Gin-erative Recipes

Michael Gamston [*] [1]

Abstract

This study investigates the application of Variational Autoencoders (VAEs), and their variants, the β-VAE and Conditional VAE, to generate gin recipes. Using a dataset containing botanicals and compounds along with their flavour profile descriptions, this research explores two generation strategies: class-based generation using prototype vectors and element-wise generation based on individual recipes. Clustering techniques were first employed to recategorise the dataset, both to expand the number of usable samples and to simplify class assignments. The generative models performance was assessed with prediction based metrics, a new disentanglement metric and visual latent space analysis. Results indicate that while the β-VAE demonstrated greater success in disentangling class features within the latent space, none of the models were able to generate high-quality, diverse recipes, with posterior collapse and dataset structure emerging as key limitations. The findings highlight both the potential and the challenges of applying generative models to recipe design, suggesting that future work should prioritise larger and more balanced datasets with the use of continuous ingredient measurements and the integration of consumer preference modelling to enhance industrial applicability.

1. Introduction

The application of artificial intelligence to culinary science has advanced rapidly in recent years, with machine learning methods being applied to diverse tasks such as food pairing, ingredient recommendation, and recipe prediction. Algorithms ranging from clustering and traditional classifiers to deep learning models have demonstrated their potential to uncover meaningful associations between ingredients to generate recipes. (Park et al., 2019) (Pyo, 2025) Despite this progress, the use of deep generative models for recipe creation remains relatively under explored. Existing approaches often rely on techniques that connect ingredients without modelling the underlying data distributions that govern flavour. This highlights the potential of generative models, particularly VAEs and their variants, to enable more structured and interpretable recipe generation.

VAEs have been successfully applied in a wide range of domains, from image generation and molecular modelling to medical imaging and data imputation (Dohi, 2020) (Richards & Groener, 2022) (Barrejon et al., 2022). Their probabilistic nature offers a way of learning latent representations, which capture the underlying structure of data distributions. Unlike adversarial approaches such as GANs, VAEs optimise a well defined loss function, allowing more stable training. As well as this, extensions such as the β-VAE and Conditional VAE (CVAE) address limitations of the standard VAE by encouraging disentanglement and enabling higher accuracy class based generation respectively.

This project investigates the potential of VAEs, β-VAEs, and CVAEs for generating gin recipes, using class descriptions and botanical features. Two generation strategies are explored: class-based generation, in which prototype vectors are used to generate new samples, and element-wise generation, which encodes the individual recipes and then reconstructs them with noise, to provide sample-by-sample based generation. By comparing the performance of these approaches across VAE variants, this study aims to evaluate the ability of generative models to capture and reproduce meaningful class distinctions within the gin dataset. This work also highlights the challenges posed by imbalanced and limited datasets, as well as the potential for future integration of consumer preference modelling to enable data driven product development.

2. Related Projects

Deep learning and tradition machine learning has already been applied across many culinary based tasks. Tasks like food pairing have been explored using algorithms types such as clustering, deep neural networks and diffusion models to predict good pairings of ingredients and produce recommendations. (Park et al., 2021) (Park et al., 2019) (Pyo, 2025) Although, actual uses of deep generative models in

UoN [1]Department of Physics and Astronomy, University of Nottingham, UK, NG72RD.

recipe design are quite limited and remain under explored. Research in this area could allow for customisable recipe generation based on desired flavour or nutritional properties. Currently most AI driven food design methods focus on associating ingredients, often lacking deeper modelling of the underlying factors of the data. (Al-Sarayreh et al., 2023)

Some of the current research includes using Variational auto encoders to generate recipes that are based off nutritional content rather than ingredients. Allowing the user to input desired nutritional content and receive a recipe describing ingredients that match the constraints. (Lei et al., 2020)

There is also research into using genetic algorithms to create recipes, then evaluating them with other machine learning algorithms such as random forest, support vector machines and neural networks. This research has boasted to have seen some success. (Zhang et al., 2019)

Deep learning with VAEs can be been used to learn the important features of recipes that customers like. Running traditional classifiers on the reduced dataset of important features produce by a VAE facilitates the prediction of customer satisfaction. Allowing producers the ability to get feedback before producing a product. (Bi et al., 2020)

Generalisation can become a issue with generation tasks when the latent space is learnt from a small dataset. Even large pre-trained generative transformers struggle to generalise when using transfer learning on a small data set.(Katserelis & Skianis, 2022) This being said, fine tuning pre-trained generative transformers with transfer learning has seen some success in recipe generation.(H. Lee et al., 2020) The benefit of using natural language processing models such as this, is that the algorithm can also be trained to provide cooking instructions with the recipe, allowing a more user friendly experience.

Generative AI has been used to produce drink recipes. An app called barGPT uses pre-trained generative transformers to create cocktails. Another project uses recurrent neural networks to successfully produce cocktail recipes. (Bojar, 2019) However, in blind taste tests consumers could usually identity the cocktail recipe generated by AI. Highlighting the need for more research in this area. (George, 2024)

Variational Autoencoders have been used across many different problem spaces. With project ranging from partial detection at CERN (Cheng et al., 2023), de-noising and populating datasets with missing values (Barrejon et al., 2022) and image generation for realistic physics simulations (Dohi, 2020). Most use-cases of Variational Autoencoders are image generation. With the MINST dataset being a benchmark for performance. (Chou, 2019) (Cao et al., 2021) (Akoury & Nguyen, 2017)

There are many variations of the Variational Autoencoder, with the β Variational Autoencoder being a popular choice. It's been used for many different tasks like medical imaging to segment images for greater interpretability (Shon et al., 2022) for example it has been used to reconstruct 3D lung nodule images to help image analysis (Li Y, 2024). As well as being used to recreated molecular structures (Richards & Groener, 2022). This method is popular due to its ability to learn the underlying factors of the data better than the standard VAE. (Burgess et al., 2018)

Research into Variational Autoencoders and their variants for generating gin recipes is a worthwhile pursuit given their success in other generational disciplines, as well as the lack of research within recipe generation. Past projects have shown that generative AI can be successfully applied to recipe generation, although the research is still in the early stages and requires further development.

3. Evaluation Metrics

This section explains the evaluation metrics that will be used in this paper.

3.1. Mean Square Error

Mean Square Error (MSE) measures the mean squared distance between estimated values and predicted values. This metric is often used with regression problems. Where Y is the prediction and $\hat{Y}$ is the true value.

$$MSE = \frac{1}{N}\sum_{i}^{N}(Y_i - \hat{Y}_i)^2 \tag{1}$$

3.2. Binary Cross Entropy

Binary Cross Entropy (BCE) measures the predicted probability $[0 \rightarrow 1]$ against the true label $[0, 1]$. This produces a loss that decreases as the predicted probability approaches the actual label. This metric is generally used with classification problems. Where Y is predicted probability and $\hat{Y}$ is the true label.

$$BCE = -\frac{1}{N}\sum_{i}^{N}(Y_i \cdot log(\hat{Y}_i) + (1 - Y_i) \cdot log(1 - \hat{Y}_i)) \tag{2}$$

3.3. Accuracy

This metric measures the percentage of correct predictions out of all predictions. This metric can sometimes be skewed and inaccurate if the data is imbalanced.

$$Accuracy = \frac{Correct\ Pedictions}{All\ Predictions} \tag{3}$$

3.4. Precision

This metric displays the percentage of correctly identified positive predictions out of all positive predictions.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (4)$$

3.5. Recall

This metric displays the percentage of positive predictions the model is correctly identifying out of all the total possible positives.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (5)$$

3.6. Silhouette Score

The Silhouette score measures how well each point in a set of data X fits within its assigned cluster. Producing a score in the range of $[-1 \to 1]$, where a higher score indicates a better fit. It does this by computing the distance $a(X_i)$ each point X_i is from every other point in the same cluster, this is the intra-cluster distance and represents how compact the cluster is. It then computes the average distance $b(X_i)$ the point X_i is from each point in every other cluster, this represents the cluster separation. The Silhouette score $s(X)$ is computed by combining these terms, as show in equation 6, and then taking the average score over every point. A good score will occur when $a(X_i)$ is small and $b(X_i)$ is large.

$$s(X) = \frac{1}{|X|} \sum_{i \in X} \left(\frac{b(X_i) - a(X_i)}{max(a(X_i), b(X_i))} \right) \quad (6)$$

(Kaufman & Rousseeuw, 1990)

3.7. Davies Bouldin Index

Davies–Bouldin Index produces a score that also measures cluster compactness and separation by measuring the average worst-case overlap between two clusters. It does this by first calculating a clusters compactness S_i, see equation 7. Where C_i is the cluster and $\bar{x_i}$ is the centroid for that cluster.

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} ||x - \bar{x_i}|| \quad (7)$$

Then calculating the inter-cluster distance between the cluster being assessed C_i and another cluster C_j from the set, with equation 8. Where $\bar{x_i}$ is centroid for C_i and $\bar{x_j}$ is centroid for C_j.

$$M_{ij} = ||\bar{x_i} - \bar{x_j}|| \quad (8)$$

After this, computing the similarity between these two clusters using equation 9.

$$R_{ij} = \frac{S_i + S_j}{M_{ij}} \quad (9)$$

Repeating the above equations until the cluster C_i has been compared with every other cluster in the set. Then selecting the comparison with highest similarity with equation 10. This finds the cluster that C_i is most blurred with.

$$R_i = max_{j \neq i}(R_{ij}) \quad (10)$$

Finally repeating the above equation until every cluster in the set X has been assessed and averaging the resulting similarities to find the Davies Bouldin score.

$$db(X) = \frac{1}{|X|} \sum_{i \in X} R_i \quad (11)$$

This results in a value in the range of $[0 \to \infty)$, where 0 indicates perfectly separated, compact clusters. A higher score indicates that clusters have poor inter-cluster separation and/or a high intra-cluster spread.

(Davies & Bouldin, 1979)

3.8. Elbow Point

The Elbow methods is used to help find the optimal value for k in KMeans. This method works by first choosing a sequential range of values for k, then calculating the within cluster sum of squared errors (WCSSE) known as inertia, see equation 12. Then plotting these in on a graph with the values of k. A sharp change in gradient indicates a optimal value for k, this is know as the elbow.

$$WCSSE = \sum_{i}^{k} \sum_{x \in C_i} (x - \bar{x_i})^2 \quad (12)$$

Where C_i is a cluster and $\bar{x_i}$ is the centroid of that cluster.

(Manuel Fritz, 2019)

3.9. t-distributed Stochastic Neighbour Embedding plots

t-distributed Stochastic Neighbour Embedding (t-SNE) plots are ways of compressing high dimensional data down into two dimensions for the purpose of visualisation. They work by converting distances between points into distributions using a modified Student-t distribution, then plotting these points into low dimensional space and computing the the distribution again. They use the Kullback Leibler (KL) divergence to minimise the difference between these distributions. The result is a plot where the distances between points are persevered, revealing clusters in an interpretable way.

(van der Maaten & Hinton, 2008)

4. Methodology

4.1. KMeans

KMeans was first published in a 1967 paper by James MacQueen (MacQueen, 1967), it was based on an algorithm developed by Stuart Lloyd at Bell labs in 1957, although Lloyd did not publish it until 1982. (Lloyd, 1982)

KMeans is a partition based clustering algorithm. Partition algorithms aim to find k number of partitions that best separate the data. (Xu & Wunsch, 2005) In the case of KMeans tries is find the optimal position of k centroids, and then assign data points into clusters based on which centroid is closest. The clustering process for KMeans is as follows:

- Step 1: k centroids are assigned randomly within the feature space.
- Step 2: Each data point is then assigned to its nearest centroid to create a cluster.
- Step 3: The average of each cluster is then computed and the centroid is moved to this position.
- Step 4: Steps 2 and 3 are then repeated until the centroids converge or the maximum number of iterations is reached.

4.2. Agglomerative Hierarchical Clustering

Hierarchical clustering was first published in 1948 by T. Sorensen. Where he introduced the Sorensen coefficient used to group plant samples based on their species content. (Sorensen, 1948)

Agglomerative refers to the bottom up approach of this hierarchical method. Agglomerative Hierarchical Clustering (AHC) starts by pairing data points together based on their distance from each other, this is usually but not exclusively measured by Manhattan or Euclidean distance, these pairs are then paired up again based on the distance metric. This continues until all data points are contained within one cluster. This creates a hierarchical set of clusters, known as a dendrogram. Figure 1 shows and example of a dendrogram. It shows the data points being joined together based on their proximity. The dendrogram can then be "sliced" at different points to produce different numbers of clusters. (Murtagh & Contreras, 2017)

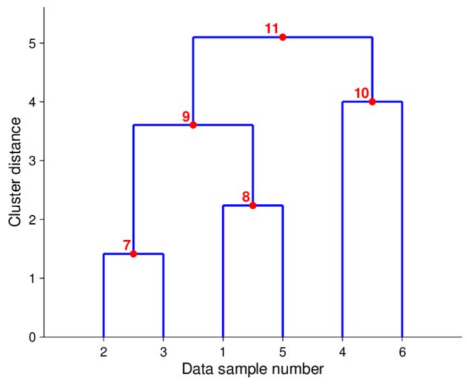


Figure 1. A example of a dendrogram. This figure has been taken from "Hierarchical Clustering for Large Data Sets" by Embrechts et al. (Embrechts et al., 2013)

4.3. DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) was first introduced in 1996 by Ester et al. It is a density-based clustering algorithm. Unlike methods such as K-Means, it does not require specifying the number of clusters in advance. Instead, it relies on two parameters:

- ϵ: the neighbourhood radius (maximum distance to consider another point a neighbour)
- minPts: the minimum number of neighbours required for a point to be considered a core point

How DBSCAN works:

- If at least minPts neighbours are within distance ϵ, mark it as a core point.
- Group all points that are reachable from any core point (through chains of neighbours) into the same cluster.
- Remaining points within ϵ of a cluster but not core points but included in the cluster and marked border points.
- Remaining points that do not meet these conditions are labelled as noise and are discard.

(Ester et al., 1996)

4.4. Bag of Words

Bag of Words is the earliest version of a vectorised representation of words. First introduced in a 1954 paper by Zellig Harris (Harris, 1954). It simple creates a unordered collection of words and their frequency within a document. It works by:

- Step 1: Building a vocabulary of all the unique words in a dataset. The number of unique words is then used to set the x dimension of the vector.
- Step 2: For each document present in the dataset, count how many times each word appears.
- Step 3: Represent the document as a vector of size x, that contains the count of each word present within document.

