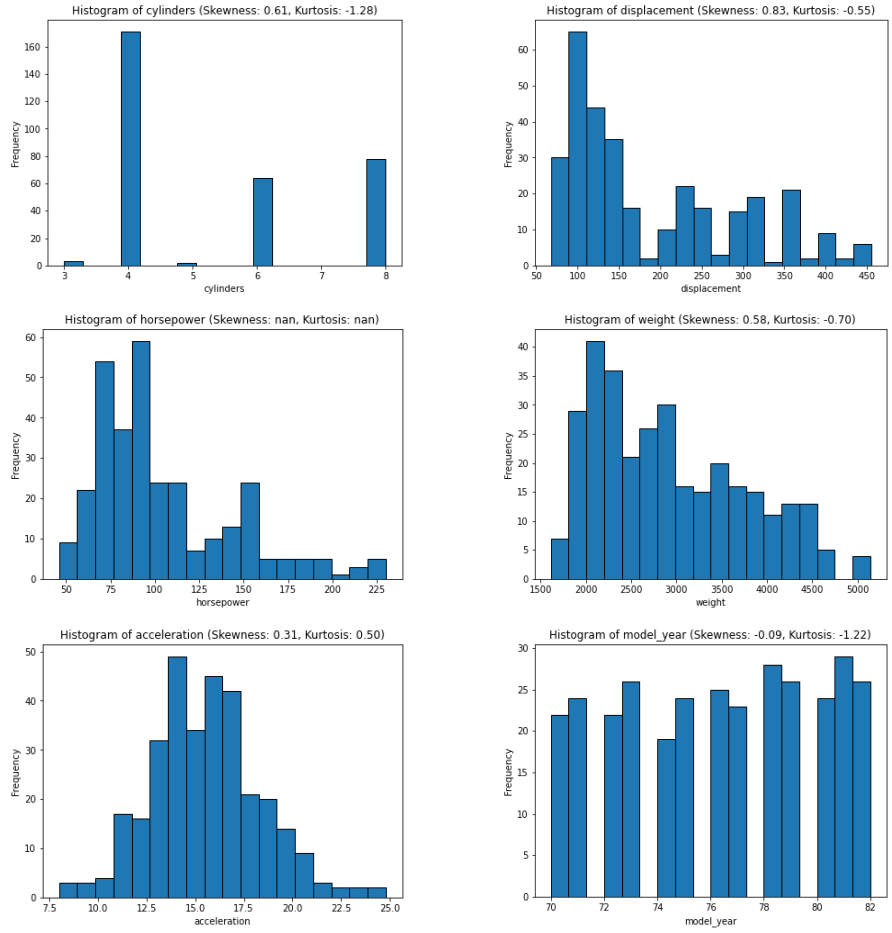
DATA302, assignment 3
Izzy Southon, 300597453

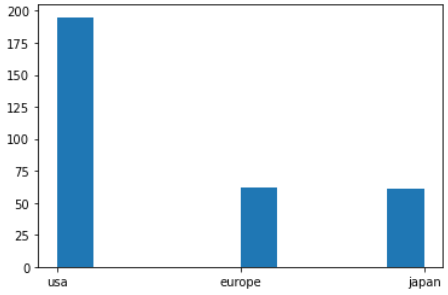
# Linear Regression

## (A) EDA

The original mpg dataset contains 9 features, where 8 features aim to predict 'mpg'. Two of these features are categorical (origin and name), three are numerical integer (cylinders, weight, and model_year) and the remaining three are numerical float64 (displacement, horsepower, and acceleration). There are 318 observations in the training data (X_train). The variable 'name' was removed from the original dataset before training because it is similar to an 'ID' column, which is likely to provide no useful information about predicting the response variable 'mpg'.

