Conformal Anomaly Detection

Mark Purtle

Submitted for the Degree of Master of Science in

MSc Data Science & Analytics

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

August 30, 2020

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 15226

Student Name: Mark Purtle

Date of Submission: 30/08/20

Signature: 

Abstract

Anomalous samples appear in all types of datasets. These may be important samples that require further attention, or unimportant samples that can be removed during data cleaning. Therefore, it is desirable to provide automated, machine learning methods to discover anomalies quickly. The objective of this project is to develop and assess the use of conformal prediction as a basis for anomaly detection algorithms. Anomaly detection using conformal prediction is a general method and so can be applied to all types of datasets. It also has the benefit of a well-calibrated false-alarm rate, so the user can intuitively alter the false alarm rate of the algorithm to their desired maximum upper bound. In this project, conformal anomaly detection is tested in the batch and online mode on the USPS handwritten digit dataset with a number of added anomalies. Conformal anomaly detection shows success at detecting these added anomalies, as well as anomalies that already existed in the dataset. It also demonstrates the well-calibrated false alarm rate property of conformal anomaly detection. The project extended into the problem of anomaly detection in time series datasets, with the methods of conformal martingales and conformal e-prediction with MUSUC. The two methods were tested on two synthetic anomaly datasets and a real-world energy consumption dataset and their results were compared. Both were successful at detecting the anomalies that existed in these datasets. The main parameter of an anomaly threshold was varied, and the two methods showed slightly different results. Generally, at lower thresholds, the delay time to detecting the anomaly was lower and the MUSUC algorithm showed slightly lower delay times compared to the conformal martingales. However, with lower thresholds also came a greater number of false alarms and at almost all thresholds, the MUSUC algorithm produced significantly more false alarms than conformal martingales. Therefore, from these experiments, conformal martingales appear to perform better overall as the false alarm rate is one of the biggest limiters for anomaly detection algorithms. However, to see if this claim is true, further experiments on a greater number of datasets and with alterations to the two algorithms, such as different non-conformity measures, need to be tested. An additional finding in this study was the use of a sliding window. The use of a sliding window greatly improved the results of both algorithms on the energy consumption dataset. However, it is unclear if this is a general improvement or specific to this dataset without carrying out experiments on more real-world datasets.

Contents

1 Introduction

Anomalous data samples are samples that do not fit the expected pattern of the rest of the samples within a dataset, and they exist in all kinds of data. Hawkins, 1980 [1] gives the definition of an anomaly as "An observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism". Anomalies can arise through many different mechanisms. They may be a result of human error during data entry or potentially a result of an error in the recording instrument, e.g. an electrical spark causing spike-like noise in the measurements recorded over time [2]. Alternatively, they may not be an error and instead may be a natural part of the dataset. For example, a badly written digit in a hand-written digit dataset or a ship changing course from regular shipping routes [3]. In some cases, anomalies can be important samples that need to be checked, i.e. anomalous bank transactions being indicative of fraud [4]. In others, they can be unimportant samples that, because they differ so much, are no longer useful, matching with the description of an anomaly in Anscombe and Guttman, 1960 [5]: "An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed.". These irrelevant observations can cause issues. For example, if there are anomalous samples in a training set, they could cause worse performance for a predictive algorithm [6]. Either way, it is important that we are able to detect these anomalies so that they may either be removed to improve performance or highlighted as an important sample.

Many different methods have been used for anomaly detection, some of which are general solutions and others which are specific to a domain [7]. These methods may also be supervised or unsupervised. Supervised anomaly detection is where samples in the training set have been labelled as anomalous and the algorithm learns a "rule" as to how to classify a new sample. However, high volumes of labelled anomaly data are rare and often expensive to collect, as they require expert knowledge to produce [3]. Additionally, it is unlikely that the labelled data will cover all possible anomalous behaviours and, therefore, it may be more beneficial to learn the "normal" behaviour from large volumes of historical data and detect deviations from this "normal" behaviour [3]. This is what unsupervised methods such as DBSCAN [8] aim to do. DBSCAN is a clustering method, it learns the "normal" behaviour of the dataset by creating clusters based on the density of points in the feature space [8]. Samples that lie within these densely populated clusters are seen as normal and samples that lie far from the main clusters, with few or no samples near them, are seen as anomalous [8]. Other examples of unsupervised methods include Kernel Density Estimation [9] and Isolation Forests [10]. Anomaly detection methods can also be semi-supervised [11], where the majority of the data is unlabelled, but a small amount of labelled data can be used to improve anomaly predictions.

The method this study will be focusing on is the use of Conformal Prediction for anomaly detection. Conformal prediction was first developed as a method for

assigning confidence values to machine learning predictions [12]. However, these confidence values can also be used in anomaly detection [13, Chapter 4]. This is an unsupervised method that is a general solution to anomaly detection (rather than domain-specific) and so is applicable to a variety of problems. One of the motivations behind the use of conformal prediction for anomaly detection is that, due to its property of guaranteed validity, it can provide a well-calibrated false-alarm rate [3]. This allows for intuitive tuning of the algorithm to the specific problem at hand, based on the false-alarm rate acceptable to the user. Additionally, it has very few parameters. Algorithms with a high number of parameters are more likely to overfit to the training set and make applying the algorithm to anomaly detection harder [3]. The only parameters of conformal prediction itself are the desired significance level and the non-conformity measure used. The non-conformity measure is typically the output of an underlying prediction algorithm, e.g. SVM, kNN or neural networks. These underlying algorithms may have parameters that need to be tuned themselves. For example, with kNN, the number of neighbours to use. Having many different underlying prediction algorithms that can be used with conformal prediction allows it to be flexible from problem to problem. Let us say we wanted confidence in our predictions for the CIFAR100 dataset, which is an image recognition problem. SVM or kNN is unlikely to work well on this particular dataset, so instead the non-conformity measure could be switched out for a convolutional neural network for better performance.

When designing an anomaly detection algorithm there are a number of ideal characteristics to consider for assessing its performance. The most obvious of these is that it minimizes the number of false negatives, i.e. it detects the majority of the anomalies in the dataset, this is the sensitivity of the algorithm. Additionally, it should minimize the number of false positives, where a normal data sample is classified as anomalous, this is the false-alarm rate. If the algorithm produces a high number of false alarms, it becomes significantly less useful as alarms may start to be ignored, causing the user to potentially miss actual anomalies. The ideal anomaly detection algorithm should aim to have low numbers of both false positives and false negatives. However, in reality this is often a trade-off for the algorithm. As you increase the sensitivity of an algorithm, the likelihood of a false positive will increase. Generally, algorithms can adjust this trade-off by tuning the parameters and will do so depending on what is more important for the problem at hand; detecting as many anomalies as possible or limiting the number of false alarms.

With time series data there is the additional goal that the anomaly detector should detect the anomalies as soon as possible. Although this may depend on the problem, in the majority of time series anomaly detection scenarios it is important to know that an anomaly is happening right away. For example, if our anomaly detection algorithm was attempting to detect anomalous seismic wave patterns that may indicate a potential earthquake happening in the near future, an algorithm that detects that these wave patterns are anomalous after receiving 10 days of more data and after the earthquake has already happened, would be of no use. A characteristic linked with this is that the anomaly detection algorithm should be computationally efficient. If the algorithm takes a significant amount of time to process each sample, it will delay the detection of anomalies.

The aim of this report is to assess the use of conformal prediction for anomaly detection in both the batch and online setting, as well as for time series data, and compare the different anomaly detection methods which are based on conformal prediction. I will start in Section 2 with background research on the conformal anomaly detection methods, covering the theory behind conformal prediction, how conformal prediction is applied to anomaly detection, martingales for online anomaly detection and, finally, conformal e-prediction for change point detection. Section 3 outlines the datasets used in this study. In Section 4 I will cover the results of my experiments to show how the different conformal anomaly detection methods perform on various datasets. Finally, in Section 5, I will discuss the effectiveness of the algorithms at detecting anomalies as well as compare the degree of success achieved by using conformal martingales and using conformal e-prediction for time series anomaly detection.

2 Background Research

2.1 Conformal Prediction

The basis for all the anomaly detection algorithms I implement in this paper is the method of conformal prediction. Therefore, I will start by introducing the theory behind it, how it can be implemented and what its benefits are. Conformal prediction was first proposed by Gammerman and Vovk, 2007 [12] for producing confidence values for machine learning predictions, along with the prediction itself. It is an unusual method compared to classical machine learning methods. Typical machine learning methods often give a point prediction of a class, in the case of classification, or a floating-point number, in the case of regression. Therefore, the accuracy of the results can simply be seen as the percentage of predictions that are correct. With conformal prediction, as will be shown next, the accuracy of the results is directly connected to the required parameter of a confidence level. This makes it difficult to compare with the classical machine learning algorithms, as we cannot simply say one has a lower accuracy than the other.

Conformal prediction provides confidence values to machine learning predictions using what are called non-conformity measures [12][14, Chapter 2]. A non-conformity measure is a function that assigns a score to a data sample based on how well it "conforms" to the other data samples. Given a non-conformity measure A and a sequence of values $z_1, z_2, \ldots, z_n$, the output will be a sequence of non-conformity scores $\alpha_1, \alpha_2, \ldots, \alpha_n$. A non-conformity measure should have the property of equivariance, i.e. the non-conformity score for a specific sample should have the same value no matter what its position in the sequence is [12]. Non-conformity measures are usually based on other machine learning algorithms. For example, it could be based on kNN, where the non-conformity score is the distance to the nearest neighbor of the same class. If the distance to the nearest sample of the same class is high, then the sample does not conform to the dataset and will have a high non-conformity score. By calculating the non-conformity score for all the samples in the data set, including the new sample, and ranking them, we get an idea of how well the new sample fits within the data set. If the new sample has a high non-conformity score, it fits the data set poorly and will have a low rank within the data set, and vice versa. This notion is defined formally in the p-value [12] (1).

$$p := \frac{|\{i=1,\ldots,n+1 : \alpha_i \geq \alpha_{n+1}\}|}{n+1} \quad (1)$$

Where α_i is the non-conformity score of a sample and n is the size of the initial data set, n+1 is, therefore, the size of the data set with the new sample.

A high p-value indicates that the sample conforms to the data set. The p-value is a ratio and will always lie between 0 and 1, as the rank of a non-conformity score cannot be greater than the number of samples in the data set. If the p-value was 1, it

would show that the sample was the least "strange" in the dataset and if the p-value was close to 0, it would show it was the most "strange".

An alteration on the p-value is the smoothed p-value, given in (2). The difference is that the smoothed p-value takes more care in breaking the ties between non-conformity scores [12]. With regular p-values if multiple samples have the same non-conformity score, they will be given the same rank. If there are many ties this can cause large changes in the rank based on small changes in the non-conformity score. As an example, say we have a set of non-conformity scores [1, 2, 3, 3, 3, 3, 5, 6]. The rank of all samples with a score of 3 will be 6 and they will have a p-value of 0.75. If one of these scores changes from 3 to 3.1, the rank of that score now changes from 6 to 3 and its p-value from 0.75 to 0.375. It is preferable that small changes in the non-conformity score do not cause this large change in p-value when there are ties. Smoothed p-values solve this by separating the rank of a sample into two parts: the number of samples that have greater non-conformity scores than it and the number of ties multiplied by τ (2). The number τ is generated randomly from a uniform distribution between [0,1]. It will be a different value for every p-value calculated. So, if we take our previous example, the rank of the sample with non-conformity score of 3 will be a value between 2 and 6, depending on the randomly generated value of τ. While it is still possible to have a rank of 6, as with regular p-values, it is much more likely that the rank will be somewhere in between 2 and 6 and, therefore, there will be less of a discrepancy in p-values when ties exist. It provides stronger validity than regular p-values [12].

$$p := \frac{|\{i=1,\ldots,n+1:\alpha_i>\alpha_{n+1}\}|+\tau|\{i=1,\ldots,n+1:\alpha_i=\alpha_{n+1}\}|}{n+1} \quad (2)$$

The method I have discussed so far assumes that all the data is labelled. However, when we are given new data, we will not know the samples' labels and instead of just seeing how well a sample conforms to the dataset, we want to predict its label. Without knowing its label, we can't compute a non-conformity score and, therefore, its p-value. So, p-values are calculated for every potential label for a new sample. For example, if the possible labels are 0, 1, 2 we calculate three p-values. Since a p-value gives a measure of how well a sample conforms, by looking at the set of p-values we can see how likely it is that each postulated label is the correct one. Therefore, if we want to make a point prediction, we can predict the label of a new sample based on which postulated label has the highest value. The credibility of this point prediction is defined by the p-value of the prediction and the confidence is defined by 1 minus the second largest p-value [12].

Another possible output mode of conformal prediction is what are known as prediction sets. Prediction sets are predictions that can contain multiple labels and will change based on the chosen significance level. The significance level can be seen as 1 minus the confidence level, so if you wanted to be 95% confident in your predictions the significance level you would set would be 5%. As previously described, in the case of classification, we loop through the potential labels for a new

sample and calculate a p-value for each label. If the p-value is greater than the significance level, then that label is added to the prediction set. For example, if a sample had p-values of 0.10 for labels 0 and 1, at a significance level of 5%, both labels would be predicted. The idea behind this is that using these significance levels, you can control the rate of error of your predictions. At a significance level of ε, the error rate should not exceed ε. This is the property of validity and it is automatically satisfied by conformal predictors. When predicting a label for a sample, the probability that its true label will not be in the prediction set should not exceed ε [12][14, Chapter 2]. For lower significance levels, you are more likely to get more multiple predictions (a prediction set with more than one label) and, therefore, the accuracy of your predictions will be higher. For higher significance levels, empty prediction sets, where none of the p-values are greater than the significance level, are more likely and the accuracy will be lower.

2.1.1 Transductive Conformal Prediction

The general algorithm for transductive conformal prediction is shown in Algorithm 1.

Algorithm 1 Transductive Conformal Prediction

INPUT: Training Set = $((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$
INPUT: Test Set = $((x_1, y_1), (x_2, y_2), ..., (x_l, y_l))$
INPUT: Non-conformity measure: A
INPUT: Significance Level: ϵ
for Each sample in Test Set **do**
  for Every possible label **do**
    Extend training set with new sample and postulated label: $z_1, ..., z_{n+1}$

    Calculate non-conformity scores for every sample in extended training set: $A(z_1, ..., z_{n+1})$

    Calculate the p-value of the test sample with this label based on the rank of its non-conformity score within the extended training set
  end for
  Prediction:= Label with max p-value
  Prediction Set:= Labels with p-value $> \epsilon$
  Credibility:= Value of max p-value
  Confidence:= 1 - second largest p-value
end for

This is the batch setting for conformal prediction. In the online setting the algorithm is slightly different as there is no fixed training or test set. Instead the new samples are input to the algorithm one by one. A prediction is made for that sample, its label is revealed and then it is added to a training set that will be used to predict the label for future samples. The training set in this case is constantly growing and predictions should improve as more samples are processed. The online setting also has a stronger case of validity [14, Chapter 2].

The transductive conformal predictor described is both valid and generally predictively efficient [12]. However, with large datasets and when data preprocessing is required, it suffers from poor computational efficiency [14, Chapter 2]. This is due to having to recalculate the non-conformity scores for the whole extended training set for every sample and for every postulated label for that sample. For example, take an image recognition dataset made up of 128x128 images with 100 possible labels and 10000 samples. Say you chose a training set of size 7000. For a single test sample, you would need to calculate 7001 non-conformity scores for each label and, since there are 100 labels, 700100 non-conformity scores in total. This would be extremely computationally inefficient. The issue of computational efficiency can be improved upon using Inductive Conformal Prediction.

2.1.2 Inductive Conformal Prediction

Inductive conformal prediction splits the training set into a training set proper and a calibration set [12][14, Chapter 4], so there are three sets in total. In regular conformal prediction the non-conformity scores of every sample in the extended training set are calculated, and they are calculated relative to all the other samples in the extended training set. With inductive conformal prediction, all non-conformity scores are calculated relative to the samples in the training set proper but not including the test sample itself. So, the non-conformity score for a sample would be the distance to the nearest neighbour of the same class in the training set proper. Non-conformity scores are also calculated in the same way for samples in the calibration set. No non-conformity scores are calculated for the training set proper.

The benefit of this is the large reduction in the number of non-conformity scores that need to be calculated [12]. Previously, all the non-conformity scores for the training set samples needed to be recalculated for every test sample, as the samples for calculating the scores from change each time with the addition of a new sample. This can change these scores slightly. However, with ICP, since the non-conformity scores are calculated relative to the training set proper, the samples we are calculating the scores from never change. That means the non-conformity scores for the calibration set never need to be recalculated.

If we take our previous example and split the training set into a training set proper of 5000 and calibration set of 2000, there is an initial calculation of 2000 non-conformity scores for the calibration set. However, now for each test sample only a single non-conformity score for each label needs to be calculated, so 100 in total for each sample. The scores for the calibration set can be constantly reused as the changing test sample will have no effect on them, unlike in regular conformal prediction.

The p-values are then calculated from the ranking of the scores for the calibration set and the test sample. This is summed up in Algorithm 2.

