UNIVERSITY OF BUCHAREST

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE

MASTER SECURITY AND APPLIED LOGIC

Practical Machine learning

REPORT UNSUPERVISED LEARNING PROJECT

Author
Dory Mathis

Professors
Radu Tudor Ionescu, Alexandru Ghita

Bucharest, January 2023

Contents

1 Presentation of the dataset **3**

2 Data preprocessing **5**

3 Kmeans approach **8**

3.1 Description of the approach 8

3.2 Results 8

4 DBSCAN approach **13**

4.1 Description of the approach 13

5 Conclusion **14**

6 Appendix **15**

6.1 Samples with PCA 15

6.2 Samples with InceptionResnet50 20

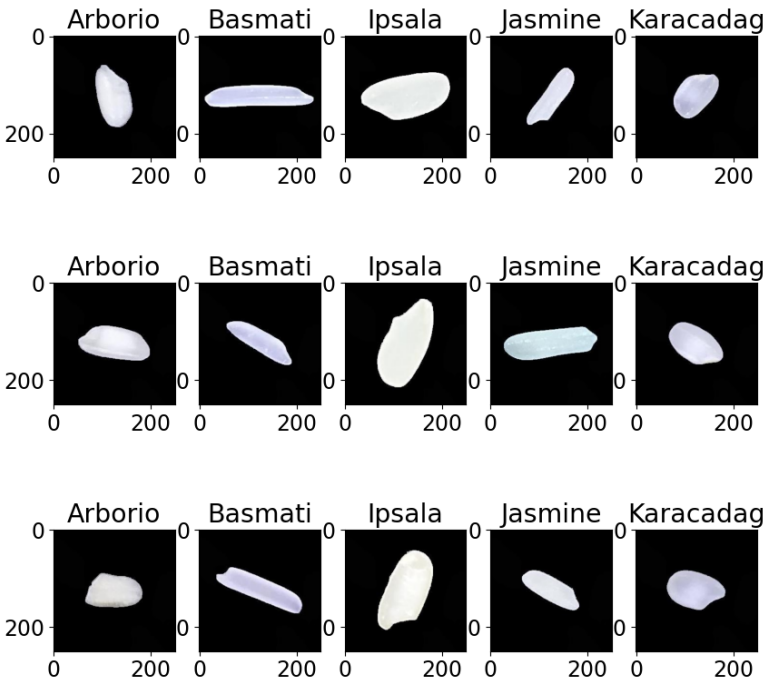
1. Presentation of the dataset

The dataset I chose contains 75 000 images of rice grains. The dataset is divided in 5 folders of 15 000 images. Each folder correspond to a specific kind of rice and the different kinds of rice are the following:

1. Arborio
2. Basmati
3. Ipsala
4. Jasmine
5. Karacadag

All images contain only a single grain of rice and are RGB images with a size of 250x250 pixels. The goal is to predict the type of rice to which a rice grain image belongs. The dataset can be found at the following link: https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset.

Figure 1.1: Sample images from the dataset



2. Data preprocessing

The first thing to do is loading every images into one single numpy array. When it is done, I reshape my array in order to get the following shape: *(number of samples, 250, 250, 3)* and I normalize everything by dividing it by 255 in order to get data between 0 and 1. I divide by 255 because it is the maximum pixel value for an RGB image. Unfortunately, due to the limitation of my computer I am not able to load every images into my array. I only use the first 1000 images of each type of rice for a total of 5000 images into my array.

In order to be able to analyze the final results, I apply a train - test split on my array with a test size of 30%. I tried to add some data augmentation by adding some rotation, flip and brightness to the images but unfortunately it hits the limit of my computer. The next step is the features extraction. I used two kinds of features extraction, the principal component analysis (PCA) and an embedding of pre trained convolutional neuronal network (InceptionResNet50) with t-SNE to reduce the dimensionality of the features (t-SNE is not the best because it can distort data unlike PCA but I already used PCA and I thought it was duplicative especially since in my case the use of one or the other did not change much). PCA will keep only the 50 most important features of each image and will reduce the dimensionality of the features. The InceptionResNet50 was trained with the imageNet dataset and is more complex than the PCA. It will learn deeper features which can potentially gives better results. Note that a lot of other pre trained model exist such as VGG16, Resnet50, VGG19 but I chose InceptionResNet50 because it has more layers, it can learn features on multiple scales and it is considered to have a better architecture. Finally, I use t-SNE after the InceptionResNet50 in order to reduce the dimensionality and in order to be able to plot the clusters because the output of InceptionResNet50 has a too high dimensionality.

