Stockholms universitet

Feature Construction of Multi-sets for Machine Learning Application

Sheila Farrahi

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2024:2
Matematisk statistik
Januari 2024

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Master Thesis **2024:2**
http://www.math.su.se

Feature Construction of Multi-sets for Machine Learning Application

Sheila Farrahi*

February 2024

Abstract

Many machine learning algorithms require inputs in the form of fixed length vectors and cannot directly process data in the form of multi-sets. In reality, raw data does not always exist in the form of fixed length vectors and can exist in the form of unordered multi-sets of scalar values, where the number of elements in each multi-set can often vary across multiple data instances. Feature construction is a technique to find a better data representation for the machine learning algorithms in case the original representation of data is not in the form of fixed length vectors.

Statistical measures and density estimation provide insights into the data, therefore they can be used as tools in constructing fixed length vectors of features from unordered multi-sets. In this report, several methods based on statistical measures and density estimation are explored for feature constructions from unordered multi-sets.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: sh.farrahi@gmail.com. Supervisor: Chun-Biu Li.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Chun-Biu Li and Ciwan Ceylan, for their invaluable guidance, support, and feedback throughout this process.

I am thankful to SEB for giving me the opportunity to carry out this thesis and for providing resources.

Last but not least, many thanks to my family and friends, and especially my partner for their love and support during this process.

Contents

1 Introduction

In the real world, raw data can exist in the form of unordered multi-sets of scalar values, where the number of elements in each multi-set can often vary across multiple data instances. Imagine a restaurant that has an average rate of 3.5 based on 1100 reviews on Google. In this example, the restaurant's review rate is an average of a collection of 1100 elements (individual rates) where these elements take value from 1 to 5 stars such as $\{3, 4, 4, \ldots, 5, 4, 2\}$. This collection of rates is an unordered multi-set, meaning that the order of the elements is not important. Also, there could be multiple instances of each element. Another example is all the incoming and outgoing transaction amounts made to and from a bank account, given a time window. The incoming and outgoing transactions form multi-sets that jointly describe the pattern of the account's income and spending pattern. For instance, the incoming multi-set could be as $\{1000, 15000, 5000, 400\}$, showing that this account has received money from 4 different accounts in the specific time window. This type of data is not unique to bank accounts but rather typical of data flows taking place on a network where each node has a varying number of neighbors with which it can make transfers or exchange information. Other examples include email networks, telecommunication networks, and data transfer networks.

Supervised learning algorithms cannot handle multi-set data as inputs. Algorithms such as support vector machines or decision trees require fixed sized vectors as inputs, with a fixed dimension for all data points [9]. Some methods circumvent this restriction by either padding each multi-set with dummy values or by sorting the multi-sets and treating them as sequences, for example, padding in natural language processing [22]. However, these methods might not be satisfying as they could introduce data or structures that are not inherent to the multi-sets. Therefore, there is a need to represent unordered multi-sets into fixed dimension vector representations.

Feature construction is a technique to manually find a better data representation to find predictive features for machine learning algorithms since the original representation might not be viable [4]. For example, data that comes in the form of unordered multi-sets can be better represented for designing a restaurant recommendation system based on Google reviews or comparing the distribution of incoming and outgoing transactions of two customers.

The goal of the feature construction methods is to represent $A = \{a_1, \ldots, a_p\}$, an unordered multi-set with p scalar elements, by $Z = [z_1, \ldots, z_m]$, a vector of m scalar features/dimension where m is a fixed number. Vector Z contains the constructed features and can be used in machine learning algorithms. For instance, in the restaurant review rate example, the input data of $A = \{3, 4, 4, \ldots, 5, 4, 2\}$ with $p = 1100$ can be represented by $[3.5, 1.25]$ with $m = 2$ which describes the first two moments of the rates given by 1100 reviewers. The size of A can be much larger compared to the dimension of Z; the information of A is being summarized. In this report, it is chosen to construct features from the information extracted from the distribution of the input data inspired by prior research [3].

One of the most widely used techniques for representing data distributions is using statistical measures, such as mean, variance, etc. Moments provide valuable information about the shape and characteristics of probability distributions of the data. The idea behind using this method is to estimate several moments of the data and use them as features.

likewise, density estimation constructs an estimate of the density function from the observed multi-set of data points [18]. Therefore, a very natural use of density estimation is in the investigation of the properties of a given set of data [18]. Density estimation provides insights into the data; hence,

it can be used as a tool in feature construction. The goal of this report is to explore different methods of density estimation for constructing fixed length vector representations from multi-sets.

A common approach in estimating density is kernel density estimation [18]. This method involves placing a smooth kernel function on each data point and summing these kernels to obtain a smoothed density estimate. To construct features using this method, the support of the estimated density function is split into equal increments. This split results in several points that are upper and lower bounds of the increments. Then, use the estimated density function at these points as features.

On the other hand, the empirical cumulative distribution function can provide useful information about the underlying distribution of the data. It can estimate the percentiles of the distribution, which describe central tendency, the spread of the distribution, the existence of outliers, etc [1]. The idea behind using this method is to use percentiles as features.

Another approach used in this report to construct features from multi-sets is the empirical characteristic function [3]. This method is motivated by the fact that the characteristic function uniquely determines the distribution function so that recognizing the characteristic function of a random variable identifies its distribution function [12]. The empirical characteristic function provides an alternative way compared to working directly with probability density functions or cumulative distribution functions.

To determine the strengths and weaknesses of the different approaches described above, the methods are empirically evaluated. This report performs such an evaluation in the following way. The features are utilized for training a classifier. This enables us to observe how effectively the constructed features capture the characteristics of the data.

Logistic regression (LR) and support vector machine (SVM) are two of the most commonly used methods for classification [17]. They are used in this report for the empirical evaluation used for different methods of feature construction. LR is a linear classifier, while SVM exists both in linear and non-linear variants.

The accuracy of the classifiers indicates that with a reasonable sample size and few features, all of the density estimation methods mentioned earlier can be utilized to successfully represent an unordered multi-set by a vector of scalar features.

This report is structured as follows: In section 2, some background on LR and SVM is provided. Section 3 goes through the details of the different statistical measures and density estimation methods and how to utilize them to construct features from unordered multi-sets. Section 4 covers the description of the data-sets used in this study, along with the results of the evaluation on how classifiers that were trained with the constructed features perform. Finally, in section 5 the conclusion and the future work on this topic are discussed.

2 Background

2.1 Classification Methods

In order to empirically compare the features constructed by the statistical measures and the density estimation methods, several synthetically generated supervised classification tasks are used. The performance of the classifiers will indicate the quality of the constructed features for classification purposes.

Classification is a supervised learning task that aims to predict the categorical response variable (class) based on predictor variables [9]. To predict the class of given data points, classification maps a function from input variables X to output variables Y as response variable [17].

For the empirical evaluation, LR and SVM are used. These are two of the most commonly used methods for classification [17]. Logistic regression is a linear classifier while SVM exists both in linear and non-linear variants depending on the used kernel. Comparing the two methods, the SVM can be both more flexible and more robust than LR, where the flexibility comes from the used kernel, and the robustness from its maximal margin loss function. Logistic regression works well on linearly separable data, in such case, SVM with kernel trick might not perform well as the model can end up being complex. Also, it is computationally more efficient compared to SVM especially if the data-set is large.

In this report, both classifiers are employed, as each of them possesses strengths and weaknesses that complement one another. Therefore, the utilization of both classifiers can provide a better understanding of how each density estimation method operates on feature construction.

The mathematical formulation of LR and SVM in the following sections are followed closely from the book, The Elements of Statistical Learning [9]. The figures of this chapter are adapted and modified from the books, The Elements of Statistical Learning [9] and An Introduction to Statistical Learning [11].

2.1.1 Logistic Regression

Logistic regression aims to find the probability of a categorical response variable belonging to a certain class based on the predictor variables and therefore is suitable to be used in classification problems. Consider X a set of n input vectors with each vector consisting of p scalar observations. X is a $n \times p$ matrix with x_i^T representing its i^{th} row.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \quad x_i^T = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix}.$$

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}.$$

Consider Y a vector of binary response variables with $y_i \in \{0, 1\}$ denoting the categorical outcome for the i^{th} observation.

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

.

Logistic regression models the log odds of each possible outcome, y_i, by a linear combination of the predictor variables, x_i. This is defined in Equation (1) where β_0 is the intercept, β is the vector of coefficients, and $\frac{p(y_i=1|x_i)}{p(y_i=0|x_i)}$ is the odds of outcome $y_i = 1$. In other words, $\log(odds)$ is the probability of occurring a possible outcome divided by the probability of not occurring that outcome.

$$\log\left(\frac{p(y_i = 1|x_i)}{p(y_i = 0|x_i)}\right) = \beta_0 + \beta x_i^T. \tag{1}$$

When using logistic regression for classification purposes, we are interested in finding a probability rather than log odds. For this purpose, Equation (1) can be transformed to Equation (2) as

$$p(y_i = 1|x_i) = \frac{1}{1 + e^{-(\beta_0 + \beta x_i^T)}}. \tag{2}$$

As Equation (2) illustrates, the probability of outcome $y_i = 1$ is not a linear function of x_i. The logit function or $\log(odds)$ allows us to move from modeling the probability with the logistic function (non-linear curve) to modeling the logit with a linear function of predictors.

The parameters of the logistic regression model are estimated by maximum likelihood estimation (MLE) [9]. This is achieved by maximizing the likelihood function of β_0 and β. The likelihood function is defined as

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(y_i = 1|x_i)^{y_i} \prod_{i=1}^{n} (1 - p(y_i = 1|x_i))^{1-y_i}.$$

Hence the log likelihood function is derived as

$$l(\beta_0, \beta) = \sum_{i=1}^{n} y_i \log(p(y_i = 1|x_i) + (1 - y_i) \log(1 - p(y_i = 1|x_i)).$$

The MLE estimates the parameters that maximize the log likelihood function, hence the cost function is

$$J(\beta_0, \beta) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))\right). \tag{3}$$

In this report `scikit-learn` library is used which is an open-source machine learning library for the Python programming language. For LR, the model `LogisticRegression()` from the `linear_model` module is used.

2.1.2 Support Vector Machine

Support Vector Machines (SVM) is a powerful machine learning algorithm for both classification and regression tasks. SVM divides the p-dimensional space into different classes using a separating hyperplane.

Consider X a set of n input vectors with each vector consisting of p scalar observations, same as in Section 2.1.1. Consider Y a vector of binary response variables with $y_i \in \{-1, 1\}$ denoting the class of the i^{th} observation.

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

The separating hyperplane is a plane that is the farthest from x_i for all i and is defined as $f(x_i) = \beta_0 + \beta x_i^T = 0$ where β_0 is the intercept and β is a vector of coefficients. The separating hyperplane classifies the data using the decision rule between classes. The decision rule is defined by

$$G(x_i) = sign(\beta_0 + \beta x_i^T) = \begin{cases} 1 & \text{if} \quad \beta_0 + \beta x_i^T > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Consider M, the distance of the closest data point to the separating hyperplane. For such a point x_i, the definition of M is

$$M = \frac{|\beta_0 + \beta x_i^T|}{\|\beta\|} = \frac{1}{\|\beta\|}. \tag{4}$$

The goal of SVM is to find a separating hyperplane that maximizes M while minimizing the classification error. The optimization problem is define as

$$\begin{aligned} &\max_{\beta_0, \beta, \|\beta\|=1} M \\ &\text{subject to } y_i(\beta_0 + \beta x_i^T) \geq M, \quad i = 1, \ldots, n, \end{aligned} \tag{5}$$

where β is the vector of coefficients of the separating hyperplane. The constraint $\|\beta\| = 1$, normalizes the vector of coefficients. This does not change the direction of the separating hyperplane, but helps with simplifying the optimization by reducing redundant degrees of freedom. The constraint $y_i(\beta_0 + \beta x_i^T) \geq M$ ensures each data point is on the correct side of the separating hyperplane, with at least a distance of M. Using the Equation (4), we can be rewrite Equation (5) as a minimization optimization problem as

$$\begin{aligned} &\min_{\beta_0, \beta} \|\beta\| \\ &\text{subject to } y_i(\beta_0 + x_i^T \beta) \geq 1, \quad i = 1, \ldots, n. \end{aligned} \tag{6}$$

Figure 1, visualizes an example where data is presented in two classes, orange and blue. The black line is the separating hyperplane between the two classes that is derived from solving an

optimization problem that maximizes M. The distance between the dashed lines and the separating hyperplane is M. The space between these dashed lines is called margin. The data points that are located inside or on the wrong side of the margin (marked by circles with black outline) are called support vectors.

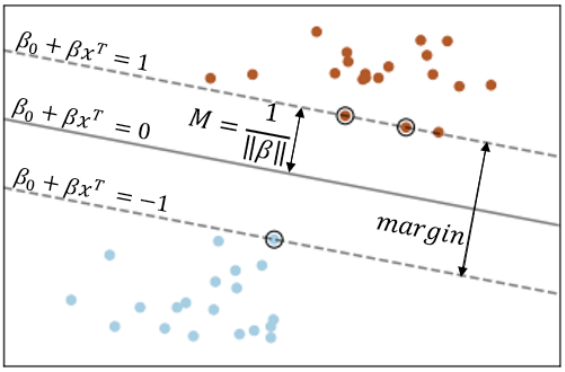


Figure 1: The black line represents the separating hyperplane between the orange and the blue class, this line is the farthest from all data points. The margin is the space between the dashed lines which has width of $2M$. The data points marked by circles with black outline are the support vectors.

SVM employs a soft margin approach, which allows a tolerance for misclassification. This approach enables SVM to classify non linearly separable data, such as in Figure 2. This tolerance is defined by slack variables, $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$, that allow individual data points to be on the wrong side of the decision boundaries or even on the wrong side of the separating hyperplane. The value of ξ_i is proportional to the amount that x_i is on the wrong side of the margin.

$$\begin{cases} \xi_i = 0 \text{ for } x_i \text{ on the correct side of the margin} \\ \xi_i > 0 \text{ for } x_i \text{ otherwise.} \end{cases}$$

In Figure 2, ξ_i which refers to the slack variable for the point x_i is illustrated. As you can see the point x_1 is classified correctly (on the correct side on the separating hyperplane) but it is inside the margin, hence the value of ξ_1 is greater than 0. Another example is the point x_2 which lies on the wrong side of the separating hyperplane, hence the value of ξ_2 is also greater than 0. The point x_3 is classified correctly and is inside the margin, hence the value of ξ_3 is 0.

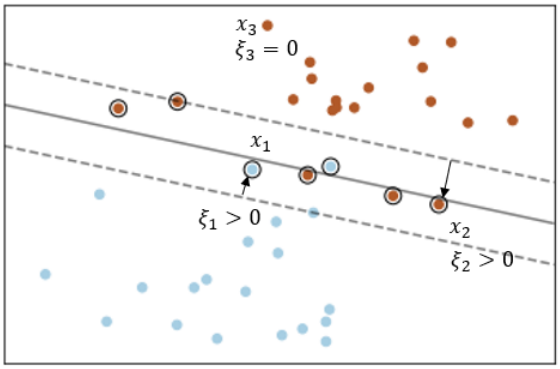


Figure 2: A non linearly separable case: The data points that are inside or on the wrong side of the margin are marked by circle with black outline. These data point have $\xi_i > 0$ while for the rest of the points $\xi_i = 0$.

Considering the tolerance for misclassification, the constraint in Equation (6) modifies from $y_i(\beta_0 + x_i^T\beta) \geq 1$ to $y_i(\beta_0 + x_i^T\beta) \geq 1 - \xi_i$. Adding an upper bound on $\sum_{i=1}^{n} \xi_i$ can control how much misclassification is allowed. Hence, the optimization problem changes to

$$\begin{aligned} &\min_{\beta_0,\beta} \|\beta\| \\ &\text{subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i \\ \xi_i \geq 0 \quad \sum \xi_i \leq \text{constant}. \end{cases} \end{aligned} \tag{7}$$

The value of ξ_i is positive for the data points that on the wrong side of the margin. Equation (7) implies that only the data points that are either on the margin or on the wrong side of the margin affect the hyperplane which makes this method robust to outliers as the points that are far away from the hyperplane cannot affect the hyperplane. Putting an upper bound on the sum of slack variables can control how much misclassification is allowed.