Algorithm 2 Inductive Conformal Prediction

INPUT: Training Set = $((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$
INPUT: Test Set = $((x_1, y_1), (x_2, y_2), ..., (x_l, y_l))$
INPUT: Non-conformity measure: A
INPUT: Significance Level: ϵ

Split Training Set into Training Set Proper and Calibration Set

Calculate non-conformity scores of Calibration Set samples based on Training Set Proper samples

for Each sample in Test Set **do**
  for Every possible label **do**
    Calculate non-conformity score for test sample with its postulated label based on Training Set Proper samples

    Calculate the p-value of the test sample with this label based on the rank of its non-conformity score within the Calibration Set scores
  end for
  Prediction:= Label with max p-value
  Prediction Set:= Labels with p-value $> \epsilon$
  Credibility:= Value of max p-value
  Confidence:= 1 - second largest p-value
end for

This increase in computational efficiency does have a cost in terms of predictive efficiency, however [12]. Since we have to split the training set into training set proper and calibration set, the number of samples we are training on to calculate the non-conformity scores will be less than in regular conformal prediction. Less samples to train on generally will result in less accurate predictions.

The online learning setting for ICP also comes with its own challenges as there is not a single training set to add new examples to after making predictions. Adding to the training set proper is a possibility but defeats the point of ICP as the calibration set non-conformity scores would have to be recalculated every time a new sample is added to the training set proper, which would cause the computational efficiency to suffer. Instead, new test samples could be added to the calibration set after prediction, which should still result in better predictions as the calibration set grows [13, Chapter 4]. An alternative to this could be that once the calibration set reaches a certain size, it becomes part of the training set proper and the calibration set starts again from the new test sample [13, Chapter 4]. This is equivalent to starting again from the beginning with a new training set proper and calibration set. It is unlikely to affect computational efficiency much as the recalculation of calibration set scores will be infrequent. It may also have the additional benefit of dealing with concept drift that can occur in datasets over time where the distribution of the new input data is changing. Since the algorithm is being retrained on new data at certain intervals, it should hopefully change the model when the distribution changes, resulting in more accurate predictions. Updating the training set proper in this way

comes with its own issue, however. For a period of time after the update trial, the number of samples in the calibration set will be low, potentially resulting in inaccurate predictions [13, Chapter 4]. This can be solved by retaining a number of samples in the calibration set at each update trial. So, instead of transferring over the whole calibration set ($z_1, \ldots, z_n$), only a fraction of the calibration set ($z_1, \ldots, z_{n-m}$, where m is the number of samples kept in the calibration set) is added to the training set proper. Theoretically, this method of online inductive conformal prediction loses the property of validity. This is because update trials are no longer independent as some samples are being used for calibration across multiple update trials [13, Chapter 4]. However, in reality it seems to have little effect on the validity [13, Chapter 4], as will be shown later in this study.

2.1.3 Non-Conformity Measures

There are many possible non-conformity measures that can be used with conformal prediction. Most machine learning algorithms can be adapted to be part of it, as long as for each sample they are able to output a "score" that is a metric of how well the sample conforms to the rest of the data and the algorithm has the property of equivariance [12]. In this study I will only focus on two of the most commonly used [12][13][14] non-conformity measures, kNN and SVM. However, non-conformity measures such as gradient boosting [14, Chapter 4][15] and neural networks [14, Chapter][16] have also had success.

2.1.3.1 kNN

The nearest neighbour NCM works upon the idea that samples of the same class should be clustered near each other and away from other classes in the feature space. A sample that conforms to the dataset should be close to samples of the same class and far from samples of other classes. It is defined in (3).

$$\alpha_i := \frac{distance\ to\ nearest\ sample\ of\ the\ same\ class}{distance\ to\ nearest\ sample\ of\ a\ different\ class} \qquad (3)$$

If a sample is far away from samples of its class and near to samples of a different class then α will be large, showing that the sample does not conform well to the data set. This can be expanded for kNN, by calculating the total distance to a sample's k nearest neighbors of the same class and k nearest neighbours of a different class [12][13, Chapter 4] (4).

$$\alpha_i := \frac{\sum_{j=1}^{k} d_{ij}^{+}}{\sum_{j=1}^{k} d_{ij}^{-}} \quad i = 1, \ldots, n \qquad (4)$$

Where d^+ is distance to a sample of the same class and d^- is distance to a sample of a different class.

A variation on this non-conformity measure that has been suggested for anomaly detection is to only use the distance to samples of the same class [13, Chapter 4] (5).

$$\alpha_i = \sum_{j=1}^{k} d_{ij}^{+} \quad i = 1, \ldots, n \tag{5}$$

It is thought to be more sensitive to outliers that are far from all classes [13]. If a sample is equidistant between two class distributions, it will have a non-conformity score of 1 with (4) no matter what class it is and no matter the actual distance from the distributions. Figure 1 illustrates this problem. The solid black sample to the far right is clearly an anomalous point as it lies far from both distributions. However, if (4) is used, it will have a similar non-conformity score to the solid black sample to the left that lies in between the two class distributions. From looking at the data it is clear that these two samples do not conform to the data equally and that (5) would perform better in this case.

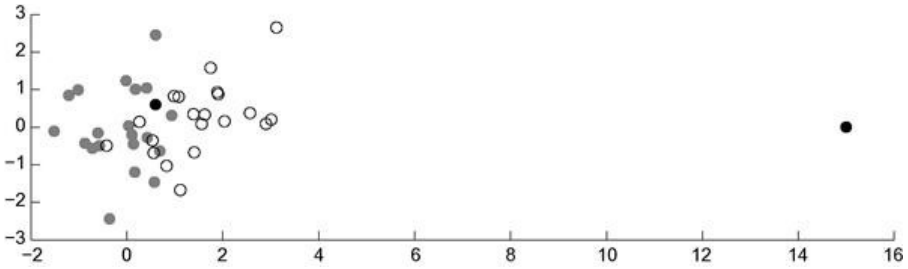

Figure 1: *An illustration of the type of data where Equation (4) would give similar non-conformity scores to two very different points. Figure taken from [13, Chapter 4].*

2.1.3.2 SVM

A support vector machine is a prediction algorithm that aims to separate classes with maximum margin hyperplanes [17]. These hyperplanes are not always linear as the SVM can use different kernels such as polynomial and radial. SVMs are typically used to make predictions for new samples. However, as well as a prediction, the SVM algorithm also outputs Lagrange multipliers. In the binary case, the Lagrange multipliers correspond roughly to the distance to the separating hyperplane and which side of the hyperplane it lies. So, if the multiplier is a high value it indicates the sample is far from the hyperplane and conforms to its class. Therefore, to make it a non-conformity measure, you multiply the multiplier by -1. In the binary case, where the labels are {-1,1}, the Lagrange multipliers will be positive or negative depending on which side of the hyperplane it lies, i.e. if it lies on the side of Class 1, the multiplier will be positive. Therefore, the Lagrange multiplier must also be multiplied by the sign of its label. The non-conformity measure is given in (6):

$$\alpha_i \coloneqq -sign(y_i)\beta_i \tag{6}$$

Where β is the Lagrange multiplier.

In the multi-class case, it is less simple as there is not a single hyperplane. However, the Lagrange multipliers output by the algorithm still give an idea of the distance to

a hyperplane in the one-vs-rest case for each possible class [14, Chapter 2]. In this case, there is no need to multiply by the class label as a Lagrange multiplier will only be negative if it lies on the wrong side of the one-vs-rest hyperplane, which only tells us it does not conform, not that it belongs to Class -1. This non-conformity measure is given in (7).

$$\alpha_i := -\beta_i \tag{7}$$

2.2 Conformal Anomaly Detection

Next, I will cover how conformal prediction can be applied to the problem of anomaly detection. In conformal prediction, a p-value is testing whether a sample conforms to the randomness assumption [12]. The randomness assumption is that all data samples are sampled independently from the same distribution [12]. When a p-value is lower than a set significance level, ε, then this assumption is violated, and the sample is thought to be generated from a different probability distribution. This fits with Hawkins [1] definition of an anomaly. Therefore, the p-values output by a conformal predictor can be used as a method of detecting anomalies [13, Chapter 4].

As previously described in the conformal prediction section, for each test sample a conformal predictor would calculate p-values for all possible labels and, if a p-value was smaller than the significance level, the label would be excluded from the prediction set. Now, if we consider the special case where at a low significance level (e.g. 1%) none of the p-values are greater than the significance level and the output prediction set is empty, this suggests for all possible labels of the sample that none of them make the sample conform to the randomness assumption. The sample was not sampled independently from the same probability distribution as the rest of the dataset and, therefore, we can classify it as being an anomaly. It may belong to another totally different class that was not included in the training set, e.g. a letter when all the possible labels are numbers, or it may be a rare event of one of the classes. There is also the case where the prediction set does not include the test sample's true label. This could happen where one digit appears similar to another, e.g. a badly written 7 looks like a 1. In this case the example can also be considered an anomaly and not generated from the same probability distribution. Since, in both cases, the p-value for the true label will be lower than the significance level, there is only ever a need to calculate the p-value for the true label rather than every possible label and classify any samples with empty prediction sets as anomalies [13, Chapter 4]. This would speed up computation time slightly as the number of non-conformity scores that need to be calculated would decrease. Only calculating the p-value for the true label assumes that the new test samples are all labelled. In the situation that new data is unlabelled, p-values will have to be calculated for every possible label and only empty prediction sets will be classified as anomalies, since it is not possible to check if the true label is not part of the prediction set. The general algorithm for conformal anomaly detection is shown in Algorithm 3.

Algorithm 3 Conformal Anomaly Detection

INPUT: Training Set $= ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$
INPUT: Test Set $= ((x_1, y_1), (x_2, y_2), ..., (x_l, y_l))$
INPUT: Non-conformity measure: A
INPUT: Significance Level: ϵ
for Each sample in Test Set **do**
    Extend training set with new sample: $z_1, ..., z_{n+1}$

    Calculate non-conformity scores for every sample in extended training set: $A(z_1, ..., z_{n+1})$

    Calculate the p-value (Eq. 1) of the test sample based on the rank of its non-conformity score within the extended training set

    if p-value $< \epsilon$ **then**
        Anomaly = True
    else
        Anomaly = False
    end if
end for

As well as being able to detect if a labelled sample is anomalous, conformal prediction provides a way to calibrate the false alarm rate [13, Chapter 4]. Due to the guaranteed property of validity of conformal prediction, the error rate of a classifier should not be greater than the significance level. This can be extended to anomaly detection where the probability of a false alarm, where the algorithm predicts an outlier where it is not, is no greater than the significance level [13, Chapter 4]. Since an anomaly will only be predicted when the prediction set is wrong, i.e. is empty or does not contain the true label, the percentage of anomalies predicted in the whole test set will not be greater than the significance level, even if every anomaly predicted was a false alarm. So, the false alarm rate's upper bound is the significance level [13, Chapter 4]. This is the concept of a well-calibrated false alarm rate. By changing the value of the significance level, the user can intuitively tune the anomaly detection algorithm to whether they want it to be more sensitive to anomalies (higher ε) or have a lower false alarm rate and higher precision (lower ε). It is up to the user to decide which is more important to them; detecting more anomalies or making fewer mistakes when detecting them.

Conformal anomaly detection can be applied easily in either the batch or online mode, as described for conformal prediction. There is a slight difference in the online mode, however, as a sample that is classified as anomalous would not be added to the training set. An issue with CAD, just like regular conformal prediction, is that it is computationally inefficient and, therefore, with large datasets becomes an undesirable method to use. A solution to this would be to use inductive conformal anomaly detection, which is the same as CAD except the non-conformity scores are calculated from the training set proper and ranked within a calibration set, as with

inductive conformal prediction. The possible methods for implementing inductive conformal anomaly detection in the online mode are the same as those discussed for inductive conformal prediction.

2.3 Conformal Martingales

A development in online anomaly detection has been to use conformal prediction to calculate martingales. Conformal martingales were developed as a method for testing the exchangeability assumption online [14, Chapter 7][18][19]. The exchangeability assumption is almost identical to the randomness assumption. It states that samples z_1, z_2, … are sampled from exchangeable probability distributions, Q in Z^∞ [19]. So, any permutations are distributed in the same way as the original samples. The motivation behind it is that the majority of machine learning results depend on the i.i.d. (independent and identically distributed) assumption [18]. Therefore, it is important to ensure that, during online training, the data a machine learning model is being trained on is remaining under the i.i.d. assumption. If the data is i.i.d., its distribution is exchangeable and, therefore, by testing the exchangeability, we are testing the i.i.d. assumption [18]. An example of the importance of this could be in running an online model to detect spam emails. Previously the model accurately detected spam emails, however, over time the content of spam emails may have changed. Now the model is predicting using old historical data with spam emails sampled from a different distribution, resulting in poor predictions. The model should begin to retrain on new data and conformal martingales aim to give insight on when the distribution has changed and, therefore, when to begin retraining your model.

The idea of testing the exchangeability online can easily be applied to online anomaly detection [20]. As previously discussed, an anomaly can be seen as a sample that is drawn from a different distribution to the training data. If it is sampled from a different distribution, then it violates the i.i.d. assumption and this change will be detected by the martingale values [20]. Martingales can be calculated based on p-values and the general formula for them is given in (8).

$$M_n := \prod_{i=1}^{n} f_i(p_i) \tag{8}$$

Where f_i is a betting function and p_i is a p-value.

A betting function is applied to the p-value. This betting function should output higher values when the p-value is low and vice-versa. Therefore, when the p-values are low the value of the martingale will increase and when they are high the martingale should decrease. Additionally, the betting function must satisfy Equation 9.

$$\int_0^1 f_i(p)dp = 1, \quad i = 1, 2, \ldots \tag{9}$$

A sequence of low p-values intuitively indicates that the samples are no longer from the same distribution as the training data and the martingale value will grow large to indicate this. Once it has reached a set user threshold, an alert will be raised that

an anomaly has occurred. Algorithm 4 covers the general process for anomaly detection using conformal martingales.

Algorithm 4 Online Conformal Martingale Anomaly Detection

```
INPUT: Z = ((x_1, y_1), (x_2, y_2), ..., (x_N, y_N))
INPUT: Non-conformity measure: A
INPUT: Betting function: f
INPUT: Threshold: C
M_0 = 1
Initialize Empty Training Set
for n=1, ..., N do
    Create extended training set with new sample: z_1, ..., z_n

    Calculate non-conformity scores for every sample in extended training
    set: A(z_1, ..., z_n)

    Calculate the p-value (Eq. 1) of the new sample based on the
    rank of its non-conformity score within the extended training set

    Bet=f(p-value)

    M_n = M_{n-1} * Bet

    if M_n > C then
        Anomaly = True
    else
        Anomaly = False
    end if

    Permanently add new sample to the training set
end for
```

2.3.1 Power Martingales

The betting function power martingales use is shown in (10), with the initial martingale $M_0^{(\varepsilon)} = 1$.

$$M_n^{(\varepsilon)} = \prod_{i=1}^{n} \varepsilon p_i^{\varepsilon-1} \tag{10}$$

The parameter ε is not the significance level that it previously represented in conformal prediction. In this scenario it is a constant with a value between [0, 1]. Typically, a value nearer to 1 is chosen, Ho et al. 2005 [21] uses a value of 0.92. Lower values of ε will cause a greater variation in the output of the betting function and very small values when the p-value is high. The smallest possible output of the betting function is equal to ε. Small output for high p-values is undesirable as, if there is a sequence of high p-values, the martingale value will become extremely small and not able to recover to higher values when the i.i.d. assumption is violated. However, if ε is too large, the betting function will not be sensitive enough to violations of the i.i.d. assumption. So, a balance needs to be found. Figure 2 shows how the martingale value changes with ε for different p-values.