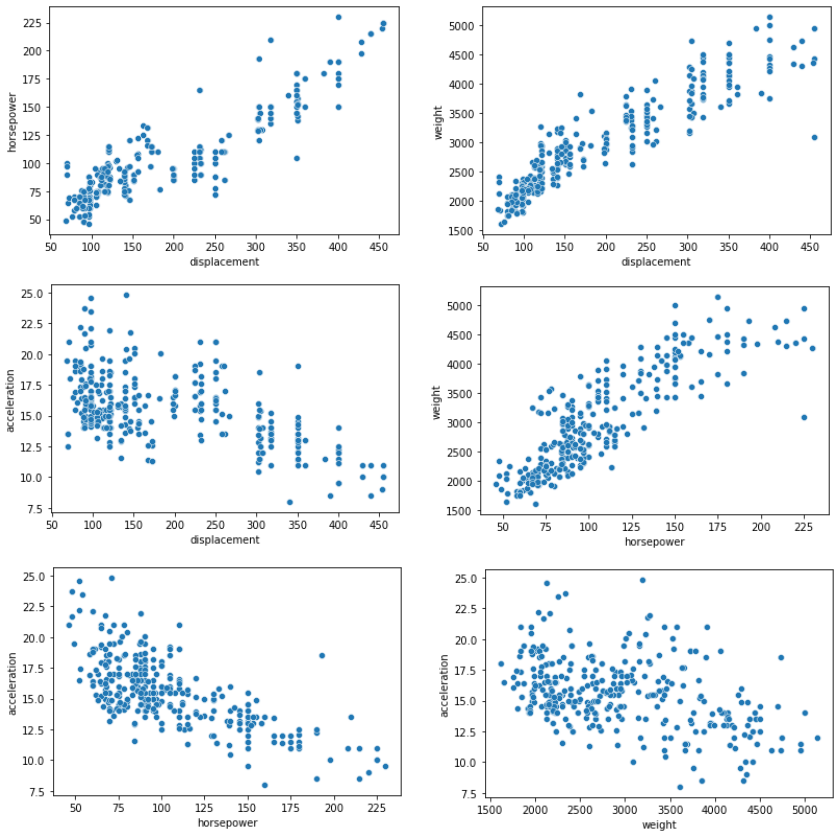



The histograms above display the distributions of the numerical variables in X_train. The number of bins to use was determined by taking the square root of the number of observations in the training set, where $\sqrt{318} = 17.8$ (*Adjusting the Number of Bins in a Histogram*, n.d.). Cylinders is an integer and has positive skewness and negative kurtosis (platykurtic). Displacement has positive skewness and negative kurtosis. Horsepower returned nan values for skewness and kurtosis, this could be due to the variable having missing values. Weight has positive skewness and negative kurtosis. Acceleration is relatively normally distributed and has positive skewness and kurtosis. Model_year has negative skewness and kurtosis and is an integer.




This histogram shows us that for the 'origin' variable, there are many more vehicles from the USA than there are from Europe and Japan. There seem to be an equal number of cars from Europe and Japan in the training dataset.

To quickly identify potential relationships between variables, a pair plot was first used, then individual scatterplots were created using the seaborn package.




There are positive relationships between horsepower and displacement, weight and displacement, and weight and horsepower. There is a negative relationship between acceleration and displacement, and acceleration and horsepower. The relationship between acceleration and weight cannot be decided from the scatterplot.

**(B) Data preprocessing before linear regression**

There are only missing values for one variable in the training set being horsepower, which has 6 missing values. To impute these missing values, KNN imputer was used because horsepower is a numerical feature. KNN is also a more robust imputation method by using neighbouring data points to estimate missing values (Brownlee, 2020a).

The only categorical variables in the original dataset are ‘origin’ and ‘name’, however the variable ‘name’ has been removed as mentioned. If ‘name’ was not removed, it would have had to be one hot encoded due to the nominal nature of the data. This would have created lots of dummy variables, which would have caused there to be many more features in the data that would have increased model complexity. The most appropriate method to encode the variable ‘origin’ is to use one hot encoding due to the nominal nature of the feature. It is also important to note that the variable model_year was put into the dataset as a numerical feature, however it is categorical. If model_year was put into the dataset as a categorical variable, it would be encoded using ordinal encoder, as years follow a natural order. Since this variable is already numerical, it will not be encoded as part of the preprocessing process.

**(C) Construct a linear regression model**

The coefficients are [-0.61456725, 0.02808721, -0.01307038, -0.00722961, 0.12042933 0.74020999, 0.74572188, 1.11420109, -1.85992297] which correspond to cylinders,

displacement, horsepower, weight, acceleration, model_year, origin_europe, origin_japan, and origin_usa respectively. The training performance of the linear model obtained MSE = 10.78 and $R^2$ = 0.78. The testing performance of the linear model obtained MSE = 10.63 and $R^2$ = 0.80.