Minimizing $\|\beta\|$ is the same as minimizing $\|\beta\|^2$, and multiplying $\|\beta\|$ by constant value of $\frac{1}{2}$ does not change the optimization problem. To make this optimization problem computationally convenient, Equation (7) can be re-written in the form of below where C is the constant in Equation (7) [9].

$$\begin{aligned} &\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i \\ \xi_i \geq 0. \end{cases} \end{aligned} \tag{8}$$

Equation (8) can be re-written in a different form. Take $C = \frac{1}{\lambda}$, dividing the objective function

of Equation (8) by C we get

$$\min_{\beta_0,\beta} \frac{\lambda}{2}\|\beta\|^2 + \sum_{i=1}^{n} \xi_i. \tag{9}$$

Since ξ_i can only take positive values, the constraint $y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$ from Equation (8) can be re-written as $[1 - y_i(\beta_0 + \beta x_i^T)]_+ \leq \xi_i$. This yields $\sum_{i=1}^{n}[1 - y_i(x_i^T\beta + \beta_0)]_+ \leq \sum_{i=1}^{n} \xi_i$. By substituting $\sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n}(1 - y_i(x_i^T\beta + \beta_0))$ in Equation (9), the optimization problem can be written as below which is in the form of hinge loss and a regularization penalty [9].

$$\min \sum_{i=1}^{n}[1 - y_i(x_i^T\beta + \beta_0)]_+ + \frac{\lambda}{2}\|\beta\|^2. \tag{10}$$

Hinge loss penalizes the model for misclassifying data points. Hinge loss incorporate the margin into the loss calculation and is defined as [9]

$$\text{Hinge loss} = [1 - y_i(\beta_0 + \beta x_i^T)]_+.$$

When a data point is correctly classified, $y_i(\beta_0 + \beta x_i^T) \geq 1$ hence $[1 - y_i(\beta_0 + \beta x_i^T)]_+ = 0$. This implies that there is no penalty in case of correct classification. On the other hand, $y_i(\beta_0 + \beta x_i^T) \leq 0$ for the misclassified data point. This implies $[1 - y_i(\beta_0 + \beta x_i^T)]_+ \geq 0$ and the loss corresponds to how far the prediction is from the separating hyperplane. To summarize, in SVM only the support vectors affect the separating hyperplane and not any other data points.

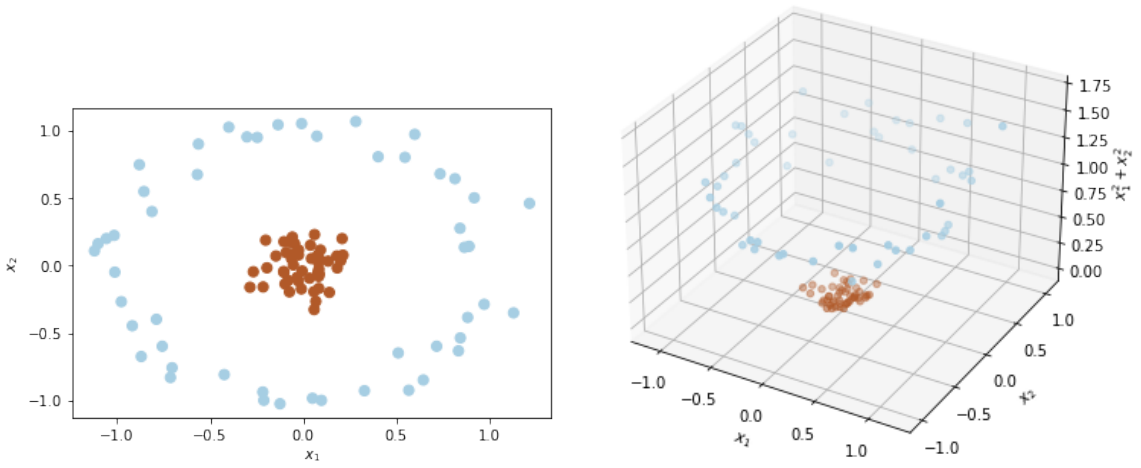
The cost function is define as

$$J(\beta_0, \beta) = \frac{\lambda}{2}\|\beta\|^2 + \sum_{i=1}^{n}[1 - y_i(\beta_0 + \beta x_i^T)]_+. \tag{11}$$

The kernel trick

The use of kernel gives SVM the ability to perform well in scenarios where the data is not linearly separable. This is useful as the features constructed by different density estimation methods might not be linearly separable. The kernel trick enables SVM to effectively separate classes by enlarging the feature space using various kernel function, such as polynomial, radial basis, sigmoidal, etc.

Figure 3(a) illustrates a scenario in 2D where the two classes cannot be separated by any linear hyperplane. Kernel enlarges the feature space by generating a new dimension which is a non-linear combination of the original dimensions. Figure 3(b) illustrates the same data with an additional dimension. This new dimension is $X_1^2 + X_2^2$ which is a non-linear combination of existing dimensions(X_1 and X_2). The kernel function transforms the data from its original space to a higher dimensional feature space, enabling the data to be separated by a linear hyperplane in the new higher dimension.



(a) A non-linearly separable data-set

(b) Kernel trick enlarges the feature space

Figure 3: How kernel tricks works: Kernel takes the non-linearly separable data and transform in to a higher dimension. The additional dimension is a non-linear combination of existing dimensions ($X_1^2 + X_2^2$). In this higher dimension data is linearly separable.

Figure 4 visualizes the non-linear separating hyperplane (the black line) in the original 2D space.

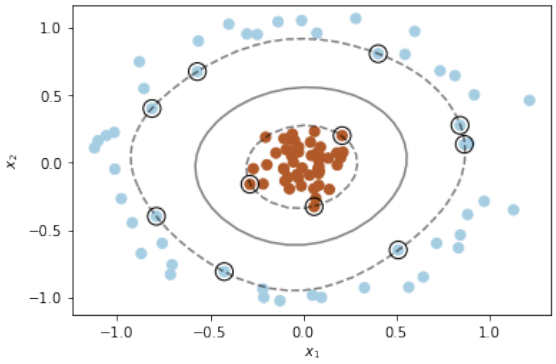


Figure 4: SVM with non-linear separating hyperplane: The black line represents the separating hyperplane between the orange and the blue class. The space between the dashed lines is the margin and the data points marked by circles with black otline are the support vectors.

In this report, the radial basis kernel which is one of the popular kernels is used [9]. Radial basis kernel for two points x and x' is defined as

$$K(x, x') = exp(-\gamma \|x - x'\|^2).$$

Kernel function compares the pairwise similarity between the point in both the original space and the transformed space to make sure that the data points that are similar in the original space will be similar in the transformed space as well. The kernel function computes the similarity between the points using the Euclidean distance, $\|x - x'\|$, between them. If the points are very close to each other, the value of the kernel function gets close to 1, the further the points are from each other, the closer the value of the kernel function to 0. The parameter γ controls which points should be considered similar to each other based on the distance between them. It describes how far the influence of each training point reaches. Low values of γ mean every point has far reach, this means that even far away points affect the decision boundary. If the value of γ is too large, every training point has a low reach, and the radius of influence gets smaller, therefore the decision boundary is only influenced by the points that are very close to it and ignores the points that are far away.

The solution function to the separating hyper parameter is defined as [9]

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i K(x, x'), \tag{12}$$

where α_i is the Lagrange multiplier vector and take values between 0 and C ($0 < \alpha_i < C$). The value of α_i is non zero only for the support vectors and zero for the other data points [11] which implies that only the support vectors affect the separating hyperplane.

To apply SVM in this report, the model `SVC()` from the `svm` module in `scikit-learn` library is used.

2.2 Binary versus multi-class classification

Both LR and SVM models are designed for binary classifications. However, they could be extended to more general cases with multiple classes k where $k > 2$. There are two approaches, one versus one and one versus all.

- One versus one: This approach breaks the k class problem into $\binom{k}{2}$ binary class cases. Each classification model is trained to classify data points between two classes. When classifying a new data point, all $\binom{k}{2}$ models are applied to that point. Each model predicts one class for the data point, the final prediction is the class that was predicted by the majority of binary classification models.

- One versus all: This approach compares one class versus the rest of the data-set and constructs k binary class cases. Each classification model is trained to classify data points for one of the classes among the rest. When classifying a new data point, k models are applied to that point. Each model predicts a class for the data point, the final prediction is the class that was predicted by the majority of binary classification models.

In this report, the one versus all approach is used as it is computationally more efficient compared to the one versus one approach. This approach requires to train k binary models for a k class problem, while one versus one requires to train $\binom{k}{2}$ binary models.

2.3 Regularization

To develop a machine learning model, the first step is to gather a data-set. This data-set should be divided into training and test subsets. The training subset is used to train the model, using this data the model learns the patterns and the relationship between the response variables (classes in this case) and the predictor variables (the constructed features). To observe the performance of the model on new data, the data that the model has not been exposed to, the model is tested by the test subset.

The goal is to build a generalized machine learning model so that the model performs well on the data it has not been trained on. The optimal model should have a low error rate on the test data-set when predicting the response variable on a new observation.

Over-fitting happens when a model is too complex or fits the training data very well, meaning that the model has a very low error rate on the training data while a high error rate on the test data. Such a model also fits the noise in the data-set it was trained on and fails to generalize any new data that it is given.

For instance, Figure 5 illustrates an example of an over-fit model and a generalized model. The orange line represents an over-fitted model, as you can see this model fits the training data very well and has a lower bias but the model has higher variability compared to the regularized model (the blue line). The over-fit model is likely to have a higher error on any new data point that it has not seen before. On the other hand, the generalized model would have a lower prediction error on any new data while higher bias on the data that it was trained on.

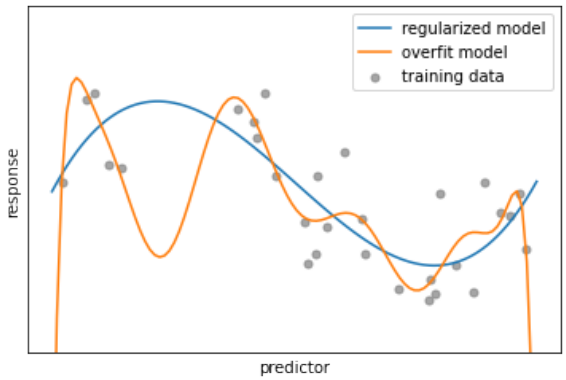


Figure 5: An over-fit versus regularized model: The over-fit model, represented by the orange line, fits the training data very well and has low bias and high variability. On the other hand, the regularized model, indicated by the blue line, has a higher bias on the training data. Nevertheless, it performs significantly better on data it has not been trained on.

When a model is too simple and under-fit and has very few parameters, the bias of the model is high and the variance is low. By increasing the model complexity the bias of the model decreases and its variance increases. An optimal model should have a balance between bias and variance. This is illustrated in Figure 6.

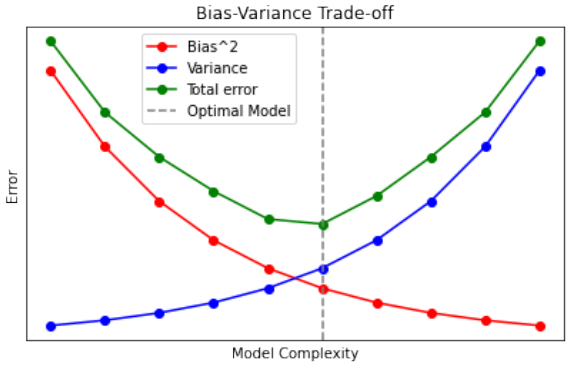


Figure 6: Bias-Variance trade off: An optimal model should have a balance between bias and variance. An under-fit model usually has high bias and low variance while an over-fit model usually has low bias and high variance.

Let Y be the response variable that we are aiming to predict, X representing the features, and $\hat{f}(X)$ be the predicted response variable given X. Bias-variance trade-off is a property of statistical learning methods that shows that the expected squared test error for a given X can be decomposed into the sum of the variance of $\hat{f}(X)$, squared bias of y and the variance of the error terms ϵ [9] as

$$E[Y - \hat{f}(X)]^2 = var(\hat{f}(X)) + [bias(\hat{f}(X))]^2 + var(\epsilon). \tag{13}$$

Equation (13) indicates that in order to minimize the expected test error we need to find a model that simultaneously achieve low variance and low bias [11].

Regularization is a method that facilitates finding a model that minimizes the prediction error by avoiding over-fitting. One of the common techniques for regularization is Ridge regression or $L2$ regularization which regularizes models by adding a penalty term into the model's cost function. By adding a small bias the estimated coefficients shrink and the model will be slightly a worse fit, however, it gives a better prediction.

In LR a small bias (shrinkage penalty) is added to the cost functions of LR in Equation (3) and the cost function changes to

$$J(\beta_0, \beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))) + \lambda \sum_{j=1}^{p} \|\beta_j\|^2. \tag{14}$$

The shrinkage penalty is $\lambda \sum_{j=1}^{p} \|\beta_j\|^2$, where λ is a regularization parameter. By adding the shrinkage penalty to the cost functions of LR, the model will have smaller coefficients which help reduce the risk of over-fitting and help the model to be more generalized. As the value of λ increases, the coefficients get smaller and smaller and approach to 0 as λ goes to ∞ [11].

In SVM, $\frac{\lambda}{2}\|\beta\|^2$ in Equation (10) is the shrinkage penalty term. The constant value $\lambda = \frac{1}{C}$ is the regularization parameter that controls the bias-variance trade-off [11]. The large values of C result in a wide margin that allows for more misclassification, while the smaller values of C result in a narrow margin that rarely allows for misclassification [11].

3 Method

Density estimation and summary statistics provide valuable insights about data, hence, they can be used to extract properties of data and construct features from those properties. The goal of feature construction is to represents A, an unordered multi-set with p scalar elements, by Z, a vector of m scalar features. In this report, four different methods are used to estimate the underlying density of given data-sets.

The choice of m significantly impacts the quality of the constructed features. Higher number of features can capture the underlying characteristics of the multi-set better, while less features might not be able to describe the characteristics of the multi-set well. Therefore, in this report features are constructed using a range of different values of m to explore the impact of m on the quality of the constructed features.

3.1 Moments Approach

Moments are a measure of the shape and variability of a distribution and therefore could be used to determine the characteristics of the distribution. Each moment provides different information about the distribution. However, the first two moments are meaningful numerical descriptive measures [20].

Let X be a continuous random variable with mean μ and finite moments, then the distribution of X is determined by the sequence of all its moments [15]. The m^{th} central moment of X is defined as [20]

$$E\left[(X-\mu)^m\right] = \int_{-\infty}^{\infty} (x-\mu)^m f(x)dx, \tag{15}$$

where $f(x)$ is the probability density function of X. Let $A = \{a_1, \ldots, a_p\}$ be a multi-set with p real valued elements. The m^{th} central moments of A is defined as

$$E\left[(A-\bar{a})^m\right] = \frac{1}{p}\sum_{i=1}^{p}(a_i-\bar{a})^m. \tag{16}$$

Equation (15) defines a general definition of the central moment for a population, where μ refers to the population mean. Whereas Equation (16) defines the estimator of the central moment for a sample, using the sample mean $\bar{a}$.

To represent an unordered multi-set by a vector of m features employing moments, m moments are calculated. The vector of features is calculated as

$$\text{Vector of features} = \left[\bar{a}, E[(a-\bar{a})^2], \ldots, E[(a-\bar{a})^m]\right].$$

This method is simple, however, one of the limitations of this method is that moments do not always exist [15]. The existence of moments depends on the properties of the distribution of data, for instance, the Cauchy distribution does not have defined moments as the integral in Equation (15) does not exist. Another limitation of this method is that larger samples are required for estimating higher moments. For the existence of m^{th} moments the number of observations must be greater or equal to m [8]. Therefore, the multi-set that is aimed to be represented by moments should have enough data points.

3.2 Kernel Density Estimation

The probability density function (PDF) denotes by $f(x)$, describes the probability density of a continuous random variable X. Kernel density estimation (KDE), is one of the well-known approaches to estimate the underlying probability density function of a data-set [5]. KDE estimates the unknown probability density function with some given data points using a kernel function K.

Let $A = \{a_1, \ldots, a_p\}$ for all be a multi-set with p real valued elements. Silverman [18] defines kernel density estimation of A at any given point a_i as

$$\hat{f}(a) = \frac{1}{ph} \sum_{i=1}^{p} K\left(\frac{a - a_i}{h}\right),$$

where h is the bandwidth of the kernel that controls the amount of smoothing. KDE puts a kernel on every data point and then sums them up to obtain density estimation. The kernel function must satisfy $\int_{-\infty}^{\infty} K(u)du = 1$ to ensure that $\hat{f}(a)$ represents a normalized PDF.