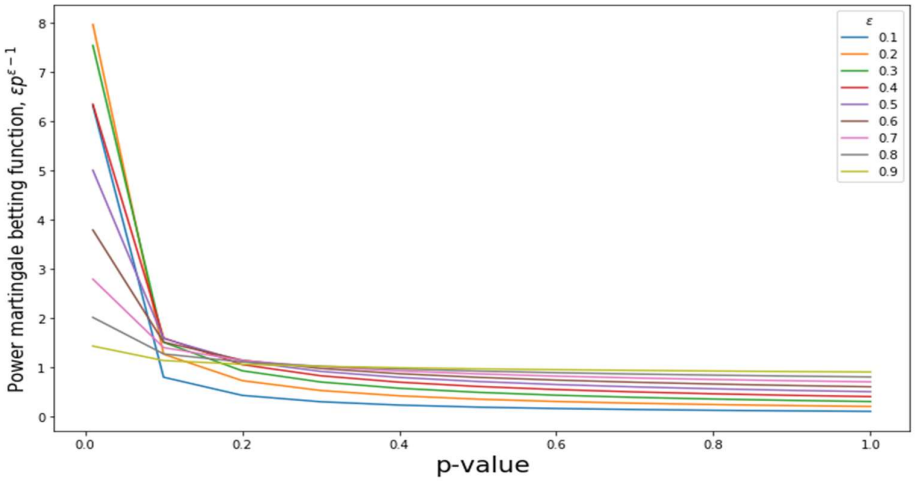

Figure 2: *Plot of how the output of the betting function ($\varepsilon p^{\varepsilon-1}$) varies for changing p-values for different values of ε.*

A variation of the power martingale is the Simple Mixture power martingale (11). It is a mixture of the power martingales for different values of ε and, therefore eliminates the dependence on ε [19].

$$M_n = \int_0^1 M_n^{(\varepsilon)} d\varepsilon \tag{11}$$

An issue that has been found with martingales is that over long sequences of data which do conform to the i.i.d. assumption, the martingale value can become extremely small [20]. The problem this causes is that when an anomaly appears, it will take a greater amount of time for the martingale to reach a high enough value to trigger an alarm, increasing the delay time for anomaly discovery. It is also possible that the martingale value will get so low that it never recovers, and the anomaly remains undetected. A proposed solution to this from Volkhonskiy et al., 2017 [20] is a modification to the calculation of the martingale (12). This modification adds the $\log_{10}$ of the betting function to the martingale, as well as making sure the martingale never falls below zero. This allows for lower mean delay time to detection [20]. They note that although theoretical validity is lost with this modification, they confirmed the validity of the method empirically [20].

$$C_n = \max\left(0, C_{n-1} + \log\left(g_n(p_n)\right)\right) \tag{12}$$

Another solution is to use a sliding window for calculating the p-values and the martingales. If there are less p-values to calculate the martingale from, the less possible it is for the martingale to decrease to very small values. An additional advantage to this is that it speeds up computation time.

2.4 E-values for Change Point Detection

A recent development in using conformal prediction for change point detection has been the use of E-values [22]. E-values are the output of conformal e-prediction and

differ slightly from the p-values of conformal prediction. Vovk, 2019 defines a conformal e-predictor as a function that maps a sequence of samples ($z_1, \ldots, z_n$) to a sequence of non-negative numbers ($\alpha_1, \ldots, \alpha_n$) of the same length and with a mean of 1 [22]. The function must also satisfy the property of equivariance. This definition matches up closely with the definition of a non-conformity measure, with the added constraints that the sequence of numbers must have a mean of 1 and all numbers must be greater than zero. Therefore, by slightly tweaking a non-conformity measure, we can produce a conformal e-predictor. For example, the kNN non-conformity measure outputs a sequence of non-negative numbers that satisfy the property of equivariance. By altering the mean of the sequence, we can turn a sequence of non-conformity scores into a sequence of E-values.

Once we have a sequence of E-values, we find the product of this sequence. This is similar to martingales where we find the product of the output of the betting function for a sequence of p-values. When the i.i.d. assumption is violated, the non-conformity score should be high and, therefore, the product of a sequence will be a large value. As with the martingale algorithm, the high values can be detected as anomalies by setting a threshold. This is known as the MUSUC procedure and is shown in Algorithm 5.

Algorithm 5 MUSUC with E-values Anomaly Detection

INPUT: Z = $((x_1, y_1), (x_2, y_2), ..., (x_N, y_N))$
INPUT: Non-conformity measure: A
INPUT: Threshold: C

Initialize Empty Training Set

for n=1, ..., N **do**
  Create extended training set with new sample: $z_1, ..., z_n$

  Calculate non-conformity scores for every sample in extended training set: $A(z_1, ..., z_n)$

  Normalize the sequence to have a mean of 1 and values all > 0 to give a sequence a E-values

  E_n = E-value of the new sample

  if $E_1 \times ... \times E_{n-1} \times E_n > C$ **then**
    Anomaly = True
  else
    Anomaly = False
  end if

  Permanently add new sample to the training set
end for

3 Datasets

I will now briefly give background information on the datasets used in this study.

3.1 USPS Handwritten Digits

The US Postal Service dataset is a series of handwritten digits taken from zip codes on mail sent through the USPS. The digits were automatically scanned from envelopes by the USPS and then deslanted and size normalized [23]. Each sample is made up of a 16x16 grayscale image of the digit, examples of the data are shown in Figure 3. There are 9298 samples in total with the data already split into training and test sets of size 7291 and 2007 respectively. The distribution of the training and the test set of this dataset is known to be different, which makes it ideal for testing how the martingale algorithm detects violations of the i.i.d. assumption [19]. No pre-processing was applied to this dataset before any of the experiments.

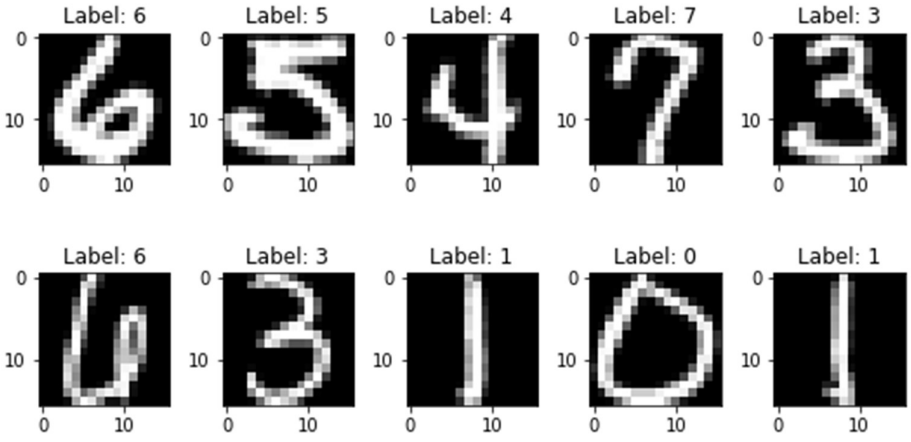

Figure 3: *Examples of the data samples in the USPS handwritten digit dataset*

3.2 Numenta Anomaly Benchmark (NAB)

Numenta, a machine intelligence company, has compiled a number of time series datasets for testing anomaly detection algorithms [24]. They provide both synthetic and real-world datasets for this purpose. Additionally, with the synthetic anomaly datasets, they provide the same datasets without the added anomaly, allowing for testing for the false alarm rate of the algorithm on data with no anomalies. In this study, two synthetic anomaly datasets have been chosen to test the anomaly detection algorithms on, as well as their no-anomaly counterparts.

3.2.1 Synthetic Dataset – Jump Up

This data is shown in Figure 4. The "normal" data consists of oscillating values between periods of high and low value. The high periods have values between ~75 and ~85, the low periods have values between ~18 and ~24. The anomaly for this

dataset is that in the high period at around Sample 3000, the values jump up to between ~140 and ~160.

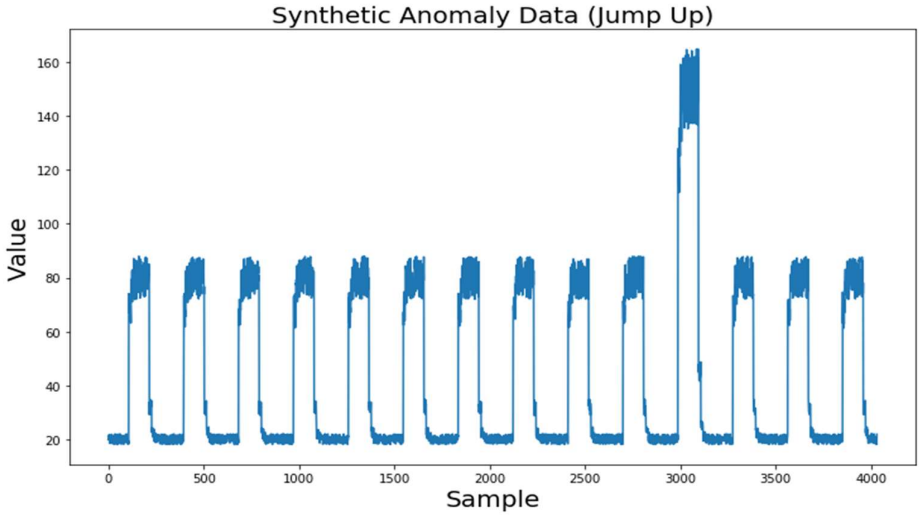

Figure 4: *Plot of the synthetic anomaly "jump up" dataset*

3.2.2 Synthetic Dataset – Jump Down

The "jump down" dataset is largely the same as the "jump up" dataset. It consists of the same "normal" data with the difference of the anomaly being a drop in the values of the high period at around Sample 3000 to between ~39 and ~44. The dataset is shown in Figure 5.

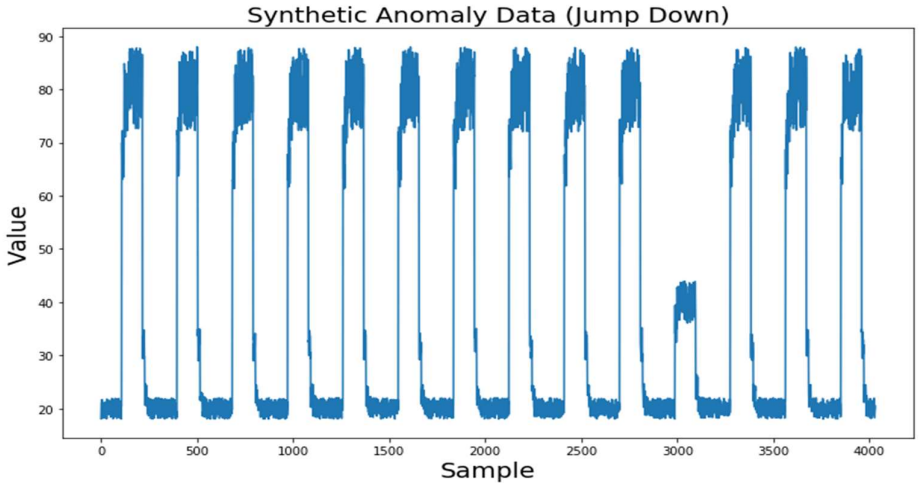

Figure 5: *Plot of the synthetic anomaly "jump down" dataset*

3.3 SmartMeter Energy Consumption Data in London Households

The dataset is obtained from the London Datastore [25] and consists of energy consumption readings for London Households that were part of the UK Low Carbon London project between November 2011 and February 2014. Energy consumption readings in kWh were taken every half hour. For my experiments I chose to test my algorithm on the energy consumption data for a single household, shown in Figure 6, with the dataset spanning the timeframe of roughly a year.

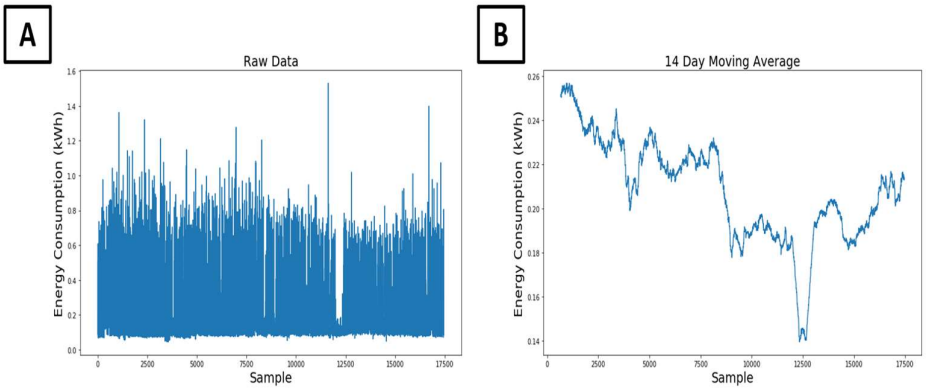

Figure 6: *Energy consumption data for a single household over a period of roughly a year,* ***a)*** *the raw data* ***b)*** *a 14-day moving average.*

The energy consumption data for this household looks relatively normal for the majority of the time period. It has high frequency variations caused by day-night cycles and potentially day-to-day weather variations. These high frequency variations are smoothed out in Figure 6b using a 14-day moving average. This begins to show longer term variations and highlights periods of lower/higher energy consumption. Since the dataset takes place over the period of a year, the higher energy consumption periods are most likely due to seasonality as more energy will be used in colder and darker months. Within these longer-term trends, however, there are still potential anomalies. The clearest of these is at around Sample 12500, where there is a significant drop in the energy consumption over a period. As this is during the long-term trend of lower energy consumption, during the summer period, we can infer that this lack of energy consumption could be due to the homeowners going on a summer holiday. This drop is an anomaly that we would aim to pick up with the anomaly detection algorithms.

4 Results

4.1 Conformal Prediction With USPS

When using transductive conformal prediction with the USPS dataset, the algorithm took ~84 seconds with the SVM non-conformity measure and ~281 seconds with the 1NN non-conformity measure to make a prediction for a single sample. Since there are 2007 test samples, this means it would take roughly 46 hours and 157 hours with the different non-conformity measures respectively to make predictions for the whole test set. Therefore, results were not collected for this algorithm and instead inductive conformal prediction was used.

Inductive conformal prediction achieved an accuracy of 0.946 with SVM and average credibility of 0.474. With 1NN, it achieved an accuracy of 0.945 and an average credibility of 0.478. The inductive conformal prediction algorithm shortened the run time of the algorithms to ~6 seconds and ~8210 seconds respectively, showing the significant increase in computational efficiency inductive conformal prediction provides over transductive conformal prediction.

However, for the purposes of anomaly detection, point predictions from conformal prediction are not the most important result. Instead it is important to look at the mode of prediction sets and, therefore, the validity of the algorithms. Validity of the conformal prediction algorithm can be shown using a validity curve. This is a plot of the significance level against the error rate (%), where an error is when a sample's true label is not in the prediction set output by conformal prediction. If the predictor is valid, it should be a diagonal line along the square [0,1]. Figure 7 shows the validity curves for SVM and 1NN inductive conformal prediction in the batch mode of learning. For both non-conformity measures, the validity curve does not exactly follow the diagonal of the square [0,1]. This is due to the batch mode of learning which has weaker validity than the online mode. They do, however, follow the line closely and provide the linear relationship between significance level and error rate. Additionally, the validity curve using smoothed p-values has been plotted for both non-conformity measures. It shows no difference from the regular p-value. This can be attributed to a lack of ties in the non-conformity scores, meaning the p-values calculated by the two different equations will be virtually the same.

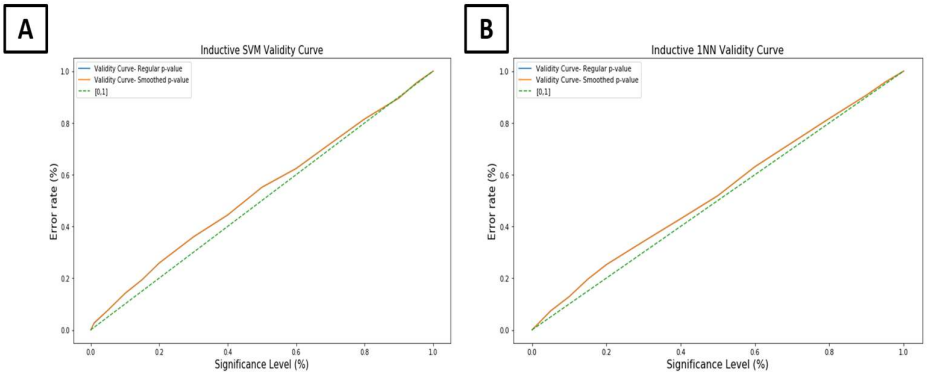

Figure 7: *Validity curves for inductive conformal prediction with* ***a)*** *the SVM non-conformity measure and* ***b)*** *the 1NN non-conformity measure. Both regular and smoothed p-values are shown.*

As discussed in Section 2, the online mode of conformal prediction has a stronger case of validity. Here I tested online inductive conformal prediction with the SVM non-conformity measure on the USPS dataset. An initial training set size of 1394 samples was used with 7904 new samples predicted for one-by-one. There are 3 different methods for implementing inductive conformal prediction in the online mode of learning. The first of these, Mode 1, is to only add new samples to the calibration set once predictions have been made for them. The cumulative number of multiple predictions, empty predictions and errors for Mode 1 at significance levels 1%, 5% and 20% are shown in Figure 8. Multiple predictions are prediction sets with more than one label, empty predictions are prediction sets with no labels and errors are prediction sets that do not include the true label of the sample. At a significance level of 1%, almost all the predictions were multiple predictions, with a total of 7902 multiple predictions, so the number of multiple predictions grows linearly over time. In addition, there were 70 errors, giving an error rate of 0.9%. For the significance level of 5% there is an initial rapid increase in the number of multiples which then levels out and increases at a slower rate, reaching a total of 649. The initial increase is due to a lack of samples in the calibration set. Unlike at significance level 1% there were a number of empty predictions too, reaching a total of 12. Finally, the plot for a significance level of 20% shows different behaviour as well. Since the significance level is so high, it is unlikely multiple p-values for a sample will be over 0.2. Therefore, the number of multiple predictions barely grows and only reaches a total of 9. Instead, the number of empty predictions and errors grow together reaching totals of 1502 and 1571 respectively. For this significance level, therefore, most of the erroneous predictions are due to empty predictions, differing from the other significance levels where an error most likely occurs due to predicting the wrong label. Figure 9 shows the validity curve for mode 1 of inductive online prediction. Unlike with batch prediction, this validity curve almost perfectly matches the diagonal of the square [0,1], demonstrating the stronger case of validity that online prediction has over batch prediction.