Figure 2.1: Test ditribution of Kmeans when using PCA

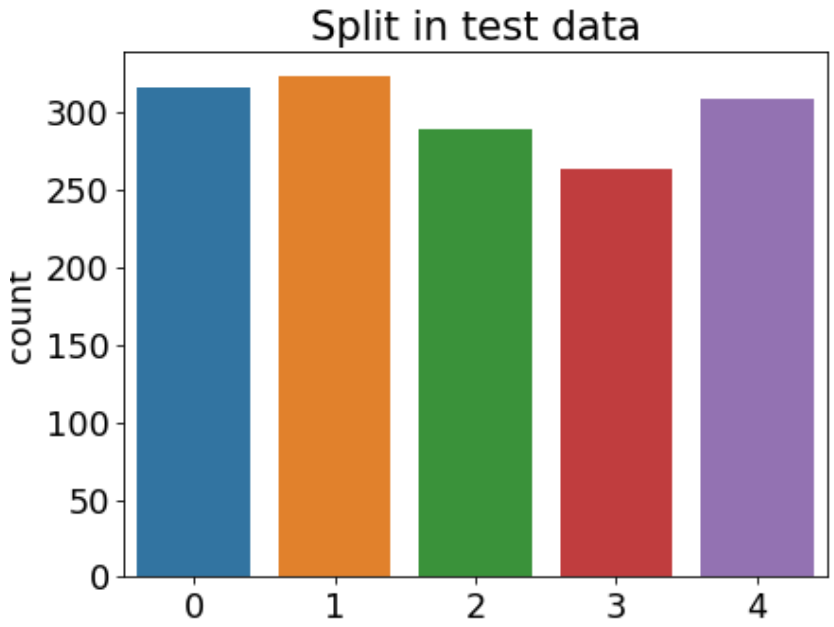


Figure 2.2: Train ditribution of Kmeans when using PCA

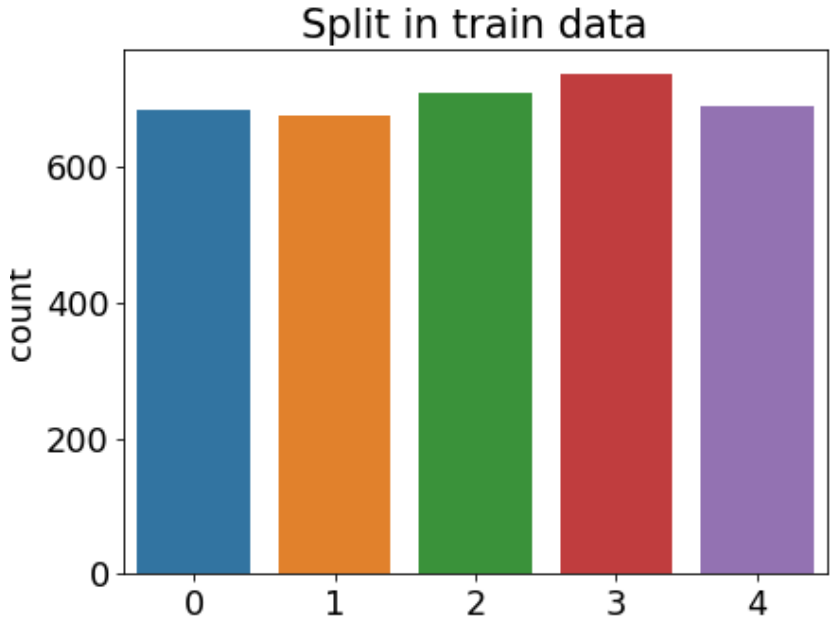


Figure 2.3: Test ditribution of Kmeans when using InceptionResNet50

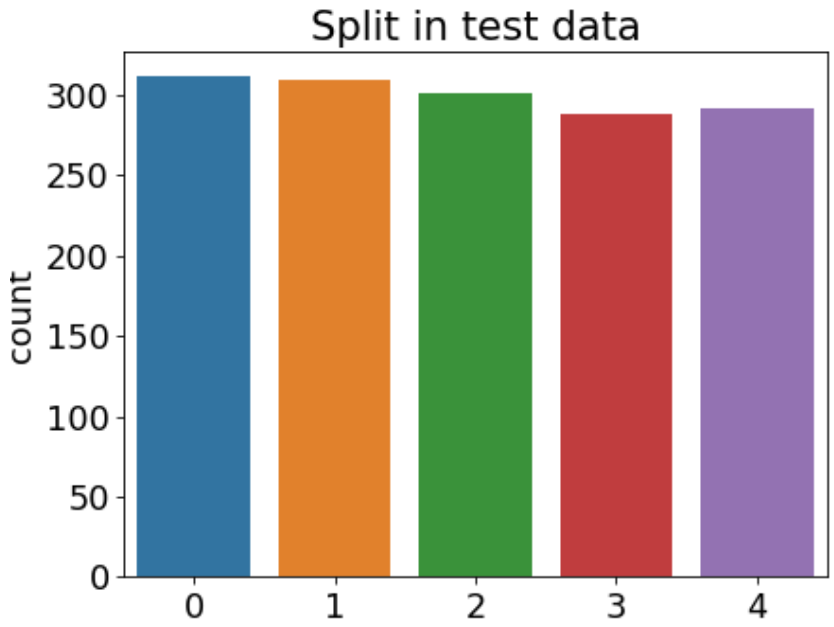
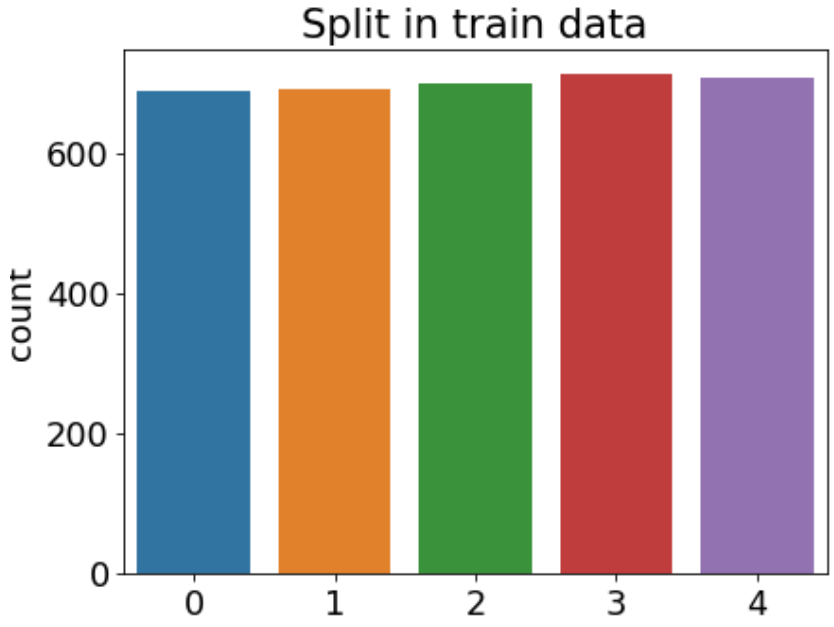


