LMU
LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK

FORSCHUNGSGRUPPE
DATA MINING IN DER MEDIZIN

Bachelor Thesis
in Computer Science

Attention in Mixed-Type Clustering

Jaanis Fehling

Aufgabensteller:	Prof. Dr. Christian Böhm
Betreuer:	Walid Durani
Abgabedatum:	tt.mm.yyyy

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

Munich, tt.mm.yyyy

. . . .
Jaanis Fehling

Abstract

This document serves as a model for the development of a thesis at the Department of Database Systems at the Institute for Computer Science at the LMU Munich. The abstract should not contain more than 300 words.

Contents

Chapter 1

Introduction

Clustering is an unsupervised machine learning method that groups similar observations together. Due to its ability to find patterns in an unlabeled dataset, its an essential task in Data Mining and Knowledge Discovery. A *cluster* is a group of similar instances that belongs to a *centroid* (center point of a cluster). Distance-based clustering algorithms use distance measures such as Euclidean distance to calculate the similarity of datapoints. Hierarchical methods partition the instances and merge (agglomerative) or split them into bigger or smaller clusters. Many other methods exist, but this work focuses on methods for clustering *mixed-type* data. [1]

Chapter 2

Related Work

Chapter 3

Foundation

3.1 Methodology

3.1.1 Datasets

In this work we use 8 mixed-type datasets from the UC Irvine Machine Learning Repository [13]. The Abalone dataset [16] contains physical measurements from abalones. It has 4177 instances, one categorial feature and seven continuous features. The Auction Verification dataset [17] has 2043 instances that contain verification runs of multi-round auctions. It is composed of six categorial and one continuous feauture. The Bank Marketing dataset [15] is related to a direct marketing campaign of a portuguese banking institution. It has 49732 instances, but was downsampled to 5000 random instances. The "age", "day" and "month" features were removed, which results in eight categorial and five continuous features. The Breast Cancer dataset [25] contains 699 instances and 9 categorial features. The Census Income dataset [9] has a total of 48842 instances. It was downsampled to 5000 random instances. It is composed of eight categorial and 6 continuous features. The Credit Approval dataset [21] contains information of applications for credit cards. It has 690 instances, nine categorial features and six continuous features. The Heart Disease dataset [8] is composed of four datasets and has 920 instances in total. It has seven categorial features and six continuous features. The Soybean (Large) Dataset [16] consists of 683 instances from soybeans with a certain disease and has 35 categorial features.

All instances containing missing values were removed. Duplicate instances were explicitly not removed, since there is no information available if they are duplicates by accident, or real duplicates. Categorial columns were standardized by removing the mean and scaling to unit variance, using the scikit-learn Python Library [19]. Formally, the standardized score z of a sample x from a feature is calculated as

$$z = \frac{(x - \mu)}{\sigma}$$

where μ is the mean of the samples $x_1, ..., x_N$ from a feature of length N, defined as

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

and σ is the standard deviation of the samples of a feature, defined as

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}.$$

The datasets were shuffled. For all random operations, a random state of integer value 0 was used to ensure reproducibility. When using the PyTorch Python Libary [18] for implementing neural networks, a flag was set to only use deterministic algorithms.

3.1.2 Evaluation

The clustering results were each evaluated with two measurents, *Accuracy* and *Normalized Mutual Information.* Accuracy is defined as

$$\frac{\text{Instances predicted correctly}}{\text{Total number of instances}}.$$

Since any clustering algorithm will assign arbitrary class labels to each instance that might not allign with the ground truth class labels, we map the predicted label classes to ground truth label classes in a way maximize the number of correctly predicted instances. This is an inversion of the *Linear Assignment Problem*, we use an implementation provided by the Scipy Python Library [24] to solve this problem and find the optimal solution.

As shown in Chapter 5, some clustering methods falsely assign almost all instances to one target class. Because a part of the datasets we used are heavily imbalanced, this leads to unjustified high accuracy scores. Therefore, we use another metric, Normalized Mutual Information. It is based on *Mutual Information*, which, for the ground truth labels U and the predicted labels V (switching both variables will not change the outcome) is defined as