There are different choices of kernels to select from, such as Gaussian, Triangular, etc [18]. In this report Gaussian kernel is used. While most kernels perform well, the choice of bandwidth plays a crucial role in the result which could be a drawback of KDE. A very small value of h in the estimation of density function can introduce a spurious structure, whereas a very large value of h may obscure the structure [18].

There are different methods for selecting the suitable bandwidth, and Silverman's Rule of Thumb is applied in this report which is based on minimizing the mean integrated squared error between the actual and the estimated density function. Silverman's Rule of Thumb suggests that the bandwidth should be proportional to the standard deviation of the data and inversely proportional to the fifth root of the number of data points as [18]

$$h = 1.06\hat{\sigma}n^{-\frac{1}{5}},$$

where $\hat{\sigma}$ is the standard deviation of the data.

To represent an unordered multi-set $A = \{a_1, \ldots, a_p\}$ by a vector of features employing KDE, the process is as follows:

1. The density of A is estimated using kernel density estimation. This is achieved by employing a Gaussian kernel with a bandwidth that is selected using Silverman's Rule of Thumb.

2. The support of $\hat{f}(a)$ is estimated as follows:

   (a) If the observations from the unordered multi-set A belong to a bounded interval, that interval is used as support. For example, the restaurant reviews on Google have rates between 1 and 5. Otherwise, the support is estimated as the interval where 95% of the observations exist where this interval is obtained as below:

      i. If only the lower bound exists, the 95^{th} percentile of the observed data is considered as the upper bound.

      ii. if only the upper bound exists, the 5^{th} percentile of the observed data is considered as the lower bound.

iii. In the absence of both lower and upper bound, the 2.5^{th} and 97.5^{th} percentiles of the observed data are used.

3. To construct m features, m equally spaced points denoted as $w_1, \ldots, w_m$ are selected from the support of the estimated density function, $\hat{f}(a)$. Here w_1 is the infimum, and w_m is the supremum of the support.

4. The value of estimated density function at points $w_1, \ldots, w_m$ are the features. The vector of features is calculated as

$$\text{Vector of features} = \left[\hat{f}(w_1), \hat{f}(w_2), \ldots, \hat{f}(w_m)\right].$$

As illustrated in Figure 7, the blue line represents the estimated density function derived from an unordered multi-set. In order to construct $m = 5$ features, five equally spaced points, highlighted in red, are chosen on the support of the estimated density function. The value of $\hat{f}(w_i)$ yields a set of m features.

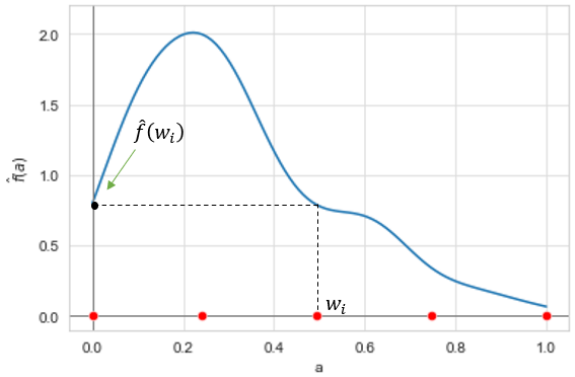


Figure 7: Feature construction using KDE: The blue line represents the estimated density function of a multi-set A by KDE. In order to construct m feature, m equally spaced points, denoted by red color, are chosen on the support of the estimated density function.

3.3 Empirical Cumulative Distribution Function

Cumulative Distribution Function (CDF), provides valuable information about the behavior and summary statistic of the data. For a continuous random variable X, the CDF accumulate all of the probabilities of the values of X up to and including a given value x which is defined as [16]

$$F(x) = \int_{-\infty}^{x} f(x)dx = P(X \leq x). \tag{17}$$

Empirical Cumulative Distribution Function (ECDF), is a non-parametric estimator to estimate CDF, it directly uses the observed data to estimate the cumulative probabilities.

Let $A = \{a_1, \ldots, a_p\}$ be a multi-set with p real values elements. The ECDF is a stepwise function that, at every data point a_i, jumps up by a magnitude of $P(a_i) = \frac{1}{p}$. The ECDF function is defined as [19]

$$\hat{F}(a) = \frac{1}{p} \sum_{i=1}^{p} \mathbb{1}_{\{a_i \leq a\}}.$$

In order to represent an unordered multi-set $A = \{a_1, \ldots, a_p\}$ by a vector of features employing ECDF the process id as follows:

1. The cumulative distribution function of A is estimated.

2. To construct m features, m equally spaced points denoted as $v_1, \ldots, v_m$ are chosen on the interval $(0, 1)$ that is the range of the estimated cumulative distribution function, $\hat{F}(a)$.

   The reason behind choosing an open interval instead of a closed one is the quality of the constructed features. By choosing the closed interval, the value of points v_1 and v_m would be equal to 0 and 1. Hence, the features constructed might not be insightful in terms of being used in classification as $\lim_{a \to -\infty} \hat{F}(a) = 0$ and $\lim_{a \to \infty} \hat{F}(a) = 1$. For instance, if there exists a data-set consists of 100 multi-sets, where each multi-set represents review ratings belong to one restaurant, and the rates are bounded between 1 and 5. Choosing $v_1 = 0$ and $v_m = 1$ results in the features related to these points to have the values of 1 and 5 for all multi-sets, which is not very insightful.

3. The values of $\hat{F}^{-1}(v_i)$ represent the features. The vector of features is calculated as

$$\text{Vector of features} = \left[\hat{F}^{-1}(v_1), \hat{F}^{-1}(v_2), \ldots, \hat{F}^{-1}(v_m)\right].$$

The steps above are illustrated in Figure 8, the blue line represents the estimated cumulative distribution function derived from an unordered multi-set A. The red points are equally spaced points on the range of $\hat{F}(a)$.

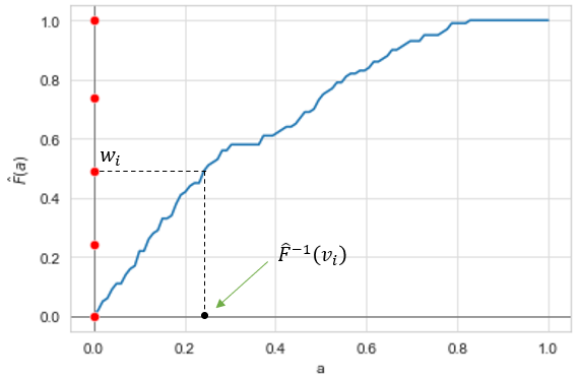


Figure 8: Feature construction using ECDF: The blue line represents the estimated cumulative distribution function derived from an unordered multi-set A. In this example there are five equally spaced points, $v_1, \ldots, v_5$, on the range of A that are highlighted in red. The black point $\hat{F}^{-1}(v_i)$ denotes the feature constructed by the point v_i

Using ECDF for feature construction has some advantages over KDE. In this methods the base for constructing features are the equally spaced points on the range of $\hat{F}(a)$. This range is fixed and always between 0 and 1 by Equation(17). However the support in the KDE method needs to be estimated.

3.4 Empirical Characteristic Function

Characteristic function (CF) for a random variable X is a Fourier transform of the probability density function of a random variable. The CF is defined as [6]

$$\varphi(t) = E[e^{itX}] = \int e^{itX} f(x)dx \quad \forall t \in \mathbb{R} \quad \text{and} \quad i = \sqrt{-1}, \tag{18}$$

where $e^{itX} = \cos(tX) + i\sin(tX)$ and the variable t is the frequency in Fourier transform. Different values of t result in obtaining insights into different aspects of the distribution.

Consider $A = \{a_1, \ldots, a_p\}$ a multi-set with p elements. Empirical characteristic function (ECF) of A is the estimate of characteristic function A and is defined as

$$\varphi_p(t) = \frac{1}{p} \sum_{j=1}^{p} e^{ita_j}.$$

In order to represent A by a vector of m features, ECF is calculated for m different values of $t = t_1, \ldots, t_m$. The features will be the value of ECF at these points. For this propose an interval with t_1 as the lower bound and t_m as the upper bound is chosen. After that m evenly spaced point are chosen in the interval to obtain m different values of t.

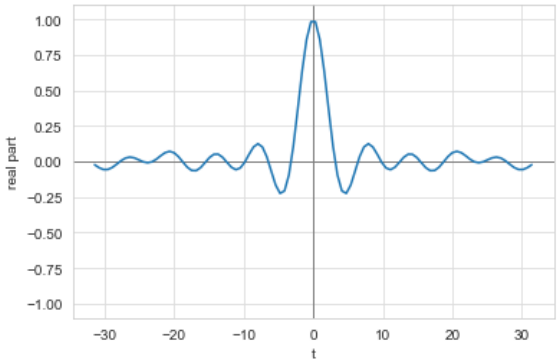
The variable t is a real number and can take any value in $\mathbb{R}$. This parameter controls the behavior of the ECF and therefore it is crucial to choose t_1 and t_m properly.

Using the inverse of Equation (18), the PDF of X can be written as

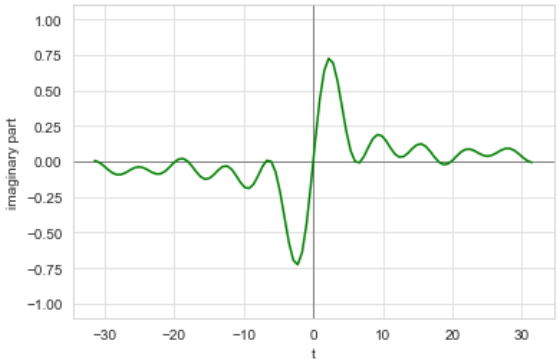
$$f(x) = \frac{1}{2\pi} \int e^{-itX} \varphi(t)dt = \frac{1}{2\pi} \int e^{-itX} E[\cos(tX) + i\sin(tX)]dt. \tag{19}$$

When the value of $|t|$ is small, the $\cos(tX)$ and $\sin(tX)$ from Equation (19) go through slow oscillations. When t approaches 0 the values on $\cos(tX)$ approaches 1 and the value of $\sin(tX)$ approaches 0. With the large values of $|t|$, the $\cos(tX)$ and $\sin(tX)$ oscillate rapidly.

Smaller values of $|t|$ correspond to a lower frequency which describes the slow variation in PDF, and larger values of $|t|$ correspond to a higher frequency which describes fast variations in PDF. This can be viewed in Figure 9 which illustrates the CF of a uniform random variable. Figure 9a shows the real part of CF for different values of t and Figure 9 b shows the imaginary part of CF for different values of t. You can see the how the value of $|t|$ affect the oscillation.



(a) The real part versus t of the CF of a uniform random variable



(b) The imaginary part versus t of the CF of a uniform random variable

Figure 9: Smaller values of $|t|$ correspond to lower frequency and larger values of $|t|$ correspond to higher frequency: When the value of $|t|$ is small, the CF oscillates slowly, while with the large values of $|t|$ the CF oscillates rapidly. When t approaches 0 the values on the real part approach 1 and the value of the imaginary part approaches 0.

It is preferable to avoid any interval for t that is symmetric around the origin. Using t and $-t$ results in the same real part, since $\cos(tX) = \cos(-tX)$. Therefore, by choosing an interval that is not symmetric around the origin we can obtain more information.

The interval $(0, 2\pi]$ is selected for this report. Note that 0 is excluded from the interval as the value of CF for every random variable at $t = 0$ is equal to 1.

In order to represent an unordered multi-set $A = \{a_1, \ldots, a_p\}$ by a vector of features employing ECF, the process is as follows:

1. Estimate the characteristics function of A.

2. To construct m features, m equally spaced points denoted as $t_1, \ldots, t_m$ are chosen in the interval of $(0, 2\pi]$. The values of t_1 is chosen very close to 0 and the values of t_m is chosen as 2π.

3. The value of $\varphi(t_i)$ generates m features that consists of real and imaginary parts. The vector of features is calculated as $[\frac{1}{p}\sum_{j=1}^{p}\cos(t_1 a_j)+\frac{1}{p}\sum_{j=1}^{p} i\sin(t_1 a_j), \ldots, \frac{1}{p}\sum_{j=1}^{p}\cos(t_m a_j)+\frac{1}{p}\sum_{j=1}^{p} i\sin(t_m a_j)]$.

Figure 26 illustrates the process of constructing features employing ECF.

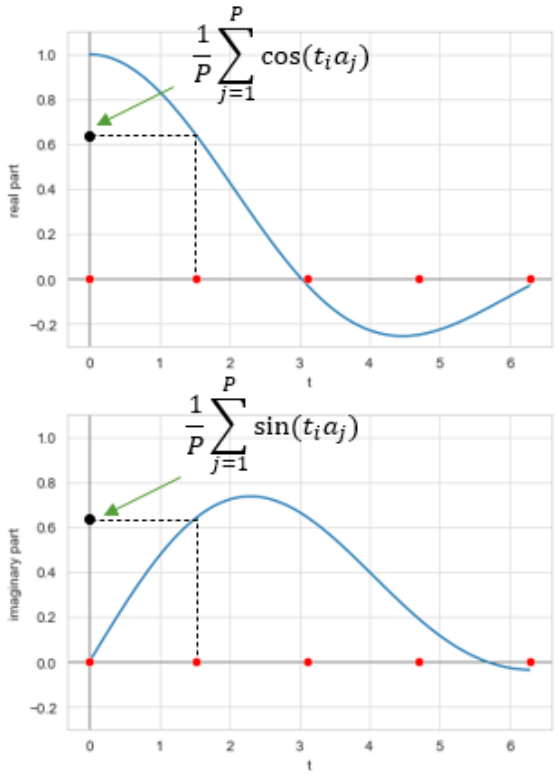


Figure 10: Feature construction using ECF: The blue lines in the top and bottom plot represent the real and the imaginary part of the ECF of an unordered multi-set $A = \{a_1, \ldots, a_p\}$ versus t respectively. To represent this multi-set by a vector of features employing ECF, the ECF of A is calculated for $t_1, \ldots, t_m$. These points are equally spaced on the interval of $(0, 2\pi]$, denoted by the red color in the figure. The value of ECF at $t_1, \ldots, t_m$ yields a set of m complex features.

The advantage of this method compared to the moment approach is that CF exists for all real valued random variables [7]. In addition to that, all features constructed by ECF are bounded since $|\varphi(t)| \leq 1$ for all t [7].

4 Results and Discussion

In this section, each density estimation method is empirically evaluated. The goal is that the constructed features represent the original data in the best way, this implies that these features are expected to be similar for similar multi-sets while being distinctly different for dissimilar multi-sets.

For this purpose, the constructed features are used in training two classifiers, LR and SVM. In this report, the evaluation of density estimation methods is done by comparing the accuracy of the classifiers that are trained with the constructed features. The objective of classifiers is to accurately classify the data-set into distinct classes.

4.1 Data

In order to empirically compare the strengths and weaknesses of each method in constructing features, four synthetic data-sets are created. These data-sets have different characteristics that aim to challenge the feature construction methods and the classifiers in different ways.

Each data-set consists of n observations where each observation is a multi-set. The multi-sets are defined in different number of classes. All data-sets are designed to be balanced, ensuring an equal representation of each class in the data-set. The number of classes is set to four or ten, and the number of representation of each class is set to 20 in the entire report. For instance if a data-set is defined in four classes, then on total it has $4 \times 20 = 80$ observations.

In order to study how the number of elements in the multi-set affects the quality of the constructed features, the experiment is done for different number of elements, p.

$$\text{Number of elements}(p) \in \{5, 10, 50, 100, 500, 1000\}.$$

4.1.1 Multi-sets with fixed length and bounded elements

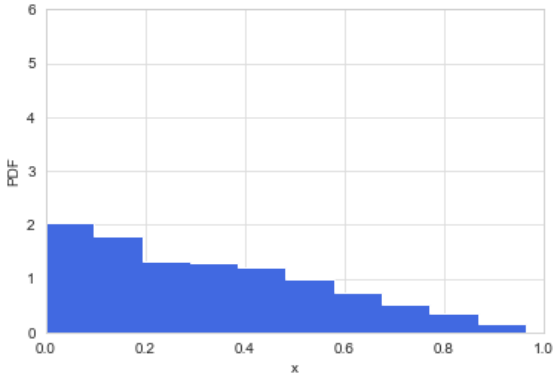
This data-set, $X_{Bounded}$, consists of $n = 80$ multi-sets, and the elements of each multi-set are drawn from the Beta distribution. There are four distinct Beta distributions used for this purpose, representing four different classes (20 multi-sets from each class). The parameters of the Beta distributions are chosen in a way to create distinct, but not too dissimilar multi-sets. The chosen Beta distributions are Beta(1,2), Beta(1,3), Beta(1,4) and Beta(1,5).

