**DMV302**

**Assesment2**

**REPORT**

**Q.A.1(b)**

**Interpreting the Cluster:**

**CLUSTER:1** represents households with moderate total assets and income. These could be middle-class families or individuals, possibly in the early or mid-stage of their career, or small business owners. Their financial profile suggests a focus on stability and steady growth in wealth.

**CLUSTER:2** includes households with significantly higher asset values but only slightly higher income compared to Cluster 1. This could represent wealthy individuals or families who have accumulated substantial assets over time, possibly including high-value properties or investments. Their high asset value relative to their income might indicate a matured stage in wealth accumulation, with a potential focus on asset management and preservation.

**CLUSTER:3** falls between Cluster 1 and Cluster 2 in terms of asset value, but its income level is comparable to the other two clusters. This group might consist of professionals or entrepreneurs who have been successful in accumulating assets more rapidly than the average middle-class household, possibly due to higher-risk, higher-return investments or successful business ventures.

**Q.A.1(C)**

Any two ways for determining the cluster are:

**Elbow Method:** Plot the sum of squared distances of samples to their closest cluster center (inertia) against the number of clusters (K). The "elbow" point, where the rate of decrease sharply changes, suggests an optimal K. This method is intuitive but sometimes subjective, as the elbow may not be well defined.

**Silhouette Score:** This measures how similar a sample is to its own cluster compared to other clusters. A high silhouette score implies well-separated, cohesive clusters. By calculating the silhouette score for different values of K, the optimal number of clusters is identified as the one with the highest score. This method provides a more objective metric compared to the elbow method.

**Q.A.1(D)**

To report the optimal K in terms of the Dunn index and inertia:

The Dunn index is a measure of how well-separated the clusters are. A higher Dunn index indicates better clustering. Therefore, look for the K with the highest Dunn index.

The Inertia measures the compactness of the clusters. A lower inertia value is preferable. Hence, find the K with the lowest inertia value.

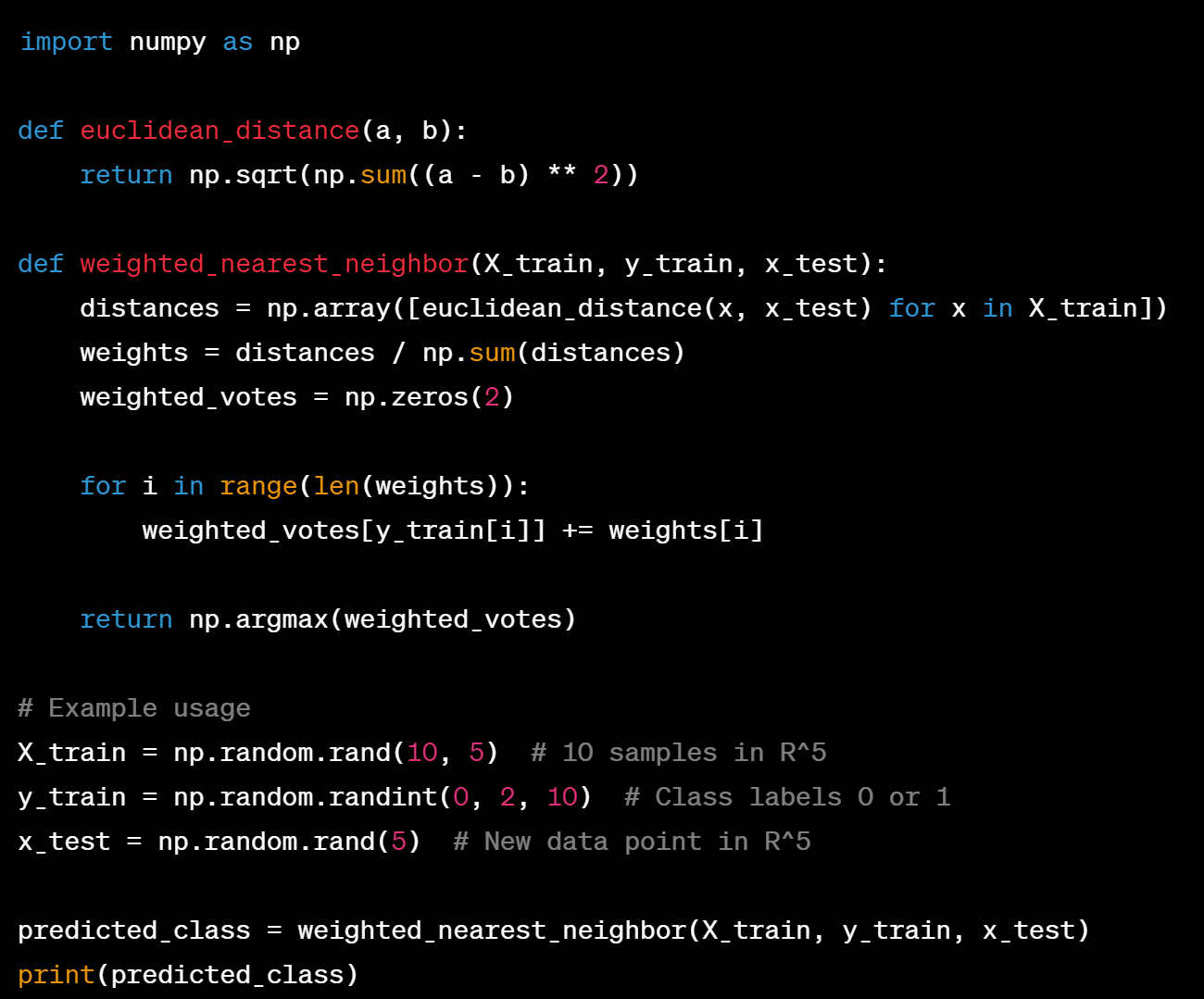
Here, is the required result:

