

IMAGE DATASET CREATION AND ANALYSIS

Name: Sushanta M Hegde

Reg No.: 2023MDTS07ALA029

Submitted to: Prof. K. A. Venkatesh

Date: 05-10-2024(Saturday)

**Overview**

This project aims at generating an image dataset consisting of 200-500 images categorized into such classes as "Apple Fruits," "Mango Fruits," "Banana," "Car," "Truck," "Train," "Pizza," "Person," "Tiger," "Cat," "Elephant," and "Bicycle." The data obtained from Bing Images was then transformed to grayscale to simplify the analysis process. It explores various machine learning techniques for image classification and clustering. The techniques include exploring both gray-scale images and RGB-converted data. This project begins with the collection and categorization of images followed by data pre-processing steps which entail conversion into gray-scales of images, scaling pixel values, and reshaping the data to feed into the classification and clustering algorithms. Class imbalance is further dealt with the help of SMOTE (Synthetic Minority Oversampling Technique), which recommends fair representation of all the categories involved.  
For classification, we use Logistic Regression as a very simple yet effective technique to classify the images into respective classes. Thus, it works as a baseline model. In addition, a Multilayer Perceptron classifier is used to identify non-linear patterns in the data to provide better flexibility. Metrics like accuracy, precision, recall, and F1-score are used to train and evaluate the MLP model. On the clustering side, the project uses K-Means clustering for the pixel-intensity-based grouping of images. Since PCA reduces the dimensionality of the dataset, the process of clustering is efficient. These are subsequently used to obtain the optimal number of clusters, resulting from the above-mentioned methods: Silhouette Score, Davies-Bouldin Score, and Elbow Method. Lastly, a confusion matrix is generated in order to compare the K-Means clusters with the original labels to measure how well the unsupervised clusters fit the true image categories.

**Dataset creation steps**

1. Images Download: Download multiple images with the help of Bing Images. These images should be kept in a structured folder hierarchy arranged in categories such as "Apple Fruits," "Mango Fruits," "Banana," "Car," "Truck," "Train," "Pizza," "Person," "Tiger," "Cat," "Elephant," "Bicycle.", and so on.

2. Folder Path Setting: Set paths for image\_folder that refers to the main folder in which original color images will be saved, and for grayscale\_folder that indicates the folder in which grayscale images are to be saved. This folder needs to be created if it doesn't exist.

3. Initalize Categories: Create a list of categories corresponding to the image\_folder's subfolders.

4. Create Grayscale Folders: For each category in the list, check to see if a corresponding subfolder exists within the grayscale\_folder. If not, initialize one.

5. Process Images: For each category subfolder within the image\_folder directory:

Open the image using PIL's Image.open().

Convert the image to grayscale with img.convert("L").

Save the greyscale version of the image in the appropriate category folder in grayscale\_folder

Print message indicating that the image was successfully processed and saved

6. Error Handling: Use error handling to skip the files, presumably corrupted form of image file or unsupported formats. Then print that a file was skipped.

7. Completion Message: At the end of processing all images in all categories, print a message that all images are now converted and saved to the grayscale\_folder.

8.Set Grayscale Folder Path: Set the path for the grayscale\_folder so further creation of datasets could take place.

9. Initialize Image Data and Labels: Initialize image\_data and labels as pixel data and corresponding category labels in lists.

10. Iterate Over Categories and Images: For each subdirectory (category) in grayscale\_folder iterate over the category folder:

Open the image from PIL using Image.open().

Resize the image to 64x64 pixels so it is the same size.

Normalize the image by turning it into a numpy array and then just divide pixel values by 255.0

Add the data from the processed image to the list image\_data and the label of the category to the list labels.

11. Handle Exceptions: Catch any exceptions that are raised when processing the images, and print messages for any files skipped if it has errors.

12.Convert Lists to NumPy Arrays: After all images have been processed convert the image\_data and labels lists to numpy arrays using np.array() so it's in a format compatible with machine learning models.

13.Set Dataset Folder Path: Store in the variable a path towards the folder where the dataset will be saved in both the CSV and NPZ formats.

Create and Save Dataset Iterate over subfolders in grayscale\_folder; process each image to make a flattened and normalized image; append to lists for image\_data and labels. After processing all images create a DataFrame from the list of image\_data and add another column called labels; save it as a CSV called grayscale\_images.csv. Save the processed image data and labels into an NPZ-file named grayscale\_images.npz.

14. Output Messages: It will print the paths at which the dataset has successfully been stored in both formats-name and NPZ, which will, therefore, confirm that the dataset is well-built.

Note on Dataset Imbalance

(This dataset exhibit imbalance or bias due to same in the number of images collected across categories. Addressing class imbalance may be necessary for model training to ensure fair representation and performance.)

**Image Classification Using Logistic Regression**

1. Importing Libraries: Libraries are imported at the very top that are required to be done for different operations in the script are shown below:

NumPy: For doing numerics and manipulation of arrays

Pandas: It is used for data operations and analysis.

Matplotlib: It is a plotting library through which data can be visualized

Scikit-learn: It is a machine learning library that offers tools to train models, evaluate them, and preprocess data.

2. Loading the Dataset

The dataset is read from a pre-defined CSV file. This process involves reading the data into a DataFrame and displaying the first few rows, just for validation purposes of the contents. There are grayscale images with their labels in the dataset.