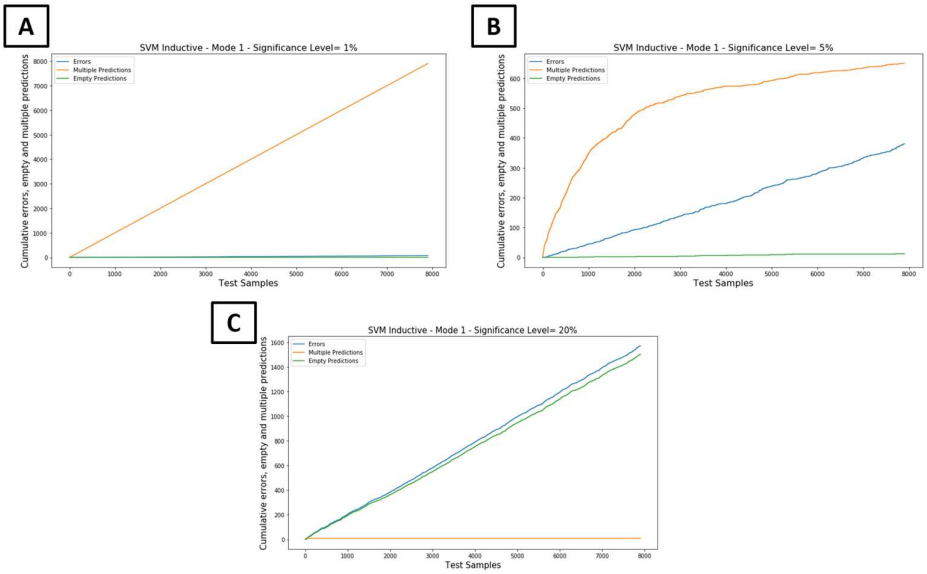

Figure 8: *Cumulative totals of multiple predictions, errors and empty predictions for inductive conformal prediction with SVM in online Mode 1 at significance levels* ***a)*** *1%* ***b)*** *5%* ***c)*** *20%*

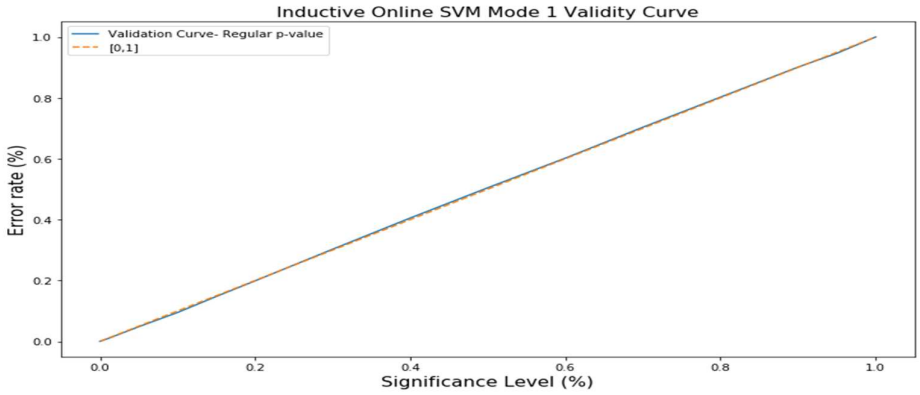

Figure 9: *Validity curve for inductive conformal prediction with SVM in online Mode 1*

The second mode of online inductive conformal prediction, Mode 2, is to add the calibration set to the training set at intervals and start the calibration set from new. The results of this are shown in Figure 10. In this experiment I used an interval of 1000 samples, so whenever the calibration set reached a size of 1000 samples, all of those samples were added to the training set at once and removed from the calibration set. The plots for significance levels 1% and 20% do not show any clear difference from Mode 1. At significance level 1% there are slightly less multiples, reaching a total of 7186. Conversely, at significance level 20% there is an increase in the number of multiples, totaling 37. However, the clearest difference between Mode 1 and Mode 2 is at significance level 5%. As can be seen in Figure 10b, there is the same initial rapid increase in the number of multiple predictions. Then, once the

calibration set samples have been added to the training set at Sample 1000, the number of multiples plateaus, no longer increasing. The only further increase in multiples is seen at the start of the training set updates. This is when the number of samples in the calibration set is low and so multiples are more likely, giving rise to this step-like pattern in the graph. The final total of multiple predictions is 552, 97 less than Mode 1. While the decrease in multiple predictions is a positive, there is a downside to using Mode 2. If this mode were to be used for anomaly detection, the small periods where there is a lack of samples in the calibration set, and most of the predictions are multiples, could cause the algorithm to miss anomalies that exist in those periods. Mode 3 for online prediction aims to provide a solution to this. Figure 11 shows the validity curve for Mode 2. As with Mode 1 the method is still valid.

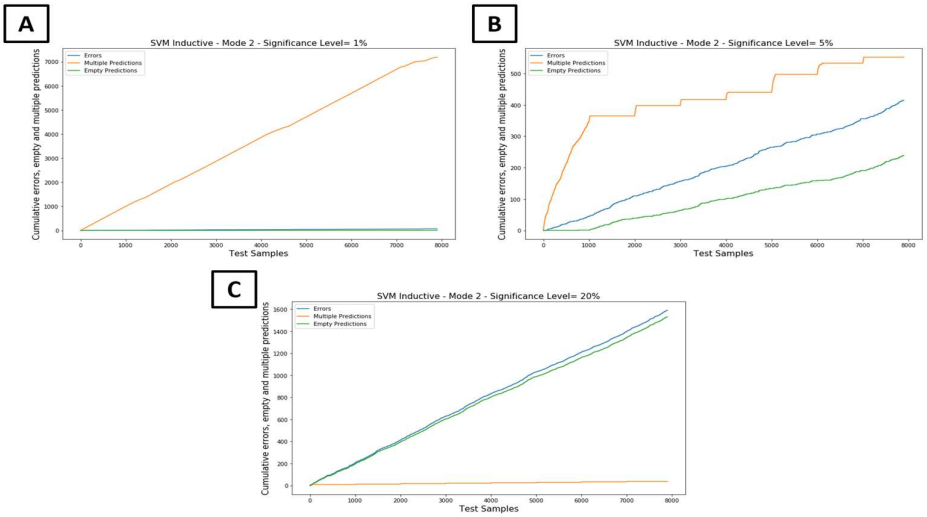

Figure 10: *Cumulative totals of multiple predictions, errors and empty predictions for inductive conformal prediction with SVM in online Mode 2 at significance levels* ***a)*** *1%* ***b)*** *5%* ***c)*** *20%*

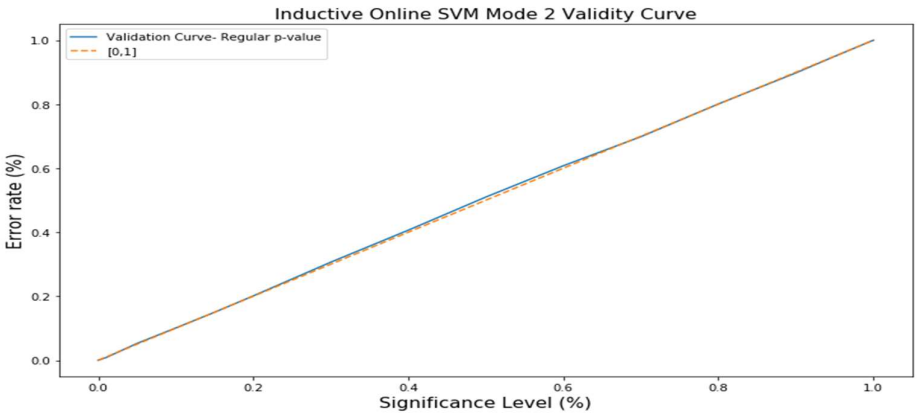

Figure 11: *Validity curve for inductive conformal prediction with SVM in online Mode 2*

Mode 3 provides a solution to the issues of Mode 2 by retaining a number of samples in the calibration set at these update intervals. In this experiment, the update interval of 1000 samples was kept, but 200 samples were retained in the calibration set. So, once the calibration set reaches a size of 1000 samples, the first 800 samples are added to the training set and the last 200 are retained within the calibration set. Therefore, when a training set update occurs, there will not be a period with few calibration samples and many multiple predictions, causing anomalies to be missed. The results of Mode 3 are shown in Figure 12. Again, the plots for significance levels 1% and 20% are similar to the other two modes, with small changes in the number of multiple and empty predictions. For significance level 5%, the difference that has been made by retaining calibration set samples is evident. Once the first update to the training set has taken place, the number of multiples plateaus, barely increasing any further. The total number of multiple predictions was 362, which is 287 less than Mode 1 and 190 less than Mode 2. As discussed in Section 2.1.2, there is a question of validity for this mode, as theoretically it is broken by using some samples for calibration across update trials. However, the validity curve was plotted for Mode 3 (Figure 13) and shows that this method is still valid.

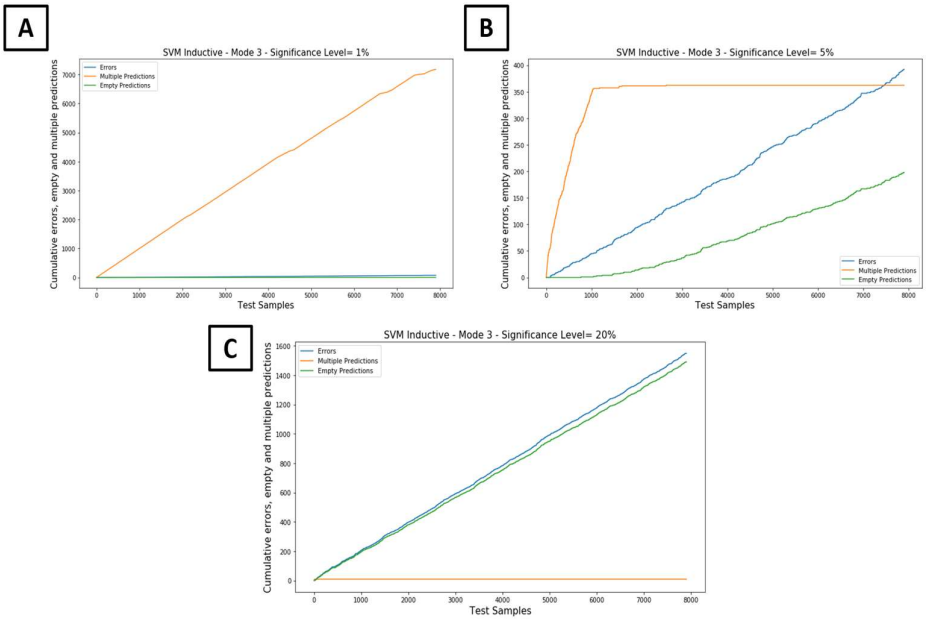

Figure 12: *Cumulative totals of multiple predictions, errors and empty predictions for inductive conformal prediction with SVM in online Mode 3 at significance levels* ***a)*** *1%* ***b)*** *5%* ***c)*** *20%*

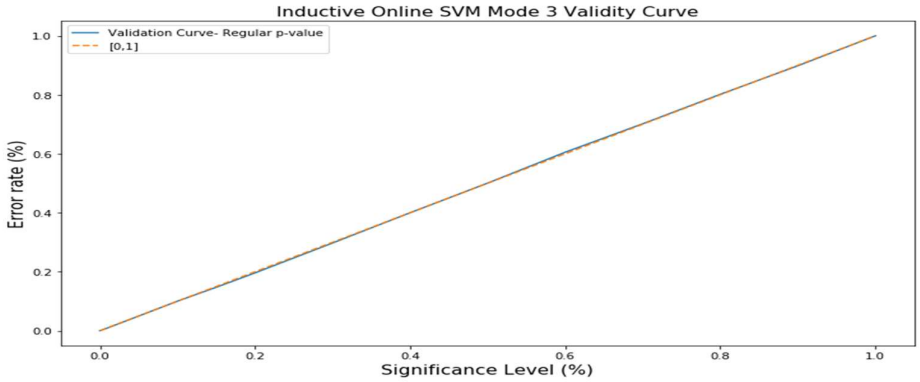

Figure 13: *Validity curve for inductive conformal prediction with SVM in online Mode 3*

4.2 Conformal Anomaly Detection With USPS

To test the method of conformal anomaly detection on non-time series data, I chose to use USPS. The handwritten digits flagged by the algorithm as anomalies can be clearly visualized and assessed as to whether they are true anomalies or false alarms. Although anomalies do already exist in the USPS as poorly written digits or digits written in a way that makes them look like another digit, I have also added a number of my own anomalies. I took 6 samples randomly and increased the value of every other pixel. The result of this was to create digits which had white bars across them and should be flagged as anomalies. The algorithm I used was the inductive conformal anomaly detector with SVM and in the online Mode 3. ICAD was chosen due to its better computational efficiency and the online mode 3 was used since it has a stronger validity than batch mode. I tested the algorithm at three significance levels: 1%, 5% and 20%. For each significance level I observed the samples that have been flagged as anomalies and aimed to classify them myself as anomaly or false alarm. This allowed me to provide a rough false alarm rate.

Examples of the anomalies output by the algorithm at a significance level of 1% are shown in Figure 14.

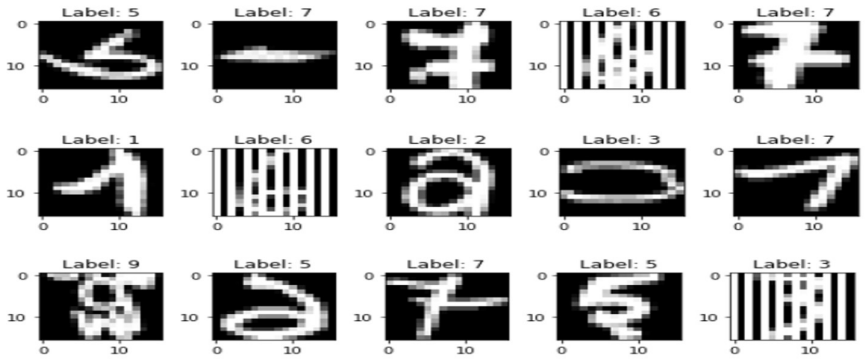

Figure 14: *Examples of the handwritten digits classified as anomalies by the online SVM ICAD at significance level 1%*

At significance level 1%, a total of 73 samples were classified as anomalies. This is proof of the well-calibrated false alarm rate. Even if all the 73 samples were false alarms, the false alarm rate would only be ~0.9%, which is lower than the significance level. From observing the anomalous samples, I classified 13 samples as being false alarms, giving a false alarm rate of ~0.2%. Where I was unsure if a sample was a false alarm or not, I classified it as a false alarm, preferring to overestimate the false alarm rate instead of underestimating it. Additionally, 4 out of 6 of the anomalies I added myself were detected. An example of these types of anomalies can be seen in Figure 14 as the digits with vertical white stripes.

For a significance level of 5%, a total of 392 samples were classified as anomalies, ~4.95% of the total samples. From these samples I classified 101 as being false alarms, to give a false alarm rate of ~1.3%. 5 out of 6 of the added anomalies were detected. Showing that as you increase the significance level, more anomalies will be detected but the false alarm rate will also increase.

Finally, for a significance level of 20%, a total of 1545 samples were classified as anomalies, ~19.5% of the total samples. I classified 477 of these to be false alarms, giving a false alarm rate of ~6%. All 6 of the added anomalies were detected at this significance level.

4.3 Conformal Martingales With USPS

To demonstrate how conformal martingales can be used for testing the exchangeability assumption in the online mode I have used the algorithm on the USPS dataset. As previously mentioned, the USPS dataset is known to be heterogenous, with the training and test set having different distributions. If the training and test sets are concatenated into one large dataset, then the conformal martingale algorithm should be able to detect this change in distribution. So, for this experiment, I first concatenated the test set to the training set, then I split the first 100 samples off into an initial training set that would further be added to through the online mode. The martingale used for this experiment was the power martingale with an epsilon value of 0.92. Figure 15 shows how the martingale changes over time for both SVM and 1NN non-conformity measures.