### (D) Enhance the model

Table 1: comparing the MSE and $R^2$ of the linear, ridge, and lasso models

| | **Linear** | **Ridge** | **Lasso** |
|---|---|---|---|
| **Training set** | MSE = 10.78<br>$R^2$ = 0.78 | MSE = 10.78<br>$R^2$ = 0.78 | MSE = 10.78<br>$R^2$ = 0.78 |
| **Test set** | MSE = 10.63<br>$R^2$ = 0.80 | MSE = 10.63<br>$R^2$ = 0.80 | MSE = 11.47<br>$R^2$ = 0.77 |

Using ridge regression, the training performance obtained MSE = 10.78 and $R^2$ = 0.78. The testing performance obtained MSE = 10.63 and $R^2$ = 0.80. Using lasso regression, the training performance obtained MSE = 10.78 and $R^2$ = 0.78. The testing performance obtained MSE = 11.47 and $R^2$ = 0.77. From this, the important findings are that performance on the test set changes when lasso regression is used. MSE increases and $R^2$ decreases. This indicates that with lasso regression, performance of the model worsens, implying that overfitting be an issue. This is because when using lasso regression, the model may be learning the training data too well, and therefore cannot generalise to new, unseen data, such as the test set in this case. The model worsens when using lasso regression, and this is also evident as MSE and $R^2$ of the models remain the same for the training set and testing set for linear and ridge regression, and only changes when lasso regression is implemented.

## Clustering

### (A) Implementing K-means clustering

Table 2: comparing silhouette scores for different values of K

| The value for K (n_clusters) | Silhouette score |
|---|---|
| K = 2 | 0.6515279371492653 |
| K = 3 | 0.838307214726647 |
| K = 4 | 0.6335381476038574 |
| K = 5 | 0.42206837621434257 |

The table above (Table 2) shows corresponding silhouette scores for values of K ranging from 2-5. The *best* K according to the silhouette scores is K=3.