The elements are continuous and bounded in the interval [0, 1] as they are drawn from the Beta distribution. The number of elements in the multi-sets is fixed across all multi-sets in each experiment and is equal to p, where $p \in \{5, 10, 50, 100, 500, 1000\}$.

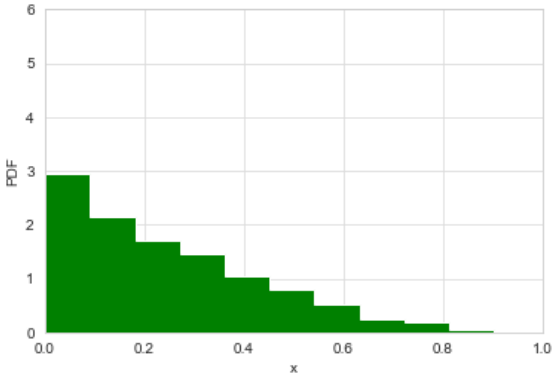
$$X_{\text{Bounded}} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix},$$

where $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,p}\}$ for all $i = 1, \ldots, n$ and $p \in \{5, 10, 50, 100, 500, 1000\}$. The elements $a_{i,j}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, p$ are drawn from one of the Beta distributions mentioned earlier.

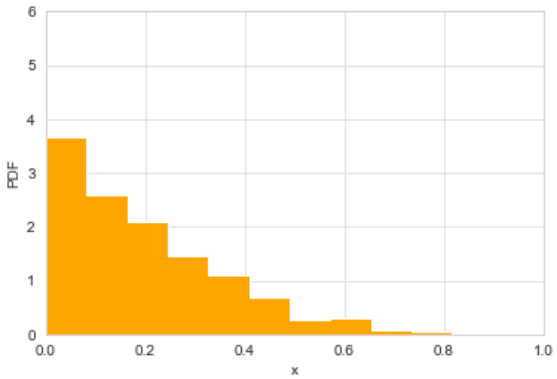
Figure 11 visualize the difference between classes, this figure shows histograms belonging to four multi-sets from $X_{Bounded}$ with $p = 1000$, one from each class, where the elements of each multi-set are drawn from a distinct Beta distribution. The objective of classifiers is to accurately classify the multi-sets from this data-set into four distinct classes using the features obtained by each feature construction method.



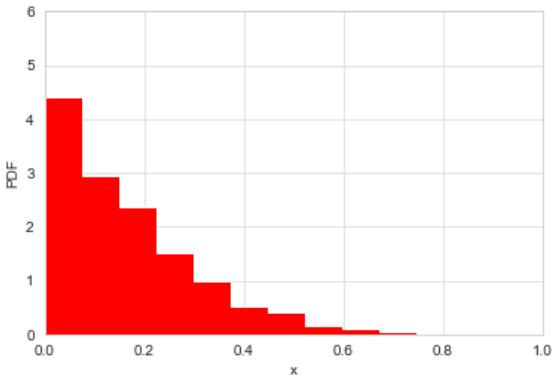
(a) Histogram of an observation from $X_{Bounded}$ where the elements of this observation are drawn from Beta$(1, 2)$



(b) Histogram of an observation from $X_{Bounded}$ where the elements of this observation are drawn from Beta$(1, 3)$



(c) Histogram of an observation from $X_{Bounded}$ where the elements of this observation are drawn from Beta$(1, 4)$



(d) Histogram of an observation from $X_{Bounded}$ where the elements of this observation are drawn from Beta$(1, 5)$

Figure 11: Histograms of four multi-sets with same number of elements(p): Each histogram describes a multi-set (an observation) from $X_{Bounded}$ with $p = 1000$ that are drawn from four distinct beta distribution.

4.1.2 Multi-sets with flexible length and bounded elements

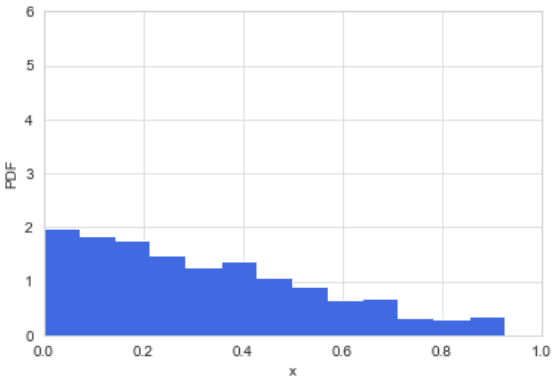
This data-set, $X_{Bounded-Flex}$, is similar to the previous data-set, $X_{Bounded}$, in having bounded elements. The data-set, $X_{Bounded-Flex}$, consists of $n = 80$ multi-sets, and the elements of each multi-set are drawn from the Beta distribution. There are four distinct Beta distributions used for this purpose, representing four different classes (20 multi-sets from each class). The chosen Beta distributions are Beta(1,2), Beta(1,3), Beta(1,4) and Beta(1,5).

To challenge the feature construction methods and the classifiers, the number of elements in multi-sets is not fixed across all multi-sets in each experiment. The number of elements in multi-sets varies from each other and is set to r, that is a number in the interval $[1, p]$, where $p \in \{5, 10, 50, 100, 500, 1000\}$.

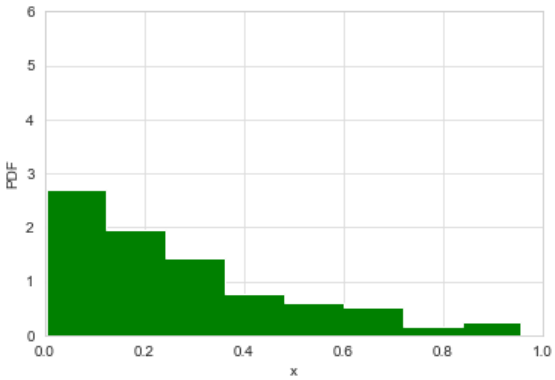
$$X_{\text{Bounded-Flex}} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix},$$

where $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}\}$ for all $i = 1, \ldots, n$ and $r_i = 1, \ldots, p$. The elements $a_{i,j}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, r_i$ are drawn from one of the Beta distributions mentioned earlier.

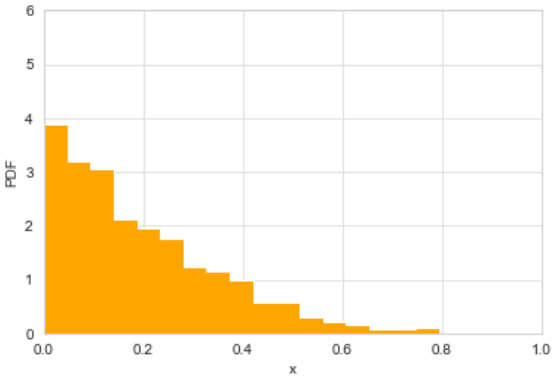
Figure 12 shows histograms belonging to four multi-sets from $X_{Bounded-Flex}$, one from each class. The objective of classifiers is to accurately classify this data-set into four distinct classes using the features obtained by each feature construction method.



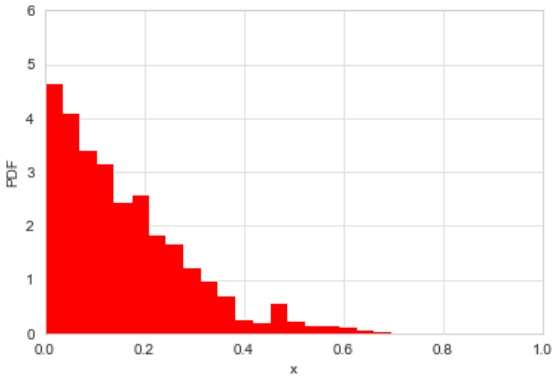
(a) Histogram of an observation from $X_{Bounded-Flex}$ with 831 elements. The elements of this observation are drawn from Beta$(1, 2)$



(b) Histogram of an observation from $X_{Bounded-Flex}$ with 99 elements. The elements of this observation are drawn from Beta$(1, 3)$



(c) Histogram of an observation from $X_{Bounded-Flex}$ with 827 elements. The elements of this observation are drawn from Beta$(1, 4)$



(d) Histogram of an observation from $X_{Bounded-Flex}$ with 953 elements. The elements of this observation are drawn from Beta$(1, 5)$

Figure 12: Multi-sets with varying number of elements: Each histogram describes a multi-set (an observation) from $X_{Bounded-Flex}$ with varying number of elements between 1 and 1000. The elements are drawn from four distinct beta distribution.

4.1.3 Multi-sets with fixed length and unbounded elements

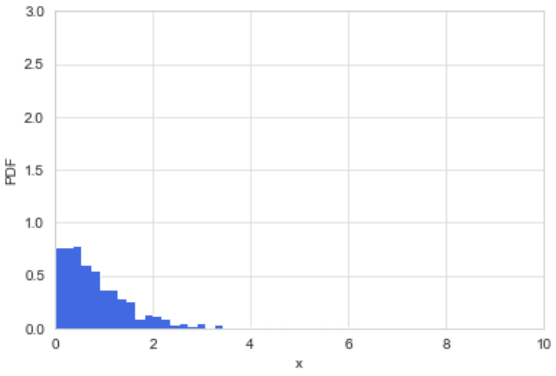
This data-set, $X_{Heavy-Tailed}$, is similar to $X_{Bounded}$ data-set when it comes to having a fixed number of elements for all the multi-sets and consists of $n = 80$ multi-sets. However, in order to challenge the feature construction methods and the classifiers, the elements are not bounded anymore. The elements of multi-sets are drawn from four distinct heavy-tailed distributions. The tail of the distribution can represent the outliers, making it advantageous to sample from heavy-tailed distributions to challenge the methods.

The data-set, $X_{Heavy-Tailed}$, consists of $n = 80$ multi-sets, and the elements of each multi-set are drawn from a heavy-tailed distribution. There are four distinct heavy-tailed distributions used for this purpose, representing four different classes (20 multi-sets from each class). The chosen heavy-tailed distributions are half-Cauchy(0,1), Lognormal(0,1), Lognormal(0,2), and Pareto(1,1.5).

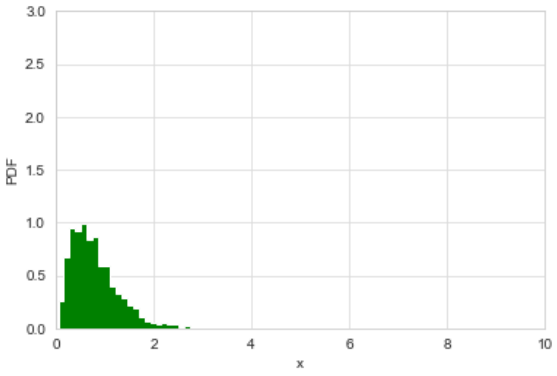
$$X_{\text{Heavy-Tailed}} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix},$$

where $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,p}\}$ for all $i = 1, \ldots, n$ and $p \in \{5, 10, 50, 100, 500, 1000\}$. The elements $a_{i,j}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, p$ are drawn from one of the heavy-tailed distributions mentioned earlier.

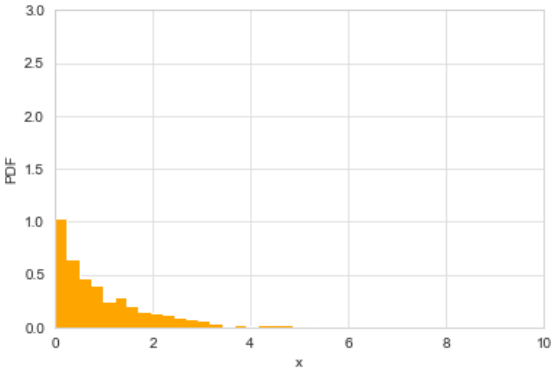
Figure 13 shows histograms belonging to four multi-sets with same number of elements, where the elements are drawn from a distinct heavy-tailed distribution. The objective of classifiers is to accurately classify this data-set into four distinct classes using the features obtained by each feature construction method.



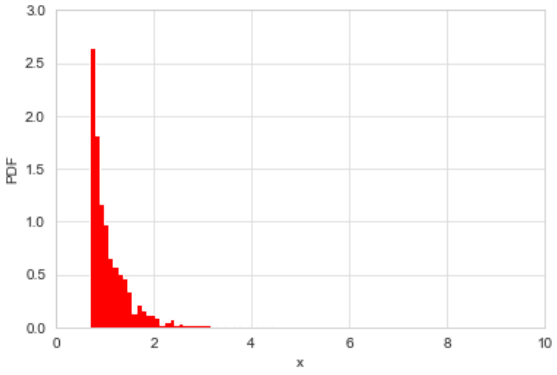
(a) Histogram of an observation from $X_{Heavy-Tailed}$ where the elements of this observation are drawn from Half-Cauchy$(0, 1)$



(b) Histogram of an observation from $X_{Heavy-Tailed}$ where the elements of this observation are drawn from Lognormal$(0, 1)$



(c) Histogram of an observation from $X_{Heavy-Tailed}$ where the elements of this observation are drawn from Lognormal$(0, 2)$



(d) Histogram of an observation from $X_{Heavy-Tailed}$ where the elements of this observation are drawn from Pareto$(1, 1.5)$

Figure 13: Multi-sets with unbounded elements: Each histogram describes a multi-set (an observation) from $X_{Heavy-Tailed}$ with 1000 elements that are drawn from four distinct heavy-tailed distributions.

4.1.4 Multi-modal with high frequency

This data-set, $X_{Multi-Modal}$, is inspired by high-frequency data with sharp peaks . For instance consider Swish transactions between people, as the transferred amount is usually rounded, and the distribution of transferred amounts has sharp peaks. In addition to sharp peaks, the intention is to achieve an overall Gaussian bell curve for this data-set, as Gaussian distribution is the most common real-world distribution.

The aim of this data-set is to challenge the feature construction methods and the classifiers with high frequency data. This data-set consists of $n = 200$ multi-sets with elements from 10 different five-modal distributions (10 classes). The number of elements in the multi-sets is fixed across all multi-sets in each experiment and is equal to p, where $p \in \{5, 10, 50, 100, 500, 1000\}$.

The process of drawing elements from a five-modal Cauchy distribution is as follows:

1. Cauchy distribution is characterized by two parameters, location(l) which defines the center of the distribution, and scale(s) which defines the spread. The first step is to generate location and scale parameters for five Cauchy distributions.

   (a) The location parameters are chosen in a manner to prevent the overlap of the peaks of the individual Cauchy distributions as much as possible. The first location is chosen as 1 and to obtain the second location, a uniformly random value between 2 and 2.5 is added to the first location. To obtain the next location, add a uniformly random value between 2 and 2.5 to the previous location. Continue this process until five locations are obtained.

   $$\text{Location parameters} = \{l_1, l_2, l_3, l_4, l_5\}.$$

   (b) The scale parameters are chosen very low, a uniformly random value between $\sqrt{0.02}$ and $\sqrt{0.2}$, to ensure higher peaks.

   $$\text{Scale parameters} = \{s_1, s_2, s_3, s_4, s_5\}.$$

2. To create an overall bell shape, the distribution that its location is in the middle should have the most number of elements compared to the rest. Moving away from the middle, the number of elements should decrease. Therefore to find the number of elements drawn from each Cauchy distribution, p is divided into 5 unequal parts, q_1 to q_5.

3. For $j = 1, \ldots, 5$, sample q_j elements from Cauchy(l_j, s_j).

4. If any of the elements is far from the location of the distribution ($|\text{element} - l_j| > 6s_j$), that element is replaced by the value of the location (l_j) of the distribution. This is done to create higher peaks in the data. Cauchy distribution has a heavier tail compared to Gaussian distribution which is the reason behind choosing the Cauchy distribution in this part. This implies a higher probability of extreme sample and a higher chance to replace an element with the value of the location, hence a higher peak.