3. Data Preprocessing

In this step, the script splits the features; that means image data, and the labels, which are categories, off the DataFrame as follows:

By deleting the column of the labels, the feature matrix will be produced.

The labels are extracted into a separate array.

Each image is reshaped to create a one-dimensional array for processing purposes

Features standardized to have zero mean and unit variance using scaling methods

4. Sample Images Visualization

To gain a better understanding of the dataset, a specified number of sample images from each category are visualized as follows,

Grid layout created in order to display all images

Each image is plotted as black and white which represents pixel intensity.

Titles of the images categorize them.

5. Training Logistic Regression Models

For every category in the dataset, the script trains a different Logistic Regression model:

In a for loop over all categories in the dataset, it creates a binary target variable for each category, i.e., whether the image belongs to that category or not.

It will split the dataset into 70% for training and 30% for testing to ensure that 70% of the data is used for training the model and 30% for testing its performance.

Logistic Regression model is trained on the training data, and predictions are made on the test set. The accuracy of the model, along with detailed classification report, is generated for each category that sums up the performance of the model.

6. Displaying Results

Finally, the results are shown in a readable format:

The accuracy of each class is printed, showing just how well the model has done

Detailed classification report for each class of averages, including precision, recall and F1 score -.

**Results and Interpretations:**

General Interpretation:

Accuracy:

For the given class, accuracy ranges between 86.30% to 95.89%. Accuracy gives an overall view about how a model has performed it, is, in fact, misleading for imbalanced data sets as well, such as banana, bicycle, and cat. Precision: Precision gives the percentage of true positives with respect to total predicted positive ones. Precision for bicycle class is 0.958; the model makes good predictions for class '0', which is the major class. For minority classes (such as 1 in banana, mango frits, and cat), precision is very low or even 0, which means the model actually has a problem with those categories. Recall:

Recall is the fraction of true positives of all actual positives. Just like precision, recall of class 0 is very high, but class 1 is very low for most categories like car, cat, pizza, and truck. That means the model has missed many positive instances of the minority class since it is low on the class 1 recall.

F1 Score:

The F1-score is the harmonic mean of precision and recall, and it is a good scoring function for models in imbalanced datasets. For example, categories like an elephant (F1 = 0.925 for class 0) indicate that the model performs very well on majority classes but attains really poor F1-scores on the minority classes across categories; thus, the model fails to cope with imbalanced data. Imbalanced data insights There is a huge imbalance of dataset concerning all classes except the majority class. Class 0 has an instance more than the instances of class 1. With such a problem comes a series of troubles:

Majority Class Dominance:

The model generally was focused on prediction of the majority class 0 with high accuracy and precision but neglected minority class 1. For instance, as we can see, categories of bike and pizzas have high precision values for class 0 but 0 values for class 1. Recall for Minority Class:

It does not detect the instances of the minority class because the recall for class 1 is relatively very low for all categories. For example, Recall = 0 for class 1 when the input is banana or cat, which means almost all the instances of the minority class are missing. F1-Score and Class Imbalance:

F1 scores for class 1 in most of the categories range from very low to 0, respectively (for example, banana, cat, mango frits). This means that although it is performing well on the majority class, it actually isn't much of a help in finding the minority class. Macro vs. Weighted Averages:

Macro avg gives same weightage to both the classes irrespective of instances and have low scores of all type like mango frits and truck indicating overall poor performance of the model. Weighted avg considers the class imbalance and shows comparatively better scores since it is getting affected by the performance of the majority class.

Category-wise Interpretation:

Accuracy for Apple Frits:

90.41% Class 0 Majority Class High precision at 92.75% and recall at 96.97% means that the model does well on class 0 classification. Class 1 Minority Class Low precision at 50% and recall at 28.57% represent that the model is classifying instances of class 1 poorly; it may just be because of class imbalance. Overall Weighted average F1-score is 89.21%, meaning that the performance of the model is shifted toward class 0 while there is much weaker performance about class 1.

Banana Accuracy:

89.04% Class 0 Strength Strong precision and recall on class 0 with values of 92.86% and 95.59%, respectively. Class 1 There are zero precision and recall for class 1, which means the model could not pick up any true positives for class 1, very likely due to the extremely low instances of class 1 samples (5). Overall: The model strongly biases toward class 0, since class 1 rarely occurs.

Bicycle Accuracy:

95.89% Class 0: Excellent performance for class 0 with precision and recall at 95.89% and 100%, respectively. Class 1: The model entirely fails to detect class 1, with zero precision, recall, and F1-score. This could be due to a small sample size of class 1 that comprises only 3 samples. In general, it shows that there is a huge imbalance between the classes where the model is biased towards predicting the major class.

Car Accuracy:

91.78% Class 0: Strong performance with a perfect recall of 100% and a high precision of 91.67% for class 0. Class 1: Precision is 100%, but the recall is only 14.29%; therefore, the model rarely detects class 1, except for a few instances. Overall: The model shows good accuracy but fails to consistently detect class 1, leading to the conclusion that the performance on the minority class is weak.

Cat Precision:

90.41% Class 0: Both precision and recall are very high (91.67% and 98.51%, respectively), so that the model is performing really well when it predicts class 0. Class 1: There were no correct predictions of class 1, because precision as well as recall are zero. Overall: It is obvious that the classes are imbalanced since the model cannot detect class 1 at all, but very biased towards class 0.