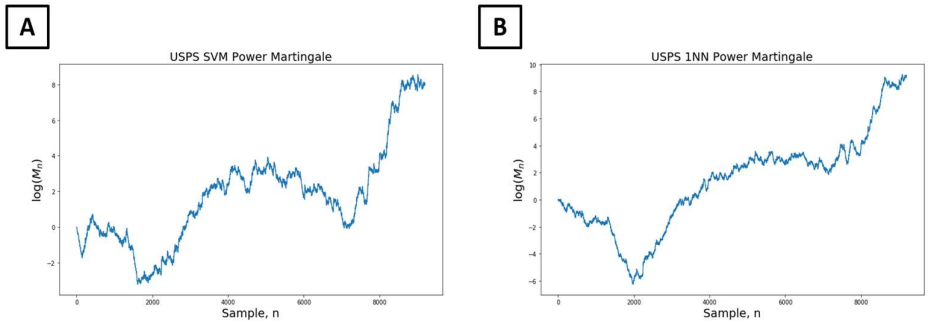

Figure 15: *Power martingale (ε=0.92) values for the whole USPS dataset using* ***a)*** *SVM non-conformity measure and* ***b)*** *1NN non-conformity measure. A* $\log_{10}$ *scale has been used for the martingale value on the y-axis.*

Both martingale plots show an initial decrease in the martingale value from Sample 0 to about Sample 2000. This is due to the first 2000 samples being drawn from the same probability distribution and, therefore, not breaking the exchangeability assumption. However, the value then begins to increase. For example, with the 1NN non-conformity measure, the martingale rises from ~$1x10^{-15}$ to ~$1x10^{5}$ between samples 2000 and 4000. This indicates a change in distribution. However, the test set does not start until Sample 7191, suggesting that the training set itself is heterogenous, not just that the training and test set have different distributions. After this increase, the martingale value stays relatively steady, even decreasing with the SVM non-conformity measure. This continues until about Sample 7191, the start of the test set. As can be seen in Figure 15, there is a sharp increase in the martingale value, reaching a final value of $3.5x10^{8}$ for SVM and $1.5x10^{9}$ for 1NN, indicating that the exchangeability assumption has been violated and showing that the test and training set have different distributions.

To show how the martingale acts when all the data is drawn from the same distribution, I randomly shuffled the data beforehand. Figure 16 shows how the martingale changes for both non-conformity measures using the randomized data. While there are small increases and decreases in the value, overall, it is steadily decreasing to a very small final value of $5.5x10^{-15}$. This shows that the exchangeability assumption has not been violated and the whole dataset is drawn from the same distribution.

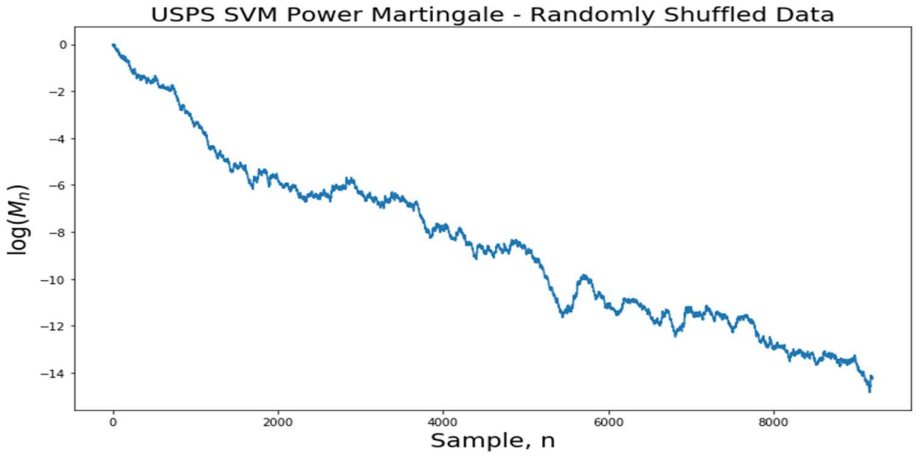

Figure 16: *Power martingale (ε=0.92) values for the whole USPS dataset, after being randomly shuffled, using the SVM non-conformity measure. A $\log_{10}$ scale has been used for the martingale value on the y-axis.*

4.4 Anomaly Detection on Synthetic Time Series Datasets

Both the conformal power martingale and the MUSUC E-values algorithms have possible applications in time series anomaly detection. To assess their effectiveness, I will begin by using relatively simple synthetic anomaly datasets, as discussed in

Section 3.2. When dealing with time series data like this, the non-conformity measures have to be adapted, since there are no classes for the samples except for the class of anomaly or normal that we assign to them. Therefore, in the case of SVM, a one-class SVM was used with the rbf kernel.

4.4.1 Synthetic Dataset – Jump Up

Figure 17 shows the martingale value and the MUSUC value for this dataset, using a $\log_{10}$ scale. Both methods show a sharp increase in value at the location of the anomaly, reaching a peak of 1.7×10^{16} for martingales and 1.2×10^{29} for MUSUC. The results of the anomalies detected using different thresholds are shown in Figures 18 and 19. In the real data the actual anomaly begins at Sample 3000. The martingale algorithm first detects this anomaly at sample 3094 for threshold 10^{15} and at sample 3023 for threshold 10^{5}. However, for a lower threshold the algorithm produces more false alarms, as can be seen in Figure 18, where it believes all the samples after the true anomaly are anomalies. For the MUSUC algorithm, it first detects this anomaly at sample 3030 for threshold 10^{15} and at sample 2999 for threshold 10^{5}. As expected, it follows a similar pattern to martingales in terms of false alarms at lower thresholds, but at the same thresholds produces more false alarms (Figure 19).

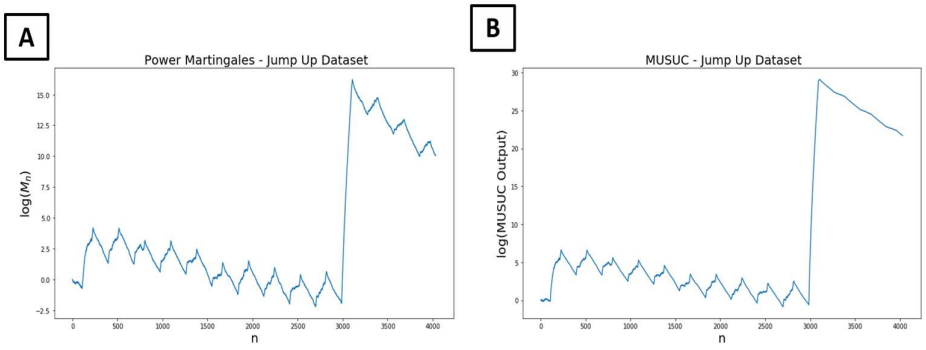

Figure 17: ***a)*** *Power martingale value and* ***b)*** *MUSUC value over time for the synthetic "jump up" dataset. A $\log_{10}$ scale is used for the y-axis.*

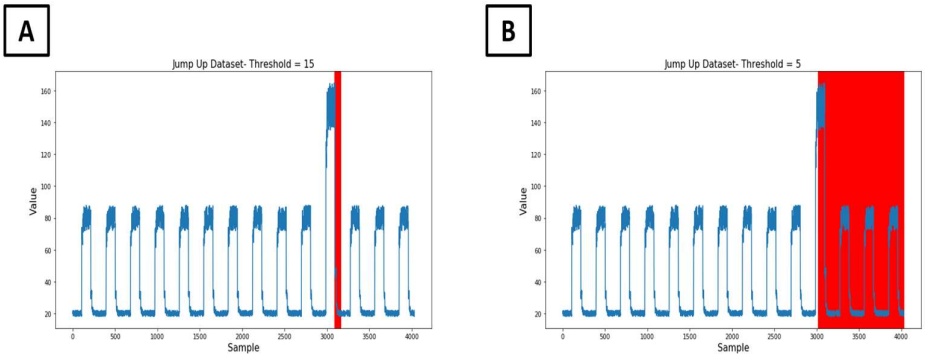

Figure 18: *Anomalies detected by the power martingale algorithm (marked by red bars) for thresholds* ***a)*** *15 and* ***b)*** *5. The thresholds are on the log scale, so the true values of the thresholds are 10^{15} and 10^{5}.*

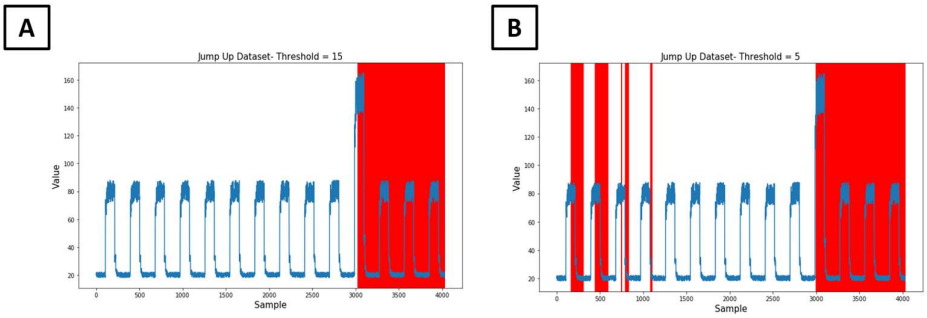

Figure 19: *Anomalies detected by the MUSUC algorithm (marked by red bars) for thresholds* ***a)*** *15 and* ***b)*** *5. The thresholds are on the log scale, so the true values of the thresholds are* 10^{15} *and* 10^{5}.

4.4.1.1 A Change to the Algorithm

For both algorithms there remains an issue which is visible in Figure 17. After the anomaly has occurred at Sample 3000, the martingale/MUSUC value remains high for data samples that are not anomalous. This causes a lot of false alarms after a true anomaly has occurred. For example, in Figure 19. A high level of false alarms like this is not ideal for an anomaly detection algorithm, so it is important to alter the algorithms to mitigate this.

Algorithm 6 Conformal Martingales - Changed Algorithm

INPUT: Z = $((x_1, y_1), (x_2, y_2), ..., (x_N, y_N))$
INPUT: Non-conformity measure: A
INPUT: Betting function: f
INPUT: Threshold: C

i=0
σ_0=0
for n=1, ..., N **do**
  Create extended training set with new sample: $z_{\sigma_i+1}, ..., z_n$

  Calculate non-conformity scores for every sample in extended training set: $A(z_{\sigma_i+1}, ..., z_n)$

  Calculate the p-value (Eq. 1) of the new sample based on the rank of its non-conformity score within the extended training set

  B_n=f(p-value)

  $M_n = B_{\sigma_i+1} \times ... \times B_{n-1} \times B_n$

  if $M_n > C$ **then**
    i = i + 1
    σ_i=n
  end if

end for

OUTPUT: Anomalies detected = $\sigma_1, \sigma_2, \sigma_3, ...$

One method is to reset the martingale/MUSUC value once an anomaly has been discovered and reset the data you are training on to be the sample after the detected anomaly. Therefore, after a change point has been detected, the martingale/MUSUC value will no longer remain high. This is the suggested procedure in Vovk, 2020 [22]. Algorithm 6 formalizes this difference in the algorithm for martingales.

Figure 20 demonstrates the change this makes to the martingale values over time, as well as the anomalies detected by the algorithm for different thresholds. A clear improvement is seen for higher thresholds, with the true anomaly being detected and few false alarms. It also improves the delay time to detection of the anomaly. At the highest threshold tested, 10^6, the anomaly is detected at Sample 3029, similar to the lowest threshold tested with the previous method and without the high number of false alarms. However, with this method it is not always as simple as assuming that a lower threshold will give a faster detection of the anomaly. For example, in Figure 20c when the threshold is 10^4, the anomaly isn't detected until Sample 3105. This is because of a false alarm occurring just before the actual anomaly, so the martingale value must increase back from 1 to the threshold again, delaying the detection. The threshold must be chosen carefully, not only to avoid false alarms but also to avoid false alarms obscuring true anomalies.

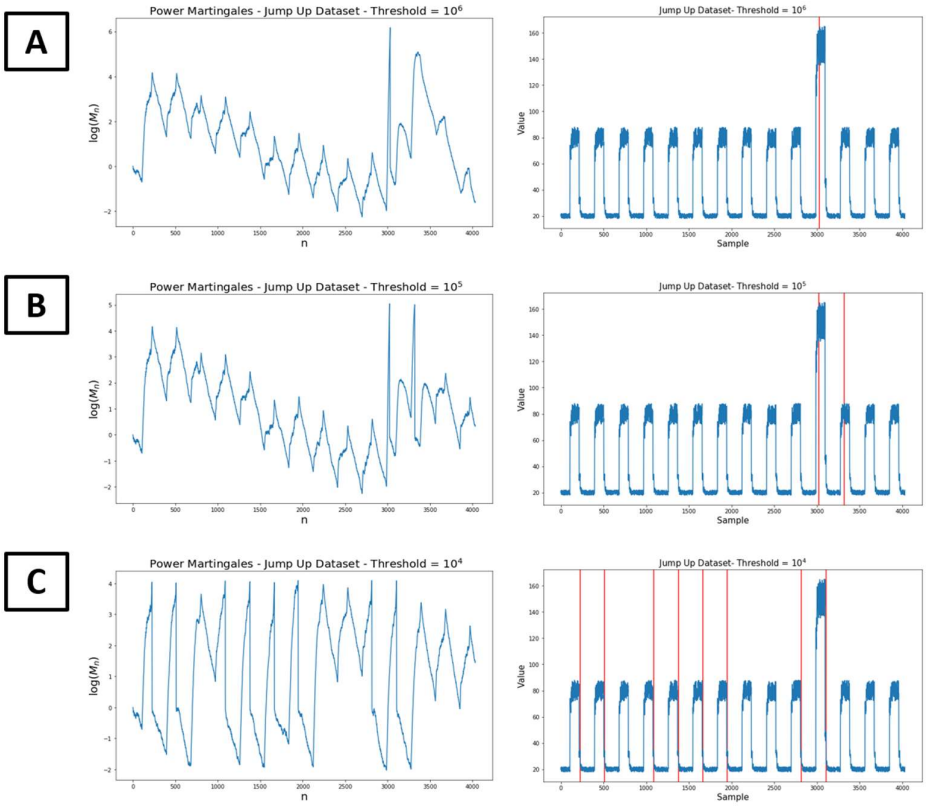

Figure 20: *Power martingale values and anomalies detected by the changed algorithm on the "jump up" dataset using a threshold of* ***a)*** 10^6, ***b)*** 10^5 *and* ***c)*** 10^4

The same change was made to the MUSUC algorithm and the results are shown in Figure 21. Similar results to Figure 20 are seen, the true anomaly is detected at all thresholds and the more you lower the threshold, the more false alarms occur. The true anomaly is detected at samples 3003, 3013 and 3003 for thresholds 10^4, 10^5 and 10^6 respectively. This is the fastest the true anomaly has been detected by the algorithms tested so far. As observed with the martingale algorithm, a lower threshold does not necessarily mean a lower delay time to detecting the anomaly. While the threshold will control the false alarm rate and sensitivity, delay time can actually be lower for higher thresholds. For example, for a threshold of 10^6 the anomaly was detected at Sample 3003, faster than at threshold 10^5. The most common false alarm that occurred during these experiments is at around Sample 3290. While this is clearly not an anomaly, it is still a change point as the distribution is changing from the anomalous distribution back to the distribution of the rest of the data.

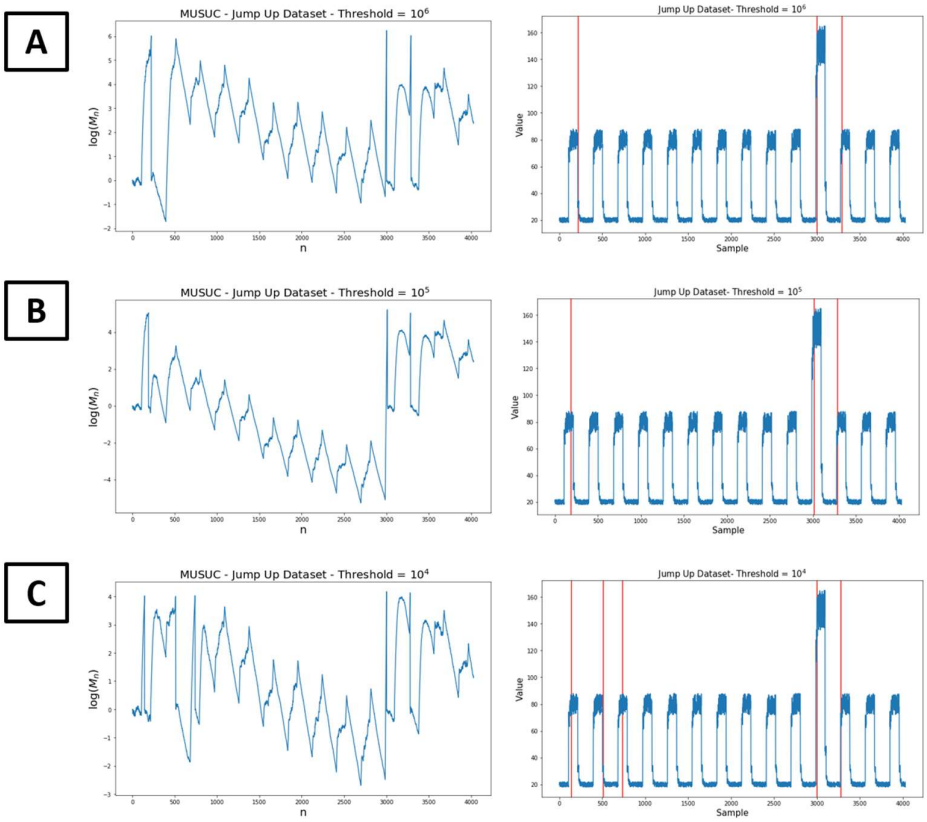

Figure 21: *MUSUC values and anomalies detected by the changed algorithm on the "jump up" dataset using a threshold of* ***a)*** 10^6, ***b)*** 10^5 *and* ***c)*** 10^4

4.4.2 Synthetic Dataset – Jump Down

Figures 22 and 23 show the martingale and MUSUC outputs respectively for the "jump down" dataset, using a $\log_{10}$ scale for the martingale/MUSUC value plot. As with the "jump up" dataset, the MUSUC values reach the threshold much faster since they grow and decrease at a greater rate than martingales. This is evident in Figure 24, where the martingale and MUSUC values are shown without a threshold. The MUSUC value increases to roughly 10^7 within the first 400 samples whereas it only rises to 10^4 for martingales. Additionally, the MUSUC algorithm reaches a higher peak, with a value of 3.9×10^{14} compared to the peak of 1.3×10^{12} for martingales.