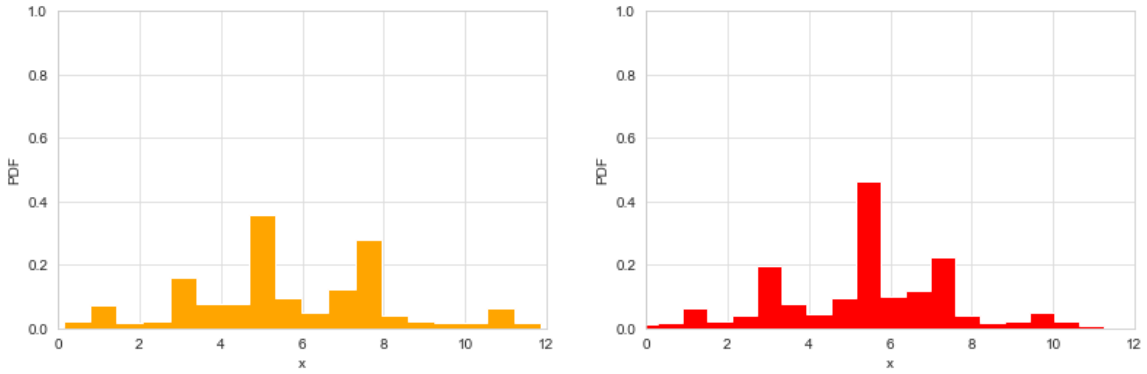
$$X_{\text{Multi-modal}} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix},$$

where $A_i = \{C_{i,1} \bigcup C_{i,2} \bigcup C_{i,3} \bigcup C_{i,4} \bigcup C_{i,5}\}$ for all $i = 1, \ldots, n$.

$$X_{\text{Multi-modal}} = \begin{bmatrix} \{C_{1,1} \bigcup C_{1,2} \bigcup C_{1,3} \bigcup C_{1,4} \bigcup C_{1,5}\} \\ \{C_{2,1} \bigcup C_{2,2} \bigcup C_{2,3} \bigcup C_{2,4} \bigcup C_{2,5}\} \\ \vdots \\ \{C_{n,1} \bigcup C_{n,2} \bigcup C_{n,3} \bigcup C_{n,4} \bigcup C_{n,5}\} \end{bmatrix},$$

where $C_{i,j} = \{a_{i,1}, a_{i,2}, \ldots, a_{i,q_j}\}$ for all $i = 1, \ldots, n$, $j = 1, \ldots, 5$ and $\sum_{j=1}^{5} q_j = p$. The elements a_{i,q_j} are drawn from Cauchy(l_j, s_j).

Figure 14 shows histograms belong to two multi-sets with same number of elements($p = 1000$), where the elements belong to two different five-modal Cauchy distributions. These two multi-sets represent two classes out of ten classes in this data-set.



(a) Histogram of an observation from $X_{Multi-Modal}$ (b) Histogram of an observation from $X_{Multi-Modal}$

Figure 14: Multi-sets with multi-modal distributions: Each histogram describes a multi-set with 1000 elements that are drawn from a certain five-modal Cauchy distribution. Both histograms have sharp peaks and the overall shape is like a bell curve.

4.2 Hyperparameter optimization

There are two types of parameters in any machine learning model. One type that can be estimated when the models are being trained such as the coefficients in the LR and SVM models. The other ones, also known as hyperparameters, are required to be set before training the models as they define the amount of regularization of the model, for instance, the parameter λ from the shrinkage penalty that is added to the cost functions of the logistic regression and the hyperparameters C and γ from the SVM model.

The hyperparameters control the learning process of the model and therefore it is important to find an optimal value for them to get the best possible performance of the model. In other words, we need to search for the hyperparameters that lead the models to a low test error.

Grid search is one of the most commonly used methods in hyperparameter optimization [10]. This method trains and validates models for all possible combinations of the values for all the hyperparameters that are used in the model. This could be a disadvantage if the search space is too large [14].

When training and validating each model via grid search, a loss function needs to be defined to compare the performance of the models. In this report, cross-entropy is selected for the loss function. The reason behind choosing cross-entropy is that this loss function is not dependent on whether the model correctly classify an observation or not. Rather, it measures the uncertainty of the model in predicting the correct class by using probability. Cross-entropy loss is defined as

$$\text{Cross-entropy loss} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(p_{ij}(x_i)),$$

where n is the number of data points and y_{ij} is the true probability of x_i belonging to class j. The value of y_{ij} is 1 if x_i is correctly classified, otherwise y_{ij} takes the value 0. And $p_{ij}(x_i)$ is the predicted probability of x_i belonging to class j.

As mentioned in Sub-section 2.1.2, SVM finds the optimal separating hyperplane and does not find a probability of each data point belonging to a class. To use cross-entropy loss in tuning the hyperparameter of SVM, the method `predict_proba` is used from the `scikit-learn` library. This method uses platt scaling to transform the output of SVM into probabilities. Platt scaling works by fitting a logistic regression model to a classifier's output [2].

For each data point x_i in the data-set, platt scaling finds the probability of x_i belonging to a certain class given the classifier scores $f(x_i)$. The classifier score is $f(x_i) = \beta_0 + \beta x_i^T$ where β_0 is the intercept and β is the coefficients of the separating hyperplane. For example in a binary classification case, the probability of x_i belong to class 1, given $f(x_i)$ is defined as

$$p(y = 1|x_i) = \frac{1}{1 + \exp(Af(x_i) + B)},$$

where A and B are the parameters to be estimated by logistic regression.

For validating models 5-fold cross-validation(CV) is used. 5-fold cross-validation is performed by randomly dividing the observations into 5 equal-size groups(folds). The first fold is the validation

set and the model is fit on the remaining 4 folds. The cross-entropy is calculated for the first fold. this is repeated 5 times, each time a different fold is used as a validation set. The 5-fold cross-validation estimates the test error by averaging the cross-entropy.

Optimal value for λ
The parameter λ from the shrinkage penalty $\lambda \sum_{j=1}^{p} \|\beta_j\|^2$ in Equation (14) controls the regularization to reduce the risk of over-fitting. The parameter λ in the L2-regularization controls the values of the coefficients in the logistic regression. The higher values of λ lead to a model with smaller coefficients, such a model has high bias and lower variance. As the value of λ decreases the models will be less biased and have higher variance.

Figure 15 illustrates the cross-validation error for the LR model for different values of λ. The model is trained by features constructed by the moments approach using X_{Bunded} where the number of features(m) is 5 and the number of elements(p) in multi-sets is 100. The blue line represent the CV error and the bars represents the standard error for CV at each point. As you can see the higher values of λ lead to lower CV error. The best model is chosen by the one standard error rule, and the value of λ for that model is the optimized value of the hyperparameter.

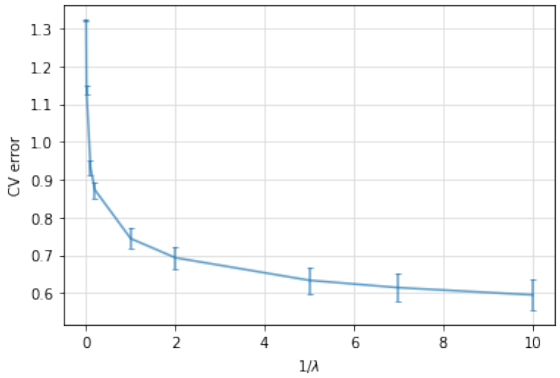


Figure 15: CV error for different values of λ in LR: The blue line represents the CV error for the different values of λ. The model is trained by features constructed by the moments approach using X_{Bunded} where the number of features(m) is 5 and the number of elements(p) in multi-sets is 100. The higher values of λ lead to a model with smaller coefficients, which has high bias and lower variance. As the value of λ decreases (moving along the x-axis) the models will be less biased and have higher variance. Using the one standard error rule, the best model is the one with $\lambda = 0.1$.

Optimal value for λ and γ
The parameter λ from the shrinkage penalty, $\frac{\lambda}{2}\|\beta\|^2$, in Equation (10) controls the regularization to reduce the risk of over-fitting. The parameter $\lambda = \frac{1}{C}$ controls the margins, higher values of C lead to higher tolerance for misclassification and result in higher bias but lower variance.

The parameter γ from the kernel of SVM in Equation (12) controls which points should be considered similar based on the distance between them. Low values of γ mean every point has far

reach, this means that even far away points affect the decision boundary and result in smoother decision boundaries and a model with higher bias but lower variance.

Figure 16 illustrates the cross-validation error for the SVM model to find the optimal values for C and γ. The model is trained by features constructed by the moments approach using X_{Bunded} where the number of features(m) is 5 and the number of elements(p) in multi-sets is 100. Each line in Figure 16 represent the CV error for different values of γ. Using one standard error rule, the model with $C = 1$ and $\gamma = 1$ which is highlighted in green is chosen.

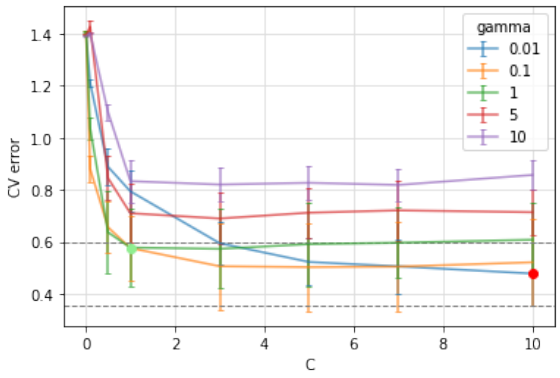


Figure 16: CV error for different values of C and γ in SVM: The different lines represent CV errors corresponding to a certain γ value. The model is trained by features constructed by the moments approach using X_{Bunded} where the number of features(m) is 5 and the number of elements(p) in multi-sets is 100. The smaller values of γ result in having smoother decision boundaries. This leads to a model with higher bias and lower variance. Higher values for C give a higher tolerance for misclassification to the model and result in higher bias but lower variance. The model with the lowest cross-validation error is highlighted in red. Using one standard error rule, the model with $C = 1$ and $\gamma = 1$ which is highlighted in green is chosen.

4.3 Evaluation of feature construction methods

In this section, the performance of classifiers that are trained by features that are constructed by methods mentioned in Section 3 is investigated. The feature construction methods applied to different data-sets is investigated to understand the strengths and weaknesses of each feature construction method.

The performance of the classifiers shows similar behaviors, indicating with an adequate number of elements(p) in each multi-set, the features constructed by all four methods can successfully train a classifier that performs well. The performance of classifiers is measured by accuracy, which is calculated as the percentage of correctly classified multi-sets. The details on the results can be found in figures in Appendix A.

All the figures in this section visualize the test accuracy, where the lines show how the accuracy of LR or SVM models changes for different numbers of constructed features(m). Each line represents a distinct number of elements(p) in the multi-sets and the shaded area around each line represents the standard error of the accuracy.

In general, the accuracy of models is very low when using only one constructed feature($m = 1$) in training them. The accuracy increases by adding more constructed features(m) in training the models, however, at some point adding more constructed features does not improve the accuracy. The number of constructed features in which the saturation occurs varies based on the number of elements(p) in each multi-set.

Figure 17 visualizes the results of LR models trained with features constructed by the moments approach. Consider the brown line that represents the accuracy of the model for $p = 1000$, the accuracy increases when the number of constructed features used in training the model increases from one to four. However, as you can see after $m = 4$, adding more constructed features does not improve the models. The models with a lower value of p saturate at a higher number of constructed features(m). For example, the model for $p = 500$, the purple line, saturates at $m = 5$.

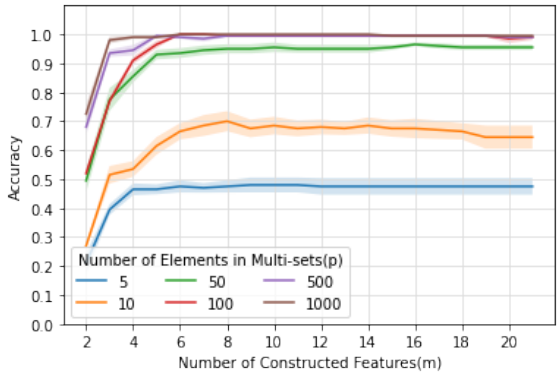
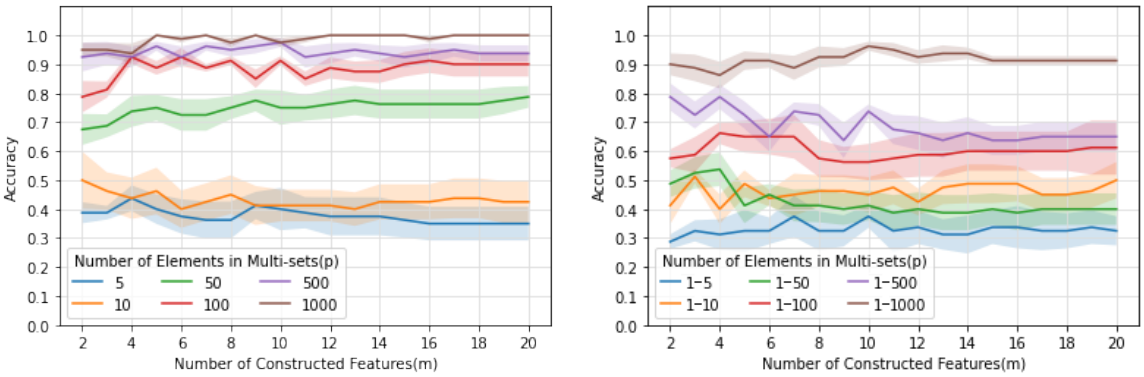


Figure 17: The accuracy increases as the number of features(m) increases, until a point of saturation: The brown line represents the accuracy of the model for $p = 1000$, the accuracy is at its lowest when using only one feature($m = 1$), but it increases as more features are used in training. However, after having four features, adding more features does not improve the accuracy.

In general, the classification results are more accurate when the number of elements(p) in the multi-sets is higher. With more data points, more information can be captured by different feature construction methods, hence the features are better with higher number of elements(p).

An example of the effect of higher values of p on the accuracy is illustrated in Figure 18 which compares the results of LR models that are trained using features constructed by KDE on two different data-sets. Both Figure 18a and Figure 18b represent the results for data-sets with elements drawn from the same distributions. The only difference between the data-sets, is that the multi-sets from Figure 18a have the same number of elements(p), while the multi-sets from Figure 18b have varying numbers of elements(between 1 to p) which are in total less.

Consider the brown lines from Figure 18a and Figure 18b. The model in Figure 18a is trained on features constructed from $X_{Bounded}$. This data-set contains multi-sets with the same number of elements($p = 1000$). On the other hand, the model in Figure 18b is trained on features constructed from $X_{Bounded_Flex}$. This data-set contains multi-sets with a varying number of elements(p from 1 to 1000).



(a) KDE using $X_{Bounded}$

(b) KDE using $X_{Bounded-Flex}$

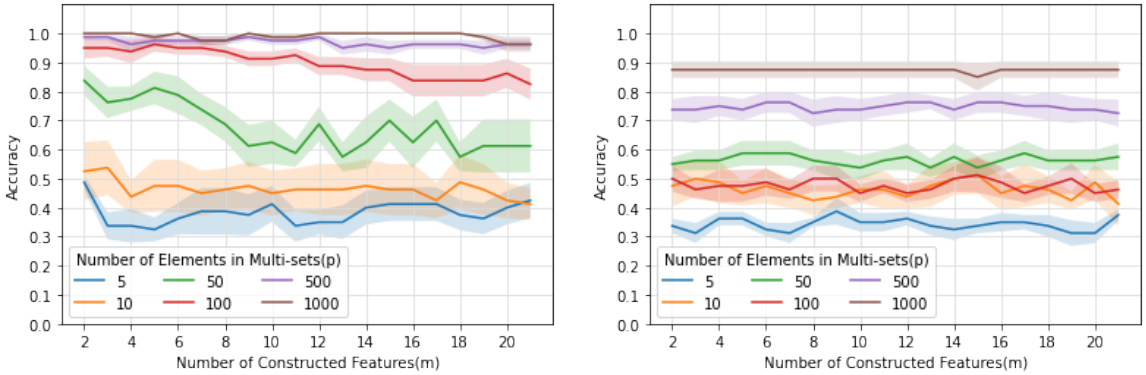
Figure 18: A higher number of elements(p) in multi-sets results in higher accuracy: With higher values of p, the density estimation methods can capture more information on the data-set, hence the constructed features would be more meaningful. Using better features in training models increases the accuracy. The data-set $X_{Bounded-Flex}$ has lower data points compared to $X_{Bounded}$ by its definition, and this explains the lower performance of the models in Figure 18b compared to the models in Figure 18a. The LR models that are trained with features that are constructed using more data points have higher accuracy.