|  |  |  |
| --- | --- | --- |
| **K** | Dunn Index | INERTIA |
| 2 | 0.673 | 493,039,335,348,780.7 |
| 3 | 0.390 | 242,796,850,441,736.3 |
| 4 | 0.264 | 134,807,065,726,074.7 |
| 5 | 0.191 | 88,464,072,120,466.6 |
| 6 | 0.149 | 68,443,507,215,233.88 |
| 7 | 0.146 | 48,211,674,876,270.945 |
| 8 | 0.118 | 37,867,884,928,564.48 |
| 9 | 0.088 | 32,850,154,089,886.79 |
| 10 | 0.068 | 30,313,641,257,305.652 |

**Q.A.2(A)**

1NN, KNN, nNN, and Weighted Nearest Neighbour classifiers each have unique advantages and disadvantages:

1. **1-Nearest Neighbor (1NN)**
   * **Advantages**: Simple, fast, and effective in cases with well-separated classes.
   * **Disadvantages**: Highly sensitive to noise and outliers; poor generalization.
2. **K-Nearest Neighbor (KNN)**
   * **Advantages**: Better generalization than 1NN; more resistant to noise.
   * **Disadvantages**: Requires choosing the right 'k'; computationally intensive for large datasets.
3. **n-Nearest Neighbor (nNN)**
   * **Advantages**: Considers all training data, potentially capturing more complex patterns.
   * **Disadvantages**: Very computationally expensive; can be biased by majority classes.
4. **Weighted Nearest Neighbor**
   * **Advantages**: Weights allow prioritization of closer points, potentially improving accuracy.
   * **Disadvantages**: Requires a well-chosen weighting scheme; computationally intensive.



**Q.A2.(C)**

The weighting factor in (b) inversely correlates with distance, which aligns with intuition: closer points are more relevant. However, it's counterintuitive as smaller distances yield larger weights, leading to an unintended emphasis on distant points. A better scheme might be A black and white math symbol

Description automatically generated is a small constant to avoid division by zero. This assigns higher weights to closer points, aligning better with common sense.