For the martingale algorithm (Figure 22) the true anomaly is detected at samples 3029, 3036 and 3044 for thresholds of 10^4, 10^5 and 10^6 respectively. For thresholds 10^5 and 10^6 only one false alarm is detected. However, for the lowest threshold of 10^4, 5 false alarms are detected. For the MUSUC algorithm (Figure 23) the anomaly is detected at samples 3031, 3051 and 3036 for thresholds of 10^4, 10^5 and 10^6 respectively. Additionally, for all the same threshold values, the MUSUC algorithm detects more false alarms. For a threshold of 10^6 it detects 2 false alarms, for 10^5 it detects 4 false alarms and for 10^4 it detects 7 false alarms. This may be due in part to MUSUC reaching higher values than martingales and so the thresholds cannot simply be compared. A higher threshold for MUSUC may show similar results to lower thresholds with martingales. To test this, we must calculate a false alarm rate for different thresholds for the two algorithms.

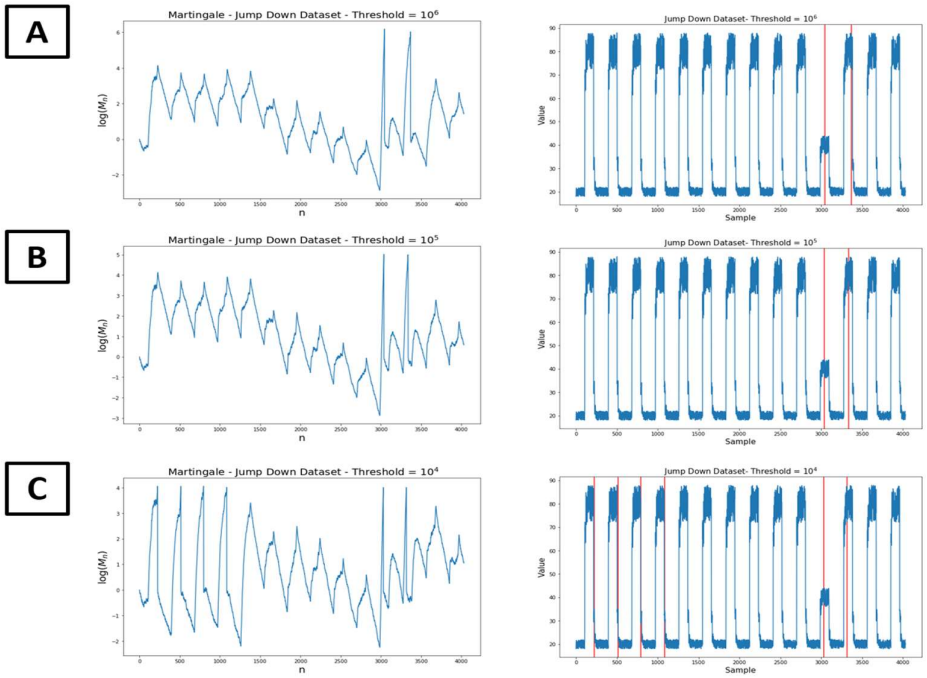

Figure 22: *Martingale values and anomalies detected by the changed algorithm on the "jump down" dataset using a threshold of* ***a)*** 10^6, ***b)*** 10^5 *and* ***c)*** 10^4

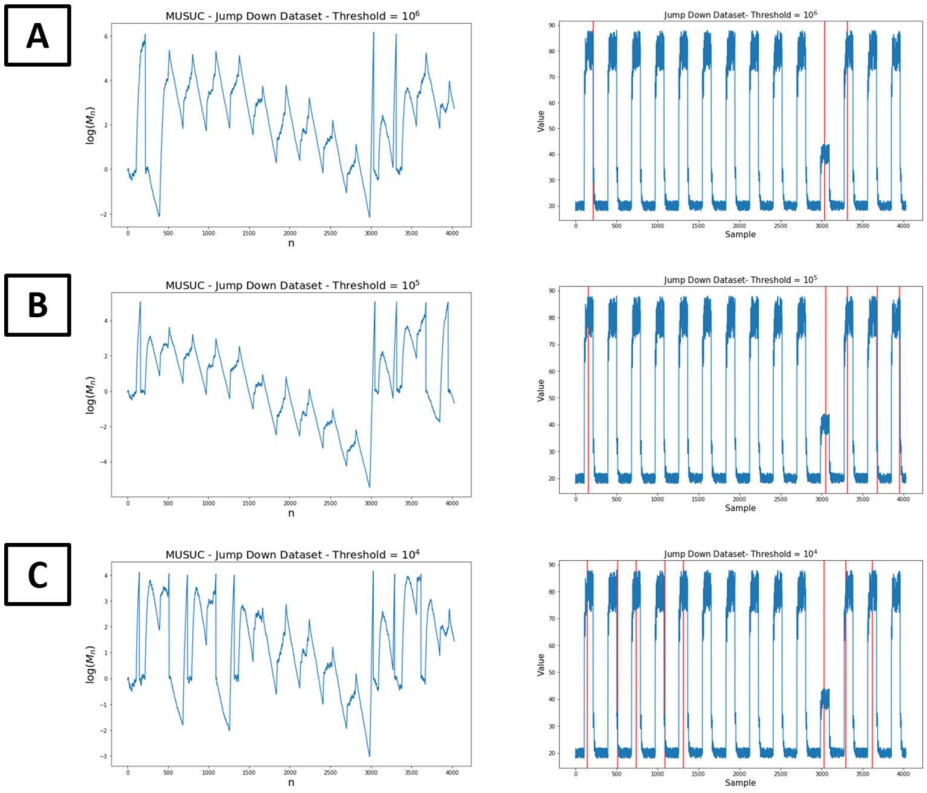

Figure 23: *MUSUC values and anomalies detected by the changed algorithm on the "jump down" dataset using a threshold of* ***a)*** 10^6, ***b)*** 10^5 *and* ***c)*** 10^4

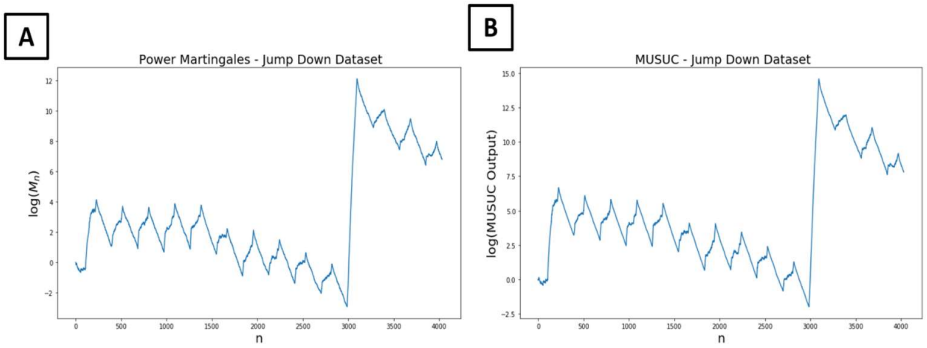

Figure 24: ***a)*** *Power martingale value and* ***b)*** *MUSUC value over time for the synthetic "jump down" dataset. A* $\log_{10}$ *scale is used for the y-axis.*

4.4.3 False Alarm Rate Test

To find a rough false alarm rate for different thresholds for these two algorithms, I tested the algorithms' performance on the same dataset but with the anomaly at

Sample 3000 removed. The "jump up" and "jump down" datasets both have the same underlying dataset, just with a different type of anomaly added. This dataset is shown in Figure 25.

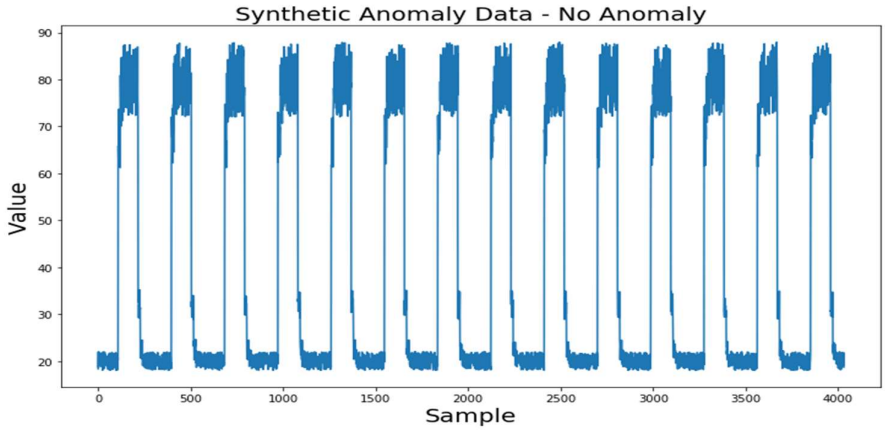

Figure 25: *Plot of the underlying data with no anomaly from the "jump up" and "jump down" datasets.*

Figure 26 shows the output of the two algorithms for the dataset. For both algorithms the output never reaches an extreme value, with a peak of 3.6×10^3 for martingales and a peak of 2.1×10^5 for MUSUC. Additionally, the value steadily decreases over time, which is expected since all of the data should be conforming to the exchangeability assumption. The two plots look very similar. However, the scale is different. As mentioned previously, the MUSUC value tends to increase and decrease much faster, resulting in a larger range of values.

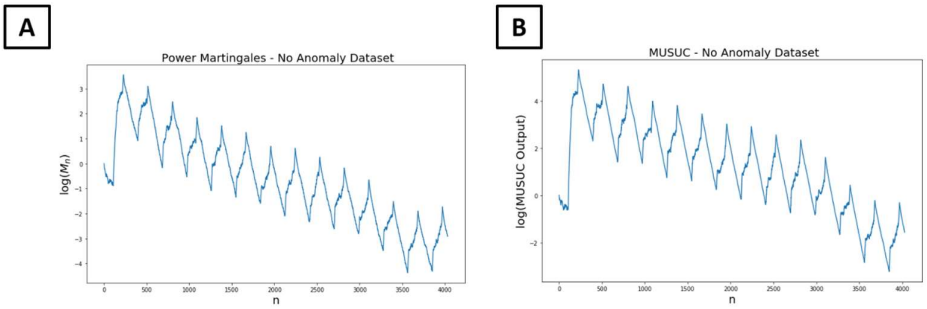

Figure 26: a) *Power martingale value and* ***b)*** *MUSUC value over time for the synthetic dataset with no anomaly. A $\log_{10}$ scale is used for the y-axis.*

Since all anomalies detected in this dataset can be considered as false alarms, we can easily calculate a false alarm rate. Figure 27 shows the false alarm rate for different thresholds for the two algorithms. MUSUC reaches a false alarm rate of 0 at a threshold of 10^6, higher than the martingale algorithm which reaches it at a threshold of 10^4. Additionally, when comparing the two methods at the same threshold, the MUSUC algorithm almost always has the high false alarm rate. It confirms the findings from experiments on the "jump up" and "jump down" datasets that the

MUSUC algorithm produces a higher number of false alarms. The false alarm rates for both algorithms are still very low, however, with this synthetic dataset. The maximum false alarm rate was 0.74% for MUSUC and 0.7% for the power martingales, and these only occurred at the very low thresholds of $10^{0.5}$ and 10^{1}.

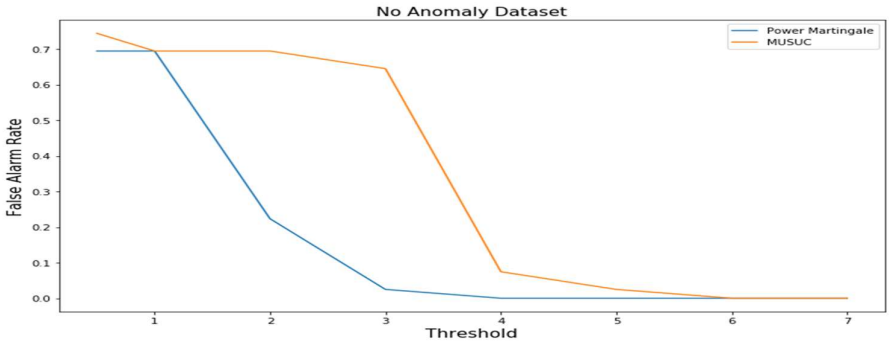

Figure 27: *Plot of the false alarm rate on the "No Anomaly" dataset when using the* ***a)*** *power martingale and* ***b)*** *MUSUC methods for different threshold values. A $\log_{10}$ scale is used for the x-axis, so a value of 4 corresponds to a threshold of 10^4.*

4.5 Anomaly Detection on the Energy Consumption Dataset

While testing the algorithms on synthetic anomaly data is useful to gain an understanding of how well the algorithms work, it is not always indicative of how successful they will be when used on real-world datasets. Therefore, the algorithm was tested on the real-world energy consumption data, as discussed in Section 3.3. The martingale and MUSUC values for this dataset when using a threshold of 10^4 are shown in Figure 28. Neither algorithm detected an anomaly in this data. This may be due to there being a high volume of "normal" data which causes the values to decrease to extremely small values. Therefore, when the anomaly appears at about Sample 12500, the martingale/MUSUC value is so small that a period of anomalous data has little effect and will not cause the value to reach a threshold. In this case using a lower threshold would not help detect the anomaly as the maximum value the MUSUC ever reaches before becoming extremely small is $3x10^1$.

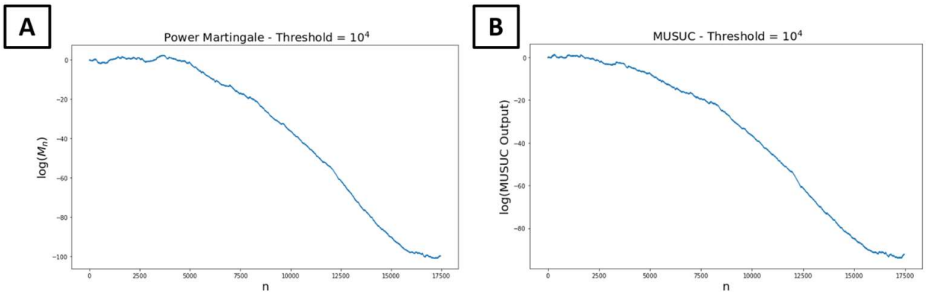

Figure 28: a) *Power martingale value and* ***b)*** *MUSUC value over time for the energy consumption dataset. A $\log_{10}$ scale is used for the y-axis.*

A possible method to deal with the value becoming too small is to use a sliding window when calculating the martingale or the E-value. This works by restricting the number of samples used in calculations, so if there are many "normal" samples the value will not become so small. For example, if a sliding window of 1000 was used, when the training data is less than 1000 samples, the algorithm runs as usual. However, once the training data reaches a size of greater than 1000, only the previous 1000 samples are used in the martingale/MUSUC calculation for a new test sample. The output values for martingale and the anomalies detected for different sliding window sizes of 1000, 500 and 250 with a threshold of 10^3 are shown in Figure 29. From my experiments, the larger the sliding window is, the longer the delay in detecting the anomaly. For martingales, the anomaly is detected at samples 12505, 12720 and 13337 for sliding windows of 250, 500 and 1000 respectively. This difference in delay time to detecting the anomaly is significant, as the difference in samples of 617 between windows of 500 and 1000 corresponds to a real time difference of almost 13 days.

The same experiments were carried out using the MUSUC algorithm, to test different sliding window sizes. The anomaly is detected at samples 12459, 12674 and 13275 for sliding windows of size 250, 500 and 1000 respectively. These results are shown in Figure 30. At each sliding window size, the MUSUC algorithm detects the anomaly earlier than the power martingale algorithm. For example, at a sliding window size of 250, the anomaly is detected 46 samples earlier, corresponding to a period of 23 hours. However, the most important factor for decreasing the delay time, in these experiments, was the size of the sliding window.