(Salton et al., 1975)

4.5. TF-IDF

TF-IDF stands for Term Frequency – Inverse Document Frequency. Using a similar vectorisation method to Bag of Words, except TF-IDF gives weight to words that are frequent in one document but rare across the dataset, and so words are represented by their importance to the document not just their frequency. This is done by first calculating the term frequency with equation 13, which measures how often a term or word t appears in a specific document d.

$$TF(t,d) = \frac{\text{count of } t \text{ in } d}{\text{number of words in } d} \tag{13}$$

The IDF term is calculated with equation 14, which measures how rare the term is across the entire collection of documents. Where N is the total number of documents and n_t

$$IDF(t) = log\left(\frac{N}{n_t}\right) \tag{14}$$

Finally these terms are combined together to create the TF-IDF score with equation 15.

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t) \tag{15}$$

(Sánchez & Cuervo-Londoño, 2024)

Words score highly if they have high frequency within the document gaining a high TF score, and/or they appear in few overall documents gaining a high IDF.

Words like ”the”, ”and” and ”with” will have high TF but very low IDF, and so their TF-IDF score will be near zero score. However, special words like ”citrus” or ”cassia” will have a high TF and high IDF, meaning they are very important for distinguishing a document.

4.6. Mean Word2Vec Representation

Word2Vec representation was first introduced by Mikolov et al. in a 2013 paper. This form of representation uses continuos vectors of an arbitrary size to represent words. These vectors are learnt from large datasets, and they encode semantic information about the word. For example, $vector(King) - vector(Man) + vector(Woman)$ results in the vector that is closest to the vector representation of the word Queen. (Mikolov et al., 2013)

Since its introduction many different Word2Vec representations have been trained. This project will use the library of en_core_web_sm from spaCy. These vectors have been trained on information available on the web. The vectors have a dimension size of 96. To allow clustering each input much be of the same size, to achieve this every vector in each documents gets averaged, resulting in a consistent dimension of 96. This averaging technique loses some of the semantic value of the vectors.

4.7. Decision Trees

Decision trees are a simple yet popular method, mainly used for classification problems. They work by recursively splitting a dataset into groups based on feature values, with the goal of making each group as pure as possible. The training process is:

- 1 : Assess which feature to use and where to split the data using information gain or Gini impurity for evaluation.
- 2 : Split the data.
- 3 : Assess the new groups of data, if they are of high enough purity stop the algorithm. If not repeat steps 1 to 3.

The splits are ordered in a branch style structure, starting with the root node (the first split) followed by the split nodes (subsequent splits), connected by branches, ending with the leaf node (which are the close to pure groups). Each leaf node is assigned a class based on what class is most ubiquitous within it.

Predictions are then generated by passing an input through the branches until it reaches a leaf node, at which point the assigned class is returned as the prediction. (Quinlan, 1986)

4.8. Bayesian Optimisation

Bayesian Optimisation works by running through the following steps:

- 1 : Approximate the unknown objective function $f(x)$ with a Gaussian process. This Gaussian is then used to produce a mean performance and variance for each point.
- 2 : To decide where to sample next, finding points along x that have a high probability of beating the current best mean.
- 3 : Evaluating the new point to find the true value of $f(x)$.
- 4 : Update the approximation by adding the output of the last evaluation.
- 5 : Repeat steps 2 to 4 until a stop condition is met or convergence is reached.

(Mockus et al., 1978)

4.9. Neural Networks

Neural networks are the backbone of modern machine learning but their theory has been around for many years. The perceptron, was first introduced by Frank Rosenblatt in 1957. (Rosenblatt, 1957) Later in 1965 Alexey Ivakhnenko and Valentin Lapa would combine these perceptrons together to form the first multi-layered perceptron, more commonly know now as an artificial neural network.

Neural networks are built of stacked layers of perceptrons, with each perceptron in a layer being connected to every perceptron in the next layer. Each perceptron in a network takes a number of inputs x and multiples each with a corresponding weight w before adding a bias value b, the resulting value z is then passed to activation function $f(z)$ before being sent to the next perception. Equation 16 displays this.

$$f\left(\sum_i x_i w_i + b\right) \tag{16}$$

Later on in the 20th century the theory of training weights in a network using partial differentials, now called backpropagation, was developed. Backpropagation updates weights by using equation 17. This takes the partial differential of the loss function with respect to the specific weight that needs to be updated $\frac{\partial L}{\partial w_t}$, it then multiplies this value by some learning rate α which is commonly set to 0.001, and then adds the result to the weight w_t to produce the update w_{t+1} .

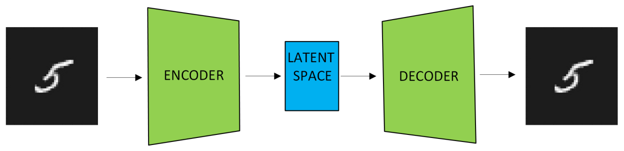


Figure 2. An image displaying the architecture of an Autoencoder

$$w_{t+1} = w_t + \alpha \frac{\partial L}{\partial w_t} \tag{17}$$

(Rumelhart et al., 1986)

Backpropagation was the first training method for neural networks, it has since been evolved into better modern alternatives. In 2017 Kingma and Ba introduced the Adam optimiser which has since become a wide spread alternative. Standard backpropagation has a known issue of getting stuck in local minima, the gradient flattens out when in these minima and the learning stops, this can cause the algorithm to stop looking for better parameters and miss the global minima. To avoid this Kinga and Ba add a momentum and velocity parameter to backpropagation. Meaning the learning only slows after the gradient has been shallow for long period of time. Allowing greater exploration around a minima before settling. (Kingma & Ba, 2017)

4.10. Variational Autoencoder

To understand how Variational Autoencoders work (VAE), it is first helpful to understand how its predecessor the Autoencoder works. Figure 2 displays an abstract architecture of an Autoencoder. Autoencoders consist of three main parts, an encoder, the latent space or bottleneck and an decoder. The encoder is a neural network that maps the input data to the latent space, producing a latent vector. This latent vector is then fed to the decoder, which is another neural network, which aims to recreate the original data. During training data is fed through the Autoencoder for a forward pass, and the resulting output data is then compared with the original data and a reconstruction loss is calculated. The reconstruction loss is either MSE[3.1] or BCE[3.2] depending on the type of data used. It is this reconstruction loss that is used on the backward pass to calculate the weight updates using backpropagation. This is the same training method described in section 4.9 for normal neural networks.

Standard Autoencoders are popular for tasks like dimensionality reduction (Hinton & Salakhutdinov, 2006) and data de-noising (Vincent et al., 2008), but they are unsuitable for generating data new data due to the learned latent space being irregular and unorganised. Sampling the latent space from near a learned vector will not necessary produce meaningful results similar to the neighbouring vector. For the purpose of data generation an organised latent space must be learnt, this is what the Variational Autoencoder does. Kingma & Welling first published the Variational Autoencoder in their 2013 paper. To achieve this change, instead of mapping an input directly to a latent vector they map the input to a distribution. Creating two vectors that represent the input's mean μ and variance σ^2, this assumes the data's prior distribution to be Gaussian. When recreating data a vector is then randomly sampled from this distribution and fed to the decoder. However, a random sampling method is not differentiable and so a trick must used to allow the network to be backpropagated. This is called the reparametrisation trick, this trick is displayed in equation 18. Where μ and σ^2 are the mean and variance and ϵ is a random sample from the standard Gaussian of $N(0,1)$. This shifts the random sampling to ϵ and allows backpropation through the now differentiable μ and σ.

$$z = \mu + \sigma \times \epsilon \tag{18}$$

Figure 3 displays how the reparametrisation trick fits into the architecture. The loss function for the VAE also differs from that used by the standard Autoencoder. The VAE tries to minimise the evidence lower bound function (ELBO) $L(\theta, \phi, x^i)$ shown in equation 19. Where ϕ are the Variational parameters and θ are the generational parameters.

$$\begin{aligned} L(\theta, \phi; x^i) = -\mathbb{E}_{q_\phi(z|x^i)}[log p_\theta(x^i|z)] + \\ D_{KL}(q_\phi(z|x^i)||p_\theta(z)) \end{aligned} \tag{19}$$

ELBO contains two parts, the first term L_1 is the log likelihood and represents the reconstruction error. In practise this is either MSE or BCE depending on the data type being used. This project uses BCE due the data being of a multi-variate Bernoulli distribution, lots of individual binary representations. This is shown in equation 20. Where x is the input data, x' is the reconstructed input from $Decoder(Encoder(x))$.

$$L_1 = -\frac{1}{N}\sum_{i}^{N}(x_i \cdot log(x'_i) + (1 - x_i) \cdot log(1 - x'_i) \tag{20}$$

The second term is the Kullback-Leibler (KL) divergence, this measures the difference between the learned latent space distribution $q_\phi(z|x^i)$ and assumed Gaussian $p_\theta(z)$. In practise this is rewritten as equation 21. Where μ and σ^2 are learned latent vectors for mean and variance. This term penalises the algorithm if it learns a distribution that differs from the standard Gaussian of $\mu = 0$ and $\sigma^2 = 1$

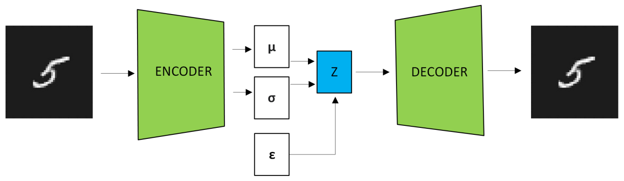


Figure 3. An image displaying the architecture of an Variational Autoencoder

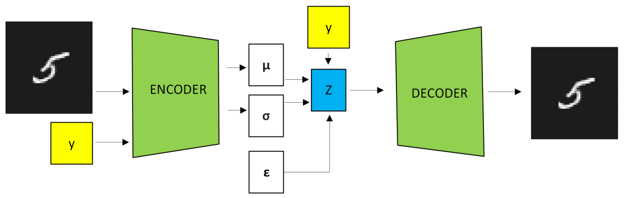


Figure 4. An image displaying the architecture of a conditional Variational Autoencoder

$$\begin{aligned} L_2 = D_{KL}[N(\mu, \sigma)|N(0,1)] = \\ -\frac{1}{2}\sum(1 + log(\sigma^2) - \mu^2 - \sigma^2) \end{aligned} \tag{21}$$

Combining these terms in equation 22, the overall loss can now be calculated. This function encourages the algorithm to map the prior distribution $p(x)$ to assumed Gaussian posterior distribution $p(z)$ whilst balancing the algorithms ability to reconstruct data accuracy.

$$L = L_1 + L_2 \tag{22}$$

(Kingma & Welling, 2013)

4.11. Conditional Variational Autoencoder

Conditional Variational Autoencoders (CVAE) were introduced in 2015 by Sohn et al. Building on the original VAE paper from Kingma & Welling, Sohn et al. making a slight change to allow a higher precision when generating class specific data. They do this by adding a one-hot encoded label Y to the sample z before passing it through the decoder, this is usually done by concatenating z and Y together at the first layer of the decoder. This allows a lot information about the sample to be carried by Y, z is then used to provide fine tuning parameters. Figure 4 shows an updated architecture displaying how this works.

(Sohn et al., 2015)

4.12. Beta Variational Autoencoder

Beta Variational Autoencoders were developed by Higgins et al. in 2017. They also use the original work done by Kingma & Willing, only adding a slight variation to the ELBO loss function. They add a β scalar to KL divergence shown in equation 23. A $\beta = 1$ keeps the operation the same as a normal VAE. A $\beta < 1$ prioritises the reconstruction accuracy over mapping the data to a Gaussian. A $\beta > 1$ pushes the algorithm to map the data closer to a Gaussian, this effect is called disentanglement creates a latent space that is more interpretable. This allows better general class based generation, as areas of the latent space correspond more meaningful features of the data. However, disentanglement happens at the cost of reconstruction accuracy.

$$L(\theta, \phi; x^i) = -\mathbb{E}_{q_\phi(z|x^i)}[logp_\theta(x^i|z)] + \beta \cdot D_{KL}(q_\phi(z|x^i)||p_\theta(z)) \tag{23}$$

(Higgins et al., 2017)

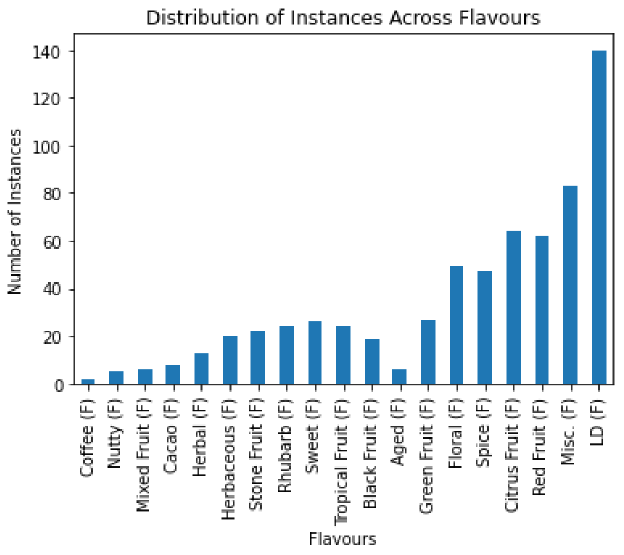


Figure 5. Distribution of Flavours

5. Data

A dataset has been provided for this project by Professor Ian Fisk at the Food Sciences department of the University of Nottingham. This dataset contains 552 gin samples. Each sample contains 723 features. Although, not every feature is relevant to the task. For instance there are 51 features that contain metadata like: date of collection, brand, name and other such descriptors that would help identify a gin. Countering this, there are 672 related features, this includes an ID, 18 different flavour profiles, 252 different ingredients or botanicals and 398 different compounds, each instance can be labelled with up to three different flavour profiles. The flavour profiles have been estimated by the staff at the Food Sciences lab and must be treated with a certain degree of leniency as they cannot be assumed to be totally accurate. The botanical features are binary and only indicate whether or not the botanical is present and not its quantity, where as the compounds are continuos and indicate the quantity detected at the time of measurement.