So far two general observations have been discussed, in the following sub-sections the discussion is more specified on each feature construction method.

4.3.1 Moments approach

Moments are sensitive to outliers, therefore is it expected that the moments approach performs better in the absence of extreme values in the data-set. Therefore, this method can construct better features that enable the classifiers to perform better for $X_{Bounded}$ compared to $X_{Heavy-Tailed}$.

Figure 19 visualizes the accuracy of SVM models on different data-sets. These models are trained using features constructed by the moments approach. Comparing Figure 19a and Figure 19b, you can see the accuracy of the models is higher for models applied on $X_{Bounded}$. The data in $X_{Heavy-Tailed}$ is right skewed due to the existence of extreme values in it. Hence, the moments have high variability and the features constructed by the moments are not able to enable the classifiers to perform well.



(a) $X_{Bounded}$ (b) $X_{Heavy-Tailed}$

Figure 19: Moments approach performs better when there are no outliers in the data: Figure 19a shows the accuracy of SVM models for $X_{Bounded}$ and Figure 19b shows the accuracy of SVM models for $X_{Heavy-Tailed}$. As the figures show, the models perform much better with $X_{Bounded}$.

Log transformation can help to avoid the limitation of the moments approach regarding the existence of outliers. Log transformation reduces the impact of outliers in the data-set. The log function changes the large values to smaller ones, this reduces the skewness of the data-set. With the absence of outliers, the moments approach can construct features that enable the model to achieve high accuracy.

In this report, log transformation is applied by using $log1p()$ function from the $NumPy$ library. This function returns $\ln(1+x)$ for all x that is given to it. The reason to choose a function that returns $\ln(1+x)$ rather than $\ln(x)$ is that $\lim_{x\to 0}\ln(x) = \infty$ and in there exists values closed to 0 in the data-sets. By adding 1 we make sure that the log transformation does not create a big negative value in the transformed data-set.

Figure 20 shows how the log transformation of the data before feature construction can increase the accuracy.

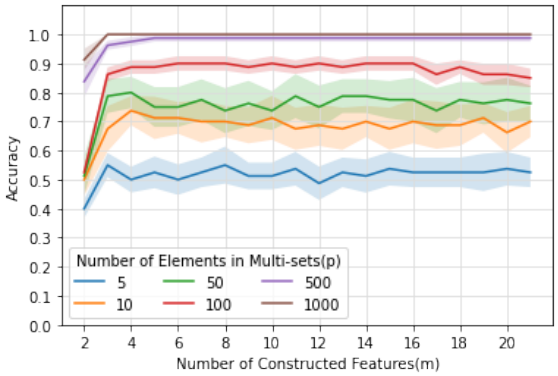
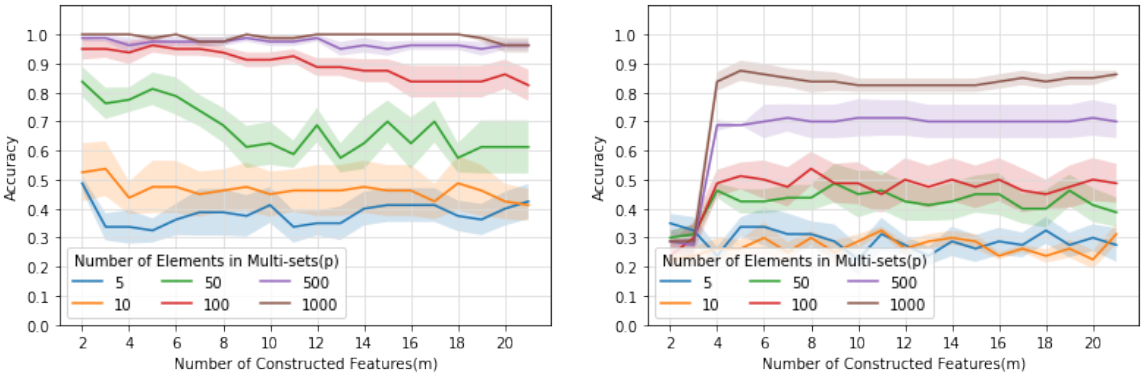


Figure 20: Log transformation of the data-set can reduce the impact of outliers in the data: Existence of outliers can impact the quality of features constructed by the moments approach. To avoid the effect outliers, log transformation can be applied to the data-set. This Figure shows how the accuracy of SVM models for $X_{Heavy-Tailed}$ can be improved by log transformation. You can compare this figure with Figure 19b. Both figures visualize the accuracy of SVM models on $X_{Heavy-Tailed}$, however log transformation was done on $X_{Heavy-Tailed}$ in Figure 20 before feature construction.

The first two moments are meaningful numerical descriptive measures of a distribution [20]. Therefore the features which are constructed by these two moments have higher importance compared to features constructed by other moments. To see the importance of the first two moments, an experiment was done on $X_{Bounded}$ by standardizing the data-set. Standardized data have 0 mean and unit variance, this makes the features that are constructed from the first two moments to be the same for all observations, meaning they cannot be used for separating different classes from each other.

Figure 21 visualizes this experiment. Figure 21a shows the accuracy of SVM models on $X_{Bounded}$. This data-set was standardized before feature construction, and the result of the models using this standardized data in training is visualized in Figure 21b. As you can see in Figure 21b, the models perform poorly when the number of features(m) used in training them is below three. However after adding more features the performance of models improves significantly. However, the accuracy of models is much lower in Figure 21b compared to the models in Figure 21a for the same number of elements(p) in the multi-sets and the number of constructed features(m).



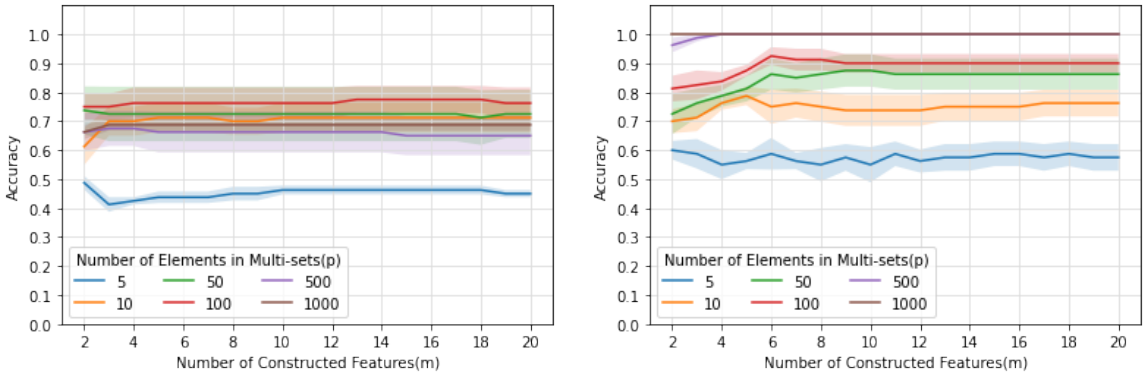
(a) $X_{Bounded}$ (b) Standardized $X_{Bounded}$

Figure 21: The first two moments construct important features: Figure 21a shows the accuracy for $X_{Bounded}$ and Figure 21b shows the accuracy for the same data-set when the data-set was standardized before feature construction. These models perform poorly when the number of features(m) used in training them is below three. By adding more features the performance of models improves significantly. However, the accuracy of models in Figure 21b does not reach the level of accuracy in Figure 21a.

4.3.2 Kernel density estimation

In KDE, the bandwidth of the kernel is fixed across all the data points belonging to a multi-set. This could lead to failure in finding an optimal kernel bandwidth in case of the existence of extreme values. Figure 22 visualizes this case.

Figure 22 shows the accuracy of LR model results with features constructed by KDE on $X_{Heavy-Tailed}$. Figure 22a shows that all the models for different number of elements(p) have relatively low performance which can be the explained by existence of outliers in $X_{Heavy-Tailed}$. To reduce the effect of outliers, log transformation is performed on $X_{Heavy-Tailed}$ before feature construction (similar to Sub-section 4.3.1). Figure 22b shows the accuracy of models on log transformation $X_{Heavy-Tailed}$. As you can see the accuracy improved significantly.



(a) $X_{Heavy-Tailed}$

(b) Log transformed $X_{Heavy-Tailed}$

Figure 22: Sensitivity of kernel bandwidth to extreme values: The existence of extreme values could lead to failure in finding an optimal kernel bandwidth. However, this could be avoided by log transformation of data-set before feature construction to reduce the effect of extreme values. Comparing Figure 22a and Figure 22b shows how log transformation of the data-set before features construction can reduce the effect of extreme values on choosing the kernel bandwidth.

4.3.3 Empirical cumulative distribution function

Based on how features are constructed using ECDF, explained in Section 3.3, the first two features contain information on the lower and upper bound of the underlying data-set. These features might not be very useful in cases where all of the observations are bounded in the same interval as these features will not enable the models to learn of the differences between observations. Figure 23 shows the accuracy of SVM models trained by features constructed by ECDF on $X_{Bounded}$. You can see the significant improvement when the number of features(m) increases from two to three.

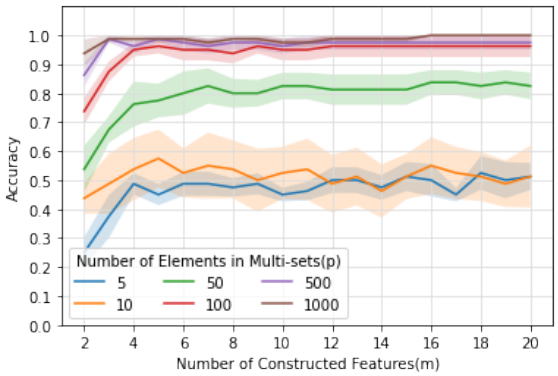


Figure 23: Using ECDF, more features are needed to reach high accuracy: The accuracy of SVM models is relatively low for models being trained by only two features. However, the accuracy increases and reaches a good level with more features. As you can see there is a big increase in the accuracy of all models in this figure when the number of constructed features(m) increases from 2 to 4.

This method is easier to use compared to KDE, as there is no need to estimate the kernel width and no need to estimate the support of data. In addition to that, ECDF performs well even with the existence of outliers in the data-set. Figure 24 shows the accuracy of SVM models trained by features constructed by ECDF on $X_{Heavy-Tailed}$. As you can see, ECDF performs well on this data-set that contains extreme values in contrast to the models from Figure 22a.

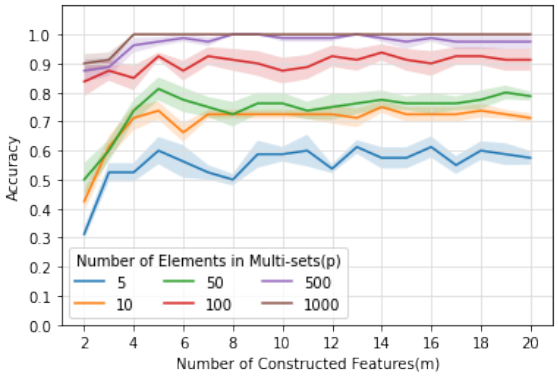


Figure 24: ECDF performs well even with the existence of outliers: With a high number of elements(p) in the multi-sets, ECDF can construct features that could train a classifier successfully even with low number of constructed features(m). When the number of elements(p) is below 100, this method requires more features to perform well. For example for $p = 50$ the models requires at least 5 features $m = 5$

4.3.4 Empirical characteristic function

The characteristic function uniquely determines the distribution function [12]. This explains why models trained by features constructed by ECF have high accuracy even with a low number of features(m). Figure 25 represents the accuracy of SVM models trained by features constructed by ECF on $X_{Bounded}$. As you can see the models in Figure 25 require few number of constructed features(m) to achieve a high accuracy.

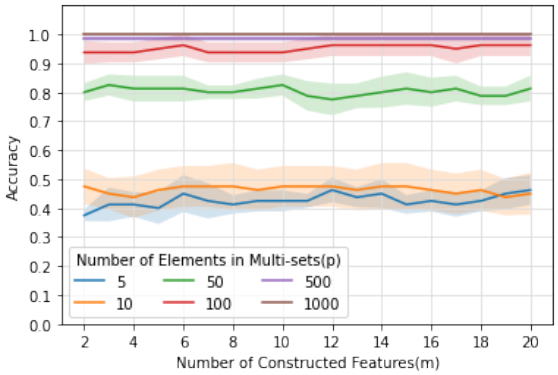


Figure 25: The models trained by features constructed by ECF, perform well with a low number of features(m): As you can see, the models with an adequate number ($p \geq 50$) of elements(p) in the multi-sets require only 2 constructed features(m) to achieve a high level of accuracy. The number of elements(p) in the multi-sets is an important factor in constructing features, as you can see the models that are presented by the blue and the orange lines never reach a high level of accuracy regardless of the number of constructed features(m) used in training them.

The existence of outliers do not affect the performance of the features constructed by ECF. Figure 26 shows the accuracy of SVM models trained by features constructed by ECF on $X_{Heavy-Tailed}$. As you can see, ECF performs well on this data-set that contains extreme values in contrast to the models from Figure 22a or models from Figure 19b.

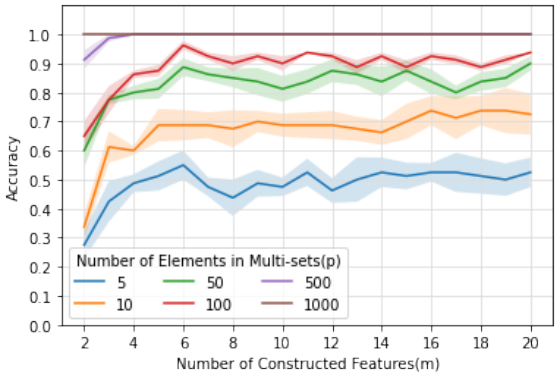


Figure 26: ECF performs well even with the existence of outliers: With higher number of elements(p) in the multi-sets, ECF can construct features that could train a classifier successfully with only few number of features(m). For example the brown line that represent models with the number of elements(p) 1000 reaches the accuracy level of 100% with only two constructed features(m).
For lower number of elements(p) in the multi-sets, the models require more constructed features(m) to perform well. For example for $p = 10$ (the models presented by the orange line) the models requires at least 5 features $m = 5$.

5 Conclusion and Future Work

In this report, constructing features with different methods was empirically compared using classifiers. The quality of the features was measured by the accuracy of the classification. Based on the results, all of the methods mentioned in this report can construct meaningful features from unordered multi-sets.

Each method has its advantages and limitations, and the best method for each data-set should be chosen according to the characteristics of the data-set. It is crucial to understand the data-set well before choosing the suitable method for feature construction.

The moments approach performs well in most cases except in the case of the existence of outliers in the data-set. The reason behind dropping the performance in this case is that moments are sensitive to outliers. Therefore, this method is not recommended to use in case of the existence of extreme values in the data-set.

Similar to the moments approach, KDE performs well in most cases but it struggles with the existence of extreme values in the data-set. However, a data-set that contains outliers could be log transformed to reduce the negative impact of outliers in feature construction. As mentioned in Section 4, both the moments approach and KDE perform well when log transformation is applied to the data-set before feature construction.

ECDF performs well in all cases. Using this method for feature construction has some advantages over KDE. In ECDF there is no need to estimate the kernel width or the support of data. ECDF uses the range of data that is fixed and always between 0 and 1. Another advantage of using ECDF compared to the moment approach and KDE is that it is not sensitive to outliers.

ECDF performs well in all cases. The performance of features constructed by ECF is not affected by the existence of outliers in the data. In addition to that, models trained by features constructed by ECF have better accuracy even with a low number of features compared to models trained by features constructed by other methods.

In this report two different classifiers were used to evaluate the different feature construction method. However, using a classifier to evaluate the quality of features, limits the evaluation result to be relevant to only classification methods.