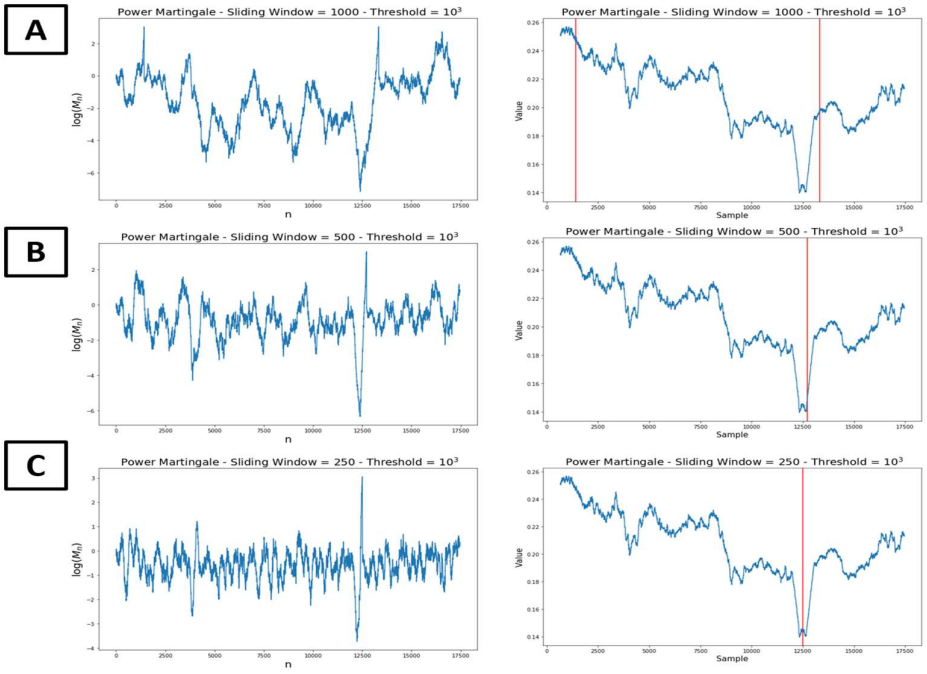

Figure 29: *Power martingale values and anomalies detected for the energy consumption dataset using a threshold of 10^3 and a sliding window size of* ***a)*** *1000,* ***b)*** *500 and* ***c)*** *250*

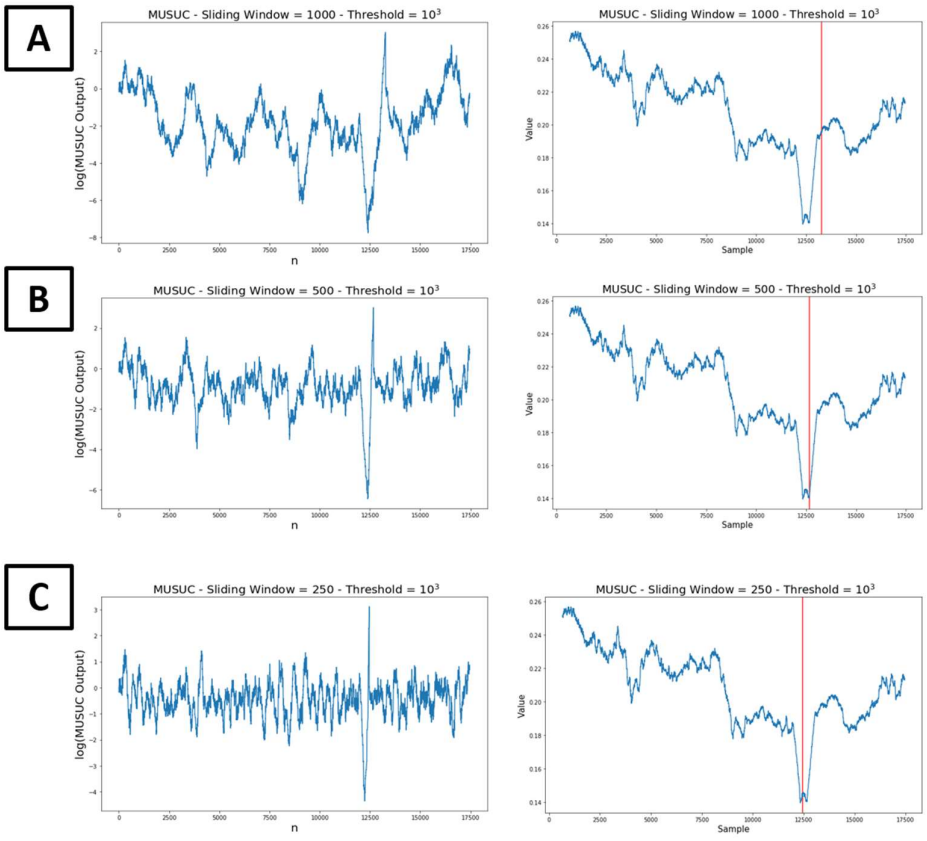

Figure 30: *MUSUC output values and anomalies detected for the energy consumption dataset using a threshold of 10^3 and a sliding window size of* ***a)*** *1000,* ***b)*** *500 and* ***c)*** *250*

Lower thresholds were tested for a sliding window size of 250 for both martingales (Figure 31) and MUSUC (Figure 32). As expected, lower thresholds produce more false alarms, with the lowest threshold of $10^{0.5}$ the martingale algorithm had 12 potential false alarms and the MUSUC algorithm had 30 potential false alarms. We are not able to say all of these alerts are definite false alarms, as they may be less obvious anomalies than the main anomaly we are focusing on, since this is real-world data. For example, at around Sample 4000 there is what looks like a smaller anomaly than the main anomaly. This is the only potential false alarm classified in Figure 31a, and so, for this threshold, there may actually be no false alarms and instead the algorithm has detected two anomalies. From these experiments it can be seen the MUSUC algorithm tends to produce more potential false alarms. It does, however, detect the main anomaly at around Sample 12500 slightly faster than the martingale algorithm. At a threshold of $10^{0.5}$ the MUSUC algorithm detects it at 12409 whereas the martingale algorithm detects it at 12430.

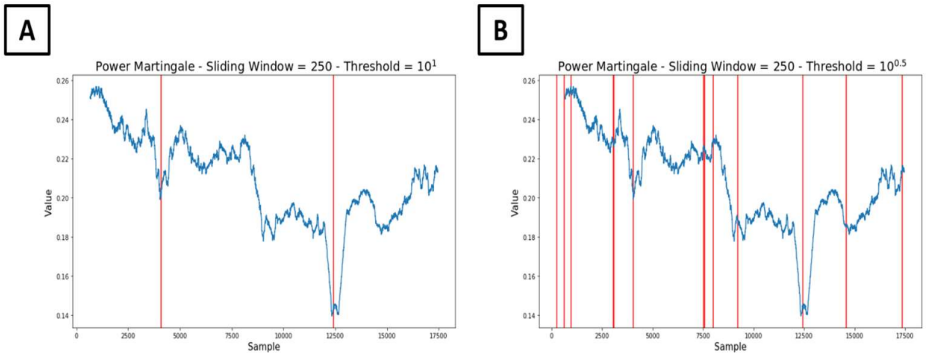

Figure 31: *Anomalies detected by the power martingale algorithm on the Energy Consumption dataset using a sliding window size of 250 and a threshold of* ***a)*** 10^1 *and* ***b)*** $10^{0.5}$

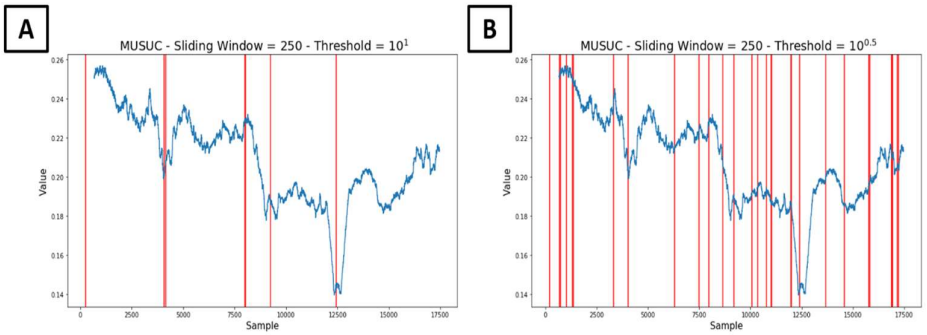

Figure 32: *Anomalies detected by the MUSUC algorithm on the Energy Consumption dataset using a sliding window size of 250 and a threshold of* ***a)*** 10^1 *and* ***b)*** $10^{0.5}$

Another method is to use the modification described in Section 2.3.1, which was proposed in Volkhonskiy et al., 2017 [20]. The modification stops the $\log_{10}$ of the martingale value from ever becoming negative. While the motivation behind this modification was to decrease the mean delay time to detection of anomalies, which would still be a useful improvement for our algorithm, the modification will also prevent the martingale value from becoming too small and unable to detect anomalies. Figure 33 shows the result of using this modification on the Energy Consumption dataset. This modification was unsuccessful at detecting the main anomaly in the dataset. It potentially allows for a number of anomalies to be detected early on between samples 1000 and 4500 if a lower threshold is used. However, overall, this modification is unsuccessful. Considering that this modification breaks the theoretical validity of the martingale method [20], it is preferable not to use it. Therefore, since it provides no clear benefit over using a sliding window, no more tests with this modification were carried out.

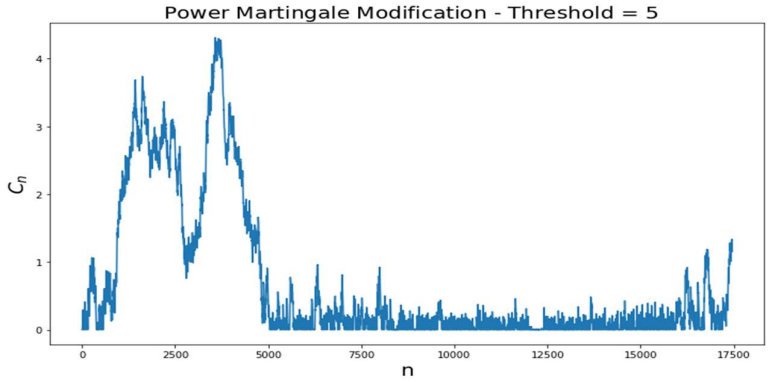

Figure 33: *Martingale value for the energy consumption data when using the modification to the algorithm. The y-axis is not a log scale but can be considered to be analogous to one since the modification adds the log of the betting function in calculating C. A threshold of 5 was used.*

4.5.1 False Alarm Rate

As with the synthetic anomaly data, it is desirable to attempt to calculate a false alarm rate for the algorithm using data with no anomalies. However, it is not possible to remove all anomalies from the energy consumption data as we do not know the location of all of them and what the "normal" value to replace them should be. Instead, we can randomly shuffle the energy consumption data. This will result in all samples being drawn from the same distribution, meaning the exchangeability assumption will not be violated and, therefore, there should be no anomalies. The randomly shuffled data and its 14-day moving average are shown in Figure 34.

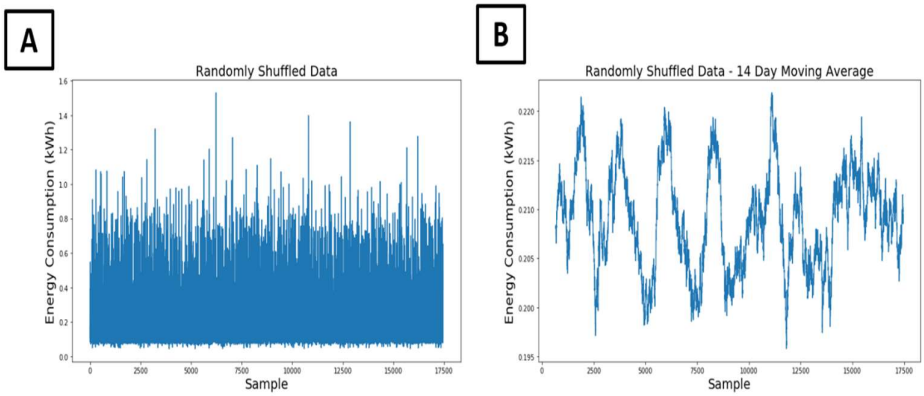

Figure 34: *Randomly shuffled Energy Consumption dataset as* ***a)*** *raw data and* ***b)*** *a 14 -day moving average*

As a sliding window size of 250 samples provided the best results for anomaly detection using both the martingale and MUSUC algorithms, it was used for the testing of the false alarm rate. The martingale and MUSUC values for this shuffled dataset are shown in Figure 35, using a threshold that is never reached. Both

algorithms never reach a value of greater than 10^1. This demonstrates how randomly shuffling the data results in a dataset with the same distribution and no anomalies.

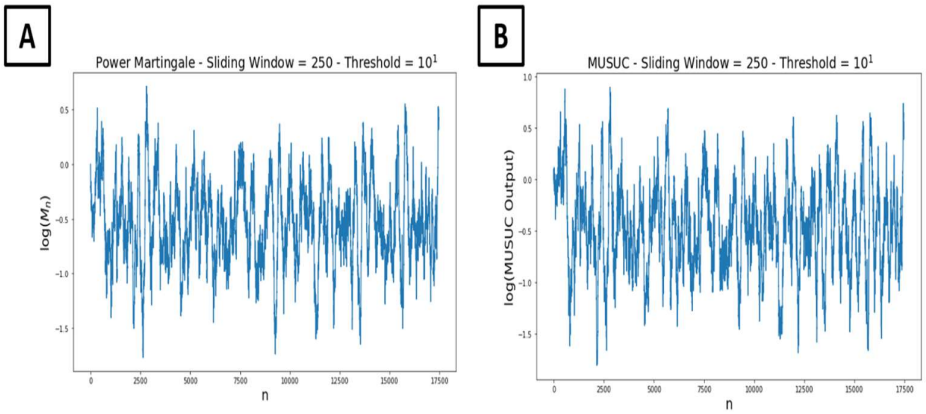

Figure 35: a) *Power martingale value and* ***b)*** *MUSUC value over time for the randomly shuffled energy consumption dataset. A $\log_{10}$ scale is used for the y-axis.*

The false alarm rates for the two algorithms at different thresholds are shown in Figure 36. The martingale algorithm decreases to a false alarm rate of 0 at the threshold value of $10^{0.8}$ and the MUSUC at the threshold value of $10^{0.9}$. Also, for thresholds lower than this, the MUSUC algorithm has higher false alarm rates. For example, at the lowest threshold tested, $10^{0.1}$, the martingale algorithm detects 29 false alarms, whereas the MUSUC algorithm detects 70.

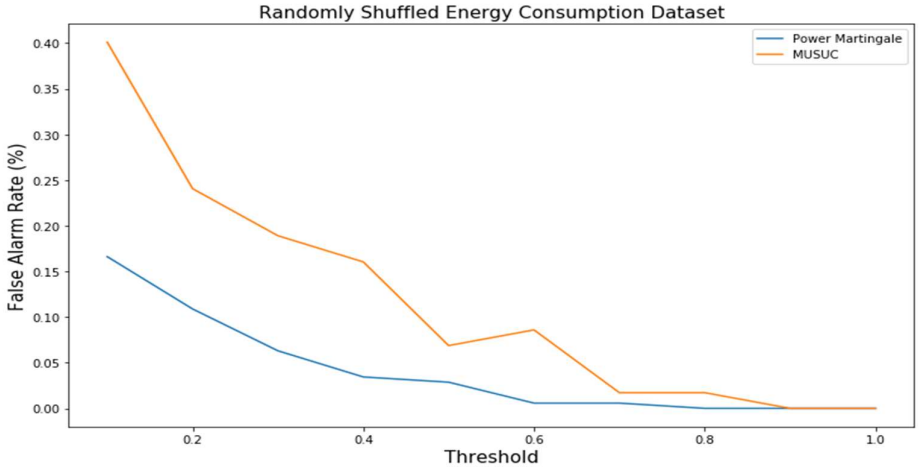

Figure 36: *Plot of the false alarm rate on the randomly shuffled energy consumption dataset when using the* ***a)*** *power martingale and* ***b)*** *MUSUC methods for different threshold values. A $\log_{10}$ scale is used for the x-axis.*

5 Discussion

5.1 USPS Anomaly Detection

Assessing the success of the algorithm on the USPS dataset robustly is not an easy task. The false alarm rate measure I calculated for assessing the effectiveness of the algorithm is obviously imperfect, as deciding whether a digit is a false alarm or not will be very subjective. Therefore, the false alarm rate may be inaccurate. However, it still gives a rough idea of the false alarm rate of the algorithm and demonstrates how this false alarm rate can be controlled by changing the significance level. Additionally, my results show the trade-off between sensitivity to anomalies and false alarm rate. At a low significance level only 4 of the 6 added anomalies were detected but the false alarm rate was low at roughly 0.2%. If it was important to detect all these added anomalies the significance level could be increased to 20% where 6 out 6 were detected, but this comes with a greatly increased false alarm rate of ~6%. Overall, I believe the algorithm was effective at finding anomalies within the dataset, whether that be the anomalies I added to the dataset or anomalies that already existed within the dataset

5.2 Sliding Windows

For the energy consumption dataset, both algorithms required a sliding window to be able to detect anomalies. This was not a feature I came across when carrying out background research on martingales or E-values for change point detection. It is possible that it may only be required for certain types of datasets, i.e. large time series dataset with high volumes of "normal" data. From observing the energy consumption data, it is clear why the concept of a sliding window is important for this particular dataset. As discussed in Section 3.3, this dataset spans the time period of roughly one year. In one year, the energy consumption will show variations from season to season, with low periods in the Summer and high periods in the Winter. Intuitively, when attempting to detect anomalies in the summer months, we are not looking for how certain data samples differ from energy consumption during Winter months. Instead, we are more interested in how certain samples differ from energy consumption in a more recent time period. Therefore, data from 8 months prior, during a different season, is not of use to our anomaly detection. It makes much more sense to use only the most recent samples in attempting to detect anomalies, which is what a sliding window does. Using all the previous samples in calculations may distort what is the "normal" value for the new sample, since it will be combining the different normal values for multiple seasons. This is the reason why I believe the sliding window improves anomaly detection so significantly for the energy consumption dataset.