Figure 5 shows the distribution of flavours throughout this dataset. It is evident that this dataset is heavily unbalanced. The large majority of samples are labelled as either miscellaneous or LD (Long Description). These two categories make up a third of the overall distribution, but provide no description for instances that have been assigned to them. This means that the 223 instances which have these labels are currently unusable. Disregarding the two unusable categories still leaves the data heavily imbalanced, with the largest category, Citrus Fruit, having 62 instances and the smallest category, Coffee, having just two. It is unlikely that meaningful class-wise generation will be successful when some classes have so few instances. As well as this each instance of this dataset can contain up to three different flavour categories, further complicating the task.

Juniper is a botanical present in every gin recipe, according to EU law if a gin does not contain juniper it cannot be called a gin. Because it is a constant, and adds no information gain between instances, it has been removed from the dataset. After generation tasks are completed juniper could then be added.

5.1. Correlation and Data Exploration

This section will review the relationship between the three main feature types, flavour, botanicals and compounds. Logically it is easy to assume that these features will all have correlation, given that each botanical will contain its own replicable list of compounds and have a certain flavour. Due to the imbalanced data distribution and the fact that flavour profiles are estimated the correlation scores need to be taken with a certain amount of tolerance and cannot be assumed to be totally accurate.

Figure 6 displays a box plot showing the number of different botanicals spread across each recipe. The plot shows that on average a recipe has three botanicals in it. Although there is a range of 17 between the upper and lower quartiles, with an outlier reaching as high as 43 botanicals. Having a large number of botanicals is likely to add a lot of complexity to the generation task, but it will also have the benefit of adding more information to the loss function, resulting in larger reductions when reconstructions are good.

The correlations between flavour and botanicals are as ex-

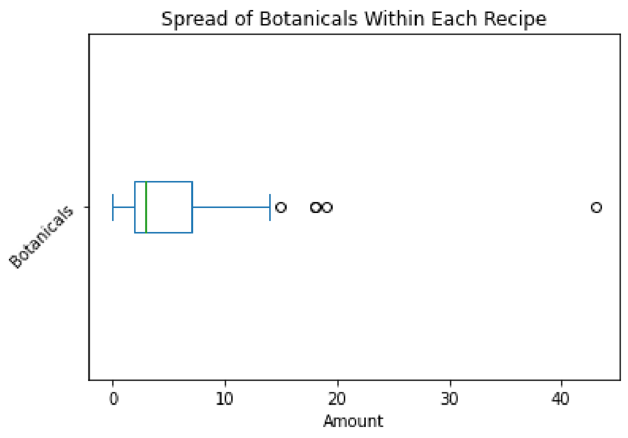


Figure 6. A box plot displays in number of botanicals present in each recipe.

pected. For example the flavour of coffee is strongly correlated with the coffee botanical and the nutty flavour is correlated with coconut, hazelnut and walnut. Because of how this data is structured with each gin having up to three flavour categories, each flavour and botanical correlation is rather a correlation between each gins flavour and corresponding botanicals present in the whole recipe. For example, the nutty flavour mentioned previously is also significantly correlated with the tarragon and celery botanicals, neither of which have a nutty flavour. Having such an imbalanced dataset further exacerbates this problem, as some categories have very few instances of data to learn from.

The correlation scores between botanicals and compounds are much less clear cut. With very few botanicals having a high correlation score of more than 50%. This could be explained by the fact that the botanicals are not measured in quantity, they are marked only if they are present or not. If a recipe has only trace amounts of a botanical then the compounds within that botanical will also only show up in small amounts, but regardless of a botanicals quantity is it marked with equal importance as the rest, so the coefficient will think that there is a high amount of this botanical and a low amount of the compound, tricking it into showing that the compound is not highly correlated with the botanical.

6. Experiments

Initial experiments trying to use classification to test if the botanicals could be used to predict the class label, and thus prove a learnable relationship between the two, showed highly variable results. A neural network[4.9] was trained to achieve this as VAEs are built of neural networks, and so this could provide insight into how well a VAE could learn the data.

Run	Val Accuracy	Test Accuracy
1	0.94	0.20
2	0.94	0.23
3	0.94	0.25
4	0.94	0.33
5	0.94	0.16

Table 1. Neural network accuracy comparison between different training run.

Before any classification could take place any rows marked at with ”Misc.” or ”LD” had to be removed, as described in section 5 these are a catch all for samples that could not or were not categorised. This reduced the data set from 552 instances to just 329 instances. These experiments were undertaken using a 70:10:20 train, validation, test split. Results from the experiments would vary depending the distribution of classes present within each data split. Table 1 displays the results from five separate training iterations of the neural network, this table shows that the test accuracy[3.3] varies with a range of 17%, which is significant. This is likely due the uneven distribution of data within each split and the overall small amount of data the model has to train on. The results shown in table 1 suggest that the model is over fitting each time to the training data and is very poorly generalised to instances it has not seen yet. This is expected given the small amount data and high class imbalance.

It is hard to gain a better picture of the classifiers performance using metrics like precision[3.4] and recall[3.5] due to the test set not being representative of the training set becuase of the imbalance. Although from the classes present it seems the classifier has confident true predictions, displayed with a high precision, but poor generalisation in identifying variety within a class, displayed with a low recall. These scores are displayed in table 2, these are taken from the run scoring a 33% accuracy. This table also displays the imbalance of the class distribution with the frequency metric in the third column, which could be another reason for the large range of test accuracy scores present in table 1. Unfortunately due to some classes only containing two instances it is not possible to stratify the classes between three sets. It also displays that classes with a high instance count generally get better results, this can be seen with Rhubarb, Floral and Red Fruit, referencing figure 5 shows that these classes are heavily populated.

7. Recategorisation

To try and counter the problems mentioned in section 6 and 5 a new set of categories needed to be developed. Two separate recategorising attempts were made, by clustering different sets of features present in the data using three different clustering algorithms;

Class	Precision	Recall	Frequency
Coffee	0	0	0
Nutty	0	0	0
Mixed Fruit	0	0	1
Cacao	0	0	0
Herbal	0	0	5
Herbaceous	1	0.25	4
Stone Fruit	1	0.28	7
Rhubarb	1	0.625	8
Sweet	0	0	8
Tropical Fruit	0	0	7
Black Fruit	1	0.5	8
Aged	0	0	1
Green Fruit	1	0.37	8
Floral	0.93	0.77	18
Spice	1	0.16	18
Citrus Fruit	0.88	0.44	18
Red Fruit	0.9	0.625	16

Table 2. Precision Recall scores for the 0.33 accuracy neural network shown in table 1 along with the number of instances present in the test set.

- KMeans[4.1], one of the most popular clustering algorithms. KMeans is accurate at clustering data that is normally distributed around some centre point. However, data that does not follow this distribution can be misclassified. The clusters are also highly depended on the quality of the initial centroid placement and the value of K. (Ahmed et al., 2020) Poor centroid initialisation drastically decreases the algorithms performance. So it is advisable to iterate through multiple initialisation points. (Cordeiro de Amorim & Makarenkov, 2023) Because of this, results displayed in this project will be that of the best centroid placement observed from multiple runs.

- Agglomerative Hierarchical Clustering[4.2] The Agglomerative method is also sensitive to the data's distribution and can suffer when it is not normally distributed. It can also be sensitive to outliers. (Gao et al., 2023) Although it does not suffer with the issue of initial centroid placement, due to it being a hierarchical based method.

- DBSCAN[4.3], DBSCAN solves the issue of distribution sensitivity due to it being a density based method, it is also more resilient to outliers. However, the quality of clusters is highly dependent on in the distance parameter. (Bhattacharjee & Mitra, 2020) (Aryani et al., 2024) The largest downside of DBSCAN in this use-case, is being unable to set the number of clusters.

To evaluate the clusters, this project implements three metrics:

- Silhouette score[3.6], this method is an industry standard, being cited as ”one of the most appropriate analyses to determine the optimal number of clusters” (Januzaj et al., 2023)

- Davies Bouldin score[3.7], another popular method, commonly used in combination with Silhouette score in research projects. (Kossakov et al., 2024) (Al-Kerboly et al., 2023) (Awong & Zielinska, 2023).

- Elbow point[3.8], this is used only for KMeans. Elbow point has been is used regularly, successful examples of its implementation can be found across clustering research. (Marutho et al., 2018)(Bholowalia & Kumar, 2014)(Syakur et al., 2018)

7.1. Recategorisation Based On Botanical and Compound Clusters

The first attempt at recategorisation, which this subsection covers, consists of clustering on the botanicals and compounds. A substantial benefit of using clustering is an increase of useable data. Instances categorised as ”LD” or ”Misc.” are now useable, so the entire dataset of 552 instances can used rather than the reduced 329 mentioned in section 5.

Initially an experiment into clustering the botanicals was undertaken. Each of the algorithms were set to iterate over a certain number of hyperparameters; different values of K and centroid placement for KMeans, different numbers of clusters for Agglomerative and different ϵ distances for DBSCAN.

The resulting metrics have been displayed in plot 8, with the elbow graph being displayed separately in plot 7. The elbow graph does not display any oblivious points at which there is a change in gradient, only showing slight changes at around six and then again at ten. This could indicate that there might not be a single optimal K value, and instead the groupings just slowing get tighter as the value for K increase. This is a commonly encountered draw back. If an elbow point is not obvious then is not possible to gain information on the optimal value of K using this method. (Yuan & Yang, 2019)

Figure 8 offers more information on the performance of the algorithms. Starting with DBSCAN, in the right column. Both the Silhouette score and Davies Bouldin score share a similar rate of change, this is counter-intuitive as a high Silhouette score suggests better clusters but a higher Davies Bouldin score suggests worse clusters, as explain in section 3.6 and 3.7. One explanation for this could be that as epsilon increases the number of clusters decreases. Therefore, with fewer clusters the Davies Bouldin score will be higher as its likely the features within each cluster will be spread out more, the Silhouette score will also be higher with fewer

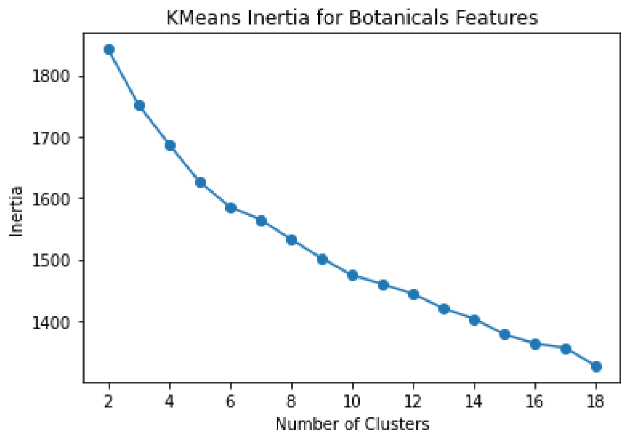
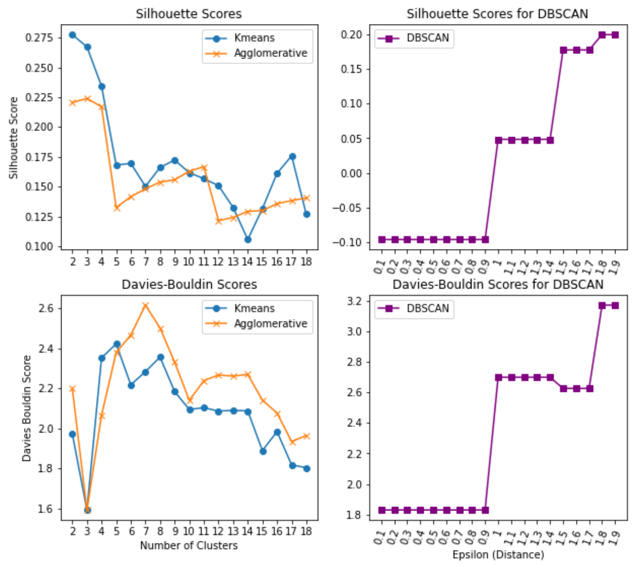


Figure 7. A graph displaying the Elbow graph for KMeans trained on the botanicals.



clusters as its less likely that cluster boundaries will overlap. Figure 9 provides some evidence for this theory. The figure displays the t-SNE[3.9] plots of cluster sets from DBSCAN with $\epsilon = 0.9$ and $\epsilon = 1.8$. As ϵ grows the clusters become spread further and become less dense, notably with cluster one. This will contribute to an increase in the Davies Bouldin score. Comparatively, when ϵ is lower clusters one, two and three are all contained within cluster zero, this will contribute to a lower Silhouette score. As ϵ increases cluster one breaks out of cluster zero and some instances occupy their own region in the top right, which would make cluster boundaries slightly more distinct increasing the Silhouette score. Taking this into consideration, it is likely that an ϵ value for epsilon would be 1.5. Figure 10 displays a bar chart and t-SNE plot from the clusters produced by DBSCAN with $\epsilon = 1.5$. The t-SNE plot displays a mid ground between the two t-SNE plots displayed in figure 9, as to be expected. The bar chart shows that clusters still remain heavily imbalanced. As well as thi,s DBSCAN only managed to find four clusters, which would not be enough for a generation task to produce anything useful. As only being able to produce four types of gin recipe would not make it a viable product.

Kmeans and Agglomerative produced similar Silhouette and Davies Bouldin scores, displayed in the left column of figure 8. Taking into account both sets of scores, and recognising that a larger number of clusters enhances the utility of the generation task, an optimal clustering solution for both cases can be reasonably proposed at ten clusters. With ten clusters, the Silhouette score for both algorithms is approximately 0.16. The Davies–Bouldin score is slightly below 2.1 for K-means and around 2.1 for Agglomerative clustering. This configuration represents a reasonable point of compromise for the two methods.

Figure 8. A graph displaying the Silhouette score and Davies Bouldin score on KMeans(left) Agglomerative Hierarchical Clustering(left) and DBSCAN(right) trained on the botanicals.

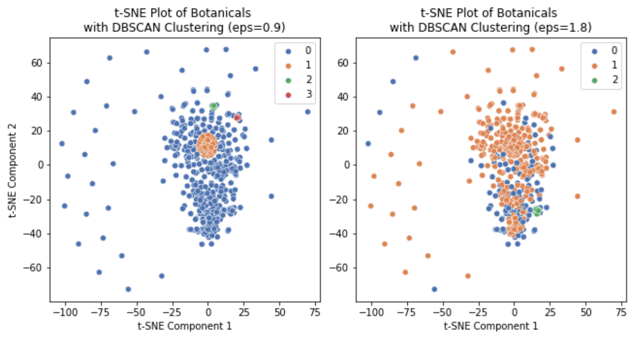


Figure 9. A comparison of the t-SNE plots for DBSCAN trained on botanicals with epsilon=0.9 (left) and epsilon=1.8 (right).