Elephant Precision:

86.30% Class 0: High precision - 88.57% and recall - 96.88%, in which the model's performance is strong on class 0. Class 1: Precision is 33.33% but recall only 11.11%, meaning the model cannot pick the positives of class 1. General: This class represents a typical case of imbalanced performance, in which the model has a tendency towards favoring class 0.

Mango Frits Precision:

91.78% Class 0: Its precision and recall are both at 93.06% and 98.53%, respectively, which is excellent performance on the dominant class. Class 1: The model does not predict any instances of the minority class; its precision, recall, and F1-score are all zero. Overall: As most other categories, the model performed terribly on the minority class.

Person Accuracy:

93.15 % Class 0: Precision and recall both are good at 95.65% and 97.06% respectively, meaning that the model performs great at class 0. Class 1: Class 1 detection occurs with a moderate improvement as precision and recall come at 50% and 40%, respectively, meaning the model has been successful in giving the correct predictions for class 1 yet recall still lags. Overall:

The model does much better on class 1 than any other types of classes but an improvement is still seen.

Pizza Precision:

Class 0 Accuracy 93.15%, Recall 100% The above indicates that the model is doing very well in class 0 with high precision and recall. Class 1 Precision 0%, Recall 0%, F1-score 0% This reflects that the model cannot identify all instances which are class 1 at all Overall Class imbalance, the model is highly biased toward predicting class 0.

Tiger Accuracy:

87.67% Class 0: Precision and Recall for class 0 are high at 90.14% and 96.97% respectively Class 1: Model never predicts class 1 with zero precision and recall Overall: it overpredicts class 0 and does not predict class 1 properly.

Train Accuracy:

87.67% Class 0: Just like in the previous case of class tigers, precision is very high for the class 0 at 90.14% and recall at 96.97%. Class 1: No class 1 was detected with zero precision and recall. Summary: The model highly biases toward class 0 and has not detected any class 1.

Truck Accuracy:

90.41 % Class 0: Precision is very high (90.41 %) with a perfect recall of 100 %. Thus, this model works like a champion for class 0. Class 1: No class 1 was correctly predicted, with zero precision, recall, and F1-score. Overall: The model is biased towards predicting class 0 as it completely fails to detect class 1.

**Image Classification Using Perceptron:**

1. Importing Libraries

The script starts by importing all the libraries needed:

Pandas: To manipulate and analyze data

Numpy: Performed for any type of numerical operations and array handling

Scikit-learn: A tool kit of machine learning to train models, evaluate the models, and preprocess data

Imbalanced-learn: Specialized library to handle imbalanced data specifically with SMOTE

2. Load the Dataset

The dataset is read from a named CSV file into a DataFrame. In this step, data and corresponding labels were read from the file.

3. Data Preprocessing

The script preprocesses the data as follows:

Feature matrix

X is created by removing the column of the labels from the DataFrame.

StandardScaler is used to standardize the features so that they have zero mean and unit variance. This will cause convergence to be faster for the model, and hence, better performance will be achieved.

4. Balancing Classes

As the classes can be imbalanced in the data set, the script assumes that class balancing will be performed using SMOTE (Synthetic Minority Over-sampling Technique)

A binary target variable y is generated for each label, wherein the label of interest is marked as 1 and all others as 0.

The dataset was split, using a 70:30 ratio into a training set and a test set, where 70% of the data were used to train and 30% to test.

The SMOTE technique is applied to create synthetic samples for the minority class for the training data set in order to make the dataset under consideration balanced before carrying out the training of the model.

5. MLP Classifier Training

Initializes and trains Multi-Layer Perceptron classifier using this script

An instance of the MLPClassifier with specified parameters like a hidden layer size of 5, a ReLU activation function, 1000 iterations maximum as well as random seed for reproducibility.

(this model is repeated for single hidden layer with 15 and 20 neurons also)

The MLP model is trained upon the balanced training data which has been produced by SMOTE

6. Model Evaluation

After training the model, its performance is measured on the test set:

Predictions are made with the help of test data.

Different performance metrics are calculated by the help of classification\_report and accuracy\_score:

Precision: The ratio of correct predictions of positive observations to total inferences as positive.

Recall: It is the ratio of true positive observations predicted to all the actual present positives.

F1 Score: The weighted average of precision and recall, thus making it balance the two.

Accuracy: Ratio of correct predictions of observations to actual observations.

7. Results Presentation

For each label, the result will be printed in structured format as follows:

Precision, Recall, F1-score, and Accuracy on each class will be shown so that one can see the performance of the model.

**Result and interpretation:**

Generalized Implication of All Models:  
Variability in Performance: The fact that every model has a peculiarity of importance implies that there is variability in performance across classes from one model to another. This means that some classes may be generally hard to classify than the others. This variability in performance might be due to several reasons such as:  
Class Imbalance: When some classes have so many examples while others do not, there is a likelihood that the model will become biased towards the more prominent class.  
Feature Complexity: Some classes may have patterns that may be very complex hence the model cannot learn these and hence low than average classification.  
Inherent Class Similarity: Classes of some cases are mostly identical especially their features hence a struggling model to classify.  
In general, the accuracy across models is impressive. However, it was up to 90% in the peak for some classes. The results may appear quite misleading, especially in a class-imbalanced scenario, if judged at face value by accuracy alone. Some classes are illustrated here with very low precisions and recalls. Thus, there is a need for more in-depth checking of model performance.  
  