An additional benefit of using sliding windows is improved computational efficiency. The smaller the sliding window size, the faster the algorithm will be able to calculate the martingale/MUSUC value.

However, sliding windows are a dataset dependent modification and will not necessarily provide improvements in anomaly detection for every dataset used. Using a sliding window will reduce the number of samples used to make a prediction and, therefore, in datasets where all previous data is useful, it will result in worse predictions. It is up to the user to decide whether using a sliding window is applicable for their dataset.

5.3 Power Martingales vs MUSUC

In this study, the majority of my results have been from applying the power martingale and MUSUC algorithms to time series datasets for the purpose of anomaly detection. These two algorithms have a lot of similarities, the main one being that they are based on conformal prediction. Therefore, it is important that I compare the effectiveness of the two algorithms in detecting anomalies. Three time series datasets have been used for the anomaly detection experiments. These are the synthetic "jump up" and "jump down" datasets and the real-world energy consumption dataset.

For the synthetic "jump up" dataset, the algorithms showed very similar results. As can be seen in Figure 17, the two algorithms output values increase and decrease in roughly the same way. The value oscillates with a small amplitude while steadily decreasing. Then, when the anomaly appears at Sample 3000, there is a large spike in the value. However, the MUSUC values have a greater magnitude than the martingale values. Therefore, the MUSUC values are increasing and decreasing at a faster rate. This has an effect on the two properties of the anomaly detection algorithms that we are assessing; the delay time to detecting the anomaly and the false alarm rate. For the "jump up" dataset, at all thresholds the MUSUC algorithm detects the anomaly earlier than the martingale algorithm. In fact, it manages to detect it only 3 samples after the anomaly has occurred, at Sample 3003. While MUSUC improves upon martingales in this aspect, it comes with the drawback of a higher number of false alarms. Even at a high threshold of 10^6, MUSUC still produces a false alarm.

The results for the "jump down" dataset are mostly the same as for the "jump up" dataset, which is to be expected as they are the same data but with the anomaly in a different form. However, there is a greater delay in detecting the anomaly, with the earliest detection from all the experiments being at 3029, 26 samples later than the earliest detection for the "jump up" dataset. This is most likely due to the anomalous values lying between the high periods and low periods in the data which makes it appear more "normal" to the underlying algorithm. In the "jump up" dataset the anomalous values are much higher than even the high periods in the data and so it is clearer they are anomalous right away. Unlike with the "jump up" dataset, the MUSUC algorithm performs worse in all aspects compared to the martingale algorithm. For the lower thresholds of 10^4 and 10^5, the anomaly is detected 2 and 15 samples later for MUSUC. Additionally, it flags a higher number of false alarms with this dataset. At a threshold of 10^5 it detected 4 false alarms, compared to 1 for martingales.

Therefore, with regard to the synthetic datasets it can be concluded that the MUSUC algorithm results in more false alarms. There may be a small benefit in faster detection times with MUSUC. However, this is not true for both datasets and for every threshold and, even when it is true, the difference in detection time can be fairly insignificant and as small as 16 samples. The conclusion is backed up by experiments using the synthetic data with no anomaly. At all thresholds, the false alarm rate was higher for the MUSUC algorithm.

When tested on the real-world energy consumption dataset, again the same conclusions can be made. The clearest anomaly in the dataset is the large drop in energy consumption at around Sample 12500. The MUSUC algorithm tends to detect this anomaly earlier, with the earliest detection at Sample 12409, 21 samples earlier than the martingale algorithm with the same parameters. The difference in false alarms for the two algorithms is much more significant with this data set than the synthetic datasets for lower thresholds. With the synthetic datasets, at the lowest thresholds tested, the difference in false alarms was often only 2-3. In the energy consumption dataset, the difference can be as large as 18. This is evidenced further when the data is randomly shuffled so that there are no anomalies. At a threshold of $10^{0.1}$, 70 false alarms are detected by the MUSUC algorithm compared to 29 by the martingale algorithm.

Overall, from the experiments carried out, the main differences between the two algorithms can be summed up as MUSUC will detect the anomalies earlier, but martingales will produce less false alarms. Balasubramainian et al., 2014 [13, Chapter 4] argues that the biggest limiting factor for an anomaly detection algorithm is its false alarm rate. Therefore, a lower false alarm rate is more important for the algorithm than minor decreases in the delay time to detection. Based on this, in my experiments I would argue that power martingales have been the more successful method for detecting anomalies. For the energy consumption dataset, the best result is seen when using a sliding window of 250 samples and a threshold of 10^{1} with power martingales. Two anomalies are detected and they both appear to be genuine anomalies.

However, a claim of superiority cannot yet be made. The two algorithms have only been tested on three datasets in this study, two of which were synthetic datasets and very similar to each other. Further experiments need to be carried out on real-world datasets to fully test if one algorithm performs consistently better than the other or, perhaps, each algorithm is better suited to certain datasets. Additionally, the problems found in this study with the MUSUC algorithm may not be to do with the general method. Instead it may be related to the function used to calculate the E-values. In my experiments I used a one-class SVM to calculate non-conformity scores. However, the non-conformity scores did not fit the requirements of having a mean of 1. By altering the sequence of non-conformity scores to have a mean of 1, some of the scores became negative. This violates the other property of an E-value in that it must be greater than 0. To solve this, I altered the standard deviation to 0.1 so the values were spread over a smaller range and did not become negative. The standard deviation of the sequence may have an effect on the algorithm's success as potentially a tighter spread of values will work better.

6 Conclusions and Further Work

Conformal prediction as a basis for anomaly detection has been tested on both a multivariate classification dataset in the USPS digit dataset and a number of time series datasets. With the USPS data, the ICAD algorithm was successful at detecting both added anomalies, where the digit had been altered, and anomalies that already existed in the dataset, where digits were written poorly. It also demonstrates the concept of the well-calibrated false alarm rate that is a feature of ICAD, meaning the false-alarm rate will never exceed the chosen significance level. For the time series datasets, the two methods of power martingales and MUSUC were implemented. Both showed success in detecting the anomalies in the synthetic datasets and the real-world Energy Consumption dataset. However, there were differences in the false-alarm rates and the delay time to detection of anomalies for the two algorithms. MUSUC generally had a shorter delay time in detecting anomalies but also a higher false alarm rate. For the energy consumption data, a sliding window for calculating the martingale/MUSUC values improved the results significantly. The larger the sliding window, the more poorly the algorithm performed, as the largest sliding windows tested showed large delays in detecting the anomalies. This modification may not be applicable to all datasets, however. Further testing on more real-world datasets with anomalies is needed for both martingale and MUSUC algorithms to be able to compare them better. Additionally, it may be of interest to test the effect of changing the underlying algorithm used to kNN or other non-conformity measures not described here, such as neural networks or gradient boosting. Finally, to properly assess success of the various conformal anomaly detection in this study, it would be important to test alternative time series anomaly detection methods, not based on conformal prediction, on the same datasets.

7 Professional Issues

As this study deals with methods for anomaly detection, I believe one of the biggest potential ethical issues will be related to false alarms. If the method was used for surveillance or another purpose related to criminal issues, then an individual being detected as anomalous when they are not could cause a lot of issues for that individual. For certain purposes like this, the detected anomalies would need to be assessed thoroughly for false alarms, rather than use an automatic system, so as to avoid causing trouble for the people detected as false positives.

Another potential issue is bias in the dataset causing certain groups to be detected as anomalies, whether it is true or false, more often than other groups. This may be an unintentional feature of the dataset or could be done maliciously. Someone may be able to alter the algorithm or the dataset to classify samples they want as anomalies and use that to further an agenda.

While the other two potential issues I have brought up deal with treating anomalies as important samples, this deals with when anomalies are treated as unimportant samples. In machine learning prediction tasks, it can be beneficial to remove anomalous data samples to improve prediction. However, great care needs to be taken when doing this as it could lead to a certain group/class being underrepresented in the dataset. For example, if a company was trying to predict which applicants should make it to the interview stage based on their CV. If the algorithm used data from previous hires from the company and anomaly detection was used to alter the dataset, then the dataset would become more homogenous. People may end up being rejected/accepted based on other qualities than their qualifications just because they are more different/similar to the previous hires.

It is also possible, if anomaly removal is carried out multiple times, that a positive feedback loop could happen. If a certain group were seen as slightly anomalous the first time round and removed without careful consideration, in the next round of removals that group would be seen as even more anomalous and removed again, leading to extreme underrepresentation.

In my project I did not come across any of these issues myself as the data I was working with was handwritten digits, synthetic data and energy consumption data. The only potential professional issue I came across during my project was that of plagiarism when writing the background research section of this dissertation. To deal with this issue all sources of information were referenced properly.

8 Self-Assessment

Overall, I believe the project went well. For the first month and a half I gained a thorough understanding of the concepts behind conformal prediction and conformal anomaly detection and was able to implement them in Python. Once I progressed onto using martingales and MUSUC for change point detection, the concepts became more complicated and for a short period it was a challenge to fully understand and implement them. However, by the end of this project I believe I have a much stronger understanding of martingales and E-values with MUSUC and their implementation. I produced a range of results for 3 different datasets with these two algorithms, but I think the study would have been improved if I had additional time to test the algorithms on more real-world datasets. During the project I had some difficulty in implementing the plug-in martingale and the simple mixture power martingale. As these were not essential to the project, I moved on from them and only compared the power martingale. With more time, I would aim to implement these in my anomaly detection algorithm so that they could be compared with power martingales and MUSUC. It would also be interesting to test other underlying algorithms as non-conformity measures for the martingale and MUSUC algorithms. While carrying out the project I was well organized and kept my workspace easy to navigate to find the results, code or datasets I needed to use. I also wrote parts of the background research section as I was implementing and reading about the concepts. This helped a lot when I came to write the background research section in this report and is something I would definitely repeat if I was to work on another project. One change I would make is that when writing small reports on the background research, I did not reference as I went along, since they were only for my own use and not to be submitted. This wasted some time when I came to the actual write up as I had to keep searching through papers to find where I had obtained certain information from. In summary, I have really enjoyed working on this project and believe it has benefited my research skills greatly. These skills and this project will aid me in my career as I aim to work in the area of data science and machine learning.

References

[1] Hawkins, D.M., 1980. *Identification of outliers* (Vol. 11). London: Chapman and Hall.

[2] Liu, Y., Liu, C. and Wang, D., 2009. A 1D time-varying median filter for seismic random, spike-like noise elimination. *Geophysics*, *74*(1), pp.V17-V24.

[3] Laxhammar, R., 2014. *Conformal anomaly detection: Detecting abnormal trajectories in surveillance applications* (Doctoral dissertation, University of Skövde).

[4] Ghosh, S. and Reilly, D.L., 1994, January. Credit card fraud detection with a neural network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on* (Vol. 3, pp. 621-630). IEEE.

[5] Anscombe, F.J., 1960. Rejection of outliers. *Technometrics*, *2*(2), pp.123-146.

[6] Liu, H., Shah, S. and Jiang, W., 2004. On-line outlier detection and data cleaning. *Computers & chemical engineering*, *28*(9), pp.1635-1647.

[7] Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), pp.1-58.

[8] Çelik, M., Dadaşer-Çelik, F. and Dokuz, A.Ş., 2011, June. Anomaly detection in temperature data using dbscan algorithm. In *2011 international symposium on innovations in intelligent systems and applications* (pp. 91-95). IEEE.

[9] Kim, J. and Scott, C.D., 2012. Robust kernel density estimation. *The Journal of Machine Learning Research*, *13*(1), pp.2529-2565.

[10] Liu, F.T., Ting, K.M. and Zhou, Z.H., 2008, December. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413-422). IEEE.

[11] Ruff, L., Vandermeulen, R.A., Görnitz, N., Binder, A., Müller, E., Müller, K.R. and Kloft, M., 2019. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

[12] Gammerman, A. and Vovk, V., 2007. Hedging predictions in machine learning. *The Computer Journal*, *50*(2), pp.151-163.

[13] Balasubramanian, V., Ho, S.S. and Vovk, V. eds., 2014. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes.

[14] Vovk, V., Gammerman, A. and Shafer, G., 2005. *Algorithmic learning in a random world*. Springer Science & Business Media.

[15] Vovk, V., 2015. Cross-conformal predictors. *Annals of Mathematics and Artificial Intelligence*, *74*(1-2), pp.9-28.

[16] Papadopoulos, H., Vovk, V. and Gammerman, A., 2007, October. Conformal prediction with neural networks. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)* (Vol. 2, pp. 388-395). IEEE.

[17] Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine learning*, *20*(3), pp.273-297.

[18] Fedorova, V., Gammerman, A., Nouretdinov, I. and Vovk, V., 2012. Plug-in martingales for testing exchangeability on-line. *arXiv preprint arXiv:1204.3251*.

[19] Vovk, V., Nouretdinov, I. and Gammerman, A., 2003. Testing exchangeability on-line. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (pp. 768-775).

[20] Volkhonskiy, D., Burnaev, E., Nouretdinov, I., Gammerman, A. and Vovk, V., 2017, May. Inductive conformal martingales for change-point detection. In *Conformal and Probabilistic Prediction and Applications* (pp. 132-153).

[21] Ho, S.S., 2005, August. A martingale framework for concept change detection in time-varying data streams. In *Proceedings of the 22nd international conference on Machine learning* (pp. 321-327).

[22] Vovk, V., 2020. Conformal e-prediction for change detection. *arXiv preprint arXiv:2006.02329*.

[23] Le Cun, Y., Matan, O., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jacket, L.D. and Baird, H.S., 1990, Handwritten Digits USPS dataset, Available at:< https://www.kaggle.com/bistaumanga/usps-dataset>

[24] Ahmad, S., Lavin, A., Purdy, S. and Agha, Z., 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, pp.134-147. "art_daily_jumpsup.csv", "art_daily_jumpsdown.csv" and "art_daily_small_noise.csv". Available at: <https://github.com/numenta/NAB/tree/master/data>

[25] UK Power Networks, 2015. SmartMeter Energy Consumption Data in London Households. "UKPN-LCL-smartmeter-sample.csv". Available at: <https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households>

Appendix A : My Code

I have presented the code I developed in a series of short Jupyter notebooks. The link to download the Jupyter notebooks is provided below, but the files are also included in my submission. In A.1 I will detail which notebooks correspond to which figures and the dataset files they used. When running the code, dataset files must be in the same directory as the Jupyter notebook.

https://github.com/markpurtle/Conformal-Anomaly-Detection

A.1 Notebooks

USPS Batch SVM Conformal Prediction.ipynb – Figure 7a, USPS dataset

USPS Batch 1NN Conformal Prediction.ipynb – Figure 7b, USPS dataset

USPS Online SVM Conformal Prediction.ipynb - Figures 8-13, USPS dataset

USPS Online ICAD SVM.ipynb – Figure 14, USPS dataset

USPS SVM Martingales.ipynb - Figure 15a and 16, USPS dataset

USPS 1NN Martingales.ipynb - Figure 15b, USPS dataset

NAB Jump Up Anomaly Results.ipynb - Figures 17, 18 and 19, NAB "Jump Up" dataset

NAB Jump Up Martingale Changed Algorithm.ipynb - Figure 20, NAB "Jump Up" dataset

NAB Jump Up MUSUC Changed Algorithm.ipynb - Figure 21, NAB "Jump Up" dataset

NAB Jump Down Martingale.ipynb – Figure 22, NAB "Jump Down" dataset

NAB Jump Down MUSUC.ipynb – Figure 23, NAB "Jump Down" dataset

NAB No Anomaly.ipynb - Figures 26 and 27, NAB "Small Noise" No Anomaly dataset

Energy Consumption Martingale.ipynb - Figure 28a, Energy Consumption dataset

Energy Consumption MUSUC.ipynb - Figure 28b, Energy Consumption dataset

Energy Martingales Sliding Window.ipynb - Figures 29 and 31, Energy Consumption dataset

Energy MUSUC Sliding Window.ipynb - Figures 30 and 32, Energy Consumption dataset

Energy Consumption Martingale Modification.ipynb – Figure 33, Energy Consumption dataset

Energy Consumption Random Shuffle.ipynb - Figures 34, 35, 36, Energy Consumption dataset

Appendix B : Microsoft Azure Report

Midway through the project I wrote a short report comparing my anomaly detection algorithm at the time with the Microsoft Azure time series anomaly detection module using the energy consumption data. The Azure algorithm is also based on conformal martingales but has a number of differences and limitations that are discussed in the report. Some of the results in this report are slightly different to the result in my dissertation. This is due to the algorithm changing by the end of the project. The report includes a small section on how the Microsoft Azure algorithm works and how to run it in the Azure Machine Learning Studio, as well as a discussion on the differences between it and my algorithm. It is included in the GitHub link with the Jupyter notebooks: https://github.com/markpurtle/Conformal-Anomaly-Detection