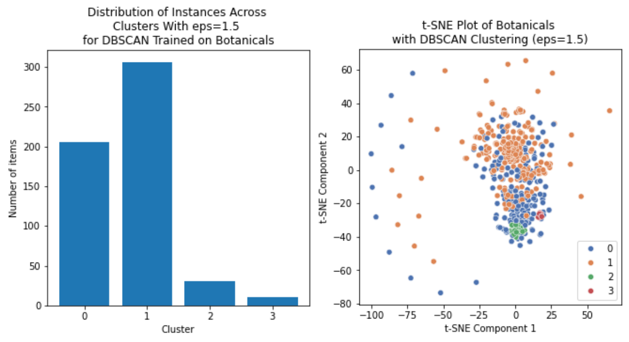


Figure 10. A bar chart(left) and t-SNE plot(right) displaying the distribution of instances in clusters found with DBSCAN with epsilon=1.5

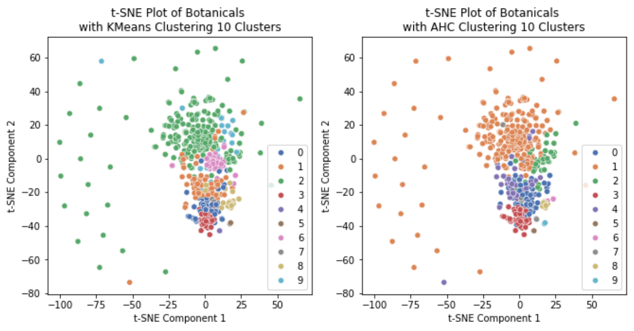


Figure 11. A comparison of t-SNE plots displaying clusters produced by Kmeans and AHC trained on botanicals.

Figure 11 displays the two t-SNE plots displaying clusters produced by KMeans on the left and AHC on the right. There is some similarity between the two plots, which is to be expected given the their evaluation scores are similar. Across both algorithms cluster 0, 3, and 8 all occupy similar spaces, with Kmeans-1 and AHC-4, KMeans-2 and AHC-1 also occupying similar spaces in their respective plots. The clusters on both plots have significant overlap which is likely to be why they both have low silhouette scores. Although certain clusters appear relatively dense, such as clusters 3, 6, and 8 in K-means, and clusters 3 and 8 in AHC, the majority are more diffused and lack a clearly identifiable distribution. This will likely lead to an increase in the Davies Bouldin score. Figure 12 displays the distribution of instances across the clusters created by both KMeans and AHC. Unfortunately, both algorithms produced a similarly imbalanced distribution.

Next an attempt would be made to cluster the compounds together. The compounds are represented by a sets of continuos floating point numbers, rather than the botanicals binary representation. Given the compounds are described in better detail it could be theorised that the they would produce better clusters. The decision was made not to normalise this data, given that each feature is a measure on the same scale and greater distances between features would likely help the clusters become better defined. A comparison of clusters created with both normalised and non-normalised data supported this decision.

Figure 13 shows the Elbow graph for KMeans trained on the Compounds. Unlike the botanicals' elbow graph, this shows an obvious change of gradient occurring between $k = 5$ and $k = 8$. This suggests that an optimal value for k could be between five and eight. Figure 14 also suggests that this range for k could be optimal. Both KMeans and AHC, displayed in the left column of figure 14, performed very similarly. A quick analysis of this column suggests eight clusters to be optimal for both algorithms. This value is determined

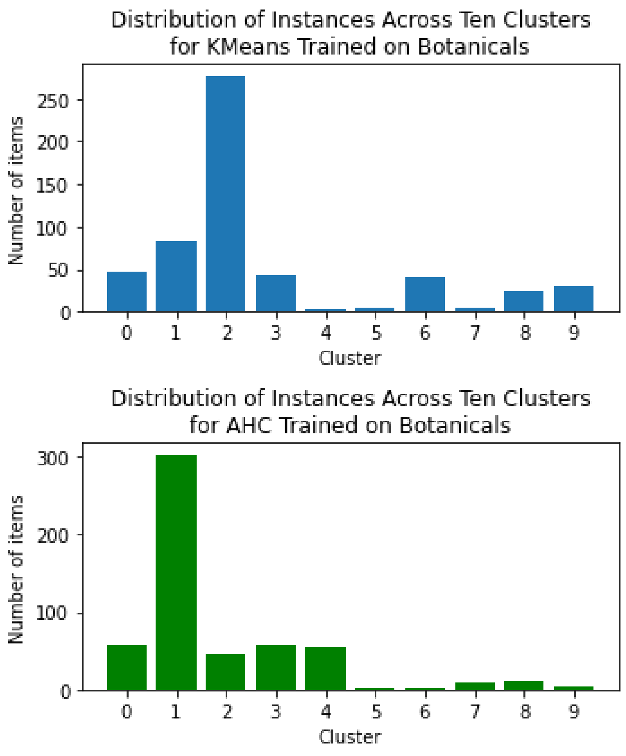


Figure 12. A comparison of bar graphs displaying the distribution of botanical instances across clusters created by KMeans (top) and AHC (bottom).

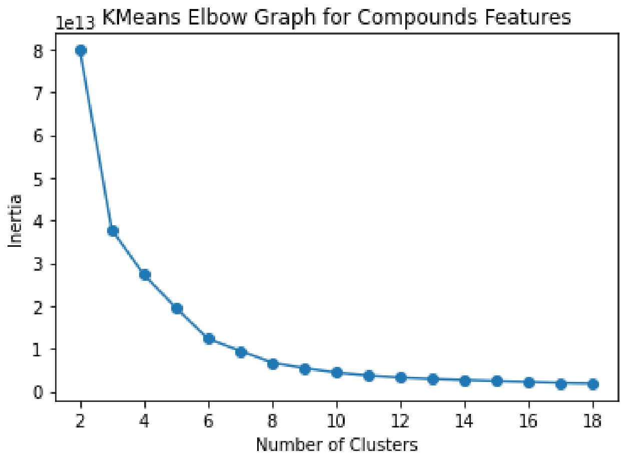


Figure 13. A graph displaying the Elbow graph for KMeans trained on the Compounds.

by seeking a sufficiently large number of clusters that simultaneously yields a relatively high Silhouette score and a low Davies Bouldin score. At eight clusters both KMeans and AHC achieve a Silhouette score of approximately 0.8 and a Davies Bouldin score of approximately 0.33. Both of these scores are considerably better than those found in the botanicals graph in figure 8. The same cannot be said for DBSCAN in the right column of figure 14. DBSCAN performed considerably worse on both Silhouette score and Davies Bouldin score, so much so that it was safe to discard it without need for further inspection.

Figure 15 displays the eight clusters created by KMeans and AHC on two separate t-SNE graphs. The clusters on both plots are much better separated than the ones seen previously with the botanicals. They occupy relatively distinct areas of the graphs and an identifiable distribution can be observed within each cluster. This reflects the higher Silhouette score and lower Davies Bouldin score observed in figure 14. However, it is evident to see that the distribution of instances across these clusters is still incredibly imbalanced. Figure 16 illustrates this point, making the cluster imbalance obvious.

The research into the data using these clustering algorithms has revealed that a large amount of the data resides in same area of feature space, so closely that the algorithms on each attempt clustered most of the data within one super cluster. Comparing the distributions created in this subsection with the distribution initially present, displayed in figure 5, shows that the new distributions have not been able to reduce class imbalance. However, this method renders the entire dataset usable, it also eliminates the multi-class categorisation of the previous system, reducing each instance to a single category assignment

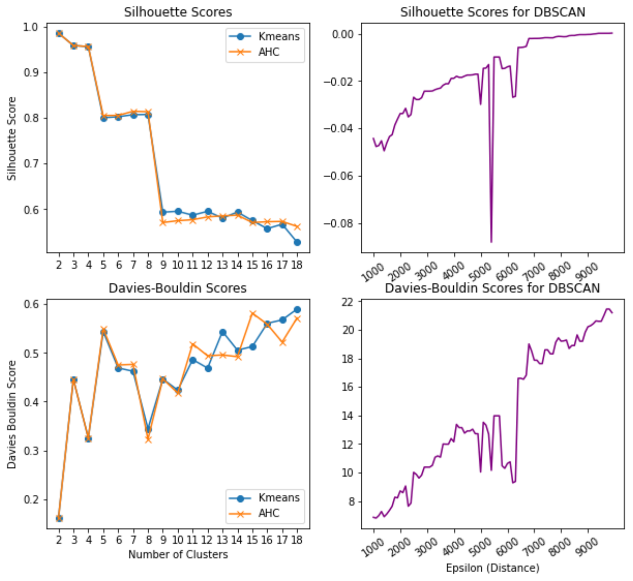


Figure 14. A graph displaying the Silhouette score and Davies Bouldin score on KMeans(left) Agglomerative Hierarchical Clustering(left) and DBSCAN(right) trained on the compounds.

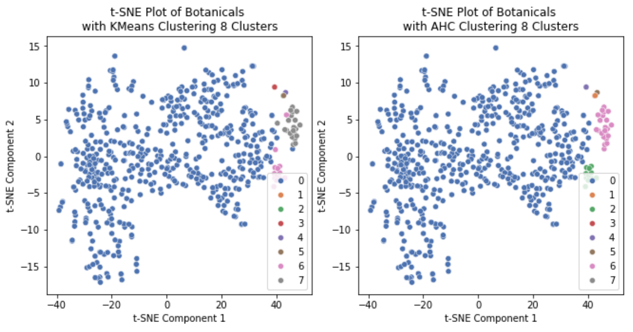
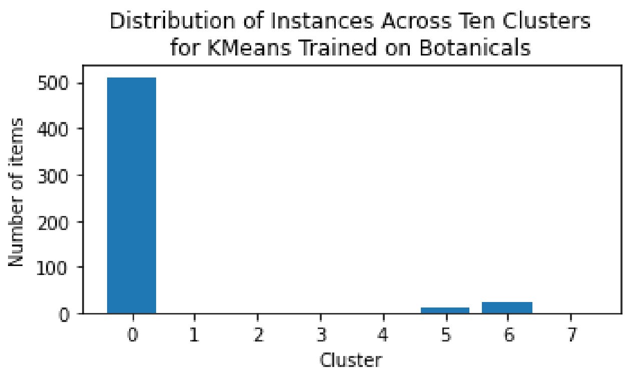


Figure 15. A comparison of t-SNE plots displaying clusters produced by Kmeans and AHC trained on compounds.



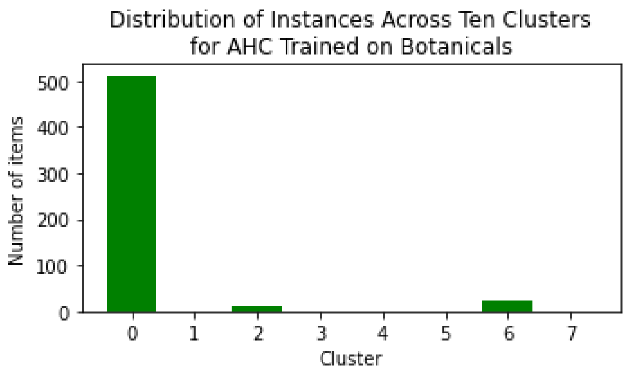


Figure 16. A comparison of bar graphs displaying the distribution of compound instances across clusters created by KMeans (top) and AHC (bottom).

7.2. Recategorisation Based On Written Descriptions

As the previous clustering approach was unable to evenly redistribute the data, a subsequent attempt was made to recategorise the dataset using the gin descriptions. Of the 556 samples, 453 contain a description, thereby increasing the number of usable samples from the original 329. Prior to clustering, these descriptions required preprocessing, with the following steps undertaken:

- Eliminating noise from the descriptions. This involves removing punctuation as well as any words that are not directly related to the gins flavour, for example; this, delicious and gin, would be removed.
- Lemmatising the remaining words. Lemmatisation is the process of reducing words to their base term or lemma. For instance, eating, eaten and eats would then become eat. By reducing semantically similar words to one stem word lemmatisation reduces noise and decreases the feature space. This allows for more accurate modelling. (Murel & Kavlakoglu, 2023)
- Vectorising the remaining words to produce data that is then compatible with the clustering algorithms. Vectorising is the process of turning human readable text into machine readable vectors.

There are many different types of vector that can be used in a natural language processing application. This project will implement the following three:

- Bag of Words[4.4]. This is one of the most basic methods of vectorisation and popular for light weight applications. One of the major downsides of Bag of Words is it considers each word in a document individually. The meaning derived from words that are spatially associated with one another is lost along with the individual semantics.(Juluru K, 2021)
- TF-IDF[4.5]. A slightly more a intelligent version of bag of words. Another popular choice for natural language processing. With many projects achieving good results. (Schmidt, 2019) (Amir Jalilifard, 2021) (Syed & Pedersen, 2025)
- Mean Word2Vec Representation[4.6]. The benefit of using Word2Vec is it captures semantic information about the words which the other methods miss. It is the most popular method for sentiment analysis and is widely viewed as giving the best results. (Kutuzov, 2018) (Hadifar et al., 2019) (Gniewkowski & Walkowiak, 2019) (Saha, 2023)

Each of these vector types has then assessed with the same three clusterings algorithms mentioned in section 7.1, figures 17 and 18 show the results.

Figure 17 displays the three inertia curves of KMeans trained on the three vector types. TF-IDF has the lowest inertia of all the vector types, suggesting that on average the clusters are tighter around their centroid than the other two vector types. The curves all displayed a relatively linear gradient, expect for Mean Embedding which has an elbow point at $k = 4$, although the gradient change is not dramatic. This Elbow graph does not offer much information on the optimal value for k.

Figure 18 displays the Silhouette scores and Davies Bouldin scores of each of the three algorithms. Starting with AHC on the top row, the plots display that Word2Vec performed consistently better than the other two vector types. With eight and seventeen clusters seeming to be two optima. Observing KMeans results on the middle row, its evident that it has performed similarly to AHC, as it did in the section 7.1. Word2Vec also performed the best with KMeans, the plots suggest an optimal k value to be ten. Finally, reviewing the last row, which displays results for DBSCAN it is evident to see that it has managed to produce the best scores with Word2Vec. However, the shape of curve for DBSCAN with Word2Vec is suspicious, and suggests that at lower the values for epsilon the algorithm could be bunching the data up into one super cluster. Figure 19 provides evidence of this, at the three lower values for epsilon DBSCAN creates the same clusters, at the higher value a new cluster is created that seemly separates the super cluster in two. Unfortunately, due the low number of clusters created and high level of imbalance these clusters are not useable.