Figure 2.4: Train ditribution of Kmeans when using InceptionResNet50



3. Kmeans approach

3.1 Description of the approach

First approach I used is Kmeans, I tried to use GridSearch or RandomSearch but I did not find any documentation about using *"automatic"* hyperparameters tuning for unsupervised models like Kmeans. It seems not to be relevant because we are not suppose to know the score of the model so I tested multiple combinations by myself and finally used *kmeans++* for init parameter and *lloyd* for the algorithm because they are the more commons parameters values even if in my case I do not really see a difference. However, it should be relevant to use the *elbow* method in order to find the best n_clusters parameter, but, in my case it was useless as I already know the number of clusters I want. For the n_clusters parameter, I set it on five because I already know that I need five clusters for my five labels. After the prediction, I loop through every clusters in order to plot the result (here I plotted only the first 40 images of each cluster). After that, I print some metrics such as accuracy or silhouette and plot the centroids of the clusters.

3.2 Results

The accuracy when using the PCA features extraction is 21,9% and it is reflected into the confusion matrix at figure 3.3 and the silhouette score is 0,187. Unfortunately, according to the accuracy score, a lot of images are not predicted in the correct cluster and according to the silhouette score, the images are poorly matched from their own cluster and can match with neighbours clusters (silhouette score is between -1 and 1, if the score is around -1 it means that a sample in a cluster is very similar to a sample in the neighbour cluster). If we look at figure 3.1, we can see that cluster number 3 overlap cluster number 0, cluster number 1 and cluster number 4. Same observation for cluster number 2 which overlap clusters 0 and 4. When using InceptionResNet50, I got an accuracy of 20,5% (again, you can see the confusion matrix at figure 3.4 and the silhouette score is 0,3. Even if the accuracy is worst compare to PCA method, the silhouette score is better

which means that samples in a cluster are "more" different than those in a neighbouring cluster compared to PCA method. We can see the results at figure 3.2. The centroids on this figure are far away from the dots which means that the cluster is not good and not well defined (the centroids are the mean of all points from a cluster).

According to the prediction samples in the chapter 6, we can observe on figure 6.3 that the majority of the cluster is made of ipsala rice but we also have two images of jasmine rice so PCA did a great job on clustering ipsala grains. Same observation for figure 6.4, it seems that we only have ipsala grains again so, unfortunately, we have two clusters having the same kind of rice. The 3 last clusters seem to have random predictions. Now, if we look at some predicted samples from InceptionResnet50, we do not have clusters with a majority of one type of grain but it seems that grains are grouped by their shape. If we take a look at figure 6.8, we can see that the majority of grains has a vertical shape while in figure 6.9 it is an horizontal shape.