Class-wise Interpretation:  
Apple Frits:  
5 Neurons: Precision: 0.15, Recall: 0.29, F1 Score: 0.20, Accuracy: 0.78  
15 Neurons: Precision: 0.50, Recall: 0.43, F1 Score: 0.46, Accuracy: 0.90  
20 Neurons: Precision: 0.33, Recall: 0.43, F1 Score: 0.38, Accuracy: 0.86  
The large performance gain with the 15-neuron model suggests that there is a threshold number of neurons beyond which the model can, in fact learn the discriminating features of the class.  
Banana:  
5 Neurons: Precision: 0.17, Recall: 0.20, F1 Score: 0.18, Accuracy: 0.88  
15 Neurons: Precision: 0.20, Recall: 0.20, F1 Score: 0.20, Accuracy: 0.89  
20 Neurons: Precision: 0.33, Recall: 0.40, F1 Score: 0.36, Accuracy: 0.90  
Comparison: For 'banana', the graph of precision and recall is an upward curve to the peak where more neurons mean it is able to extract and represent the data in such a way that this class could be distinguished better.  
Bicycle:  
All Models: Precision, Recall, and F1 Score: 0.00; Accuracy: 0.82 (5 Neurons), 0.90 (15 Neurons), 0.90 (20 Neurons)  
Comparison: All models catastrophically fail to classify this class well. This means that there is a huge amount of scope for improvement in either the data representation or in the training of the model. In fact, it could also indicate that characteristics of bicycles are not very well represented in the dataset.  
Car:  
5 Neurons: Precision: 0.22, Recall: 0.29, F1 Score: 0.25, Accuracy: 0.84  
15 Neurons: Precision: 0.00, Recall: 0.00, F1 Score: 0.00, Accuracy: 0.85  
20 Neurons: Precision: 0.33, Recall: 0.29, F1 Score: 0.31, Accuracy: 0.88  
Comparison: The performance of the 'car' class is much improved at 20 neurons, which suggests more complex structures would better capture this class; failure of the 15 neuron model could be due to overfitting or a lack of informative features at that architecture level.  
Elephant:  
5 Neurons: Precision: 0.67, Recall: 0.44, F1 Score: 0.53, Accuracy: 0.90  
15 Neurons: Precision: 0.67, Recall: 0.33, F1 Score: 0.43, Accuracy: 0.89  
20 Neurons: Precision: 0.67, Recall: 0.22, F1 Score: 0.33, Accuracy: 0.89  
Comparison: This class is consistent, though not strong, on most models, meaning that the model can pick out elephants even if recall fluctuates. Since the findings are based on some neuron threshold that supports sustained performance, they highlight that probably more neurons do not necessarily give any better performance but maintain for specific classes.  
Mango :  
5 Neurons: Precision: 0.08, Recall: 0.20, F1 Score: 0.11, Accuracy: Higher on other classes  
15 Neurons: Precision: 0.12, Recall: 0.20, F1 Score: 0.15, Accuracy: 0.90  
20 Neurons: Precision: 0.12, Recall: 0.20, F1 Score: 0.15, Accuracy: 0.85  
Comparison: There is no more than minimal improvement across models, indicating that there are still problems with the classification of this category, and improvements in this area should be more targeted; perhaps these could be achieved through feature enhancement or through an increase in the training set.  
Person:  
5 Neurons: Precision: 0.16, Recall: 0.60, F1 Score: 0.25, Accuracy: 0.75  
15 Neurons: Precision: 0.15, Recall: 0.40, F1 Score: 0.22, Accuracy: 0.81  
20 Neurons: Precision: 0.17, Recall: 0.40, F1 Score: 0.24, Accuracy: 0.82  
Comparison: even though precision fluctuated, it still appears to be higher for the 5-neuron network so it doesn't really miss the true instances but instead indicates that sometimes fewer neurons may achieve better detection results for certain classes.  
Pizza  
5 Neurons: Precision: 0.17, Recall: 0.20, F1 Score: 0.18, Accuracy: 0.88  
15 Neurons: Precision: 0.17, Recall: 0.20, F1 Score: 0.18, Accuracy: 0.88  
20 Neurons: Precision: 0.33, Recall: 0.40, F1 Score: 0.36, Accuracy: 0.90  
Comparison: The 20-neuron model shows a dramatic impact on performance; hence higher complexity contributes better in this class's subtleties.  
Tiger:  
5 Neurons: Precision: 0.18, Recall: 0.29, F1 Score: 0.22, Accuracy: 0.81  
15 Neurons: Precision: 0.40, Recall: 0.29, F1 Score: 0.33, Accuracy: 0.89  
20 Neurons: Precision: 0.14, Recall: 0.14, F1 Score: 0.14, Accuracy: 0.84  
Comparison: The performance is model-specific, but one can easily get the feel that a better architecture of neurons and proper tuning may help improve the performance.  
Train:  
5 Neurons: Precision: 0.33, Recall: 0.43, F1 Score: 0.38, Accuracy: 0.86  
15 Neurons: Precision: 0.50, Recall: 0.43, F1 Score: 0.46, Accuracy: 0.90  
20 Neurons: Precision: 0.43, Recall: 0.43, F1 Score: 0.43, Accuracy: 0.89  
Comparison: The amount of the rise in precision with the 15-neuron model is quite enough to have the confidence that higher complexity goes well with the better class representation and so prove the hypothesis that has been proposed; there is an optimal neuron configuration as a threshold.  
Truck:  
5 Neurons: Precision: 0.25, Recall: 0.29, F1 Score: 0.27, Accuracy: 0.85  
15 Neurons: Precision: 0.40, Recall: 0.29, F1 Score: 0.33, Accuracy: 0.89  
20 Neurons: Precision: 0.33, Recall: 0.29, F1 Score: 0.31, Accuracy: 0.88  
Comparison: Even for 15 neurons, the model's accuracy is slightly better than what it had initially achieved when it captured more cases of this class with appropriately tuned architecture.