Figure 20 shows t-SNE pots comparing between KMeans with $k = 10$ and AHC with eight clusters. Eight was chosen over seventeen, as seventeen clusters spreads the data too thin and results in some clusters being poorly populated. In both of t-SNE plots the clusters intermingle with each other, blurring the cluster boundaries. This is likely to be why there such a poor performance in terms of Silhouette score. Even though the clusters do not have distinct boundaries, they generally are clustered within their own area of the graph. This is probably why the Davies Boudlin score is comparability low. Figure 21 helps visualise the distribution between clusters on both the AHC and KMeans clusters. Its shows a set that is still imbalanced, although significantly more balanced than previous attempts in section 7.1. It could be argued that the these clusters are not any more balanced than the original data set is, but using this method there are an extra 103 instance available across a few number of clusters, leading to each cluster to be more populous.

7.3. A Discussion Concerning the Clustering Results

Given the results presented in Sections 7.1 and 7.2, it would be reasonable to assume that the most appropriate categories were those generated by KMeans or AHC using the VAEs

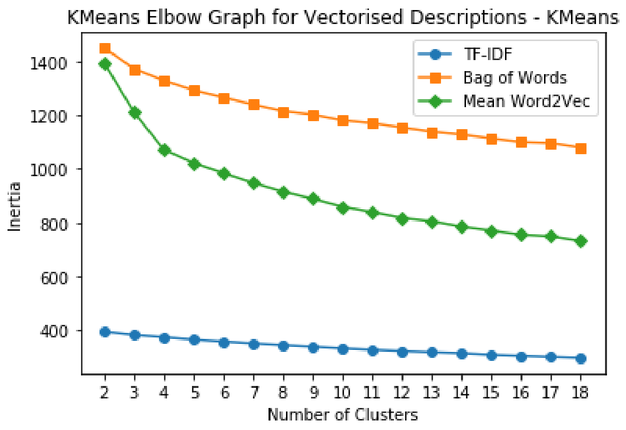


Figure 17. A graph displaying the Elbow graph for KMeans trained on the Bag of Words, TF-IDF and Mean Embedded vectors.

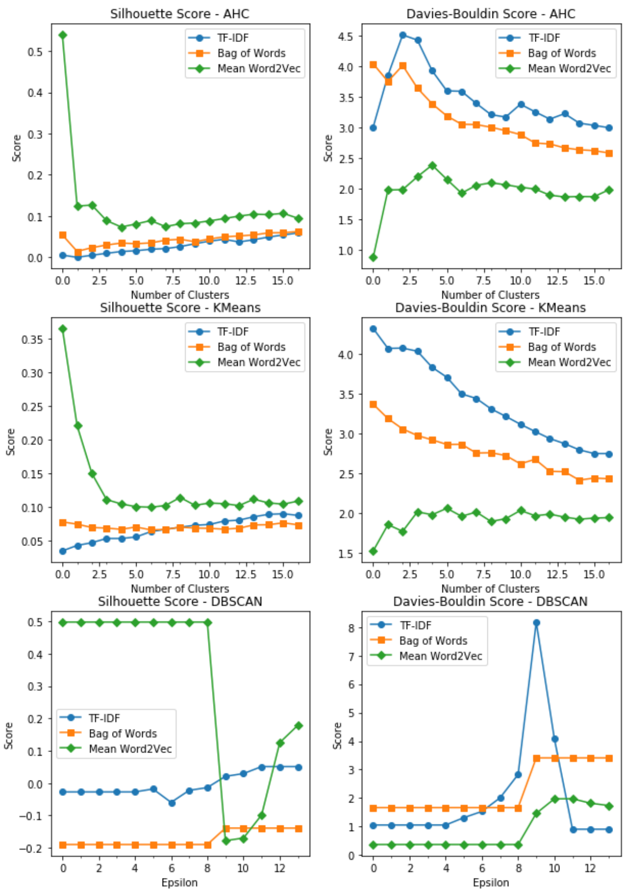


Figure 18. A graph showing the Silhouette score (left column) and Davies Bouldin score (right column) of ACH (top row), KMeans (middle row) and DBSCAN (bottom row) trained on three vectors types.

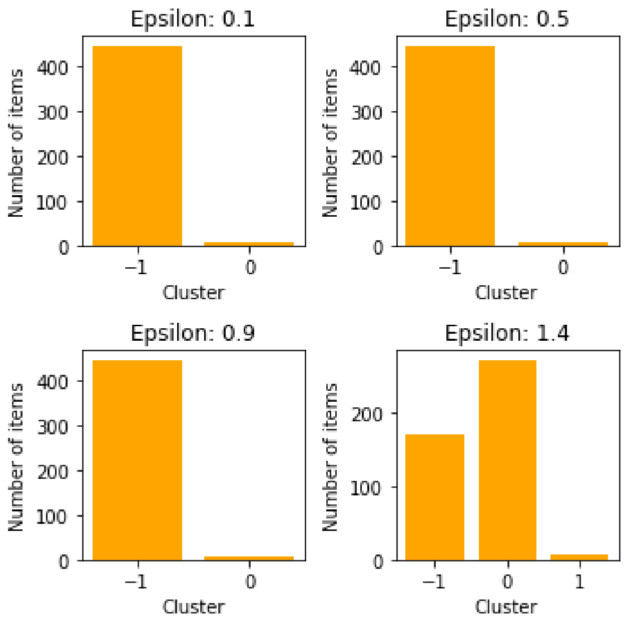


Figure 19. A comparison of distributions between clusters created by DBSCAN trained on Mean Embedded Vectors with different values of epsilon.

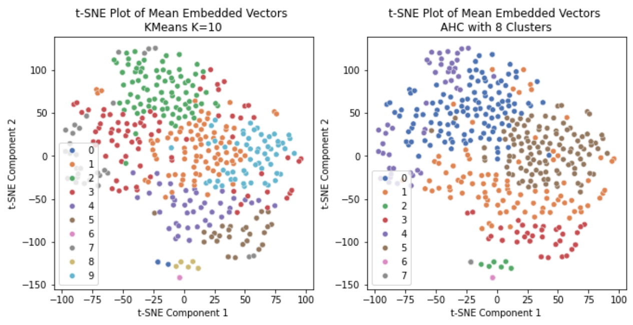


Figure 20. A comparison of t-SNE plots displaying clusters produced by KMeans and AHC trained on Mean Embedded vectors with ten and eight clusters respectively.

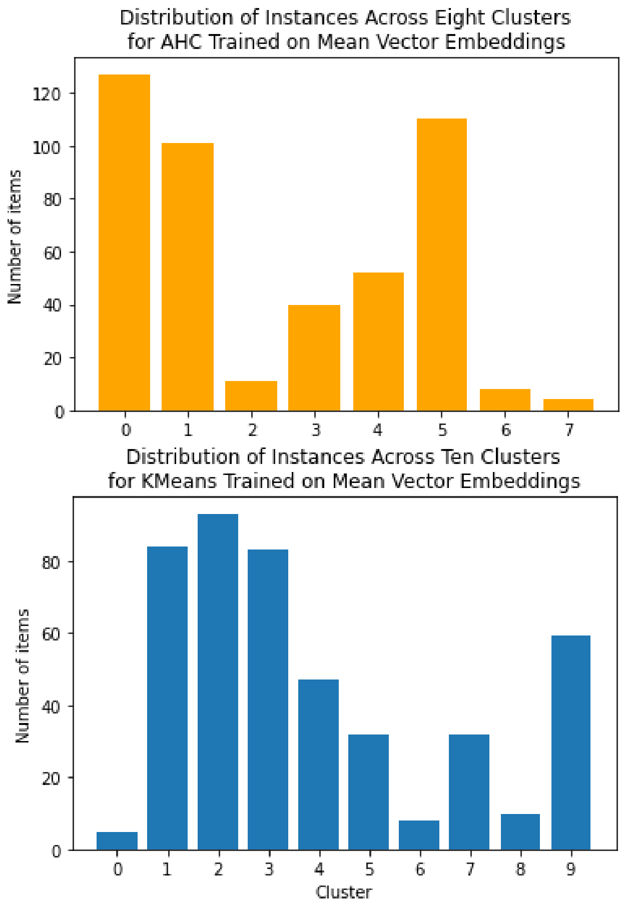


Figure 21. A comparison of bar graphs displaying the distribution of instances across clusters created by AHC (top) and KMeans (bottom).

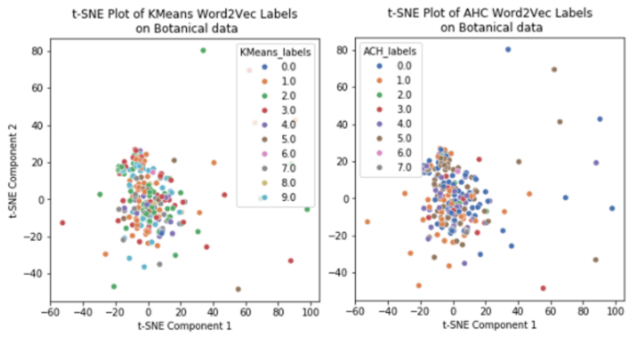


Figure 22. Two t-SNE plots displaying the category labels learnt from KMeans and AHC using Word2Vec embeddings displayed on the botanical data.

Word2Vec representations trained on the descriptions. However, further investigation challenges this assumption. As illustrated in Figure 22, the category labels derived from these methods, when projected onto the botanical data, fail to produce meaningful separation. While the algorithms succeed in partitioning the descriptions into distinct clusters, this structure does not carry over to the botanical features. Figure 22 reveals no organisation within the clusters when mapped to the botanicals. This outcome is disappointing, as it suggests that the approach is unsuitable. It is unlikely that a VAE could effectively learn a latent space capable of meaningfully representing these clusters. These results are likely because the written descriptions are too vague or inaccurate.

Based on these findings, this project adopts the categories identified by KMeans clustering on the botanical data. Although this approach did not produce evenly distributed classes, it offers two significant advantages. First, it replaces the multi-class categorisation of the original dataset with a single class system, thereby simplifying subsequent analyses and generation tasks. Second, it increases the volume of usable data from 329 to 552 instances, substantially enhancing the dataset's overall coverage and mitigating some of the effects of the imbalance.

The following list is a description of the clusters based on the compiled individual gin description present within each cluster. This must be taken with some leniency as it can only be as accurate as the written descriptions are. These cluster descriptions were compiled using chatGPT.

- Cluster 0 :

  Fresh and citrus-forward

  Strong lemon presence

  Peppery spice notes

  Herbal undertones

  Tangy, refreshing character

- Cluster 1

  Prominent citrus and orange

  Coriander spice

  Earthy undertones

  Aromatic hints

  Clean, balanced finish

- Cluster 2

  Sweet citrus profile

  Orange and lemon highlights

  Fresh brightness

  Smooth, zesty balance

- Cluster 3

  Robust pine influence

  Bright citrus freshness

  Lively, strong character

- Cluster 4

  Coriander-led

  Floral notes

  Citrus support

  Ginger spice

  Peppery finish

- Cluster 5

  Warm spice focus: cinnamon and cardamom

  Orange lift

  Aromatic complexity

  Slight sweetness

  Lingering finish

- Cluster 6

  Zesty citrus, especially grapefruit

  Tangy and clean

  Aromatic spice

  Refreshing brightness

- Cluster 7

  Citrus core

  Earthy depth

  Light floral hints

  Lemon freshness

  Subtle spice

- Cluster 8

  Citrus backbone

  Cardamom spice

  Herbal freshness

  Creamy texture

  Smooth balance

- Cluster 9

  Classic citrus style

  Orange and lemon dominate

  Fresh and zesty

  Bright and lively

8. Implementation

This section outlines the implementation of the VAE, providing a rationale for its selection, detailing the specific VAE variants employed, and discussing the architectural design decisions underpinning the model.

8.1. Algorithm Selection

VAEs offer a distinct advantage over alternative generative frameworks, such as Generative Adversarial Networks (GANs), due to their stable probabilistic training procedure. Unlike GANs, VAEs optimise a well-defined and traceable loss function, enabling consistent performance evaluation and facilitating direct comparisons across experimental configurations.

Although VAEs provide several advantages, they are not without limitations. A common issue is the introduction of noise into the generated outputs, which often manifests as blurred regions when generating images. This effect is caused by the KL divergence term in the loss function, which encourages the model to approximate the posterior distribution. This approximation process can oversimplify complex distributions, reducing the model's ability to accurately reproduce highly structured features (Bond-Taylor et al., 2022). However, given the simple binary nature of the gin data, and the small number of features, the noising issue is unlikely to be a problem, as the prior distribution is likely simple and easily approximated.

VAEs also present advantages when compared with diffusion based generative architectures. Diffusion models require many iterations of de-noising, which is the process of gradually injecting Gaussian noise into the data and then learning a de-noising procedure to recover the original data. (Yang et al., 2024) Whilst diffusion models are known for their high-fidelity generation (Hanqun Cao, 2023), their process requires substantially higher computational costs, as generation requires multiple iterative steps. In contrast, VAEs can generate new samples in a single forward pass from a latent vector, offering far greater efficiency. Given the relative simplicity of the gin data, particularly when compared with image data, diffusion models are arguably excessive, as the task does not require complex generation techniques to learn the data distribution.