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} log \frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

where $|U_i|$ is the number of instances assigned to cluster U_i and $|V_i|$ is the number of instances assigned to cluster V_i [19]. Normalized Mutual Information is Mutual Information normalized by the mean of the Entropies of U and Y [19]:

$$NMI(U,V) = \frac{MI(U,V)}{\frac{1}{2}(H(U)H(V))}.$$

3.2 Traditional Methods for Mixed-Type data

3.2.1 k-means

The most well known distance-based clustering method is k-means [11]. The goal is defined as follows: Suppose we have a finite set of n instances $S = \{p_1, p_2, ..., p_n\} \in \mathbb{R}^m$ for a dataset with m features, the target of k-means is to find optimal centroids $B = \{b_1, b_2, ..., b_k\} \subseteq \mathbb{R}^m$ for a given $k(\leq n) \in \mathbb{N}$ that minimize the sum of the squared Euclidean distance of each point in S to its nearest centroid. Formally

$$\sum_{i=1}^{n} d(p_i, B)$$

has to be minimized, where d is the Euclidean distance from a point $p_i \in S$ to the nearest centroid in B [12]:

$$d(p_i, B) = min_{1\leq j\leq k} d(p_i, b_j).$$

The Euclidean distance between two points p and q in an n-dimensional Euclidean space is defined as

$$d(p,q) = ||p - q|| = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}.$$

Finding the optimal centroids is a NP-hard problem, even for $d = 2$, as shown by Mahajan et al. [12]. The most common algorithm used for the k-means problem is a iterative refinement technique proposed by Lloyd [10]. It is defined as follows

1. Randomly set k initial cluster centroids $b_1^{(1)}, ..., b_k^{(1)}$.

2. Assign each obseration p_i to the nearest centroid using squared Euclidean distance. This splits our instances into S into k sets $\{S_1^{(t)}, ..., S_k^{(t)}\}$.

3. Recalculate the optimal position of each centroid using the mean distance to each instance assigned to the centroid:

$$b_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{p_j \in S_i^{(t)}} p_j.$$

4. Repeat steps 2. and 3. until the centroid assignments no longer change.

3.2.2 k-modes

In many real-world scenarios, besides continuous, numerical data, *categorial* data exists. While Euclidean distance or other distance measures work well with continuous data, categorial data is different. Suppose we have categories $\{A, B, C\}$ of a given feature, we would encode them into numeric values to allow for computation of a distance measure:

$$\{A, B, C\} \equiv \{1, 2, 3\}.$$

While A and C can share the same semantic similarity as A and B, numerically category A and Category C are now $|1 - 3| = 2$ apart, while Category A and B are only $|1 - 2| = 1$ apart. During clustering, this could lead to instances being assigned to centroids based on a wrong distance assumption.

A possible solution is to use *one-hot encoding*, also known as *dummy coding* in classical statistics. One-hot encoding turns a discrete feature containing k mutually exclusive categories into a vector x of length k, in which only one of the elements x_k equals 1 and all remaining elements equal 0 [3]. For an instance B of a feature having $k = 3$ separate categories $\{A, B, C\}$, the one-hot vector x would be represented by $x = (0, 1, 0)^\intercal$.

According to Huang [6], one-hot encoding has two drawbacks:

1. In real-world applications, categorial features with hundreds or thousands of categories are encountered. This would result in a large number of binary features in the one-hot encoded representation, which will increase cost and space of computation.

2. The centroid value of a certain one-hot encoded feature, given by a real value between 0 and 1, cannot indicate the characteristics of the according cluster, since the feature only describes the presence or absence of one category.

Therefore, Huang [6] proposed using the Kronecker-Delta as a dissimilarity measure between multiple categorial columns. Formally, if we have two instances X and Y of a dataset with m categorial features, d_1 will count the number of mismatches between the categorial features of both instances, defined as

$$d_1(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_j)$$

where the Kronecker delta $\delta(x_j, y_j)$ is defined as

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}.$$

If we have a finite set of n instances $S = \{p_1, p_2, ..., p_n\}$ for a dataset with m categorial features, the goal of *k-modes* [6] is to find optimal *modes* $B = \{b_1, b_2, ..., b_k\}$ for a given $k(\leq n) \in \mathbb{N}$ that minimize

$$\sum_{i=1}^{n} d_1(p_i, B)$$

where

$$d_1(p_i, B) = min_{1 \leq l \leq k} \sum_{j=1}^{m} \delta(p_{i,j}, b_{l,j}).$$

Similar to k-means, we can use an iterative algorithm for efficient computation [6]:

1. Randonly choose k instances from the dataset as initial modes for the clusters.

2. Assign each instance to their nearest mode using the proposed dissimilarity measure one by one and update the mode of each cluster after each assignment.