**Clustering on image dataset:**

1. Data Loading and Preprocessing:

Reads in grayscale image dataset from a CSV file. In this case, each row represents a flattened grayscale image; the 'label' column defines its label and the label column will be dropped since we want to apply our clustering to this data which is just the pixel values:

The second step is using StandardScaler in order to standardize data where every feature must have a mean value of 0 and standard deviation of 1. This is an important step since many clustering algorithms, like K-Means, are sensitive to scale of data.

2. Do a little dimensionality reduction with PCA:

Considering that the dimension of the image data is very high (like 4096 pixels for 64x64 images), the technique used here is PCA to reduce the number of features with maximum variance. In this case, the dimension is reduced to 50 components such that the process of K-Means clustering can be sped up and further reduce noise in the data.

3. Clustering with K-Means:

Next, apply K-Means on the data reduced by PCA. The number of clusters might be equal to the number of categories found distinct in images. K-Means attempts to divide the data into 11 clusters, where each image will end up assigned to the closest cluster centroid. random\_state for reproducibility

The output of clustering is stored in labels, with each image being assigned a number of clusters.

4. Clustering Image Visualization

All images within a cluster are visualized in a grid and up to 5 images per a cluster. Additionally, the title of each cluster (cluster ID), and the original label of each image are included to compare against.

This visualization will assist in understanding whether the clustering has successfully grouped similar images together and if the clusters align with their true labels.

5. Confusion Matrix to Compare Clusters with True Labels:

This confusion matrix is used to compare how well the K-Means cluster assignments match with the actual labels of the images. Since the true labels are unknown in the K-Means algorithm, this comparison will be used to check how well the unsupervised clustering performs with respect to the original classes.

The labels are then encoded as numerics in order to calculate the confusion matrix. Heats of confusion matrix visualize what clusters correspond to true labels and which ones do not correspond.

6. Cluster Evaluation Metrics:

Several metrics will be employed to establish how good the quality of K-Means clusters is at a variety of cluster sizes ranging from 2 to 12:

Silhouette Score: It measures how close or different an image is from its own cluster rather than other clusters. Here, in high-scored clusters, the images form well-defined and separate groups of clusters.

Davies-Bouldin Score: This value measures the overlapping amount of clusters and their compactness. The smaller the score, the better the clustering with a minimum of overlap.

Inertia (Elbow Method): Inertia is the total sum of squared distances of all data points to their corresponding cluster centers. The Elbow Method is useful for determining the number of clusters since the inertia for different numbers of clusters can be plotted versus the number of clusters. The "elbow" point of the curve in this plot is where additional clusters added has diminishing returns.

7. Graphical Results:

A silhouette score plot vs. number of clusters plots: The optimal number of clusters is found at the peak silhouette score.

Davies-Bouldin Score Plot: Plot Davies-Bouldin vs. number of clusters to find the setting producing the minimum score, thereby yielding better clustering.

Elbow Method Plot. It plots the inertia as a function of the number of clusters. The number of clusters that best explains this curve is typically at the "elbow" of the curve, where it no longer substantially reduces inertia to increase further the number of clusters.

