figure 2: Visualization for $d = 2$ of Scenario 2 (Heavy-Tailed), $\nu = 1$.

In Figure 2 we have a visualization of the 2D case with $\nu = 1$. This is the parameter with the most amount of extreme outliers.

5.3 Scenario 3: Uniform Outlier Noise

For Scenario 3, data from the normal class are drawn from a standard normal distribution, whereas data from the outlier class are drawn from a rectangular, uniform distribution. More specifically, we have that

$$X_{\text{normal}} \sim \mathcal{N}(\mu, \Sigma), \quad X_{\text{outlier}} \sim \text{Uniform}([a_1, b_1] \times \cdots \times [a_d, b_d])$$

$$\mu = (0, \ldots, 0), \quad \Sigma = I_d.$$

Varying the size of the rectangles sides $a = b$, which forms the outlier class, to values $[3, 4, 6, 8]$ while keeping other parameters and variables fixed.

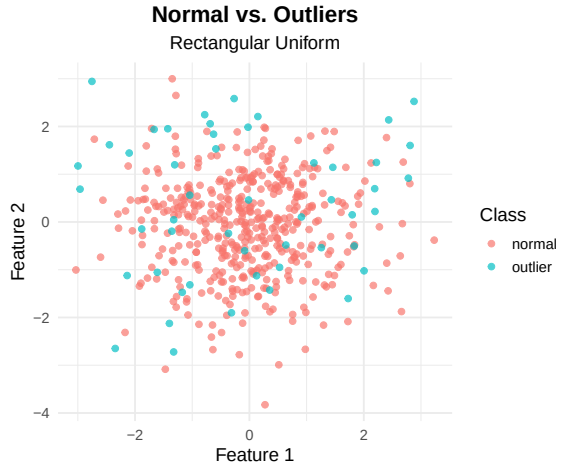

Figure 3: Visualization for $d = 2$ of Scenario 3 (Uniform Noise), $a = b = 3$.

The aim of this scenario is to simulate uniform noise outliers and how their spread impact method performance. Increasing the rectangular box size seen in Figure 3 makes the outliers more distant and should lead to improved metrics, depending on the method.

5.4 Scenario 4: Nonlinear Boundary

For Scenario 4, data is generated from a standard normal distribution and a radial score is computed from which an outlier label is assigned probabilistically from a soft radial boundary. More specifically we have that

$$X \sim \mathcal{N}(0, I_d), \quad r = \|X\|$$

where r is the computed radius in the soft radial boundary

$$P(Y = 1 \mid X) = \sigma\left(\frac{r - r_0}{\tau}\right).$$

Here $Y = 0$ ($Y = 1$) represents the normal (outlier) class, r_0 controls the location of the boundary (the radius where $P(Y = 1 \mid X) = 0.5$) and $\tau > 0$ determines the thickness in overlap. In this scenario all values are fixed, including $r_0 = 4$, and the parameter $\tau \in [0.4, 0.6, 0.8, 1.0]$ is varied.

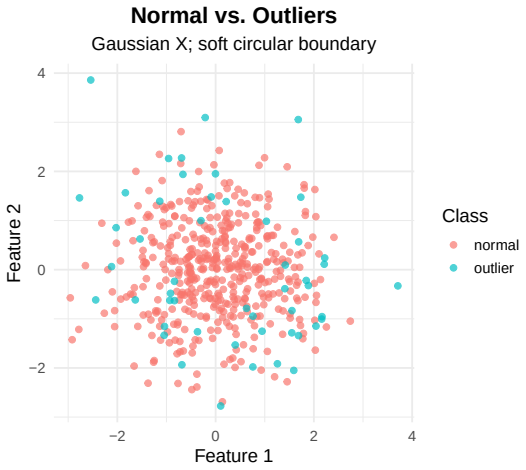

Figure 4: Visualization for $d = 2$ of Scenario 4 (Nonlinear), $\tau = 0.6$.

The Figure 4 illustrates Scenario 4, where the outliers are more likely at the edge of the radius. As τ increases the boundary becomes less sharp and the classes overlap more which should make the classification task more difficult. This scenario is similar to Scenario 2 in that separation is mostly radial but has labels assigned probabilistically to represent overlap and uncertainty near the boundary.