3. Test if each instance still belongs to its assigned mode, i.e. if each instance is assigned to its nearest mode. If the instance would belong to a different mode, reassign the instance and update the modes of both clusters.

4. Repeat step 3. until the mode assignments no longer change.

3.2.3 k-prototypes

As proposed by Huang [6], it is straightforward to combine the k-means and k-modes algorithms into the *k-prototypes* algorithm, which can be used to cluster *mixed-type* data (consisting of numerical, continuous and categorial features). The dissimilarity between two instances X and Y with features $A_1^r, A_2^r, ..., A_s^r, A_{s+1}^c, ..., A_m^c$, where features $A_1^r, ..., A_s^r$ are continuous and features $A_{s+1}^c, ..., A_m^c$ are categorial, is defined as

$$d_2(X, Y) = \sum_{j=1}^{s} (x_j - y_j)^2 + \gamma \sum_{j=s+1}^{m} \delta(x_j, y_j).$$

The first part of the equation is the Euclidean distance as used in k-means, while the second part is taken from the k-modes algorithm. Huang [6] states: "The weight γ is used to avoid favouring either type of attribute".

Again, we need to find k optimal centroids $B = \{b_1, b_2, ..., b_k\}$ and therefore have to minimize

$$\sum_{i=1}^{n} d_2(p_i, B)$$

where

$$d_2(p_i, B) = min_{1 \le l \le k} \sum_{j=1}^{s} (p_{i,j} - b_{l,j})^2 + \gamma \sum_{j=s+1}^{m} \delta(p_{i,j}, b_{l,j}).$$

We can minimize both distance measures at the same time since they are nonnegative. Therefore, we can use the same algorithm as defined in 3.2.2. [6]

3.2.4 Gower distance

Gower distance [5] is a general similarity measurement between instances containing mixed-type features. It is defined as follows: When comparing instances x_i and x_j, for each feature k of p total features, we calculate a score $s_{ijk} \in [0, 1]$. The score will be close to 1 for two instances x_{ik} and x_{jk} of a feature k if they are similar, and close to 0 they are not similar. Gower distance is also computable between instances with missing values, therefore a quantity δ_{ijk} is calculated, which is equal to 1, when feature k can be compared across the two instances x_i and x_j, and 0 otherwise (illustrated in Figure 3.1). Gower distance then is the average of the known score

$$S_{ij} = \frac{\sum_{k=1}^{p} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}}.$$

The Score s_{ijk} is calculated differently according to the type of feature [5]:

1. For *dichotomous* (when a value is either present or absent) features, the score s_{ijk} is 1 when the value is present in both features and 0 otherwise, as shown in Figure 3.1.

2. For categorial features, the score s_{ijk} is 1 if they both instances match on feature k and 0 otherwise.

3. For continuous features the score is calculated as

   $$s_{ijk} = 1 - \frac{|x_i - x_j|}{R_k}$$

   where R_k is the range of feature k in the dataset or in the sample.

SCORES AND VALIDITY OF DICHOTOMOUS CHARACTER COMPARISONS

	Values of character k			
Individual i	+	+	−	−
j	+	−	+	−
s_{ijk}	1	0	0	0
δ_{ijk}	1	1	1	0

Figure 3.1: Score s_{ijk} and quantity δ_{ijk} of a feature k on two instances x_i and x_j. Presence of a feature is denoted by "+" and absence by "-". [5]

Gower [5] has shown that $\sqrt{1 - S_{ij}}$ is a valid distance representation for two instances x_i and x_j. We can now convert our similarity matrix S into a distance matrix and are able to use Hierarchical clustering methods [7]. Philip and Ottaway [20] used Gower distance with agglomerative clustering. Agglomerative clustering places each instance into its own cluster and recursively merges the clusters together using the given distance matrix, until only the specified number of clusters is remaining [7].

3.3 Deep Clustering

3.3.1 Neural Networks

The idea of computation by neurons inspired by the human brain was first formalized into a mathematical model by McCulloch [14]. A *neuron* was defined as a element that takes multiple boolean inputs and has one boolean output. The neuron *fires*, meaning the output is set to true, when the sum of the input values extends a certain threshold.