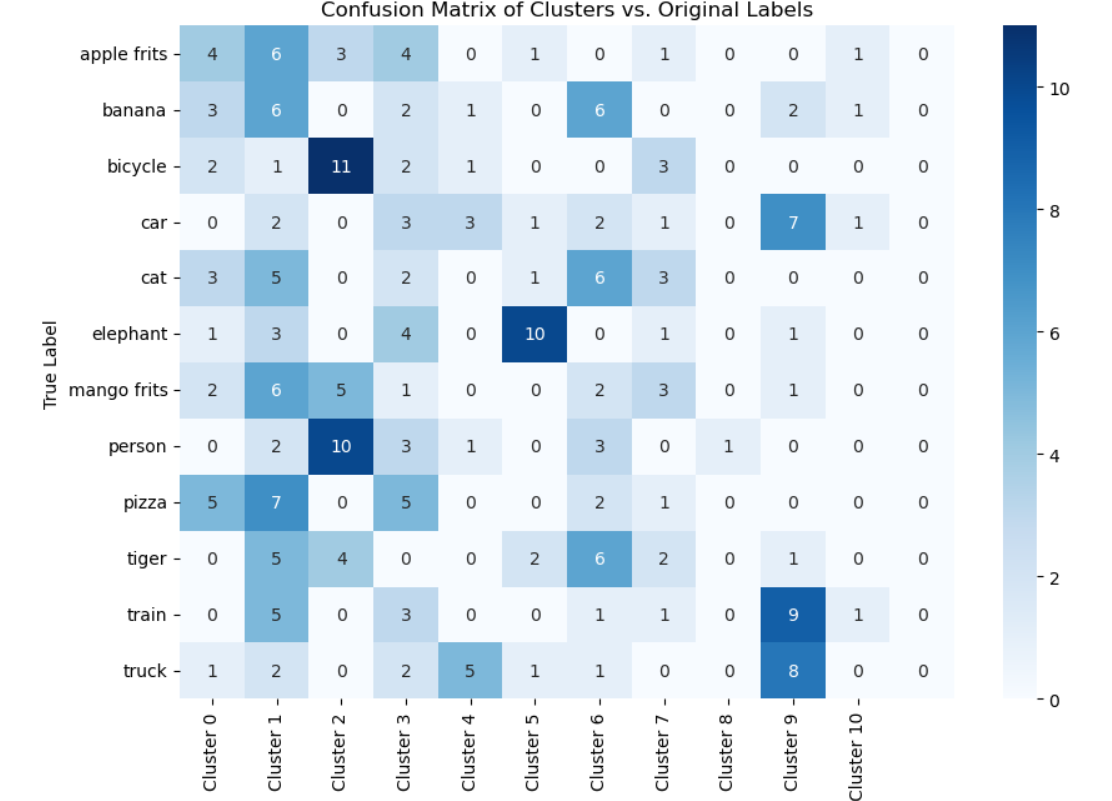
**Result and interpretation:**

these are the plot of valuations where it gives that how for every different values of k the score is varies

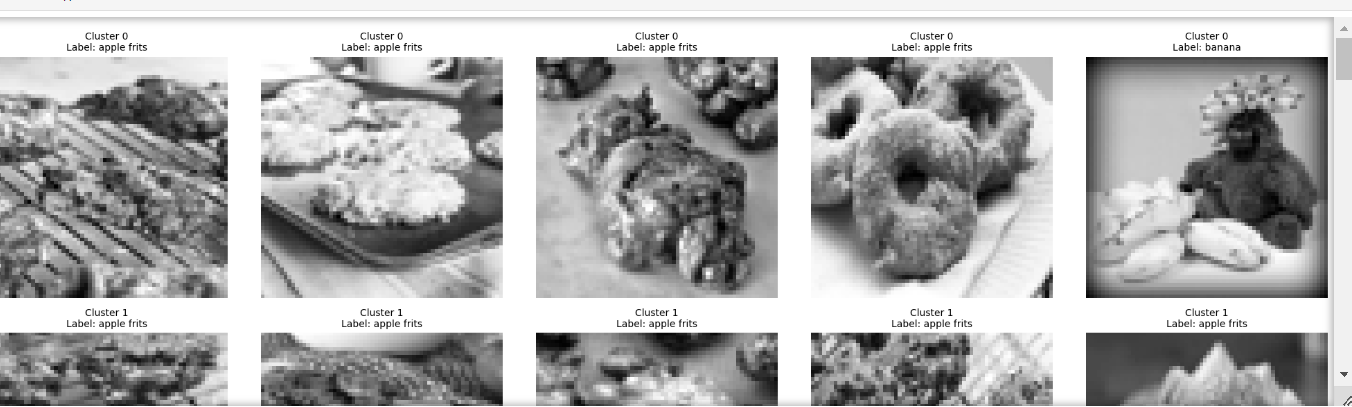
silhotte score says the optimal no. of k is 4