6 Results

6.1 Overview

In this chapter we present simulation results for each scenario. Performance is evaluated using *Balanced Accuracy* (BA), the *F1-Score* (F1) and the *Area Under the (ROC) Curve* (AUC), as defined in Chapter 3.7. For each scenario, we report an overall summary table with mean metric values per method averaged over all simulation repetitions, a figure with boxplots showing the distribution of metrics from individual simulations and a table of mean metric values for each parameter tested in the scenario.

6.2 Results Scenario 1

In Scenario 1, with two overlapping clusters, the three methods perform differently depending on how large the overlap is. Table 2 reports mean AUC, BA and F1 for each method averaged over number of simulation repetitions and all values of δ. Random Forest achieves the highest average performance across the reported metrics.

Table 2: Summary of averaged performance metrics of Scenario 1.

Method	AUC	F1 Score	Balanced Accuracy
Random Forest	0.801	0.436	0.677
SVM	0.727	0.365	0.644
kNN	0.776	0.410	0.666

Figure 5 shows how the distribution of AUC, BA and F1 across simulations for each δ. The main trend is performance gain as δ increases, across all metrics. Support Vector Machines has larger variability and lower median performance which is most clearly seen in AUC. For $\delta = 0.5$, the boxplots indicate a clear drop in performance.

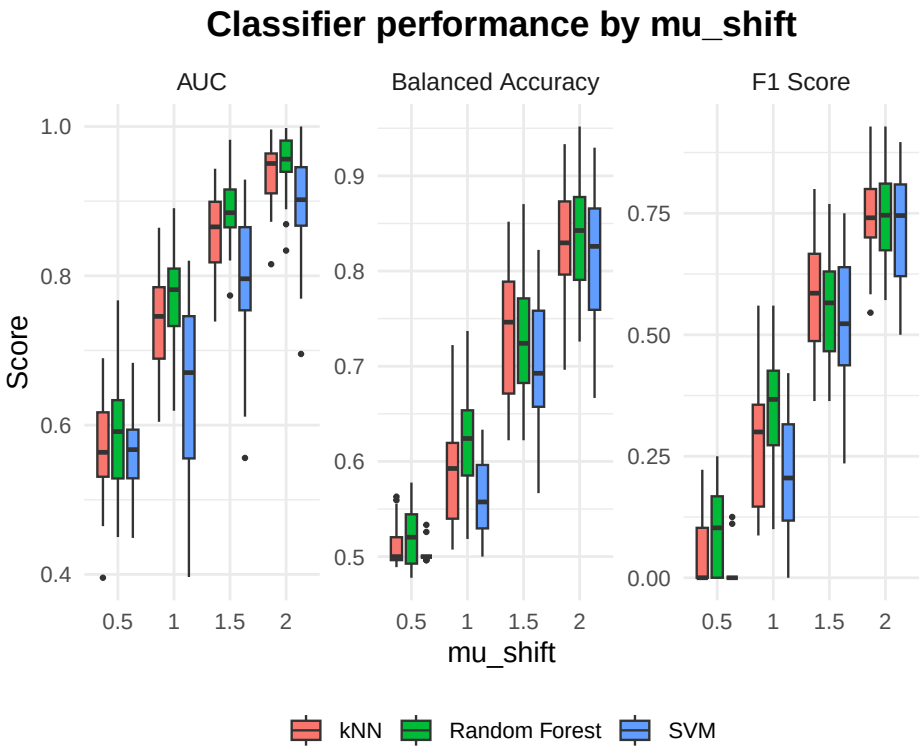

Figure 5: Performance metrics for Scenario 1.

The mean performance of the methods for each δ is presented in Table 3 where the overall performance gain is most clear in F1 between $\delta = 0.5$ and $\delta = 1$.