### (B) Visualisation of clusters with PCA

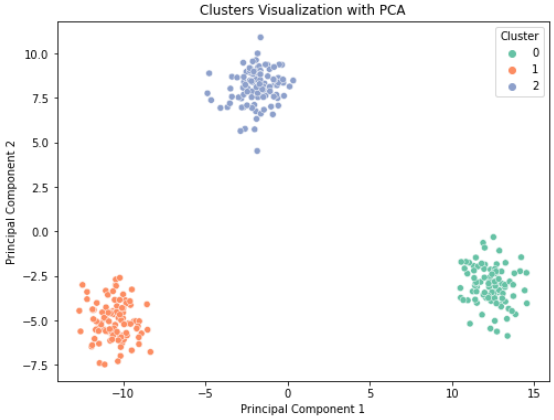



*Figure 1: visualisation of clusters with PCA*

## (C) Hierarchical clustering

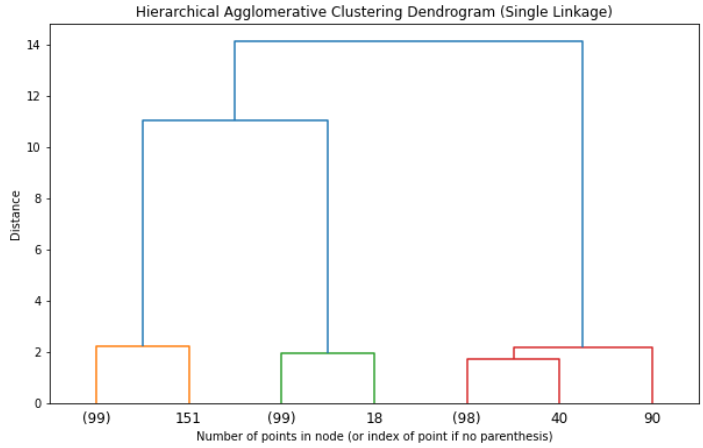



*Figure 2: Agglomerative clustering using single linkage method*

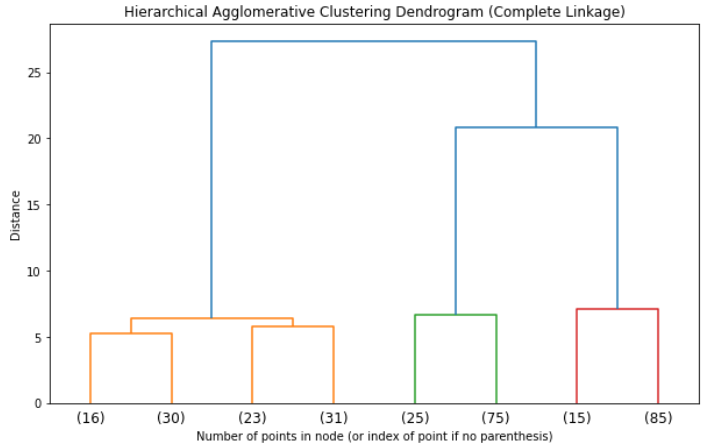



*Figure 3: Agglomerative clustering using complete linkage method*

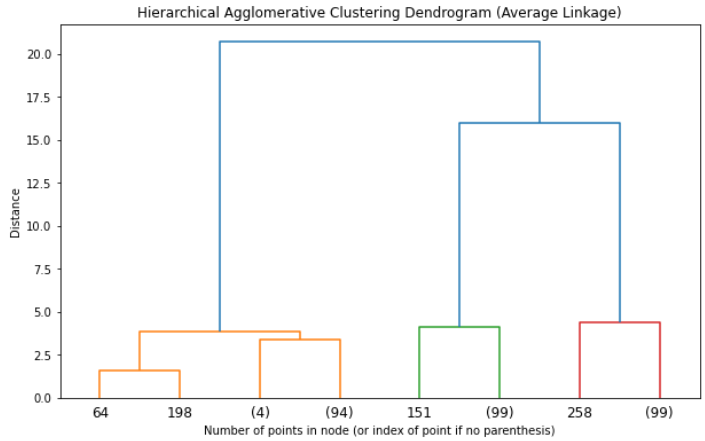



*Figure 4: Agglomerative clustering using average linkage method*

The agglomerative method begins by assuming that every single data point or instance is a cluster (Gadakari, 2019). The two instances closest to one another are combined into one cluster, and this process continues until a cluster is formed containing all objects (Fernandes & Solimun, 2022). In other words, each item is its own cluster, and we want to find the best pair to merge into a new cluster, until all clusters are joined together (Fernandes & Solimun, 2022). The dissimilarity between two observations is related to the vertical height at which they first get merged into the same cluster. The greater the height, the greater the dissimilarity.

The single linkage method computes the minimum pairwise dissimilarity where one observation is in cluster A and the other is in cluster B. The complete linkage method computes the maximum pairwise dissimilarity where one observation is in cluster A and the other is in cluster B. The average linkage method computes the average pairwise dissimilarity where one observation is in cluster A and the other is in cluster B.

In this scenario, the different linkage methods of single, complete, and average created clusters differently. In figure 2 (single linkage method), the orange cluster is quite far from the red cluster, indicated by the length of the blue line between them. The red cluster can also spilt into more clusters. Figure 3 (complete linkage method) still shows that the orange cluster is far from the red cluster, but the orange cluster can be spilt into more clusters. Figure 4 using the average linkage method also can spilt the orange cluster into further clusters. The dendrograms produced using the complete and average linkage methods produced similar looking plots, this may be because these methods tend to produce more balanced dendrograms. It is important to note that although the plots look the same, they would need to be cut at different values of height to create clusters (Geetha, 2022).

**(D) Comparing hierarchical clustering with K-means clustering**

K-means is a very simple algorithm that classifies datapoints to a particular class. It starts with K centroids and assigns each instance to the nearest centroid. It then updates by computing the new centroid for each cluster as the mean of the object is assigned to the cluster. Each instance is reassigned to the nearest centroid until there is no change in the centroids. Although K-means is useful due to its simple and flexible nature and ability to scale well with large datasets, it has limitations. These limitations are that K needs to be specified in advance and the algorithm needs to be re-run to obtain clustering with different numbers of clusters. Additionally, K-means needs features to be numerical, therefore categorical data needs to be pre-processed as a mean is required. Lastly, it is also a stochastic algorithm, and therefore needs to be re-run many times to be confident about the conclusion.