The single-layer *perceptron*, the first neural machine learning algorithm, was invented by Rosenblatt in 1957 [22]. Formally, the binary valued output o_j given an input vector x_i is calculated as

$$o_j = \begin{cases} 1 & \sum_i w_{ij} x_i + b > 0 \\ 0 & otherwise \end{cases}$$

where w is the learnable weight matrix, and b is a predefined bias. For training, the weights w are simply incremented when the output is 0 but the ground truth is 1, and decremented if the output is 1 but should be 0. If the output was predicted right, no weights are changed.

The idea of neural networks was revived three decades later, using multiple perceptron layers and a differentiable error function [23]. In a multi-layer neural network, we have an input layer, an output layer and multiple hidden layers. Each layer is a collection of neurons, that acquire the outputs of each neuron from the previous layer (or from the input in case of the input layer) as their input, and produce a new output. Formally, the output a_j of the jth neuron of layer n that gets d input values from the previous layer is defined as

$$a_j = \sum_{i=1}^{d} w_{ji}^{(n)} x_i + b_j^{(n)}$$

where $w_{ji}^{(n)}$ is the learnable weight of the input x_i going through neuron j in layer n [2]. A bias b_j is also added. This output is commonly refered to as an *activation* of a neuron [2]. Because every neuron is a linear function, in order to avoid the collapse of each neuron from all layers into a single linear function, we pass the activation a_j into a non-linear activation function f

$$z_j = f(a_j)$$

before being passed to the next layer of neurons [4].

As shown in a recent survey [4], there are many different activation functions used in neural networks. A simple example for an activation function, which naturally alligns with the idea of a biological neuron firing is the *step function* (also known as *Heaviside step function*) [2], that ouputs 0 for negative values and 1 for positive values:

$$\text{step function}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}.$$

In order to stay between 0 and 1, but also utilize all real values inbetween, another activation function that has historicaly been popular is the *logistic sigmoig* function [4], that squashes any input in $]0, 1[$:

$$\text{logistic sigmoid}(x) = \frac{1}{1 + e^{-x}}.$$

Because the ouputs are in a range close to 0, it is prone to the *vanishing gradient problem*. In the vanishing gradient problem, a gradient that is very close to 0 leads to almost no update in the weights of the network during training. Moreover, using a exponential value naturally leads to a greater computational complexity. [4]

The current state-of-the-art activation function, the *Rectified Linear Unit* (ReLU) [4], is not limited by the above disadvantages. It is simple and computationally performant. It is defined as the identity for positive values and as 0 for negative values:

$$\text{ReLU}(x) = max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}.$$

One downside of the ReLU activation function is that there will be a substantial amount of dead neurons in the network (neurons with output 0 will not affect the neurons of the next layer). The *Leaky Rectfied Linear Unit* (Leaky ReLU or LReLU) [4] utilizes negative values aswell, but with a small coefficient, usually set to 0.01. It is defined as

$$\text{Leaky ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0.01 \times x & x < 0 \end{cases}.$$

All four activation functions are illustrated in Figure 3.2.