Table 3: Full summary of averaged performance metrics of Scenario 1.

delta	Method	AUC	F1	Balanced Accuracy
0.5	Random Forest	0.591	0.096	0.519
0.5	SVM	0.570	0.008	0.502
0.5	kNN	0.572	0.044	0.509
1	Random Forest	0.773	0.348	0.623
1	SVM	0.649	0.195	0.558
1	kNN	0.736	0.279	0.590
1.5	Random Forest	0.889	0.553	0.728
1.5	SVM	0.791	0.530	0.699
1.5	kNN	0.861	0.577	0.735
2	Random Forest	0.953	0.746	0.839
2	SVM	0.899	0.726	0.817
2	kNN	0.935	0.739	0.829
Overall	Random Forest	0.801	0.436	0.677
Overall	SVM	0.727	0.365	0.644
Overall	kNN	0.776	0.410	0.666

6.3 Results Scenario 2

In Scenario 2 the main point of interest is how the methods behave as the outlier distribution becomes less extreme. As degrees of freedom ν (df) increase, the heavy tails weaken for the outlier distribution thus making the classification task harder. Table 4 reports overall mean performance per method and shows low performance for our metrics, most notably in F1, indicating that correctly identifying the outlier class is difficult in this scenario.

Table 4: Summary of averaged performance metrics of Scenario 2.

Method	AUC	F1	Balanced Accuracy
Random Forest	0.600	0.162	0.547
SVM	0.602	0.135	0.542
kNN	0.553	0.083	0.521

Figure 6 shows the distribution of AUC, BA and F1 across repetitions for each value of ν tested. The results indicate an increase of difficulty as ν increases and for $\nu \in [5, 8]$ there is a large decline in BA and F1. Across ν, kNN tends to performe the worst overall with a particularly low AUC where the lower tail of the distribution falls below 0.5 when difficulty increases.

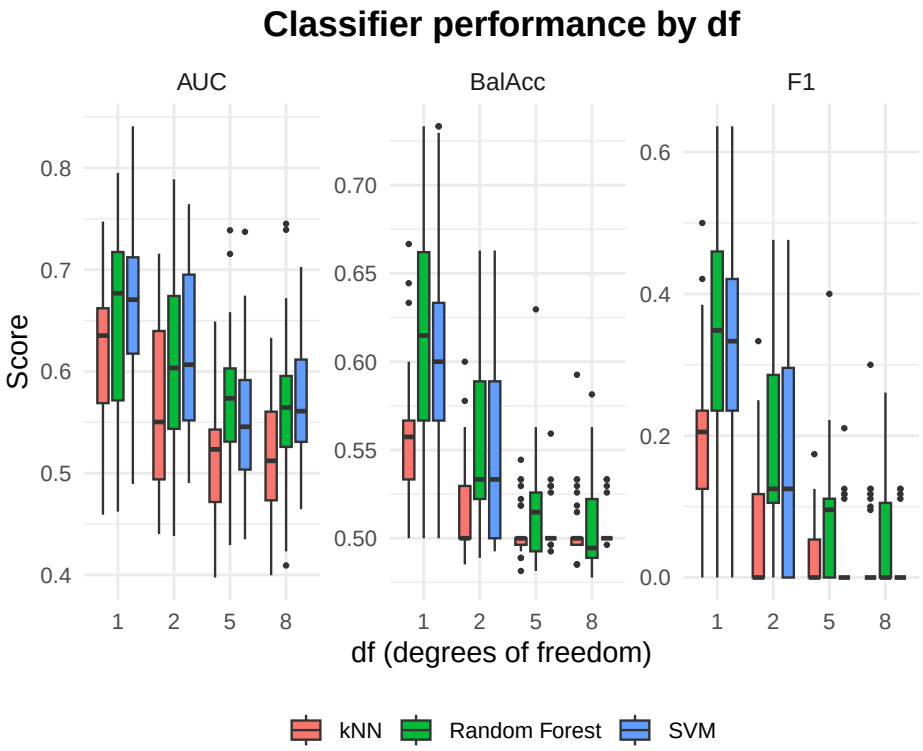

Figure 6: Performance metrics for Scenario 2.

Mean performance by ν is presented in Table 5 where a consistently low F1 is visible for $\nu > 1$.