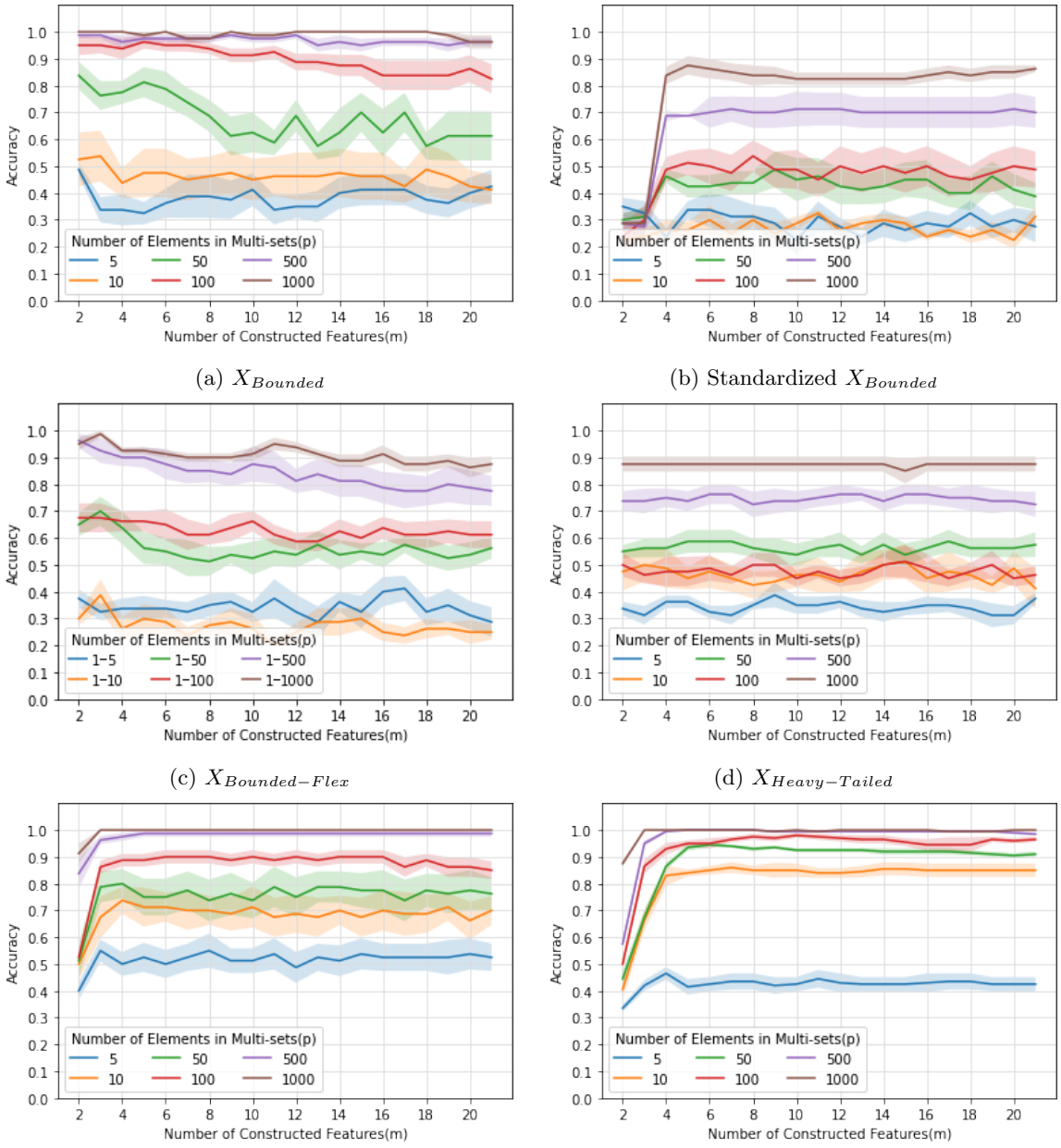
As a future work, more generalized ways for evaluating the feature construction methods can be explored. One method is to use a similarity measure. The idea is that if two multi-sets are similar (belong to the same class), the feature vectors constructed from those multi-sets should also be similar.

Similarity can be measured by distance and there exist different statistical distance measures, such as Kullback–Leibler (KL) divergence or Wasserstein distance. Such a measure enables the evaluation of how each method preserves the information in the data in a more general way. Using a similarity measure, a feature construction method preserves the information well if the distance between multi-sets is highly correlated with the distance between the feature vectors constructed from them.

A Appendix

Moments approach

SVM results



(a) $X_{Bounded}$

(b) Standardized $X_{Bounded}$

(c) $X_{Bounded-Flex}$

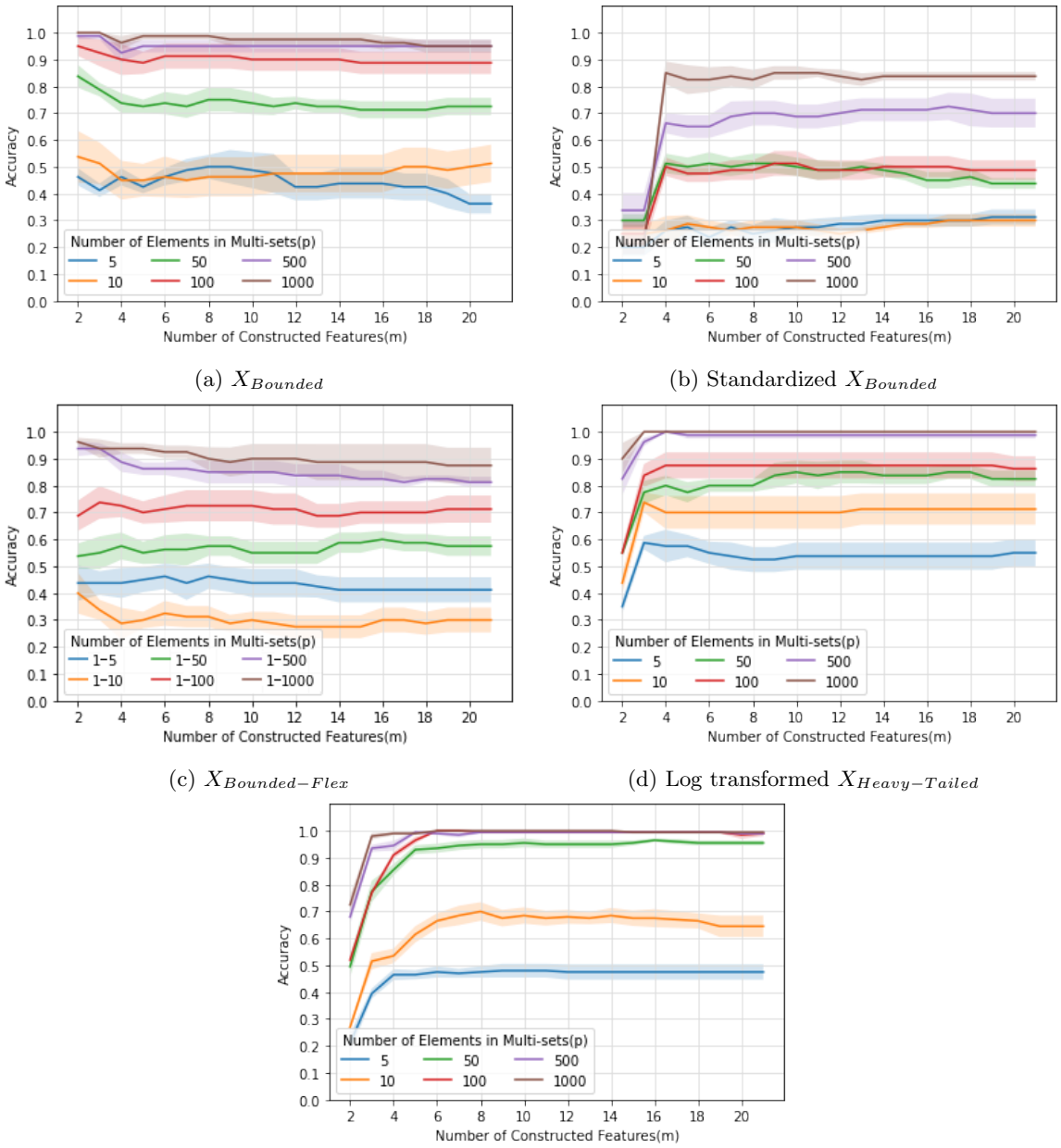
(d) $X_{Heavy-Tailed}$

(e) Log transformed $X_{Heavy-Tailed}$

(f) $X_{Multi-Modal}$

Figure 27: SVM results: Accuracy of SVM models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by moments approach.

Logistic regression results



(a) $X_{Bounded}$

(b) Standardized $X_{Bounded}$

(c) $X_{Bounded-Flex}$

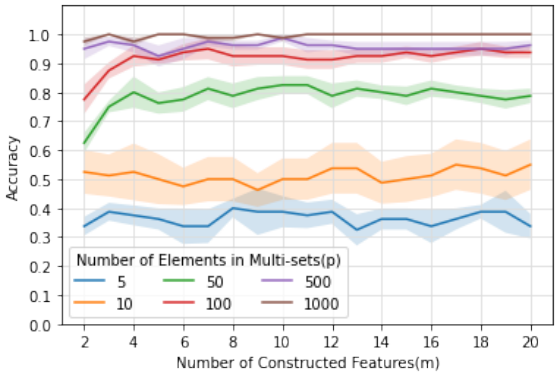
(d) Log transformed $X_{Heavy-Tailed}$

(e) $X_{Multi-Modal}$

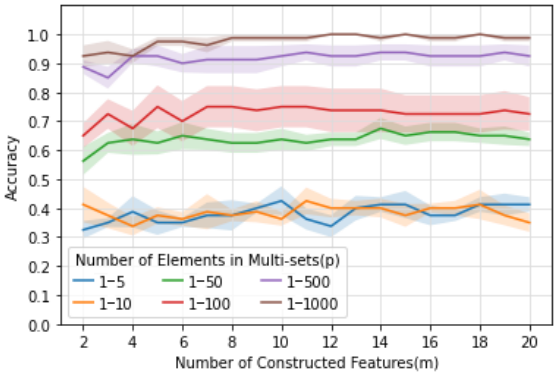
Figure 28: Logistic regression results: Accuracy of logistic regression models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by moments approach.

Kernel density estimation

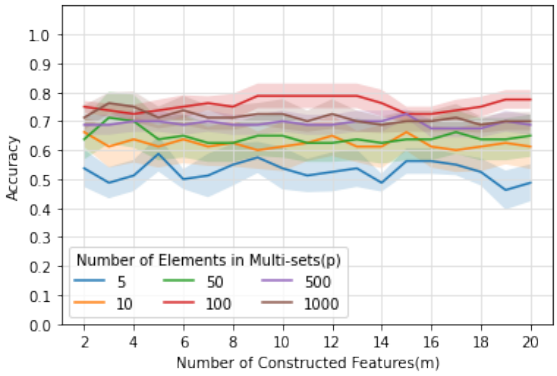
SVM results



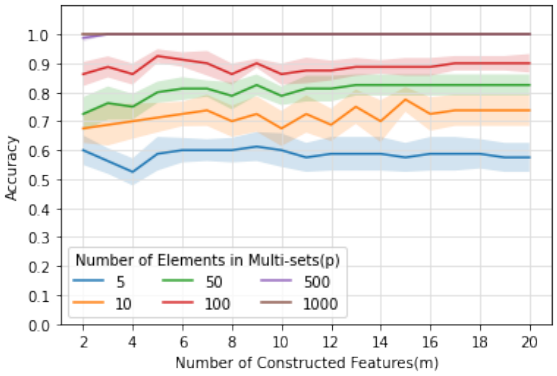
(a) $X_{Bounded}$



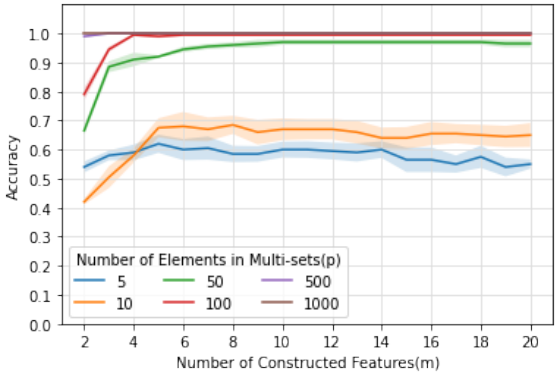
(b) $X_{Bounded-Flex}$



(c) $X_{Heavy-Tailed}$



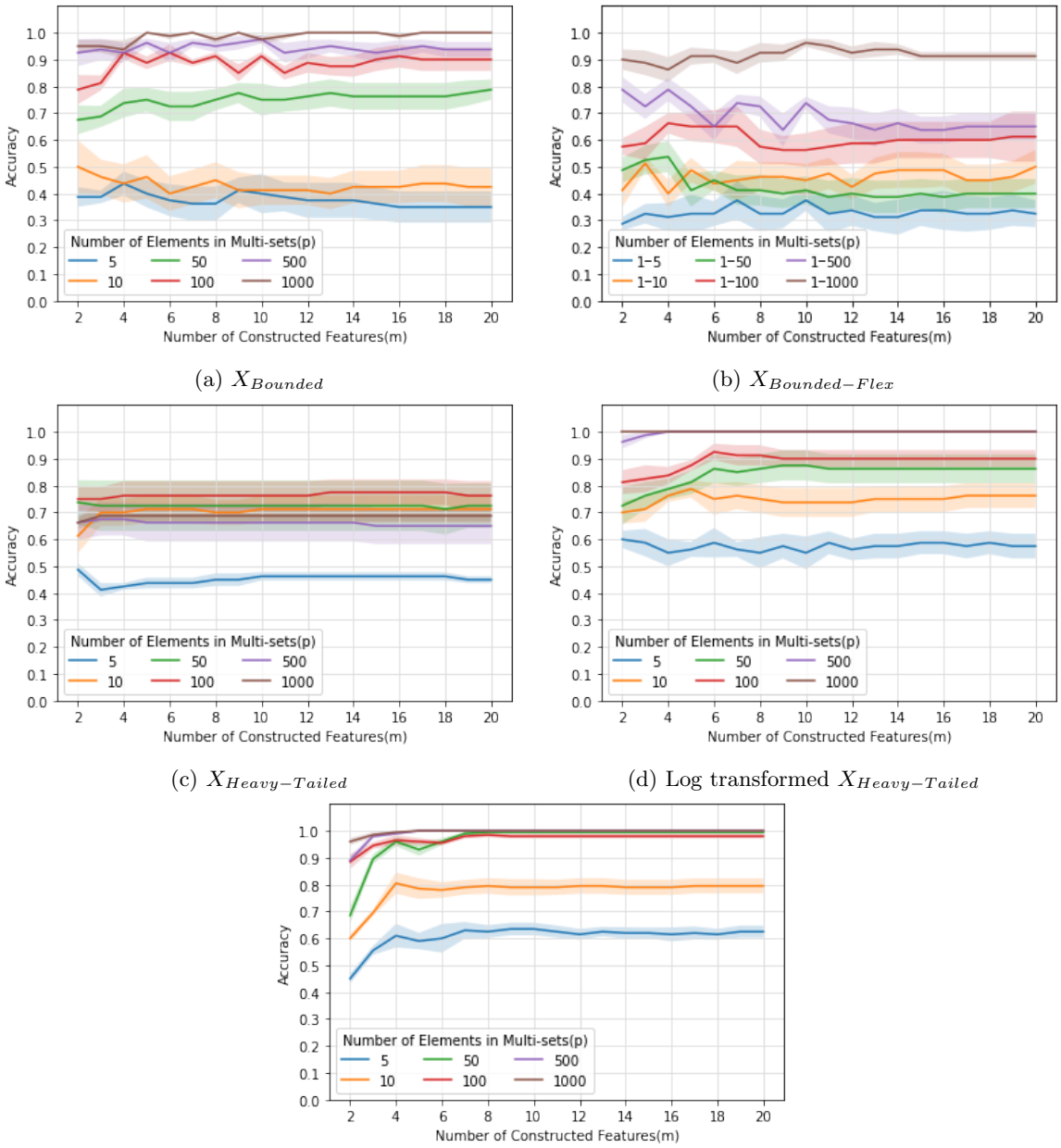
(d) Log transformed $X_{Heavy-Tailed}$



(e) $X_{Multi-Modal}$

Figure 29: SVM results: Accuracy of SVM models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by KDE.