In contrast, hierarchical clustering can help address the limitations associated with K-means clustering. Hierarchical clustering does not require a specific value of K to be specified like K-means does, as it has natural clustering. The number of clusters is determined by the height of the cut. Additionally, hierarchical clustering does not need to be re-run to get clustering with different numbers of clusters like K-means does. Like K-means has a distance measure, hierarchical clustering also has a distance measure, defined by its linkage method. Like K-means, hierarchical clustering also needs categorical data to be transformed in some way to be applicable or use a specific distance metric such as Gower distance which can handle categorical data directly. Additionally, another advantage to hierarchical clustering is that it is a deterministic algorithm which means that it produces the same output for a given input each time it runs. Despite the advantages of hierarchical clustering, there are disadvantages of the algorithm. These are that clustering done by cutting the dendrogram at a particular height is nested within the clustering obtained by cutting at a greater height. This is important because where the dendrogram is cut can impact the interpretation of clustering results. Another disadvantage is that hierarchical clustering can be more computationally expensive than the K-means algorithm, meaning that it may take longer for the hierarchical clustering algorithm to run.

K-means might be preferred over hierarchical clustering when there is a large dataset, and results need to be obtained quickly. This is because in this case K-means would be more computationally efficient than hierarchal clustering. Additionally, K-means might be preferred

when *K* is known in advance or is predefined. This is because hierarchical clustering might not align with a predefined structure.

Hierarchical clustering may be preferred to K-means if the data naturally has a hierarchical structure. This is because it would preserve the hierarchical relationship between clusters better than K-means would. Hierarchical clustering may be preferred to K-means when there are nested or overlapping clusters in data because K-means is a flat clustering algorithm that may struggle to capture complex data structures.

## Neural Networks

### (A) Define a neural network class

The number of neurons used in the input layer was 64 because the number of neurons in the input layer is typically equal to the number of features in the input data. Images in the digits dataset are of size 8x8 pixels, and therefore need 64 (8x8) neurons, each representing one pixel value. In this classification problem the output layer corresponds to the number of classes which are 0-9 for the digits dataset. This means that the output layer is 10 neurons. The hidden layer has 128 neurons as specified.

### (B) Loss function and optimizing network

Cross Entropy has been utilised for the loss function. The optimizer that has been selected is from torch.optim.Adam. This is because it combines the best properties of the AdaGrad and RMSprop and works by keeping two moving averages for each parameter. One for the square of the gradients like AdaGrad, and one for the gradients like RMSprop. It uses these estimates to adjust the learning rate for each parameter individually. Adam also tends to perform well with a range of hyperparameter values and is less sensitive to the choice of value for the learning rate. It is important to set an appropriate learning rate because when it is too large, we can jump over a deep valley, but when it is too small, we can slowly descend into a local minimum and miss the deeper valley. Therefore, an initial learning rate of 0.001 was chosen as a starting point, but the Adam optimizer automatically adjusts the learning rate based on the gradients observed during training.

```
Epoch 1: Loss = 1.859, Accuracy = 65.52%
Epoch 2: Loss = 1.705, Accuracy = 65.52%
Epoch 3: Loss = 1.325, Accuracy = 62.07%
Epoch 4: Loss = 0.919, Accuracy = 79.31%
Epoch 5: Loss = 0.941, Accuracy = 75.86%
Epoch 6: Loss = 0.571, Accuracy = 93.10%
Epoch 7: Loss = 0.422, Accuracy = 82.76%
Epoch 8: Loss = 0.504, Accuracy = 86.21%
Epoch 9: Loss = 0.347, Accuracy = 93.10%
Epoch 10: Loss = 0.327, Accuracy = 89.66%
Epoch 11: Loss = 0.383, Accuracy = 93.10%
Epoch 12: Loss = 0.315, Accuracy = 93.10%
Epoch 13: Loss = 0.089, Accuracy = 100.00%
Epoch 14: Loss = 0.305, Accuracy = 89.66%
Epoch 15: Loss = 0.207, Accuracy = 93.10%
```

*Figure 5: training loss and accuracy of the training set*

### (C) Evaluate model on the test set

```
Test Accuracy: 91.67%
Test Image 1: Predicted Label = 4, Actual Label = 4
Test Image 2: Predicted Label = 5, Actual Label = 5
Test Image 3: Predicted Label = 1, Actual Label = 1
Test Image 4: Predicted Label = 4, Actual Label = 4
Test Image 5: Predicted Label = 2, Actual Label = 1
```