Table 5: Full summary of averaged performance metrics of Scenario 2.

df	Method	AUC	F1	Balanced Accuracy
1	Random Forest	0.645	0.342	0.614
1	SVM	0.657	0.339	0.610
1	kNN	0.619	0.208	0.561
2	Random Forest	0.611	0.175	0.550
2	SVM	0.629	0.163	0.548
2	kNN	0.562	0.061	0.515
5	Random Forest	0.574	0.076	0.516
5	SVM	0.555	0.023	0.505
5	kNN	0.512	0.031	0.504
8	Random Forest	0.569	0.054	0.507
8	SVM	0.569	0.016	0.504
8	kNN	0.517	0.032	0.506
Overall	Random Forest	0.600	0.162	0.547
Overall	SVM	0.602	0.135	0.542
Overall	kNN	0.553	0.083	0.521

6.4 Results Scenario 3

Scenario 3 has the outlier class scattered uniformly over a square area controlled by the sides $a = b$ of the region. As the length of the sides (L) increases, the outliers spread over a wider region and classification of the outliers generally become an easier task. Table 6 reports overall mean performance per method averaged over all simulations and tested values of L. Random Forest has the highest overall averaged performance across the reported metrics.

Table 6: Summary of averaged performance metrics of Scenario 3.

Method	AUC	F1	Balanced Accuracy
Random Forest	0.867	0.631	0.778
SVM	0.831	0.611	0.762
kNN	0.833	0.514	0.701

Figure 7 show the distribution of AUC, BA, F1 across repetitions for each tested L. Overall there is a steady increase in median and a decrease in variance as L becomes larger, for all methods. For SVM and Random Forest, a small number of simulations at the smallest tested L yield AUC and BA below 0.5 which is visible in the lower tail of the boxplots.

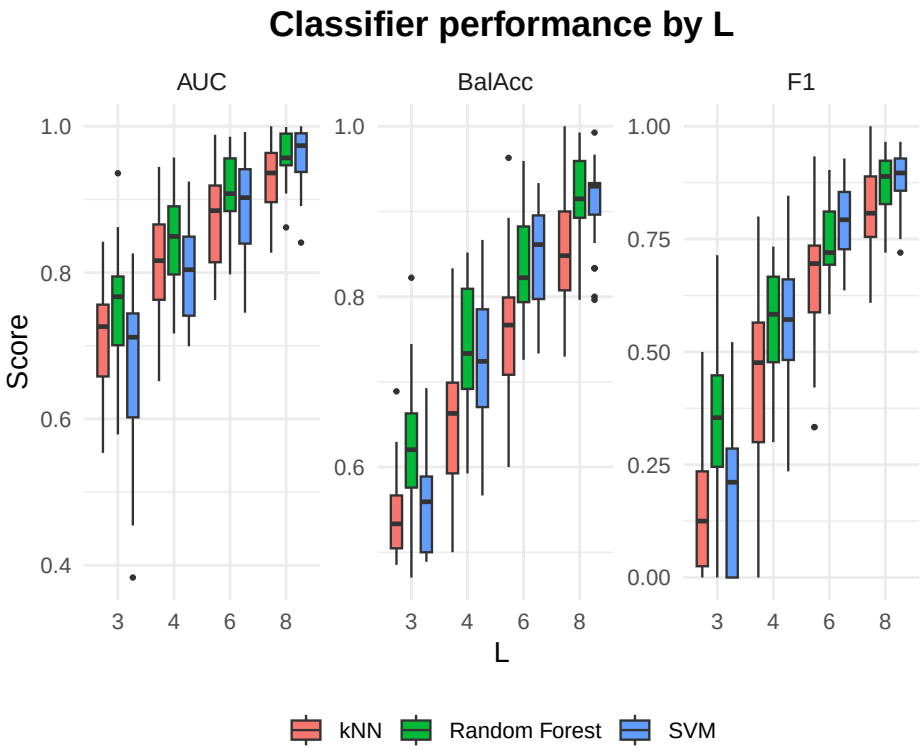

Figure 7: Performance metrics for Scenario 3.

The mean performance by L is shown in Table 7. In particular, an increase in F1 is visible when L increases from 3 to 4 for SVM and kNN.