8.2. VAE Variations

This paper will implement three different types of VAE; The traditional VAE described in section 4.10, the β-VAE described in section 4.12 and the Conditional VAE described in section 4.11. The β-VAE and Conditional VAE have been implemented to try and mitigate some of the downfalls of the original VAE. VAEs have a known issue of entanglement when learning the latent space. Entanglement happens when more than one latent dimension is learnt to represent the same underlying factor of the data. This can lead to an over complicated and disorganised latent space where changes in dimension values do not result in meaningful changes in generation data. This can also lead to samples from different classes becoming inter populated with each other, blurring cluster boundaries and reducing class based generation accuracy. (Shakya et al., 2024)

β-VAEs introduce the beta variable to scale the KL divergence term. This has been widely documented as a good disentanglement method. Disentanglement encourages the latent space to learn the underlying factors of the data, encoding each latent dimension with its own distinct factor. β-VAEs constantly outperform traditional VAEs in generation quality and latent space interpretability. (Higgins et al., 2017)

The conditional VAE improves upon the standard VAE framework by incorporating class labels into the latent representation prior to decoding. This label provides class specific information, facilitating more accurate class based generation (assuming the labels appropriately capture the underlying data structure). The latent vector can then be used for fine-tuning, enabling subtle variations around a class baseline to produce diverse individual instances. (Wang et al., 2022)

8.3. Architecture Optimisation

Selecting an optimal architecture for a neural network based algorithm is key when trying to achieve high quality results. There is no set structure to follow for VAEs, each must be designed to fit the problem. While designing networks manually can lead to high performing algorithms, finding well optimised networks can be challenging (Radosavovic et al., 2020), and generally requires a lot experience. To overcome this search algorithms can be used to find optimal hyperparameters.

There are many choices for optimisation methods such as genetic algorithms, reinforcement learning, Bayesian optimisation and random search. (Elsken et al., 2019)(White et al., 2023) Bayesian optimisation methods are particularly promising for hyperparameter search, as they incorporate the memory of past evaluations to probabilistically find areas of the search space that are likely to yield good results. (Bergstra & Bengio, 2012). This property reduces the number of evaluations required compared with uninformed search methods. Bayesian optimisation is advantageous when the loss function is computationally expensive to evaluate (Frazier, 2018), as is the case with training neural network architectures such as VAEs.

The architectural optimisation search space is defined across three primary dimensions: the number of layers L constrained to the range $1 >= L >= 4$, the number of neurons per layer N within the range of $56 >= N >= 560$, and the dimensionality of the latent space Z where $10 >= Z >= 60$. These hyperparameters have be chosen using previous personal experience building neural networks combined with knowledge gain whilst researching VAEs.

In the case of the β-VAE, the search space for β is defined by $0 >= \beta >= 30$. This range has been identified in previous research as suitable for balancing reconstruction quality and disentanglement performance. (Bonheme & Grzes, 2022) (Fil et al., 2021) (Kim & Mnih, 2019)

The loss function for the architectural optimisation task is simple, as it can use the performance of the algorithm as a guide, but evaluating β will be challenging. As the standard loss function cannot be used accurately assess disentanglement. There are current known methods, such as:

- Intervention based: A measure of how latent dimensions respond to changes in factors.
- Predictor based: Which uses supervised models for the predicting of factors from latent vectors.
- Information based: Which measure the mutual information difference between latent dimensions, to display how useful they are.

(Carbonneau et al., 2022)

Unfortunately, existing disentanglement metrics require knowledge of ground truth factors, the underlying features that structure the data. In this use-case that could be things like sweetness, bitterness or fruitiness. Unfortunately, like in most real word dataset, these are unknown and would be difficult to accurately estimate. In response to this, this project will propose an alternative for assessing the latent space structure and disentanglement effectiveness. This proposed method uses the clustering analysis metrics Silhouette score $s(x)$ and Davies Bouldin score $db(x)$, combining them to produce an optimisable metric that will work with Bayesian optimisation. This metric measures the intra-class compactness and inter-class separation of the latent space, providing evidence of how well the latent space organises into distinct regions associated with different classes. Unlike the standard disentanglement metrics, this approach requires only class labels, making it applicable to use-cases where ground truth factors are unknown. This composite metric is defined in Equation 24. The first term transforms the Silhouette score from its original maximization form into a minimization form by reversing its sign and rescaling its range from $[-1 \rightarrow 1]$ to $[0 \rightarrow 2]$. The second term incorporates the Davies Bouldin score but reduces its impact by applying a logarithmic transformation. This adjustment compensates for the unbounded range $[0 \rightarrow \infty)$ of $db(x)$ and the tendency of this score to change sharply as cluster quality improves. The addition of 1 is to avoid this term producing a negative value.

$$L(x) = (1 - s(x)) + (ln(db(x) + 1) \qquad (24)$$

9. Results

9.1. VAE Architecture

An analysis of the Bayesian optimisation results revealed a tendency to prioritise architectures that enforced a latent space closely aligned with the assumed Gaussian. Due to the simplistic nature of the data, improvements in reconstruction quality led to only marginal reductions in the reconstruction term of the loss function. Consequently, the most substantial decreases in the overall loss came from reductions in the KL divergence term. As a result, the optimisation process favoured architectures that promoted minimisation of the KL divergence disregarding enhancements in reconstruction accuracy. This lead to unusable architectures.

A manual search for hyperparameters was then undertaken. Starting with a small architecture and slowly adding layers until the loss function became unbalanced. Figure 23 shows a visualisation of the final architecture. The input and reconstruction layer are both equal to the size of the data, which for the botanicals is 252, with one hidden layer in both encoder and decoder with 350 neurons and latent dimension of 52. A drop out layer has been added to the decoder, as this has shown to reduce the risk of posterior collapse. (Petit & Corro, 2021)

9.2. Finding β

Figure 24 shows a sausage plot displaying the Bayesian optimisation results from the β search, where $\beta = x$ and $f(x)$ represents the loss function. The red dots indicate values which have been searched, the blue line indicates the approximate line of best fit, the surrounding blue bubble rep-

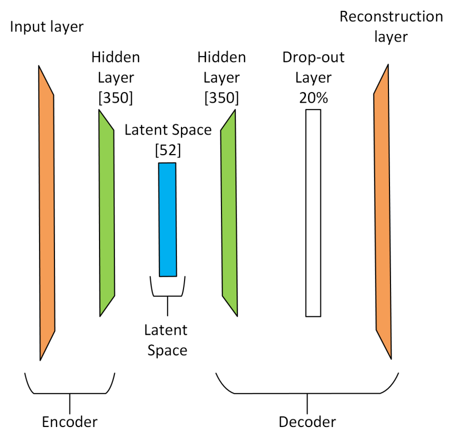


Figure 23. A visualisation of the architecture developed with Bayesian optimisation.

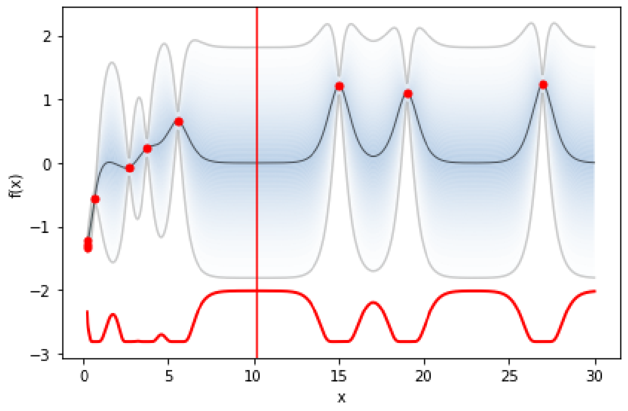


Figure 24. A sausage plot displaying the results of the search for β using Bayesian Optimisation.

resents the confidence of the algorithm in the line of best fit, the red curved line indicates places where a search would be beneficial (the higher the line the more beneficial it would be) and the vertical red line indicates where the algorithm would search next. These plots can be difficult to interpret, but the main conclusion that should be drawn from them is the what value of x best minimises $f(x)$. In this case $f(x)$ decreases as x decreases. This result was unexpected, as it suggests that smaller values of β can, in this case, facilitate a more distinct organisation of similar samples within the latent space. This contrasts with findings in the literature, where larger β values are typically associated with more structured and disentangled latent representations. This outcome can be attributed to the simplicity of the dataset and the relatively flat gradient of the reconstruction term. As the reconstruction error shows only minimal change, the KL divergence dominates the optimisation process as it contributes the largest reductions to the overall loss. When this happens the latent space gets continually pushed towards the assumed Gaussian, which can ultimately lead to a loss of meaningful structure, this is known as posterior collapse.

Figure 25 illustrates the previous point. It displays four t-SNE plots that have compressed and displayed the latent space from a β-VAE with β set to different values. This graph show that as β increases the latent space becomes closer to a true Gaussian as expected. However, clusters become more intermingled with each other, and the latent space becomes messy. The Beta=10 plot displays complete posterior collapse.

A $\beta = 0.1$ was chosen, given the results displayed by figure 24 and the risk of posterior collapse at higher values.

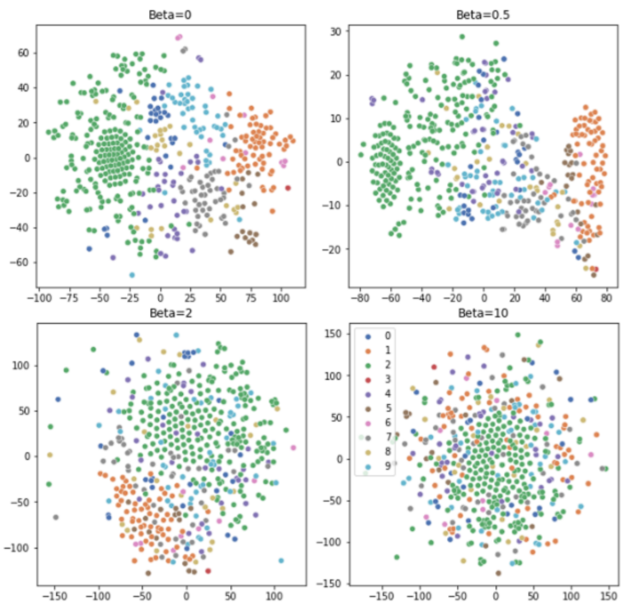


Figure 25. A set of t-SNE plots displaying the latent space produced by β a four different levels.

9.3. Latent Space Comparison

Figure 26 displays a comparison of the latent spaces between the three VAE types.

The VAE and CVAE plots both display a relatively compressed latent space, where classes intermingle. In both cases, a dense grouping of green data points forms a compact cluster, which may reflect the presence of a subset of gins within the dataset that share highly similar botanical structure. The β-VAE plot shows a latent space that is broader, where classes occupy their own space. Although there is still a significant amount inter population between classes.

Latent space organisation can be visualised by calculating the prototype vector μ_k for each class k. This is done by averaging each μ vector present across each class to produce a central vector that represents the class mean, known as a prototype vector. (Tang et al., 2023) (Zhang et al., 2019) μ_k is calculated using equation 25, where S_k is the set of samples present in class k and $\mu(x)$ is the individual μ vector produced by the input sample x.

$$\mu_k = \frac{1}{|S_k|} \sum_{x \in k} \mu(x) \tag{25}$$

These vectors are then transformed into heat maps and compared to measure latent space organisation and class distinction. If the latent space is organised the prototypes will look distinct from one another. If there is little structure the prototypes will look homogeneous. Figure 27 shows the three sets of prototype vectors produced by each VAE type. The results indicate that, among the three algorithms, the β-VAE produces the most distinct prototype representations. By contrast, the VAE and CVAE exhibit broadly similar behaviour, which may be attributed to their comparable approaches in structuring the latent space. An important note; Prototype 3 looks so distinct because its the average of three μ vectors, and therefore is much more likely to be located in a distinct part of the latent space. This class is underpopulated.

The latent space comparison reveals that the VAE exhibits limited structural organisation, suggesting it is unlikely to yield high quality generative results. Similarly, the CVAE demonstrates weak latent structuring. However, the inclusion of class labels prior to decoding may still enable the generation of viable samples. In contrast, the β-VAE demonstrates greater success in promoting class distinction, producing the most coherent and organised latent space among the three models.

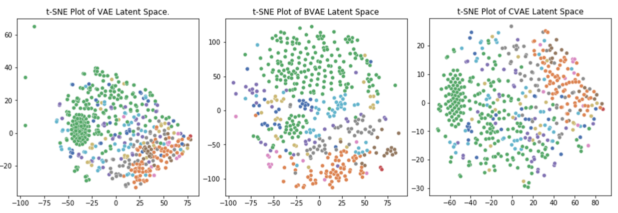


Figure 26. Three t-SNE plots displaying the latent space of the VAE, β and CVAE

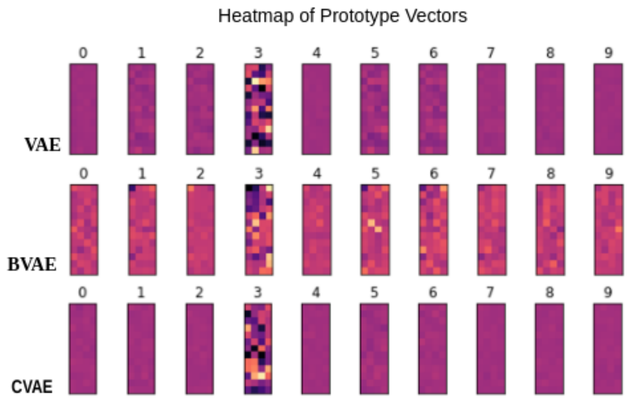


Figure 27. A set of heat maps, displaying prototype vectors of each class, generated by the VAE, β-VAE and CVAE

9.4. Quality of Generated Samples

A decision tree[4.7] was trained on the gin data using the KMeans labels with the objective of predicting the classes of generated samples to provide a quantitative metric of their quality. For each VAE variant, two sets of samples were subsequently generated. One set generated by a class dependent method using the prototype vectors, and the other an element-wise generation method.

The class generation task used prototype mean μ_k and prototype variance σ_k^2 to produce samples according to a class wide mean and variance. σ_k^2 has been calculated using the law of total variance written out in equation 26. Where $\sigma^2(x)$ is the variance of a sample present in class k generated from an individual input x. The first term is the within-sample variance and finds the average of the variance vectors for class k. The second term calculates the between-sample variance of the spread of mean vectors across the class k.

$$\sigma_k^2 = \frac{1}{|S_k|}\sum_{x\in k}\sigma^2(x) + \frac{1}{|S_k|}\sum_{i\in k}(\mu(x) - \mu_k)^2 \qquad (26)$$

μ_k and σ_k^2 were then sampled using the parameterisation trick explained in section 4.10 to produce a set of gin recipes of equal size to the original dataset containing the same distribution of recipes per class.

For the element-wise method, each recipe present in the original data was encoded to their corresponding μ and σ^2 and then sampled using the parameterisation trick. Creating another dataset of equal size and class distribution to the original.