Figure 3.1: Centroids and clusters of Kmeans using PCA

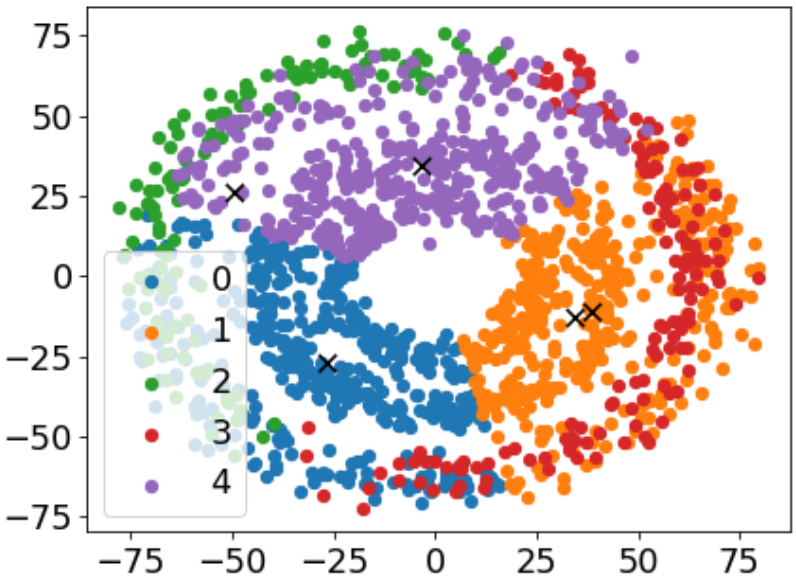


Figure 3.2: Centroids and clusters of Kmeans using InceptionResnet50

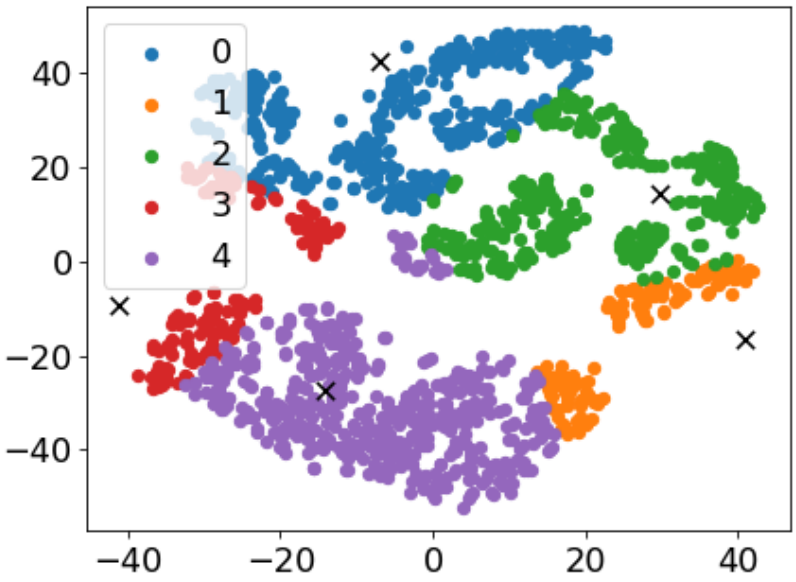


Figure 3.3: Confusion matrix of Kmeans with PCA

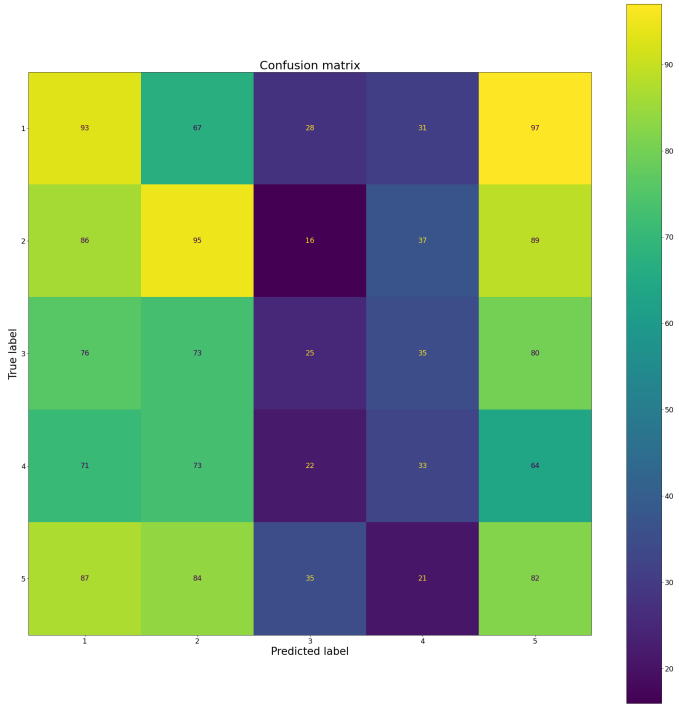
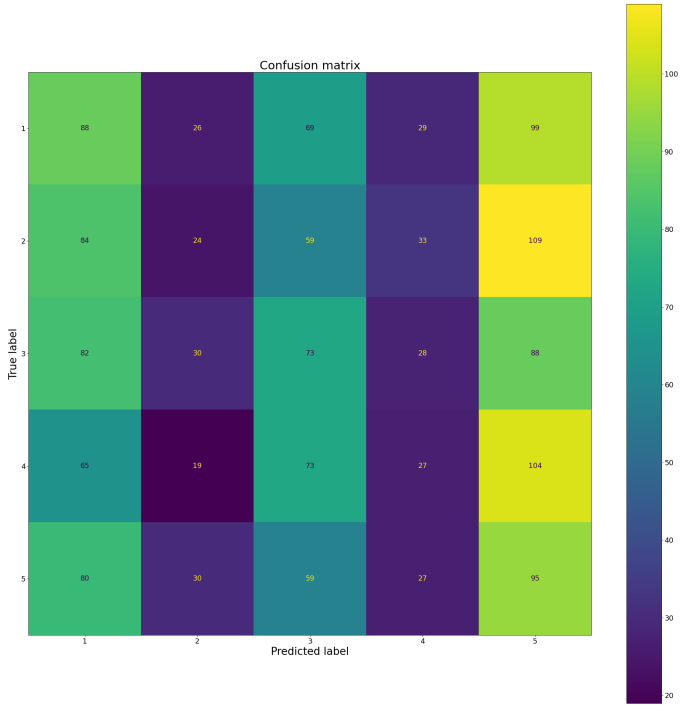