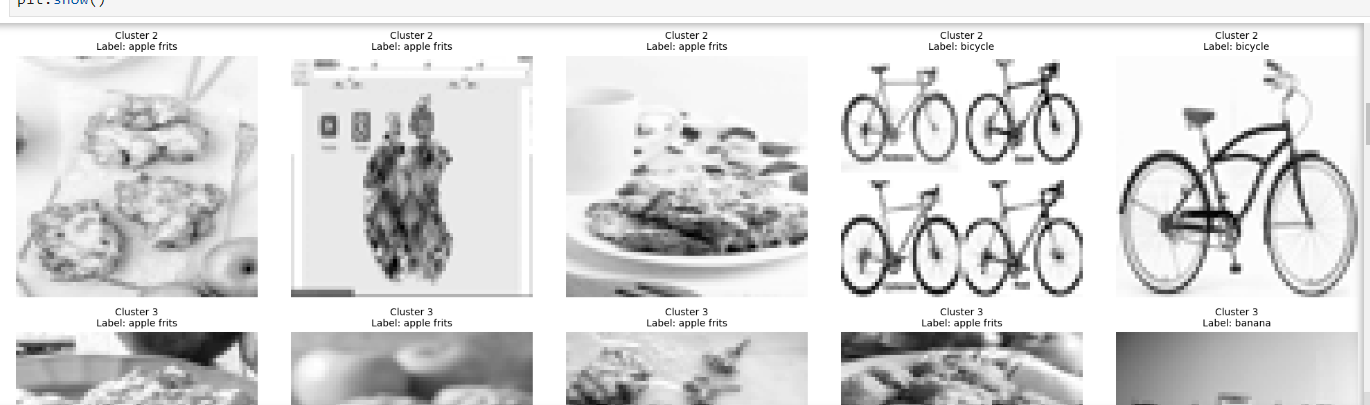
Davies-Bouldin score says 4

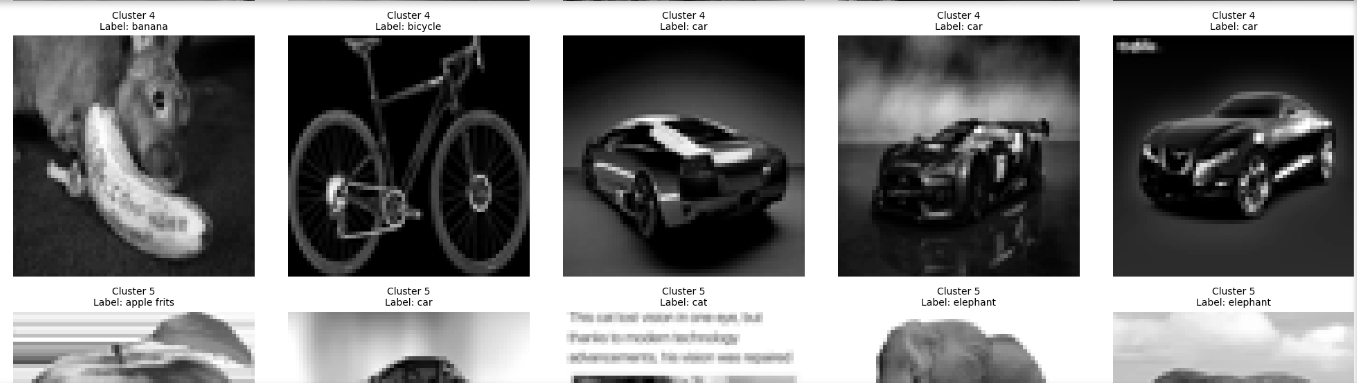
Even the ELBOW method also says k=4 will be the optimal



In this confusion matrix we can see that model is not well designed for the imbalance dataset. Among the 20 images from each category true positive is very less so for the clustering very few similar kinds (same category) of images are lies in the same cluster. I know that clustering cannot be done on the basis of categories because the images are not going to assigned as per their labels. But I want to know that how the original labels and assigned clusters are related so that’s the reason I made this confusion matrix. Like same category images should not varies too much in their value because if we take apple images picture contain same object (single objected images) then it should be lies in the same clusters. That’s the idea.







Clustering on the images is difficult task. In the images if multiple objects are present then the same picture comes under multiple categories for example if a person eating a pizza, it is description for the image, when we categorise it, this image comes in both pizza and person category because it has both objects similarly, we will get lots of that kind of objects.

**Comparison between all the models:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Model | Accuracy | Precision | Recall | F1 Score |
| apple frits | Logistic Regression | 90.41% | 0.93 | 0.97 | 0.95 |
| apple frits | MLP (5 Neurons) | 78% | 0.15 | 0.29 | 0.20 |
| apple frits | MLP (15 Neurons) | 90% | 0.50 | 0.43 | 0.46 |
| apple frits | MLP (20 Neurons) | 86% | 0.33 | 0.43 | 0.38 |
| banana | Logistic Regression | 89.04% | 0.93 | 0.96 | 0.94 |
| banana | MLP (5 Neurons) | 88% | 0.17 | 0.20 | 0.18 |
| banana | MLP (15 Neurons) | 89% | 0.20 | 0.20 | 0.20 |
| banana | MLP (20 Neurons) | 90% | 0.33 | 0.40 | 0.36 |
| bicycle | Logistic Regression | 95.89% | 0.96 | 1.00 | 0.98 |
| bicycle | MLP (5 Neurons) | 82% | 0.00 | 0.00 | 0.00 |
| bicycle | MLP (15 Neurons) | 90% | 0.00 | 0.00 | 0.00 |
| bicycle | MLP (20 Neurons) | 90% | 0.00 | 0.00 | 0.00 |
| car | Logistic Regression | 91.78% | 0.92 | 1.00 | 0.96 |
| car | MLP (5 Neurons) | 84% | 0.22 | 0.29 | 0.25 |
| car | MLP (15 Neurons) | 85% | 0.00 | 0.00 | 0.00 |
| car | MLP (20 Neurons) | 88% | 0.33 | 0.29 | 0.31 |
| cat | Logistic Regression | 90.41% | 0.92 | 0.99 | 0.95 |
| cat | MLP (5 Neurons) | 81% | 0.21 | 0.50 | 0.30 |
| cat | MLP (15 Neurons) | 85% | 0.14 | 0.17 | 0.15 |
| cat | MLP (20 Neurons) | 81% | 0.00 | 0.00 | 0.00 |
| elephant | Logistic Regression | 86.30% | 0.89 | 0.97 | 0.93 |
| elephant | MLP (5 Neurons) | 90% | 0.67 | 0.44 | 0.53 |
| elephant | MLP (15 Neurons) | 89% | 0.60 | 0.33 | 0.43 |
| elephant | MLP (20 Neurons) | 89% | 0.67 | 0.22 | 0.33 |
| mango frits | Logistic Regression | 91.78% | 0.93 | 0.99 | 0.96 |
| mango frits | MLP (5 Neurons) | 78% | 0.08 | 0.20 | 0.11 |
| mango frits | MLP (15 Neurons) | 90% | 0.25 | 0.20 | 0.22 |
| mango frits | MLP (20 Neurons) | 85% | 0.12 | 0.20 | 0.15 |
| person | Logistic Regression | 93.15% | 0.96 | 0.97 | 0.96 |
| person | MLP (5 Neurons) | 75% | 0.16 | 0.60 | 0.25 |
| person | MLP (15 Neurons) | 81% | 0.15 | 0.40 | 0.22 |
| person | MLP (20 Neurons) | 82% | 0.17 | 0.40 | 0.24 |
| pizza | Logistic Regression | 93.15% | 0.93 | 1.00 | 0.96 |
| pizza | MLP (5 Neurons) | 88% | 0.17 | 0.20 | 0.18 |
| pizza | MLP (15 Neurons) | 88% | 0.17 | 0.20 | 0.18 |
| pizza | MLP (20 Neurons) | 90% | 0.33 | 0.40 | 0.36 |
| tiger | Logistic Regression | 87.67% | 0.90 | 0.97 | 0.93 |
| tiger | MLP (5 Neurons) | 81% | 0.18 | 0.29 | 0.22 |
| tiger | MLP (15 Neurons) | 89% | 0.40 | 0.29 | 0.33 |
| tiger | MLP (20 Neurons) | 84% | 0.14 | 0.14 | 0.14 |
| train | Logistic Regression | 87.67% | 0.90 | 0.97 | 0.93 |
| train | MLP (5 Neurons) | 86% | 0.33 | 0.43 | 0.38 |
| train | MLP (15 Neurons) | 86% | 0.00 | 0.00 | 0.00 |
| train | MLP (20 Neurons) | 89% | 0.43 | 0.43 | 0.43 |
| truck | Logistic Regression | 90.41% | 0.90 | 1.00 | 0.95 |
| truck | MLP (5 Neurons) | 85% | 0.25 | 0.29 | 0.27 |
| truck | MLP (15 Neurons) | 85% | 0.00 | 0.00 | 0.00 |
| truck | MLP (20 Neurons) | 88% | 0.33 | 0.29 | 0.31 |