If a VAE type had successfully managed to map the original data to latent space and then train a decoder capable of accurately mimicking the data, relatively similar prediction accuracies would be observed from the decision tree. However, due to the Gaussain noise VAEs use for sampling it would be unlikely to see results as good or better than the original.

Figure 28 presents the decision tree precision and recall scores for the prediction task on data generated by all three algorithms across both generation tasks. For comparison, the top row reports the decision tree's baseline performance on the test set derived from the original data. The results demonstrate that none of the algorithms succeeded in producing meaningful outcomes. Across models, only generated samples within class two were identified, with little evidence of broader class diversity. Although the β-VAE achieved marginally higher scores, its performance remained insufficient. A likely explanation is that the decoder converged towards producing highly similar outputs irrespective of the latent input, a symptom of posterior collapse.

	Average Dist	Most Ubiquitous
Original Data	2.67	12.5%
CVAE Prototype	1.15	60.8%
CVAE Element-wise	1.20	60.6%
β-VAE Prototype	1.32	41.4%
β-VAE Element-wise	2.05	33.6%
VAE Prototype	0.47	62.8%
VAE Element-wise	0.48	59.9%

Table 3. A table displaying the accuracy scores of element-wise generated samples and prototype generated samples from the VAE, β-VAE and CVAE.

Table 3 displays the results of a further investigation into this issue. The first column contains the average euclidean distance between samples in the data, this can be used to describe how different one sample is from another on average. The second column describes how much of the data set is made up of the most ubiquitous recipe. It shows the original dataset has an average distance of 2.67. Although 12.5% of this data is made of the same recipe. This could explain the tight grouping of green vectors present in some of the latent space visualisations. Both CVAE and VAE perform exceedingly poorly, having approximately 60% of their generated data made up of the same sample. This again provides evidence of posterior collapse. The β-VAE does comparatively better, with element-wise seeing the best results. However, these results are still poor when compared to the original data set.

9.5. Discussion

The poor results seen from the VAEs are due to posterior collapse. When a decoder becomes too powerful or the reconstruction term doesn't output much change, then decoder learns to ignore the latent vector and reproduce the same output each time. The training loss still decreases as the KL term takes over and forces the latent distribution further towards a Gaussain. (He et al., 2019) (Bowman et al., 2016)

This outcome is likely a consequence of how the reconstruction term evaluates the generated sample quality. Specifically, reconstruction quality is assessed using BCE[3.2] applied element-wise across the original and generated recipes, comparing all 252 individual pairs of botanicals. Given that, on average, a recipe contains only three botanicals, 249 elements remain zeros. Consequently, if the decoder learns to generate outputs consisting only of zeros, the reconstruction loss remains artificially low, as 249 of the predictions are still correct. This encourages the model to converge towards producing a single solution rather than learning to decode the actual botanical combinations. As a result of this, the re-

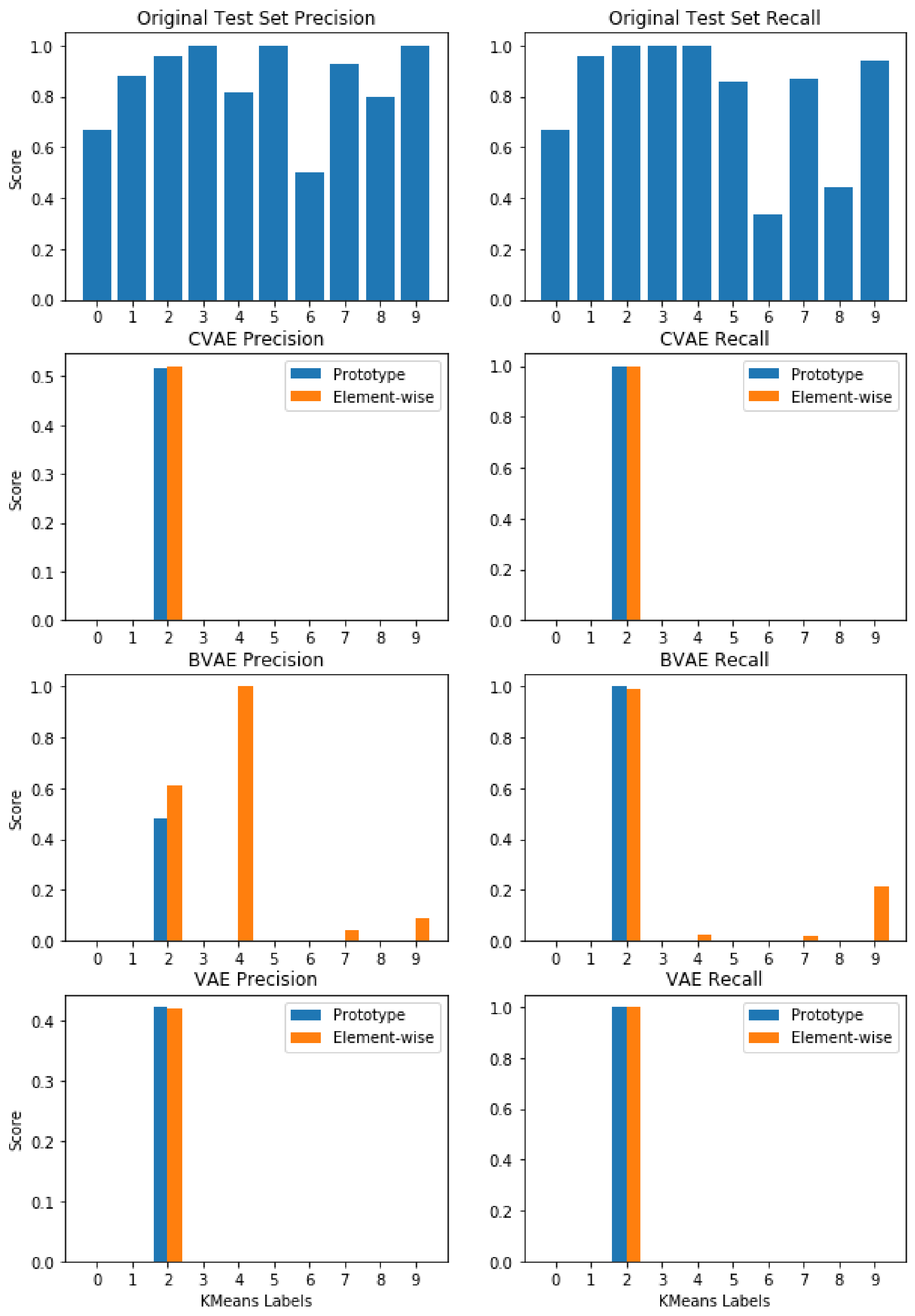


Figure 28. A comparison for precision and recall scores of element-wise generated samples and prototype generated samples from the VAE, β-VAE and CVAE.

construction term will stop learning and the KL divergence will dominate the loss, forcing the latent space further into a Gaussain until it losses structure. This is why a $\beta = 0.1$ was found to be optimal for the β, as this drastically scales down the KL divergence, allowing small changes in the reconstruction loss to be noticed. Although this did not seem to be enough to reactivate the reconstruction in a meaningful way. It is important to note that increasing the contribution of the reconstruction loss by adding a scalar made little difference.

The results indicate that the decoder converged toward generating an identical sample, irrespective of variations in the latent variable. Notably, this recurring sample consisted entirely of zeros. Providing evidence of posterior collapse. Figure 29 illustrates this by presenting a heatmap of the most frequently generated sample for each of the generation tasks. In this figure, each of the 252 binary botanicals is represented as a pixel within a 14x18 grid.

These results show that with the current data it is unlikely this form of generation will produce meaningful results. Because the botanicals are represented with binary descriptors and not quantities, and given that on average there are only three of them present in each recipe, it means the loss function doesn't change much between accurate and meaningful generation vs meaningless generation where most if not all botanicals are marked with 0.

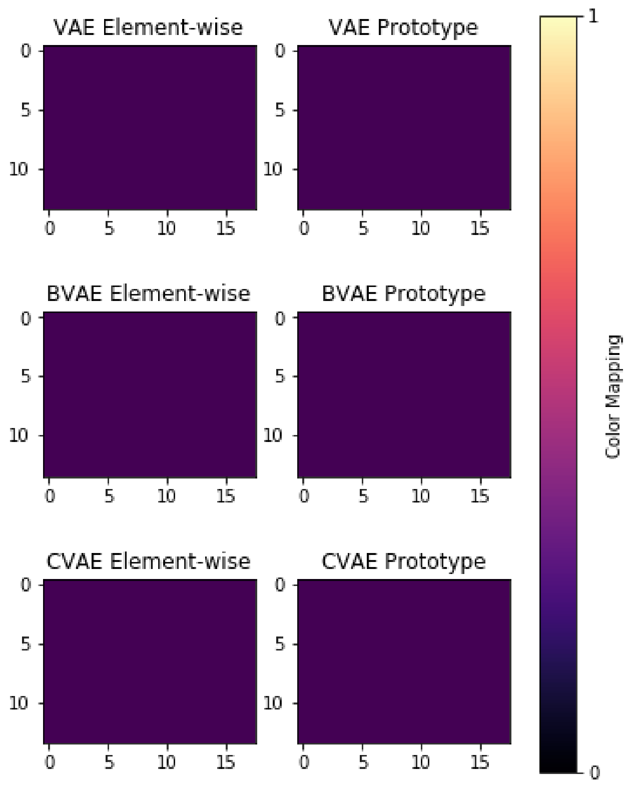


Figure 29. A heatmap of the most ubiquitous sample generated by each algorithm

10. Further Work

This section outlines the improvements required to enhance the effectiveness of the project and discusses its potential industrial application.

10.1. Improvements

The current methods of element-wise and prototype based generation have limits on their usefulness. The prototype based method only allows a finite amount of customisable variation with generation as its limited by the number of classes. When the intra-class variation is large, class specific generated samples vary significantly. Whilst element-wise allows for more precision when generating samples it still does not solve the customisability problem. As generated samples are directly linked to instances that already exist, and will only offer a small amount of variation from the original. To resolve this, further revisions need to be made. The latent space needs to be structured in a meaningful way such that generations can be customised in a predictable way by changing the values in a latent vector, producing predictable changes. To do this the latent space needs to be completely disentangled with the underlying factors of the data discovered.

There are papers that attempt to find these underlying factors. One uses a mutual information evaluation between the input and each latent variable as a measure of influence. Higher mutual information implies that a latent factor is more informative about the input, and therefore a factor. The authors found success with identifying factors on datasets like datasets like MNIST and CelebA.(Liu et al., 2019) Only keeping latent dimensions with a high mutual information could result in factor identification. Another project proposes an alternative that finds underlying factors by measuring the degree of dependence among all dimensions of a latent vector, they called this measurement the total correlation. This is a form of mutual information, but for multiple variables not just a pair. This dependence term is then incorporated into the loss function to encourage the VAE to learn a disentangled latent space. The authors also managed to find success with datasets like CelebA. (Kim & Mnih, 2019) Tweaking the dimension size of the latent space and comparing loss outputs until an optimal is found could reveal the number of underlying factors.

These disentanglement techniques would likely still struggle with the data in its current form. A key improvement for this project would be the use of an enhanced dataset, comprising of continuous data that represents the quantity of each botanical, rather than binary indicators of a botanicals presence. Access to such continuous data would substantially improve the effectiveness of the reconstruction loss, enabling it to more reliably distinguish between meaningful and meaningless generations. In addition, further extensive data collection is required, incorporating enough gins to span the full spectrum of possible flavour profiles. Ideally, this process should ensure sufficient sampling across the flavour space to achieve a more balanced and representative distribution. Once this is complete, factors of the latent space could then be linked to the flavour of the gins that reside at the extremes, allowing changes of direction in latent space to be predictably linked with changes of flavour in the recipe.

10.2. Industry Application

Provided the improvements outlined in section 10.1 are implemented, this project has significant potential for industry application. With sufficient knowledge of consumer preferences for the sampled gins, a predictive model could be trained to identify regions of the latent space that correspond to products favoured by consumers. Such a model could then recommend latent vectors with a high likelihood of producing popular gins. These latent vectors could be adjusted and customised in accordance with the outputs of the recommender model, enabling the generation of highly appealing products without the need for extensive consumer testing, reducing research and development time. Similar approaches have already been successfully employed in research, where machine learning models are used to predict consumer preferences and provide personalised recommendations. (Yang et al., 2017) (Gao et al., 2019) (Powell et al., 2024) (Bi et al., 2020) A model inspired by these existing applications could be integrated into the present framework extending its potential for industrial deployment.

11. Conclusion

This project implemented recategorisation techniques, using clustering algorithms to improve on the original dataset. It then demonstrated clustering techniques could increase the volume of usable data and enable the generation task by replacing the original multi-class labelling system with a single-class system. However, despite improvements in data availability, none of the clustering methods succeeded in producing balanced or clearly separated classes. The results suggest that while recategorisation offered benefits, the underlying imbalance and overlap within the dataset remained unresolved, limiting the possible effectiveness of subsequent generative modelling.

This project then investigated the use of VAEs and their variants for the generation of gin recipes, based on the recategorised data. While the models were able to capture certain structural patterns in the data, particularly in the case of the β-VAE, none succeeded in producing high quality or diverse generations. The main issues came from the imbalance of the reconstruction loss compared to the KL divergence, which led to posterior collapse, with the decoder converging on meaningless outputs and primarily producing empty recipes.

The findings highlight the constraints of applying deep generative models to small, imbalanced, and binary encoded datasets. They also suggest that progress in this domain will depend largely on improving data quality, through larger, more balanced datasets with continuous measures of botanical quantities. This project demonstrates the potential of generative modelling to support recipe development. With improved data and the integration of consumer preference modelling, this project could guide product design and reducing research and development time.

References

Ahmed, M., Seraj, R., and Islam, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8), 2020. ISSN 2079-9292. doi: 10.3390/electronics9081295. URL https://www.mdpi.com/2079-9292/9/8/1295.

Akoury, N. and Nguyen, A. Spatial pixelcnn: Generating images from patches, 2017.

Al-Kerboly, D. M. A., Hamad, M. M., and Dawood, O. A. Clustering algorithms comparison for university of anbar researchers' google scholar profiles. In *2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT)*, pp. 91–96, 2023. doi: 10.1109/AICCIT57614.2023.10217951.