3.3.2 Autoencoders

3.3.3 Deep clustering methods

3.3.4 Attention

3.3.5 Deep clustering methods

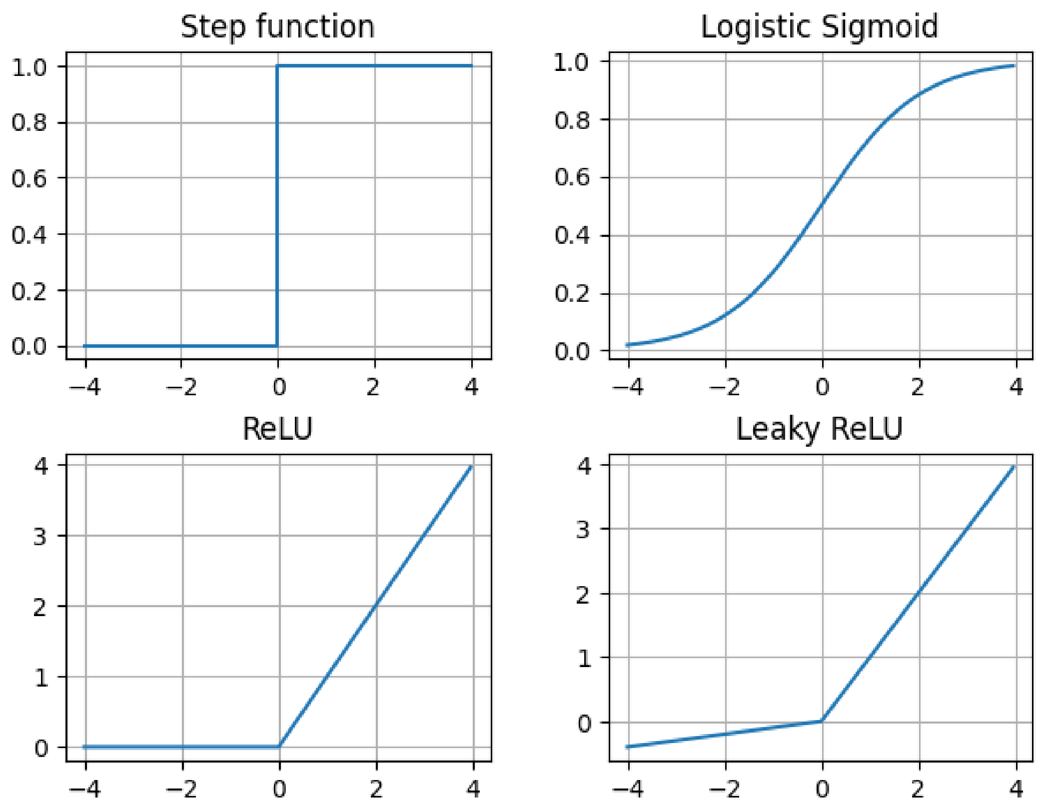

Figure 3.2: An illustration of the step function, the logistic sigmoid, the ReLU and the Leaky ReLU activaiton functions.

Chapter 4

Attention in Mixed-Type Clustering

Chapter 5

Experiments

5.1 Comparison of classical Clustering Methods

Our first comparison is between classical clustering methods for clustering mixed-type data. We evaluated k-means, k-means with one-hot encoding, k-prototypes and gower distance with agglomerative clustering, using average linkage (average distance between each instance of two sets).

	k-means	k-means one-hot	k-prototypes	Gower distance
Abalone	0.1718	**0.1740**	0.1716	0.1614
Auction Verification	**0.0162**	0.0071	0.0077	0.0062
Bank Marketing	0.0198	**0.0261**	0.0195	0.0013
Breast Cancer	**0.7468**	0.7363	0.5925	0.5537
Census Income	0.1080	**0.1850**	0.1417	0.0043
Credit Approval	**0.3131**	0.1710	0.1166	0.0035
Heart Disease	**0.2046**	0.1645	0.1893	0.1408
Soybean Disease	0.6722	**0.7102**	0.5676	0.6695

Figure 5.1: Comparsion of Normalized Mutual Information of various classical methods on clustering mixed-type datasets.

	k-means	k-means one-hot	k-prototypes	Gower distance
Abalone	0.1353	0.1314	0.1343	**0.1954**
Auction Verification	0.6647	0.5761	0.5805	**0.8008**
Bank Marketing	0.7796	0.7866	0.7872	**0.8842**
Breast Cancer	**0.9605**	0.9502	0.9151	0.9004
Census Income	0.6082	0.6976	0.6256	**0.7684**
Credit Approval	**0.8086**	0.7060	0.6662	0.5482
Heart Disease	0.3344	0.3211	0.4247	**0.5652**
Soybean Disease	0.5765	**0.5996**	0.4715	0.501

Figure 5.2: Comparsion of Accuracy of various classical methods on clustering mixed-type datasets.