Figure 3.4: Confusion matrix of Kmeans with InceptionResnet50



4. DBSCAN approach

4.1 Description of the approach

The second approach I tried is DBSCAN or Density-Based Spatial Clustering of Applications with Noise. I also used PCA and InceptionResnet50 in order to predict my images with DBSCAN. Unfortunately I do not have any results with DBSCAN because all my predictions are equal to -1 and according to the DBSCAN documentation, -1 is labeled as a noise. I tried to edit the hyperparameters of DBSCAN such as *eps* and *min_samples* in order to get something else than -1 but it was not successful.

5. Conclusion

After analyzing the results I got with Kmeans, the accuracy is very poor whether with PCA or with InceptionResnet50 but it seems that PCA is clustering the images based on the size of the grains and maybe it is why it performs well with ipsala rice which is a big grain. InceptionResnet50 seems to use the shape and the orientation of the grains in order to create the clusters. For DBSCAN, the difference between the grains seems to be too small and the model can not perform. Obviously, my models do not beat the random chance because the accuracy is too small. Something that I could improve is using the same train/test split for PCA and InceptionResnet50, I agree that this might be more relevant when comparing the results of the two approaches. Using unsupervised models like Kmeans or DBSCAN is a bad choice because the images are too similar, they have the same color, almost the same shape and size. Further more "regular" machine learning algorithms are really sensitive to images orientation. Maybe it is for all those reasons that I only found solutions which used CNN, ANN or other kinds of supervised deep learning which perform with a 99% accuracy.