*Figure 6: test set accuracy and 5 example predictions with their actual labels*

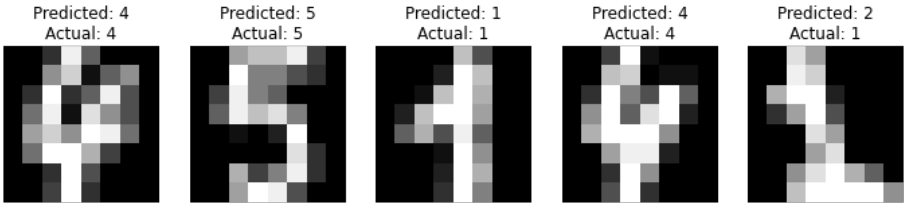



*Figure 7: example test images with their predicted and actual labels*

**(D) Discuss different learning rates**

Having different learning rates impacts the training process. If the learning rate is set too high, then there may be an unstable training process where the model converges too quickly to a suboptimal solution. Larger learning rates require fewer training epochs due to rapid changes. A learning rate that is too low can cause training to get stuck, causing a long training process. Lower learning rates need more training epochs due to smaller changes made to the weights in each update (Brownlee, 2020b).

**(E) Comparing different activation functions**

The test accuracy increased from 91.67% with the ReLu activation function for the hidden layer to 91.94% for the Sigmoid activation function. The test accuracy using the Tanh activation function for the hidden layer is 91.67%, which is the same as the ReLu activation function. This shows that the ReLu or Tanh activation function are most effective for the digits dataset. The ReLu activation function is a non-linear function, that returns the input if it is positive, and zero otherwise. In contrast, the sigmoid activation function maps input values to the range (0, 1) using a smooth S-shaped curve, making it suitable for binary classification tasks. The sigmoid activation function is also non-linear. The Tanh activation function is similar to the sigmoid, but maps input values to (-1, 1). The test accuracy obtained from the ReLu or Tanh activation functions suggest that either are suitable for the digits dataset. It suggests that the networks test performance is likely to be influenced by other factors such as the models architecture and optimization methods.

**(F) The network architecture**

Adding more hidden layers to this networks architecture will allow for the neural network to make more complex calculations. In the context of this classification task, adding more hidden layers may allow for the network to capture more features from the input data. In the case of handwritten digits, adding more hidden layers into the neural network may mean that higher level patterns such as curvature or stroke orientation are captured in the model. This would mean that model performance on the training set would increase. However, this could lead to overfitting, as the test accuracy may decrease due to the model learning the training data too well. This means that the models performance on the test set could worsen if more hidden layers were added, meaning that the neural network would not generalise well to new, unseen data.

Changing the number of neurons in a layer may impact the models performance depending on whether neurons are added or if neurons are removed. If neurons are added in a layer, features can be classified with greater resolution. The layer also becomes more flexible in its ability to model non-linear relationships, leading to better performance on the training and test set.

With fewer neurons in a layer, the layer has less capacity to learn complex patterns, which could lead to underfitting. Reducing the number of neurons may mean the neural networks accuracy decreases and isn't able to classify handwritten digits as well as more neurons can.

**References**

*Adjusting the number of bins in a histogram* . (n.d.). Campus.datacamp.com. Retrieved April 9, 2024, from https://campus.datacamp.com/courses/statistical-thinking-in-python-part-1/graphical-exploratory-data-analysis?ex=7#:~:text=The%20%22square%20root%20rule%22%20is

Brownlee, J. (2020a, August 17). *kNN Imputation for Missing Values in Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/

Brownlee, J. (2020b, September 12). *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Fernandes, A. A. R., & Solimun, S. (2022). Comparison of the Use of Linkage in Cluster Integration With Path Analysis Approach. *Frontiers in Applied Mathematics and Statistics*, *8*. https://doi.org/10.3389/fams.2022.790010

Gadakari, M. (2019, August 1). *Hierarchical Agglomerative Clustering (HAC) with Single linkage method*. Medium. https://medium.com/@MaheshGadakari/hierarchical-agglomerative-clustering-hac-with-single-linkage-method-1159fa623d52#:~:text=The%20hierarchical%20agglomerative%20clustering%20uses,tree%20structure%20as%20named%20dendogram

Geetha, S. (2022, January 16). *Hierarchical Clustering - Types of Linkages*. Sai's Data Website. https://www.saigeetha.in/post/hierarchical-clustering-types-of-linkages#:~:text=Average%20Linkage%3A