Table 7: Full summary of averaged performance metrics of Scenario 3.

L	Method	AUC	F1	Balanced Accuracy
3	Random Forest	0.749	0.344	0.623
3	SVM	0.676	0.196	0.562
3	kNN	0.714	0.162	0.547
4	Random Forest	0.844	0.562	0.741
4	SVM	0.801	0.568	0.724
4	kNN	0.810	0.423	0.648
6	Random Forest	0.913	0.750	0.833
6	SVM	0.887	0.791	0.849
6	kNN	0.873	0.652	0.755
8	Random Forest	0.960	0.868	0.913
8	SVM	0.961	0.889	0.913
8	kNN	0.935	0.818	0.855
Overall	Random Forest	0.867	0.631	0.778
Overall	SVM	0.831	0.611	0.762
Overall	kNN	0.833	0.514	0.701

6.5 Results Scenario 4

In Scenario 4 the main point of interest is performance of classification when there is a soft circular boundary between inliers and outliers. The overlap between classes, which is controlled by τ, widens the region of the boundary and results in a more difficult task. Table 8 shows the overall mean of AUC, F1 and BA for each method. There is an overall low F1 and BA however RF is performing better than the other methods.

Table 8: Summary of averaged performance metrics of Scenario 4.

Method	AUC	F1	Balanced Accuracy
Random Forest	0.702	0.215	0.568
SVM	0.654	0.109	0.535
kNN	0.658	0.107	0.530

In Figure 8 the distribution of the metrics for the simulation show a declining median performance as τ increases with kNN yielding some simulations with AUC < 0.5, shown in the tail of the boxplot. A sharper decline is observed for kNN and SVM in the metrics BA and F1 when $\tau > 0.4$ while RF performs better.

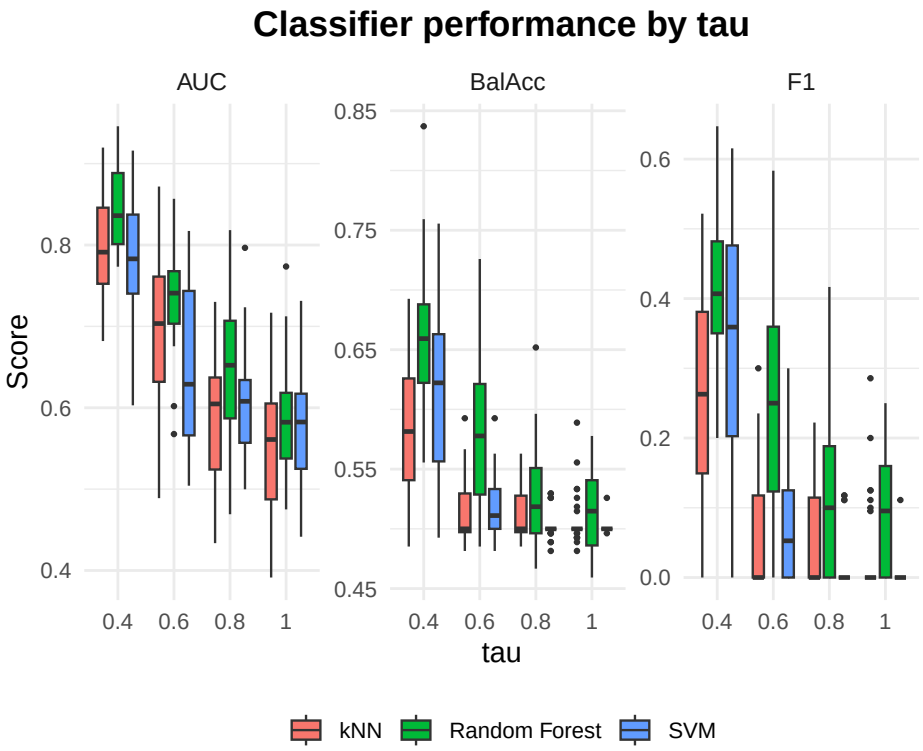

Figure 8: Performance metrics for Scenario 4.