6. Appendix

6.1 Samples with PCA

Figure 6.1: Prediction samples from Cluster 1 of Kmeans with PCA

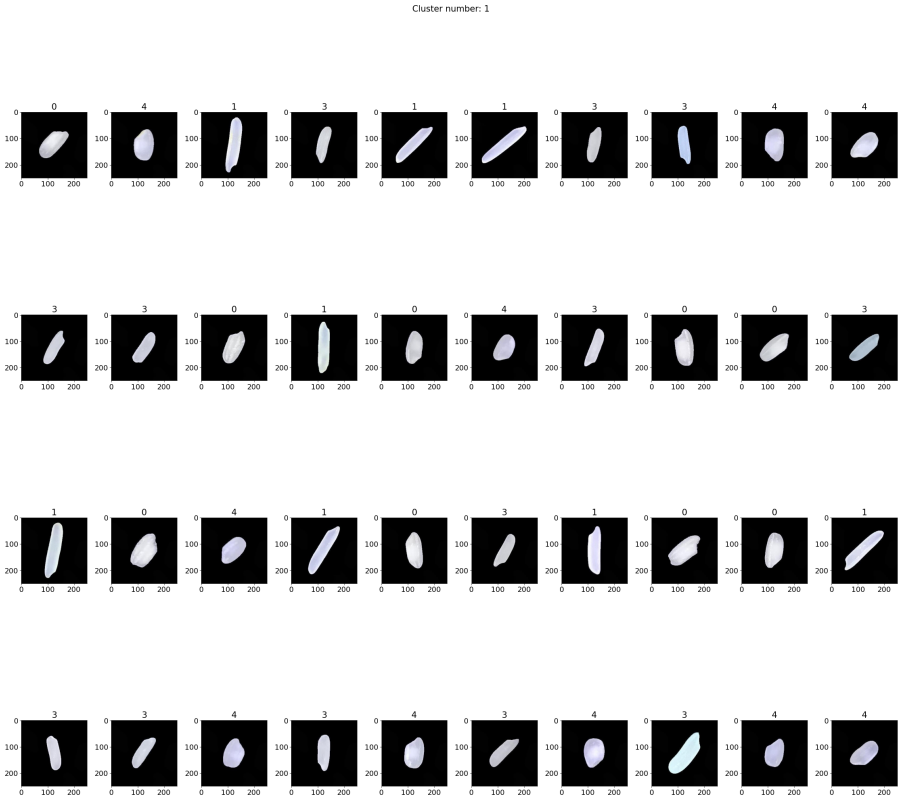


Figure 6.2: Prediction samples from Cluster 2 of Kmeans with PCA

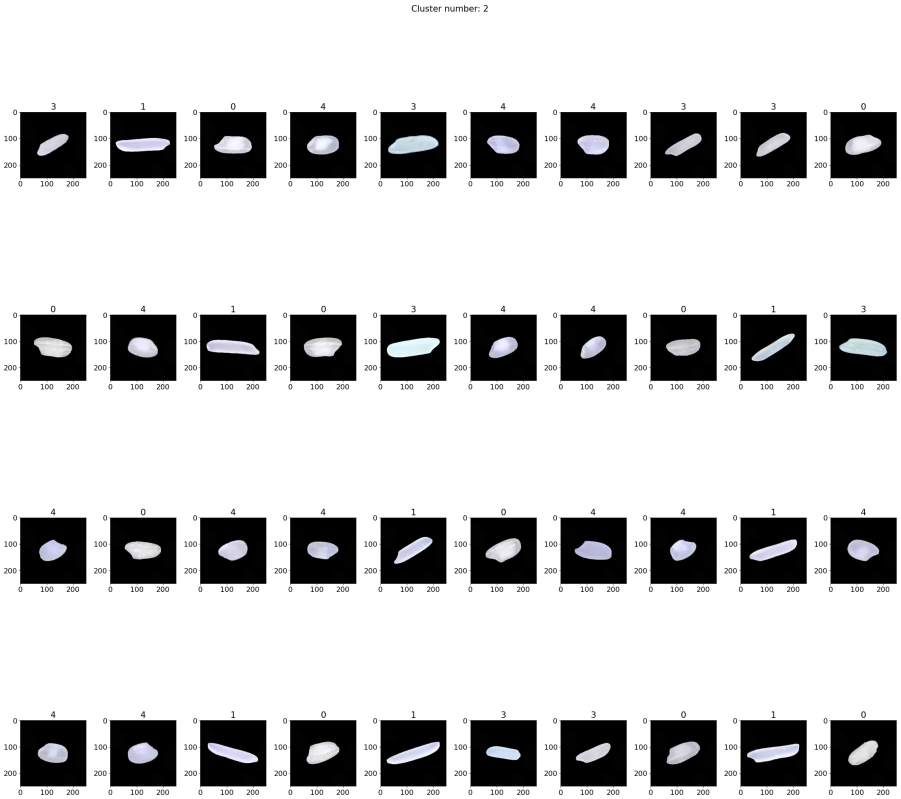