**Performance Metrics Overview:**

The table presents multiple performance metrics for each model: Accuracy, Precision, Recall, and F1 Score.

Accuracy measures the overall correctness of the model.

Precision indicates the proportion of positive identifications that were actually correct.

Recall measures the ability of the model to find all the relevant cases (true positives).

F1 Score is the harmonic mean of precision and recall, providing a balance between the two.

Model Comparison:

Logistic Regression consistently shows high accuracy across most categories, often reaching above 90%. It also has strong precision and recall, indicating that it performs well in identifying true positives without many false positives.

MLP Models (Multi-Layer Perceptrons) exhibit varying performance depending on the number of neurons:

5 Neurons: Generally, these models perform poorly in terms of precision and recall, indicating they struggle to identify positive cases effectively.

15 Neurons: Performance improves in several categories but still lacks robustness in identifying some classes.

20 Neurons: This configuration yields mixed results; while some categories show improved precision and recall, others remain low, particularly in classes like bicycle and cat, which show zero scores in certain metrics.

Class-Specific Performance:

Some categories, like bicycle and cat, perform poorly across the board, with zero precision and recall in certain models. This suggests that the models struggle to distinguish these classes effectively, which may be due to similarities with other categories or a lack of training data for those classes.

**Future Improvement**

Advanced Image Augmentation: The size of the given dataset is relatively small in terms of number and increasing advanced data augmentation including random cropping, rotation, flipping, and zoom would help to make further variations in the data for generalizing the model better. This will reduce reliance on synthetic oversampling techniques like SMOTE.

Hyperparameter Optimization: Further, the MLP classifier and K-Means clustering model can be improved by using techniques like Grid Search or Random Search for hyperparameter tuning. An advanced version of Bayesian Optimization can help optimize models much more

effectively.

Multimodal Learning: In addition to including RGB data, including grayscale might pave the way for multimodal learning in which models learn from color as well as variations in intensities, enhancing the classification and clustering.

Evaluation Metrics Extension: Currently, the evaluation metrics are informative but there can also be extensions that may include ROC-AUC for classification or ARI (Adjusted Rand Index) for clustering to have a better insight of the model performance.

Using Autoencoders and LSTM.

**Conclusion:**

In this project, we have built a dataset of images and applied machine learning techniques for classification and clustering of images. We have employed Logistic Regression and MLP to find the ability of both linear and nonlinear classifiers over grayscale images, respectively. The MLP classifier was more flexible and sensitive to the complex patterns in data images. For studying unsupervised learning, we used K-Means clustering assisted by PCA for dimensionality reduction. We also tested the clustering performance in comparison with other metrics, such as Silhouette Score, Davies-Bouldin Score, and the Elbow Method.

Results from this project have shown that, at a small dataset size, machine learning models are capable of image classification and clustering. The use of different kinds of evaluation techniques gave insight into the model's performance and also confusion matrix analysis helped in evaluating the quality of clustering relative to its original label. Future work can be in the direction of increasing the size of the dataset, including more complex models like CNNs, employing transfer learning, and researching deep learning-based clustering methods for further improvements. This project finally establishes a strong basis for continued study into machine learning techniques applied in the context of image analysis.