The mean performance of the methods for each τ is presented in Table 9, which highlights how the metrics decline when overlap increases, particularly in F1 when $\tau > 0.4$.

Table 9: Full summary of averaged performance metrics of Scenario 4.

tau	Method	AUC	F1	Balanced Accuracy
0.4	Random Forest	0.843	0.411	0.658
0.4	SVM	0.783	0.338	0.618
0.4	kNN	0.798	0.268	0.585
0.6	Random Forest	0.732	0.244	0.574
0.6	SVM	0.647	0.079	0.520
0.6	kNN	0.695	0.067	0.515
0.8	Random Forest	0.644	0.120	0.526
0.8	SVM	0.601	0.015	0.502
0.8	kNN	0.587	0.057	0.512
1	Random Forest	0.589	0.087	0.514
1	SVM	0.583	0.004	0.501
1	kNN	0.550	0.035	0.507
Overall	Random Forest	0.702	0.215	0.568
Overall	SVM	0.654	0.109	0.535
Overall	kNN	0.658	0.107	0.530

7 Discussion

7.1 Summary and overall conclusions

In the simulations we compared the methods kNN, SVM and Random Forest, for outlier detection framed as binary classification, on four simulation scenarios. The metrics used were BA, F1 and AUC which show Random Forest to be slightly more stable and having higher typical performance across the scenarios. SVM is competitive in some scenarios but shows higher variability in metrics across repetitions in harder tasks. kNN is generally weakest and is most sensitive to class overlap when the local neighborhood structure becomes mixed. This is reflected in lower medians and wider tails in the more difficult settings. Overall Random Forest is the preferred method in this study due to its more robust performance across scenarios comparatively. While kNN is least reliable in the most difficult regimes and SVM slightly more competitive.

7.1.1 Scenario difficulty

The scenarios differ mainly in their approach to how they separate the inlier and outlier classes. Scenario 1 becomes easier as δ increases as this corresponds to larger separation between classes. Small δ implies strong overlap and a gradual performance drop as δ shrinks. Scenario 3 becomes easier as L increases due to the uniformly scattered outliers being spread over a wider region, which makes the separation from the inlier cluster larger and reduces

variability. In contrast, Scenario 2 becomes harder as ν increases due to the outlier distribution being less heavy-tailed and outliers less extreme. This reduces the ability to capture outliers and leads to low F1 and declining BA. The classes are not correctly identified. Scenario 4 is also challenging because as τ increases, the transition region widens and overlap increases along a nonlinear boundary. This results in low medians and higher variability in distribution of metrics across runs.

7.1.2 Scenario 1: Effect of overlap δ

The overlap changes separability which directly affects classification performance. Performance differs strongly across δ with a sharp drop in the most difficult parameter $\delta = 0.5$ and high performance in $\delta = 2$. RF performs best overall while SVM shows larger variability and lower medians most clearly seen in AUC. An AUC < 0.5 indicates worse than random ranking for some simulations for SVM in $\delta = 0.5$ and also a near zero F1 which indicates difficulty to reliably identify outliers correctly.

7.1.3 Scenario 2: Effect of degrees of freedom ν

When ν increases the heavy tails weaken and the outlier distribution becomes more similar to the inlier. kNN performs the weakest overall, however the notable decline in BA and F1 for $\nu \in [5, 8]$ indicates that the methods are performing close to chance level performance for these parameter values. In the most simple setting, when $\nu = 1$, the robustness of correctly identifying outliers is weak since the variability of F1 is large, even if BA shows better performance.

7.1.4 Scenario 3: Effect of noise spread L

A larger L spreads outliers farther around the inlier cluster which improves separability. The median increase and variability decrease in all metrics and methods show a good overall performance. Although at $L = 3$ a small number of runs fall below BA/AUC 0.5 which shows difficulty when the outliers are close to the inlier cluster. Random Forest has the strongest overall mean performance and there is a marked increase for $L \in [3, 4]$, easiest seen in F1 which shows more correctly classified outliers.