**Q.A.3.(a)**Here is a description of how to implement a multi-layer neural network with one hidden layer using a machine learning library.

**Implementation Description**

**Neural Network Library**: The implementation uses **scikit-learn**, a popular machine learning library in Python. This library offers a range of tools for data mining and data analysis and is built on NumPy, SciPy, and matplotlib. Specifically, the **MLPClassifier** class is used for creating a multi-layer Perceptron (MLP) that operates as a neural network for classification tasks.

**Activation Function and Neurons**:

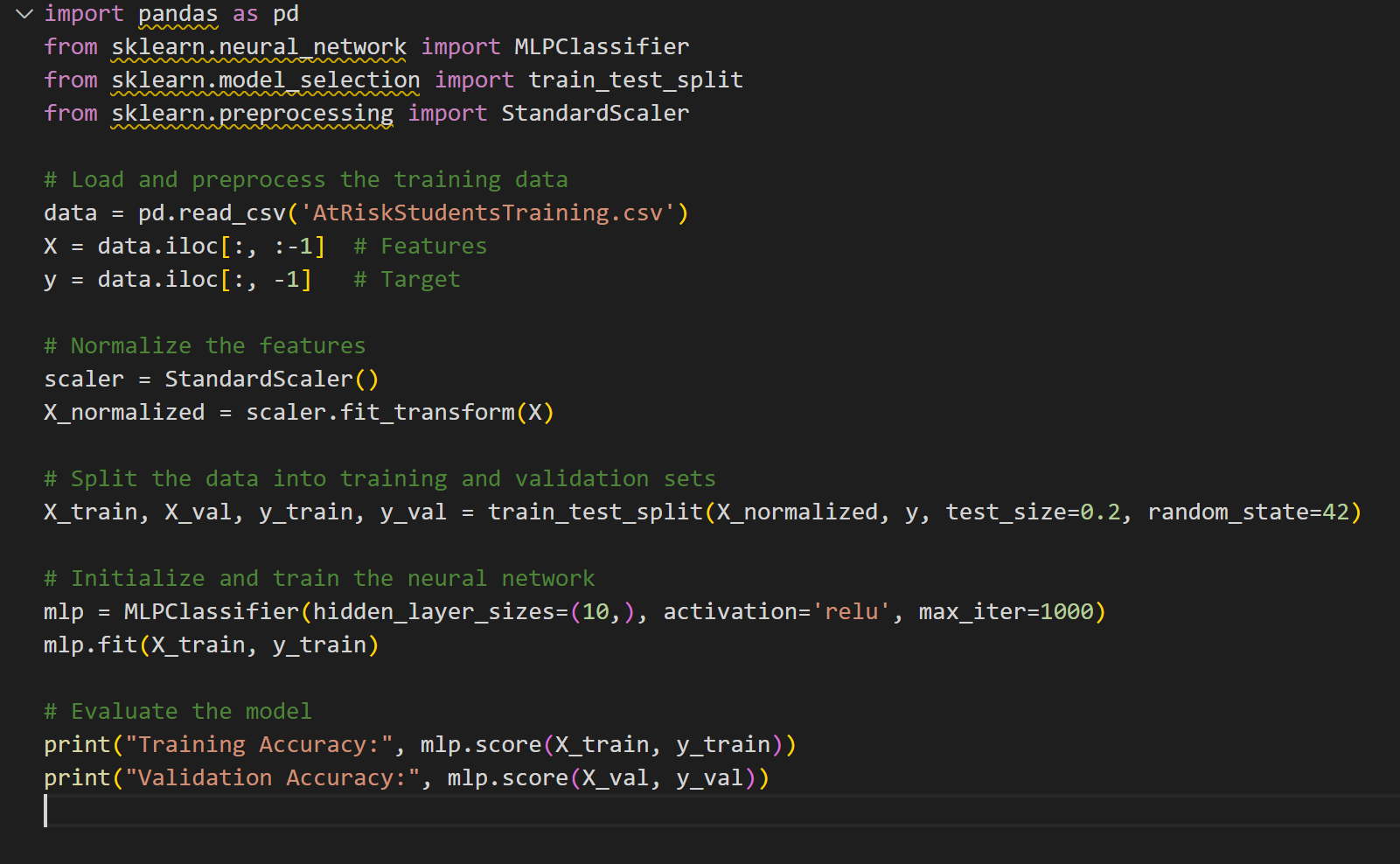
* **Activation Function**: The ReLU (Rectified Linear Unit) activation function is chosen for the hidden layer. ReLU is commonly used in neural networks because it helps with the vanishing gradient problem, allowing models to learn faster and perform better. It is defined as A black text on a white background