Al-Sarayreh, M., Gomes Reis, M., Carr, A., and dos Reis, M. M. Inverse design and ai/deep generative networks in food design: A comprehensive review. *Trends in Food Science Technology*, 138:215–228, 2023. ISSN 0924-2244. doi: https://doi.org/10.1016/j.tifs.2023.06.005. URL https://www.sciencedirect.com/science/article/pii/S0924224423001693.

Amir Jalilifard, Vinicius F. Caridá, A. F. M. R. S. C.-F. P. C. d. F. *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 1*. Springer Singapore, 2021. ISBN 9789813369771. doi: 10.1007/978-981-33-6977-1.

Aryani, A., Petrie, S., Nambissan, A., Astudillo, A., and Cao, S. A rapid review of clustering algorithms. *arXiv*, 2024.

Awong, L. E. E. and Zielinska, T. Comparative analysis of the clustering quality in self-organizing maps for human posture classification. *National Library of Medicine*, 2023.

Barrejon, D., Olmos, P. M., and Artes-Rodriguez, A. Medical data wrangling with sequential variational autoencoders. *IEEE Journal of Biomedical and Health Informatics*, 26(6):2737–2745, June 2022. ISSN 2168-2208. doi: 10.1109/jbhi.2021.3123839. URL http://dx.doi.org/10.1109/JBHI.2021.3123839.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 2012.

Bhattacharjee, P. and Mitra, P. A survey of density based clustering algorithms. *Frontiers of Computer Science*, 2020.

Bholowalia, P. and Kumar, A. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9):17–24, November 2014. ISSN 0975-8887. doi: 10.5120/18405-9674. URL https://ijcaonline.org/archives/volume105/number9/18405-9674/.

Bi, K., Qiu, T., and Huang, Y. A deep learning method for yogurt preferences prediction using sensory attributes. *Processes*, 8(5), 2020. ISSN 2227-9717. doi: 10.3390/pr8050518. URL https://www.mdpi.com/2227-9717/8/5/518.

Bojar, D. Generating novel cocktail recipes with a specific style through recurrent neural networks, 2019.

Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347, November 2022. ISSN 1939-3539. doi: 10.1109/tpami.2021.3116668. URL http://dx.doi.org/10.1109/TPAMI.2021.3116668.

Bonheme, L. and Grzes, M. How do variational autoencoders learn? insights from representational similarity, 2022. URL https://arxiv.org/abs/2205.08399.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space, 2016. URL https://arxiv.org/abs/1511.06349.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in β-vae, 2018.

Cao, S., Li, J., Nelson, K. P., and Kon, M. A. Coupled vae: Improved accuracy and robustness of a variational autoencoder, 2021. URL https://arxiv.org/abs/1906.00536.

Carbonneau, M.-A., Zaidi, J., Boilard, J., and Gagnon, G. Measuring disentanglement: A review of metrics, 2022. URL https://arxiv.org/abs/2012.09276.

Cheng, T., Arguin, J.-F., Leissner-Martin, J., Pilette, J., and Golling, T. Variational autoencoders for anomalous jet tagging. *Physical Review D*, 107(1), January 2023. ISSN 2470-0029. doi: 10.1103/physrevd.107.016002. URL http://dx.doi.org/10.1103/PhysRevD.107.016002.

Chou, J. Generated loss and augmented training of mnist vae, 2019. URL https://arxiv.org/abs/1904.10937.

Cordeiro de Amorim, R. and Makarenkov, V. On k-means iterations and gaussian clusters. *Neurocomputing*, 553:126547, 2023. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.126547. URL https://www.sciencedirect.com/science/article/pii/S0925231223006707.

Davies, D. L. and Bouldin, D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979. doi: 10.1109/TPAMI.1979.4766909.

Dohi, K. Variational autoencoders for jet simulation, 2020. URL https://arxiv.org/abs/2009.04842.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey, 2019. URL https://arxiv.org/abs/1808.05377.

Embrechts, M., Gatti, C., Linton, J., and Roysam, B. Hierarchical clustering for large data sets. *Studies in Computational Intelligence*, 410:197–233, 01 2013. doi: 10.1007/978-3-642-28696-4_8.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231. AAAI Press, 1996.

Fil, M., Mesinovic, M., Morris, M., and Wildberger, J. Beta-vae reproducibility: Challenges and extensions, 2021. URL https://arxiv.org/abs/2112.14278.

Frazier, P. I. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

Gao, C. X., Dwyer, D., Zhu, Y., Smith, C. L., Du, L., Filia, K. M., Bayer, J., Menssink, J. M., Wang, T., Bergmeir, C., Wood, S., and Cotton, S. M. An overview of clustering methods with guidelines for application in mental health research. *Psychiatry Research*, 327:115265, 2023. ISSN 0165-1781. doi: https://doi.org/10.1016/j.psychres.2023.115265. URL https://www.sciencedirect.com/science/article/pii/S0165178123002159.

Gao, X., Feng, F., He, X., Huang, H., Guan, X., Feng, C., Ming, Z., and Chua, T.-S. Hierarchical attention network for visually-aware food recommendation, 2019. URL https://arxiv.org/abs/1810.05032.

George, A. S. The human touch: Exploring the synergy between bartenders and ai in cocktail creation, 04 2024.

Gniewkowski, M. and Walkowiak, T. Evaluation of vector embedding models in clustering of text documents. 09 2019. doi: 10.26615/978-954-452-056-4_149.

H. Lee, H., Shu, K., Achananuparp, P., Prasetyo, P. K., Liu, Y., Lim, E.-P., and Varshney, L. R. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In *Companion Proceedings of the Web Conference 2020*, pp. 181–184. ACM, April 2020. doi: 10.1145/3366424.3383536. URL http://dx.doi.org/10.1145/3366424.3383536.

Hadifar, A., Sterckx, L., Demeester, T., and Develder, C. A self-training approach for short text clustering. In Augenstein, I., Gella, S., Ruder, S., Kann, K., Can, B., Welbl, J., Conneau, A., Ren, X., and Rei, M. (eds.), *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pp. 194–199, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4322. URL https://aclanthology.org/W19-4322/.

Hanqun Cao, Cheng Tan, Z. G. Y. X. G. C. P.-A. H. S. Z. L. A survey on generative diffusion models. *arXiv:2209.02646v10*, 2023.

Harris, Z. S. Distributional structure. *WORD*, 10 (2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL https://doi.org/10.1080/00437956.1954.11659520.

He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. Lagging inference networks and posterior collapse in variational autoencoders, 2019. URL https://arxiv.org/abs/1901.05534.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL https://www.science.org/doi/abs/10.1126/science.1127647.

Januzaj, Y., Beqiri, E., and Luma, A. Determining the optimal number of clusters using silhouette score as a data mining technique. *International Journal of Online and Biomedical Engineering (iJOE)*, 19:174–182, 04 2023. doi: 10.3991/ijoe.v19i04.37059.

Juluru K, Shih HH, K. M. K. E. P. Bag-of-words technique in natural language processing: A primer for radiologists. *National Library of Medicine*, 2021.

Katserelis, K. and Skianis, K. Towards fine-dining recipe generation with generative pre-trained transformers. *Athens University of Economics and Business*, 2022. URL https://arxiv.org/abs/2209.12774.

Kaufman, L. and Rousseeuw, P. Finding groups in data: An introduction to cluster analysis. *Biometrics*, 1990. URL https://www.researchgate.net/publication/220695963_Finding_Groups_in_Data_An_Introduction_To_Cluster_Analysis.

Kim, H. and Mnih, A. Disentangling by factorising, 2019. URL https://arxiv.org/abs/1802.05983.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

Kossakov, M., Mukasheva, A., Balbayev, G., Seidazimov, S., Mukammejanova, D., and Sydybayeva, M. Quantitative comparison of machine learning clustering methods for tuberculosis data analysis. *Engineering Proceedings*, 60(1), 2024. ISSN 2673-4591. doi: 10.3390/engproc2024060020. URL https://www.mdpi.com/2673-4591/60/1/20.

Kutuzov, A. Russian word sense induction by clustering averaged word embeddings, 2018. URL https://arxiv.org/abs/1805.02258.

Lei, Z., ul Haq, A., Dorraki, M., Zhang, D., and Abbott, D. Composing recipes based on nutrients in food in a machine learning context. *Neurocomputing*, 415:382–396, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.08.071. URL https://www.sciencedirect.com/science/article/pii/S0925231220313771.

Li Y, Sadée CY, C.-P. F. S. H. T. A.-G. O. A 3d lung lesion variational autoencoder., 2024.

Liu, S., Liu, J., Zhao, Q., Cao, X., Li, H., Meng, H., Liu, S., and Meng, D. Discovering influential factors in variational autoencoder, 2019. URL https://arxiv.org/abs/1809.01804.

Lloyd, S. Least squares quantisation in pcm. *IEEE Trans. Information Theory, 1982*, 1982.

MacQueen, J. Some methods for classification and analysis of multivariate observations. *Berkeley Symp. on Math. Statist. and Prob.*, 1967.

Manuel Fritz, Michael Behringer, H. S. Log-means: Efficiently estimating the number of clusters in large datasets, 2019.

Marutho, D., Hendra Handaka, S., Wijaya, E., and Muljono. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 International Seminar on Application for Technology of Information and Communication*, pp. 533–538, 2018. doi: 10.1109/ISEMANTIC.2018.8549751.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Mockus, J., Tiesis, V., and Zilinskas, A. The application of bayesian methods for seeking the extremum. In *Towards Global Optimization*, volume 2, pp. 117–129. Elsevier, 1978.

Murel, J. and Kavlakoglu, E. What are stemming and lemmatization?, 2023. URL https://www.ibm.com/think/topics/stemming-lemmatization.

Murtagh, F. and Contreras, P. Algorithms for hierarchical clustering: an overview, ii. *WIREs Data Mining and Knowledge Discovery*, 7(6):e1219, 2017. doi: https://doi.org/10.1002/widm.1219. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1219.

Park, D., Kim, K., Park, Y., Shin, J., and Kang, J. Kitchenette: Predicting and ranking food ingredient pairings using siamese neural network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 5930–5936. International Joint Conferences on Artificial Intelligence Organization, August 2019. doi: 10.24963/ijcai.2019/822. URL http://dx.doi.org/10.24963/ijcai.2019/822.

Park, D., Kim, K., Kim, S., Spranger, M., and Kang, J. Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. *Scientific Reports*, 2021.

Petit, A. and Corro, C. Preventing posterior collapse in variational autoencoders for text generation via decoder regularization, 2021. URL https://arxiv.org/abs/2110.14945.

Powell, C., Zhu, E., Xiong, Y., and Yang, S. A data-driven approach to predicting consumer preferences for product customization. *Advanced Engineering Informatics*, 59, 2024.

Pyo, S. J. Flavordiffusion: Predicting food pairings and chemical interactions using diffusion models, 2025. URL https://arxiv.org/abs/2502.06871.

Quinlan, J. R. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces, 2020. URL https://arxiv.org/abs/2003.13678.

Richards, R. J. and Groener, A. M. Conditional β-vae for de novo molecular generation, 2022.

Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain1. In *Psychological Review*. Cornell Aeronautical Laboratory, 1957.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 1986. URL https://doi.org/10.1038/323533a0.

Saha, R. Influence of various text embeddings on clustering performance in nlp, 2023. URL https://arxiv.org/abs/2305.03144.

Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Commun. ACM*, 18(11): 613–620, November 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL https://doi.org/10.1145/361219.361220.

Schmidt, C. W. Improving a tf-idf weighted document vector embedding, 2019.

Shakya, S., Maharjan, B., and Shakya, P. From entanglement to disentanglement: Comparing traditional vae and modified beta-vae performance. *International Journal on Engineering Technology*, 2:38–48, 12 2024. doi: 10.3126/injet.v2i1.72491.

Shon, K., Sung, K. R., Kwak, J., Shin, J. W., and Lee, J. Y. Development of a -variational autoencoder for disentangled latent space representation of anterior segment optical coherence tomography images. *Translational Vision Science Technology*, 11(2):11–11, 02 2022. ISSN 2164-2591. doi: 10.1167/tvst.11.2.11. URL https://doi.org/10.1167/tvst.11.2.11.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Sorensen, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter*, 1948.

Syakur, M., Khusnul Khotimah, B., and Rohman, E. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. *IOP Conference Series: Materials Science and Engineering*, 336:012017, 04 2018. doi: 10.1088/1757-899X/336/1/012017.

Syed, S. and Pedersen, T. Duluth at semeval-2025 task 7: Tf-idf with optimized vector dimensions for multilingual fact-checked claim retrieval, 2025.

Sánchez, J. and Cuervo-Londoño, G. A. A bag-of-words approach for information extraction from electricity invoices. *AI*, 5(4):1837–1857, 2024. ISSN 2673-2688. doi: 10.3390/ai5040091. URL https://www.mdpi.com/2673-2688/5/4/91.

Tang, W., YANG, B., Li, X., Liu, Y.-H., Heng, P.-A., and Fu, C.-W. Prototypical variational autoencoder for 3d few-shot object detection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=fljrZsJ2I8.

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

Wang, C., Sharifnia, E., Gao, Z., Tindemans, S. H., and Palensky, P. Generating multivariate load states using a conditional variational autoencoder. *Electric Power Systems Research*, 213:108603, 2022. ISSN 0378-7796.

White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., and Hutter, F. Neural architecture search: Insights from 1000 papers, 2023. URL https://arxiv.org/abs/2301.08727.

Xu, R. and Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16:645 – 678, 06 2005. doi: 10.1109/TNN.2005.845141.

Yang, L., Hsieh, C.-K., Yang, H., Dell, N., Belongie, S., Cole, C., and Estrin, D. Yum-me: A personalized nutrient-based meal recommender system, 2017. URL https://arxiv.org/abs/1605.07722.

Yang, S., Chen, Y., Wan, L., Liu, S., and Chen, Y. Denoising diffusion step-aware models. *arXiv:2310.03337v4*, 2024.

Yuan, C. and Yang, H. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2):226–235, 2019. ISSN 2571-8800. doi: 10.3390/j2020016. URL https://www.mdpi.com/2571-8800/2/2/16.

Zhang, X., Zhou, T., Zhang, L., Fung, K. Y., and Ng, K. M. Food product design: A hybrid machine learning and mechanistic modeling approach. *Industrial & Engineering Chemistry Research*, 58(36):16743–16752, 2019. doi: 10.1021/acs.iecr.9b02462. URL https://doi.org/10.1021/acs.iecr.9b02462.