Figure 6.3: Prediction samples from Cluster 3 of Kmeans with PCA

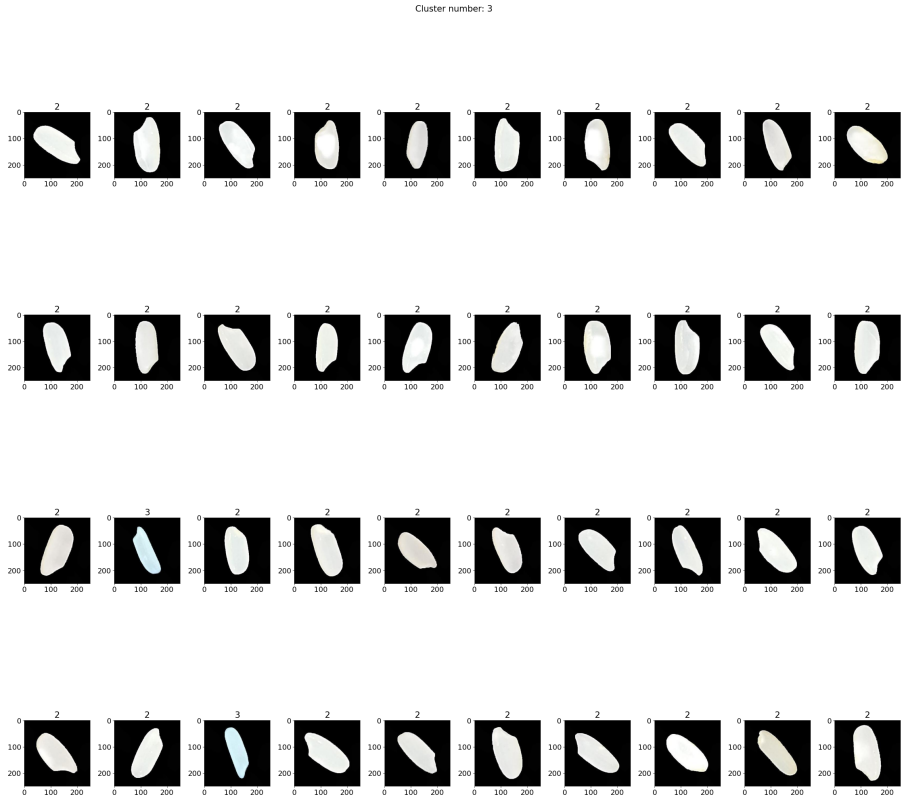


Figure 6.4: Prediction samples from Cluster 4 of Kmeans with PCA

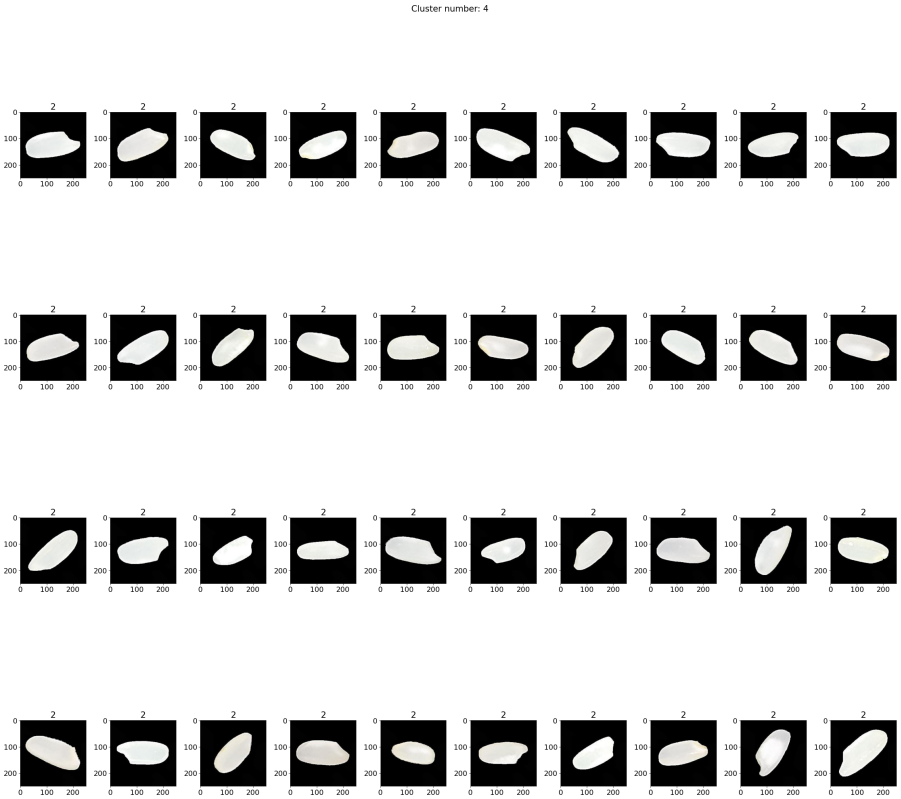
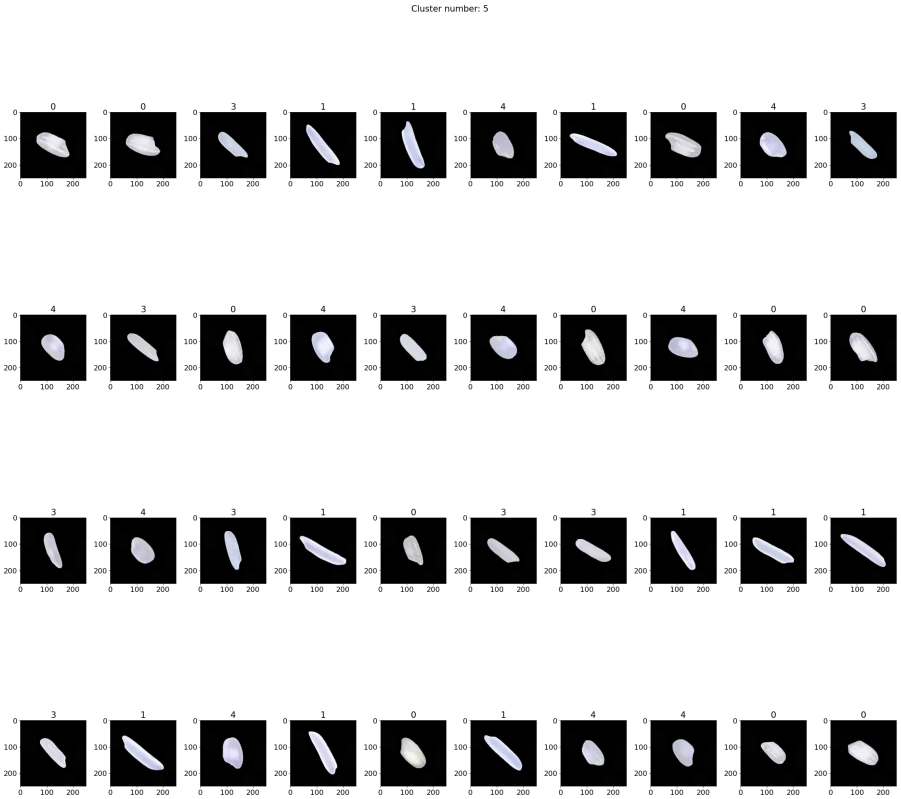