7.1.5 Scenario 4: Effect of boundary overlap τ

When τ increases, the region between inliers and outliers widens, which increases overlap and reduces class separability. The difficulty of the scenario is seen with the declining median performance as τ increases, particularly a pronounced decline in BA and F1 when $\tau > 0.4$, indicating near guess,

or worse, performance for all methods in high overlap. Random Forest performs comparatively better although variability remains high across repetitions. The distribution of AUC shows some kNN runs below 0.5 indicating occasional worse than random ranking and low separation.

7.2 Limitations

The simulations were done with 30 repetitions per parameter setting and restricted to two dimensions to limit the scope of the study. As a consequence, conclusions may not be generalized directly to all real world scenarios where feature structure and data generation may differ. Furthermore, the choice of parameter grids and tuning strategy has an effect on outcomes and must typically be tailored for each task. In this study the same tuning grids were used across scenarios which is not optimal for all settings. The variability observed across repetitions indicates that conclusions when deciding the strength of one method over another should be based on the distribution of performance measures rather than the result of single runs.

7.3 Suggestions for further simulations

Since many of the scenarios have the performance drop/increase noticeably more between some tested parameter values, it may be of interest to focus on the interval between them. For example $L \in [3, 4]$ in Scenario 3 and $\tau \in [0.4, 0.6]$ in Scenario 4 could benefit from being more granular in the parameter interval to gain more insight into when and how performance deteriorates significantly. The area of interest would be these boundary regions where performance between methods may diverge. Although this may require redesigning scenarios, for example Scenario 2, since degrees of freedom are integers in most applications.

In addition to a fixed scenario setup, several simulation parameters can be further varied in future experiments and other scenarios can be considered:

- **Dimension(d)** Number of features used. Initial scenarios use $d = 2$, but higher dimensions ($d = 5$, 10) can also be of interest.
- **Outlier proportion** In this study simulations use a fixed imbalance 10%, but this can be varied (3,5,15%) to study sensitivity to the proportion of outliers, although this comes with more challenges in choosing and interpreting metrics as outlined in Chapter 3.7.
- **Cluster separation (δ)** This is the distance between the normal and outlier class means when both classes correspond to multivariate normal distributions can be further looked at. Where δ quantifies the amount of overlap between the two classes.

- **Outlier distribution shape** Heavy-tailed distribution (t-distribution), uniform distribution, or normal distributions with different, or similar, covariance matrices may be of interest and can mimic real populations.
- **Covariance (Σ)** Instead of using identity matrices, correlated features could be introduced, not only for the outlier class, but also for the normal class. This may also further simulate real instances of problems.
- **Nonlinear boundary** Outliers that examine how the methods handle nonlinear boundaries could further be studied.

References

[1] Andrea Dal Pozzolo et al. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert Systems with Applications* 41.10 (2014), pp. 4915–4928.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Computing Survey* (2009).

[3] David M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[4] R. Gnanadesikan and J. R. Kettenring. "Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data". In: *Biometrics* 28.1 (1972), pp. 81–124. ISSN: 0006341X, 15410420.

[5] Markus M. Breunig et al. "LOF: identifying density-based local outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808.

[6] Cédric Poutré, Didier Chételat, and Manuel Morales. "Deep unsupervised anomaly detection in high-frequency markets". In: *The Journal of Finance and Data Science* 10 (2024), p. 100129. ISSN: 2405-9188.

[7] Paul Primus et al. *Anomalous Sound Detection as a Simple Binary Classification Problem with Careful Selection of Proxy Outlier Examples*. 2020.

[8] Colin Bellinger, Shiven Sharma, and Nathalie Japkowicz. "One-Class versus Binary Classification: Which and When?" In: *2012 11th International Conference on Machine Learning and Applications*. Vol. 2. 2012, pp. 102–106.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer, 2009.

[10] James Gareth et al. *An Introduction to Statistical Learning: With Applications in R*. Springer, 2013.

[11] Andreas Lindholm et al. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[12] George Forman and Martin Scholz. "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement". In: *SIGKDD Explor. Newsl.* 12.1 (Nov. 2010), pp. 49–57. ISSN: 1931-0145.

[13] Eve Richardson et al. "The receiver operating characteristic curve accurately assesses imbalanced datasets". In: *Patterns* 5.6 (2024).