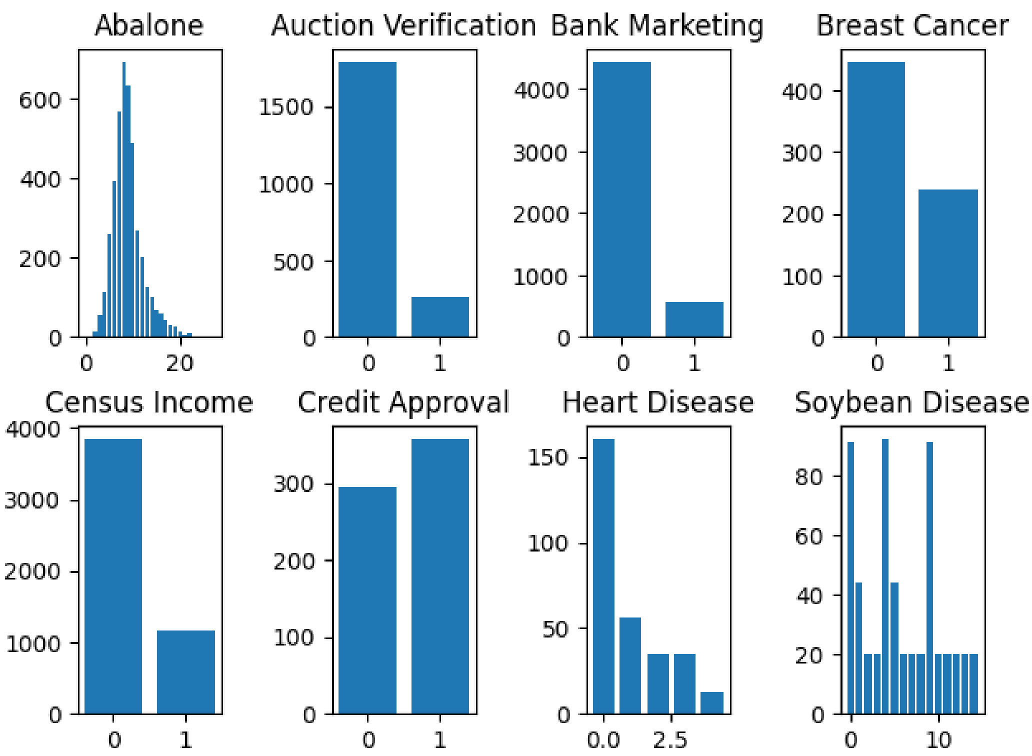

Figure 5.3: Number of instances of each target class of each dataset used.

	No Column Embedding AE	All Columns AE	Categorial Columns AE	All Columns to Categroial Columns AE
Abalone	0.1560	0.1588	**0.1854**	0.1709
Auction Verification	0.0003	0.0162	0.0045	**0.0174**
Bank Marketing	0.0015	0.0168	**0.0182**	0.0003
Breast Cancer	**0.7602**	0.7203	0.7180	0.6309
Census Income	0.0000	0.0048	**0.0460**	0.0025
Credit Approval	0.0072	**0.0111**	0.0004	0.0034
Heart Disease	0.1852	0.1581	**0.1858**	0.0989
Soybean Disease	0.4793	**0.5955**	0.4261	0.4041

Figure 5.4: Comparsion of Normalized Mutual Information of various Autoencoder architectures combined with k-means on clustering mixed-type datasets.

	No Column Embedding AE	All Columns AE	Categorial Columns AE	All Columns to Categroial Columns AE
Abalone	0.1245	0.1439	0.1448	**0.1513**
Auction Verification	0.5507	0.6074	0.6466	**0.6676**
Bank Marketing	0.6000	0.5918	0.5968	**0.6684**
Breast Cancer	**0.9634**	0.9531	0.9546	0.9283
Census Income	**0.7198**	0.5004	0.6930	0.6102
Credit Approval	**0.5544**	0.5115	0.5360	0.5421
Heart Disease	0.4080	0.3579	**0.4114**	0.2943
Soybean Disease	0.3879	**0.4466**	0.3577	0.3665

Figure 5.5: Comparsion of Accuracy of various Autoencoder architectures combined with k-means on clustering mixed-type datasets.

Chapter 6

Conclusion

Bibliography

[1] Amir Ahmad and Shehroz S. Khan. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*, 7:31883–31902, 2019.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[5] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971.

[6] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2:283–304, 1998.

[7] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988.

[8] Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. Heart Disease. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C52P4X.

[9] Ron Kohavi. Census Income. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5GP7S.

[10] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[11] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[12] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. *Theoretical Computer Science*, 442:13–21, 2012. Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009).

[13] Kolby Nottingham Markelle Kelly, Rachel Longjohn. The UCI Machine Learning Repository. `https://archive.ics.uci.edu`. Accessed: 2023-07-16.

[14] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[15] S. Moro, P. Rita, and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5K306.

[16] Warwick Nash, Tracy Sellers, Simon Talbot, Andrew Cawthorn, and Wes Ford. Abalone. UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C55C7W.

[17] Elaheh Ordoni, Jakob Bach, Ann-Katrin Fleck, and Jakob Bach. Auction Verification. UCI Machine Learning Repository, 2022. DOI: https://doi.org/10.24432/C52K6N.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

[20] G. Philip and B. S. Ottaway. Mixed data cluster analysis: An illustration using cypriot hooked-tang weapons. *Archaeometry*, 25(2):119–133, 1983.

[21] Quinlan Quinlan. Credit Approval. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5FS30.

[22] F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[25] William Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5HP4Z.