Logistic regression results



(a) $X_{Bounded}$

(b) $X_{Bounded-Flex}$

(c) $X_{Heavy-Tailed}$

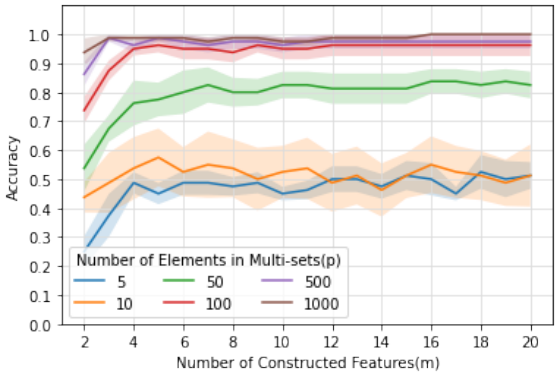
(d) Log transformed $X_{Heavy-Tailed}$

(e) $X_{Multi-Modal}$

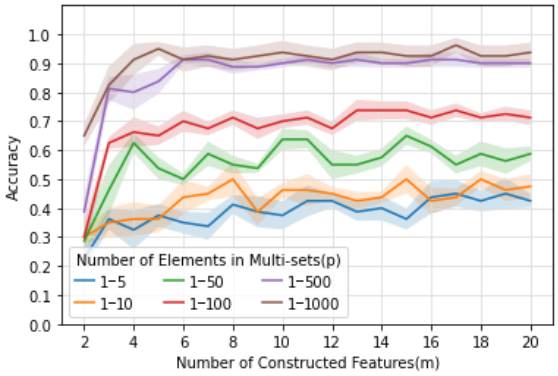
Figure 30: Logistic regression results: Accuracy of logistic regression models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by KDE.

Empirical cumulative distribution function

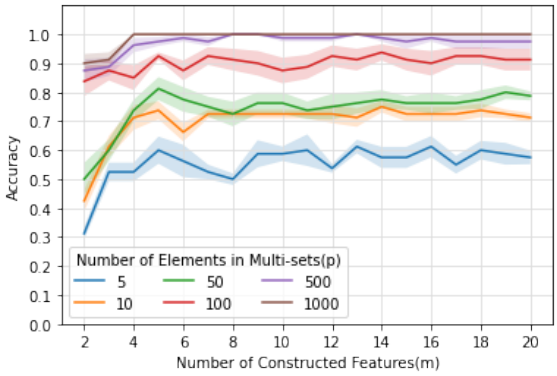
SVM results



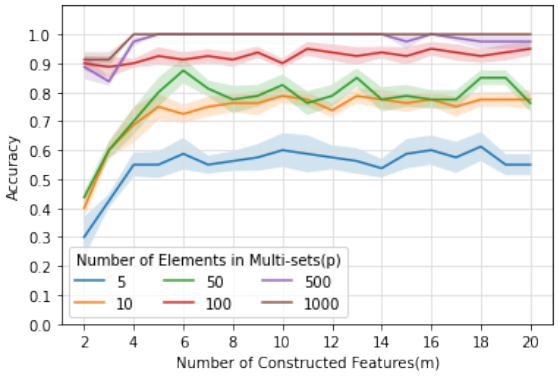
(a) $X_{Bounded}$



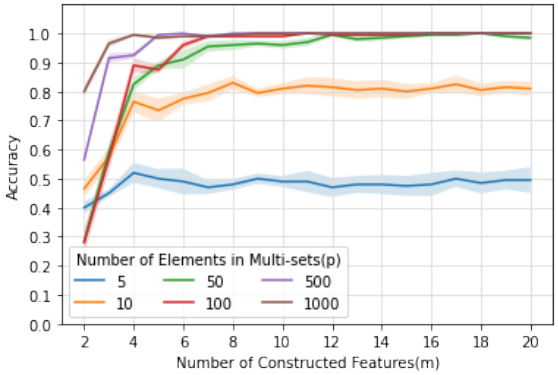
(b) $X_{Bounded-Flex}$



(c) $X_{Heavy-Tailed}$



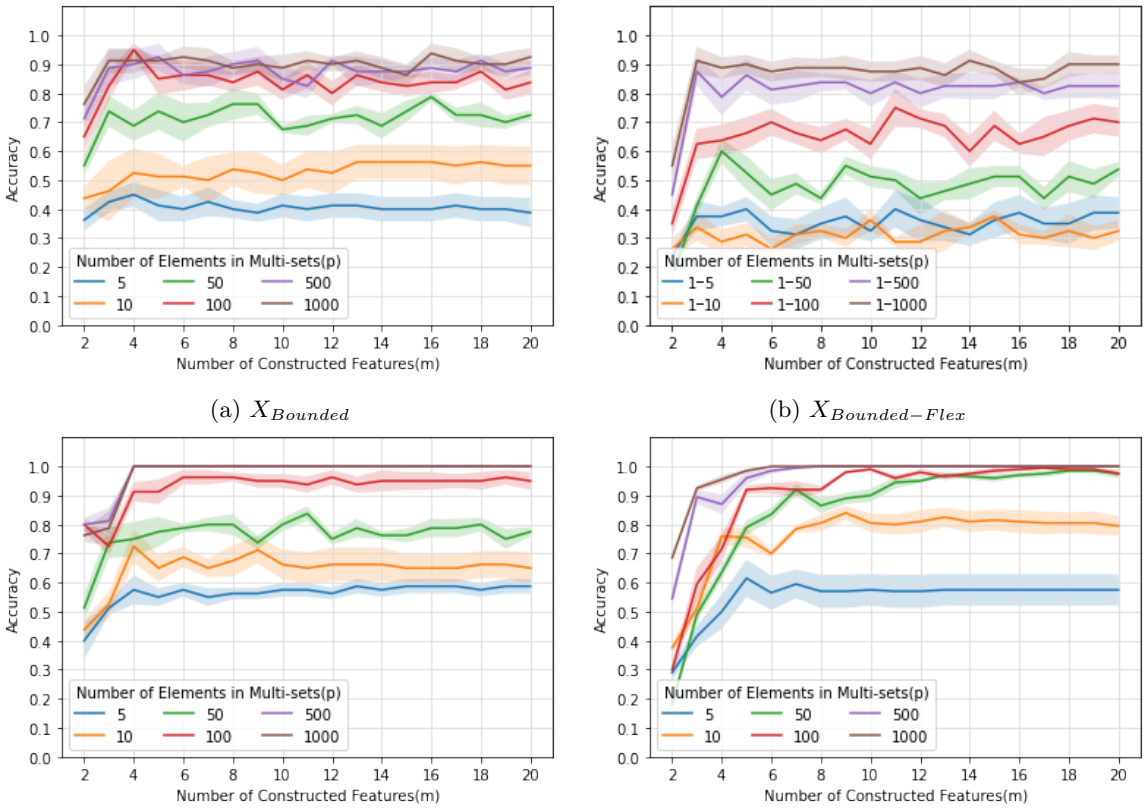
(d) Log transformed $X_{Heavy-Tailed}$



(e) $X_{Multi-Modal}$

Figure 31: SVM results: Accuracy of SVM models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by ECDF.

Logistic regression results



(a) $X_{Bounded}$

(b) $X_{Bounded-Flex}$

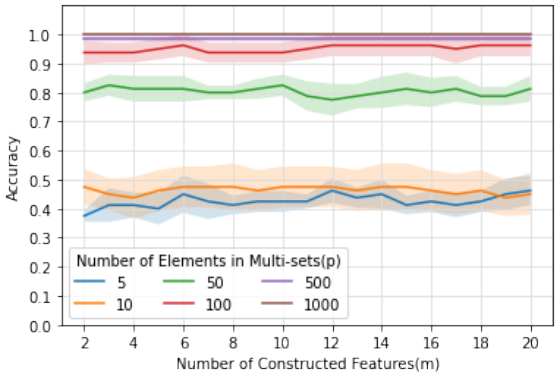
(c) Log transformed $X_{Heavy-Tailed}$

(d) $X_{Multi-Modal}$

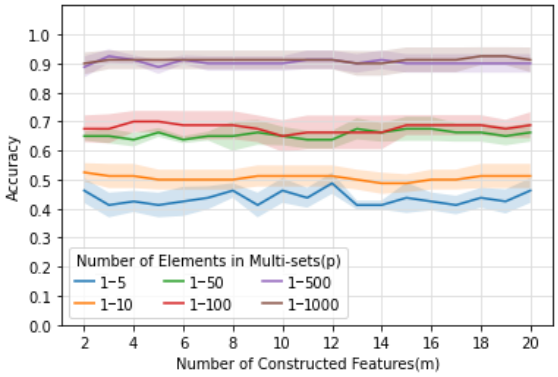
Figure 32: Logistic regression results: Accuracy of Logistic regression models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by ECDF.

Empirical characteristic function

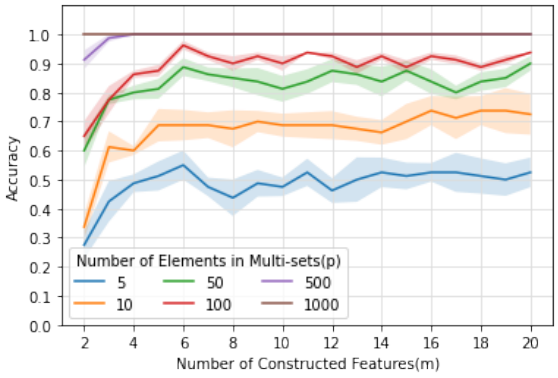
SVM results



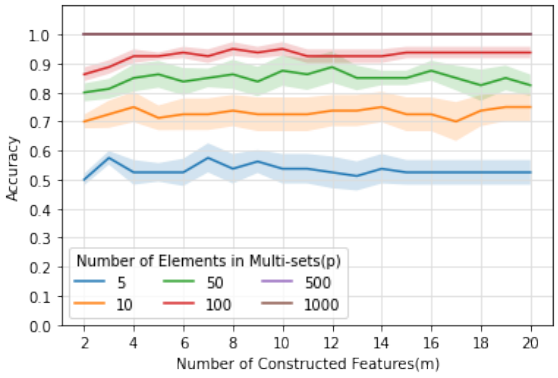
(a) $X_{Bounded}$



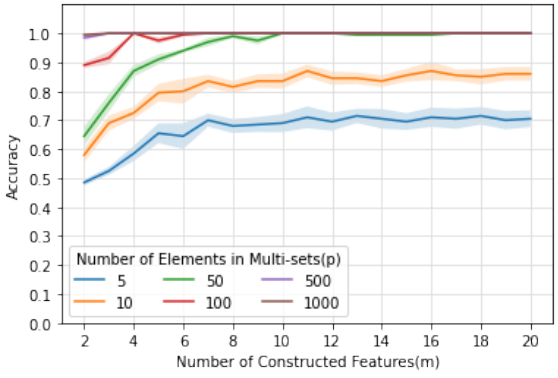
(b) $X_{Bounded-Flex}$



(c) $X_{Heavy-Tailed}$



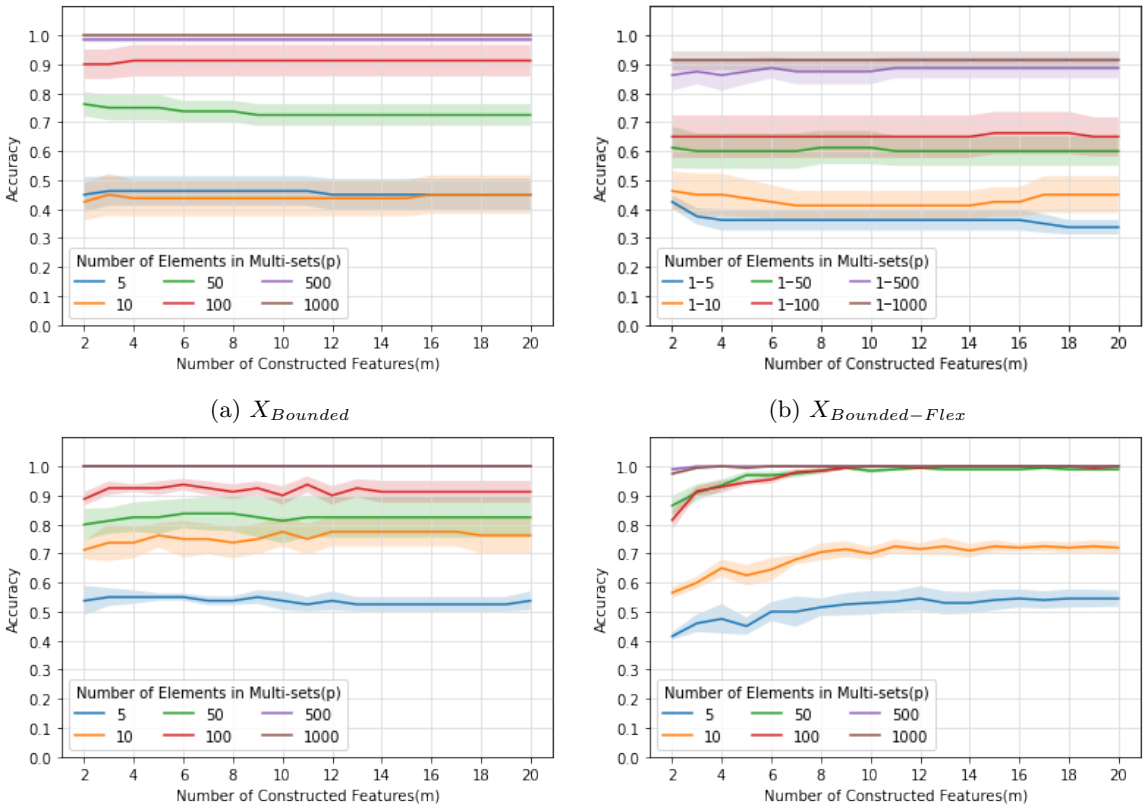
(d) Log transformed $X_{Heavy-Tailed}$



(e) $X_{Multi-Modal}$

Figure 33: SVM results: Accuracy of SVM models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by ECF.

Logistic regression results



(a) $X_{Bounded}$

(b) $X_{Bounded-Flex}$

(c) Log transformed $X_{Heavy-Tailed}$

(d) $X_{Multi-Modal}$

Figure 34: Logistic regression results: Accuracy of Logistic regression models for different number of elements(p) in multi-sets and number of features(m) constructed, where the features are constructed by ECF.

References

[1] Lutz Bornmann, Loet Leydesdorff, and Rüdiger Mutz. The use of percentiles and percentile rank classes in the analysis of bibliometric data: Opportunities and limits. *CoRR*, abs/1211.0381, 2012.

[2] Björn Böken. On the appropriateness of platt scaling in classifier calibration. *Information Systems*, 95:101641, 2021.

[3] Ciwan Ceylan, Kambiz Ghoorchian, and Danica Kragic. Digraphwave: Scalable extraction of structural node embeddings via diffusion on directed graphs, 2022.

[4] David Charte, Francisco Charte, Salvador García, María José del Jesus, and Francisco Herrera. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *CoRR*, abs/1801.01586, 2018.

[5] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances, 2017.

[6] T. W. Epps. Characteristic functions and their empirical counterparts: Geometrical interpretations and applications to statistical inference. *The American Statistician*, 47(1):33–38, 1993.

[7] Allan Gut. *An Intermediate Course in Probability*. 01 2009.

[8] Paul R. Halmos. The Theory of Unbiased Estimation. *The Annals of Mathematical Statistics*, 17(1):34 – 43, 1946.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[10] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Systematic ensemble model selection approach for educational data mining. *Knowledge-Based Systems*, 200:105992, 2020.

[11] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.

[12] Alan F. Karr. *Characteristic Functions*, pages 163–182. Springer New York, New York, NY, 1993.

[13] Jingyi Jessica Li and Xin Tong. Statistical hypothesis testing versus machine learning binary classification: Distinctions and guidelines. *Patterns*, 1(7):100115, 2020.

[14] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for nas, 2019.

[15] Gwo Dong Lin. Characterizations of distributions via moments. *Sankhyā. Series A. Methods and Techniques*, 54, 01 1992.

[16] D. C. Montgomery and G. C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley and Sons, 2003.

[17] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2, 2021.

[18] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, 1986.

[19] A. W. van der Vaart. *Empirical Processes*, page 265–290. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[20] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury Advanced Series, sixth edition edition, 2002.

[21] Geoffrey I. Webb. *Overfitting*, pages 744–744. Springer US, Boston, MA, 2010.

[22] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning, 2023.