  Description automatically generatedand is particularly effective for models dealing with non-linear data.
* **Number of Neurons**: The hidden layer is configured with 10 neurons. This number is a starting point that balances complexity and computational efficiency. It's enough to capture complex patterns in the data but not so large as to cause significant overfitting.

**Training the Neural Network**:

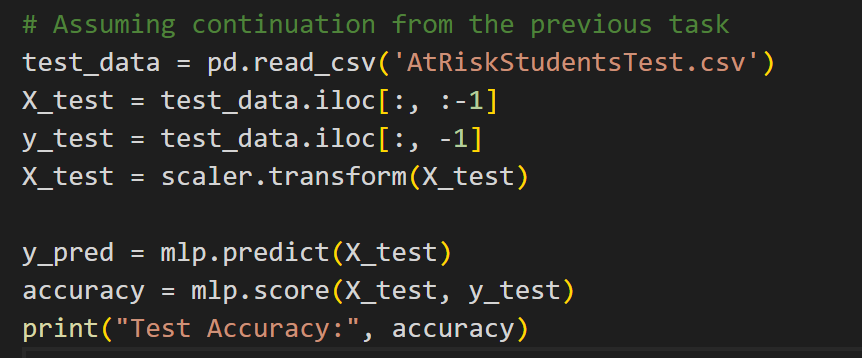
* The training process involves feeding the **AtRiskStudentsTraining.csv** dataset into the neural network. This dataset includes features such as average GPA, attendance record, duration of access to the learning system, and language test results. These features are used to predict whether a student is at-risk (binary classification).
* The model's performance is primarily determined by its ability to accurately classify the at-risk students.

Here is the Coding part:



Q.A.3.(b)

Validating the code:



**Q.A.3.(d)**

For the discussion on drawbacks:

* **Limited Exploration**: The process described in (c) involves a limited exploration of possible configurations. There are potentially better configurations that were not tested due to the practical constraints of computational resources and time.
* **Risk of Overfitting**: Increasing the complexity of the network (more layers/neurons) might lead to overfitting, especially if the dataset is not large enough. Overfitting results in high accuracy on training data but poor generalization to new data.
* **Computational Cost**: Training multiple models with varying configurations can be computationally expensive and time-consuming, particularly with more complex networks.
* **No Systematic Hyperparameter Tuning**: The approach lacks a systematic way of tuning hyperparameters like learning rate, batch size, or regularization, which can also significantly impact model performance.

**Q.A.4**

The code shopping.py does the following:

* Reads the CSV file to get the transactions.
* Implements functions to calculate the support and confidence.
* Provides a menu-driven interface for the user to select options and input items.
* Continuously prompts the user for input until the exit option is chosen.

Result :

Based on the transactions, the results are as follows:

1. **Support for 'eggs'**: Approximately 17.97%. This means that eggs are present in about 17.97% of the transactions.
2. **Confidence for purchasing 'chocolate' given that 'cookies' were bought**: Approximately 12.94%. This indicates that there is about a 12.94% chance of chocolate being bought when cookies are in the transaction. ​