Figure 6.5: Prediction samples from Cluster 5 of Kmeans with PCA



6.2 Samples with InceptionResnet50

Figure 6.6: Prediction samples from Cluster 1 of Kmeans with InceptionResnet50

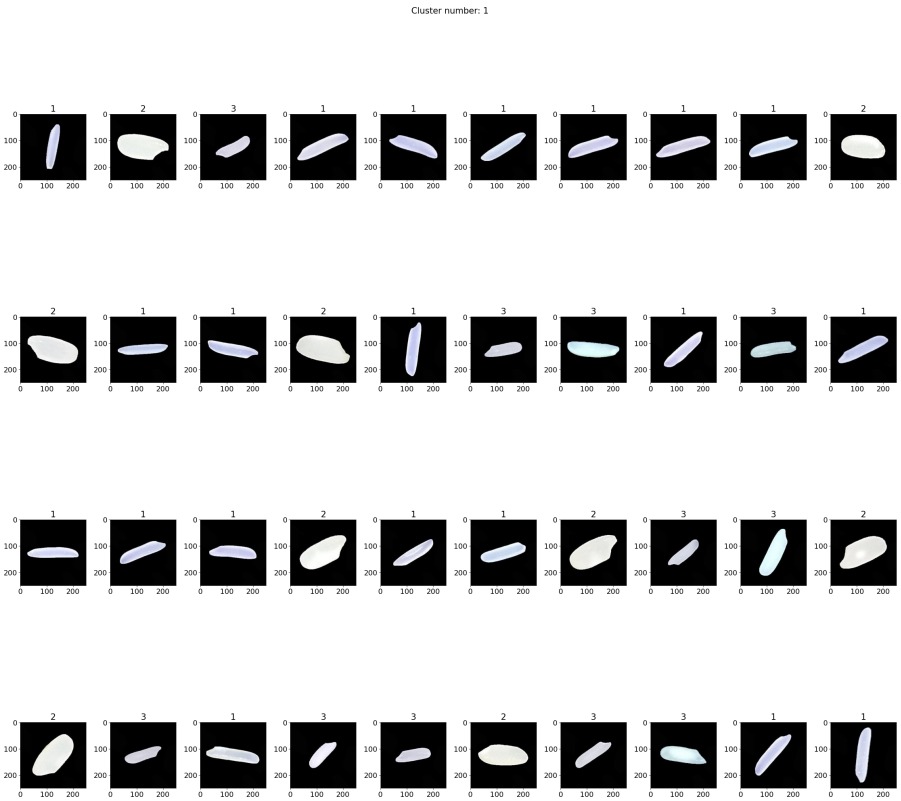


Figure 6.7: Prediction samples from Cluster 2 of Kmeans with InceptionResnet50

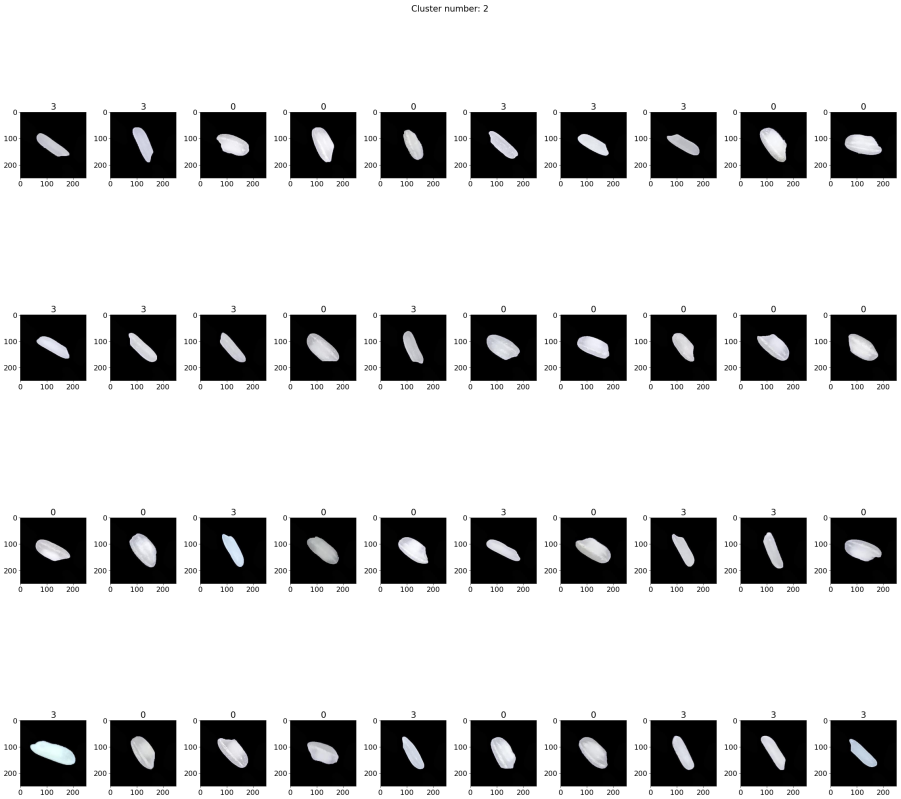


Figure 6.8: Prediction samples from Cluster 3 of Kmeans with InceptionResnet50

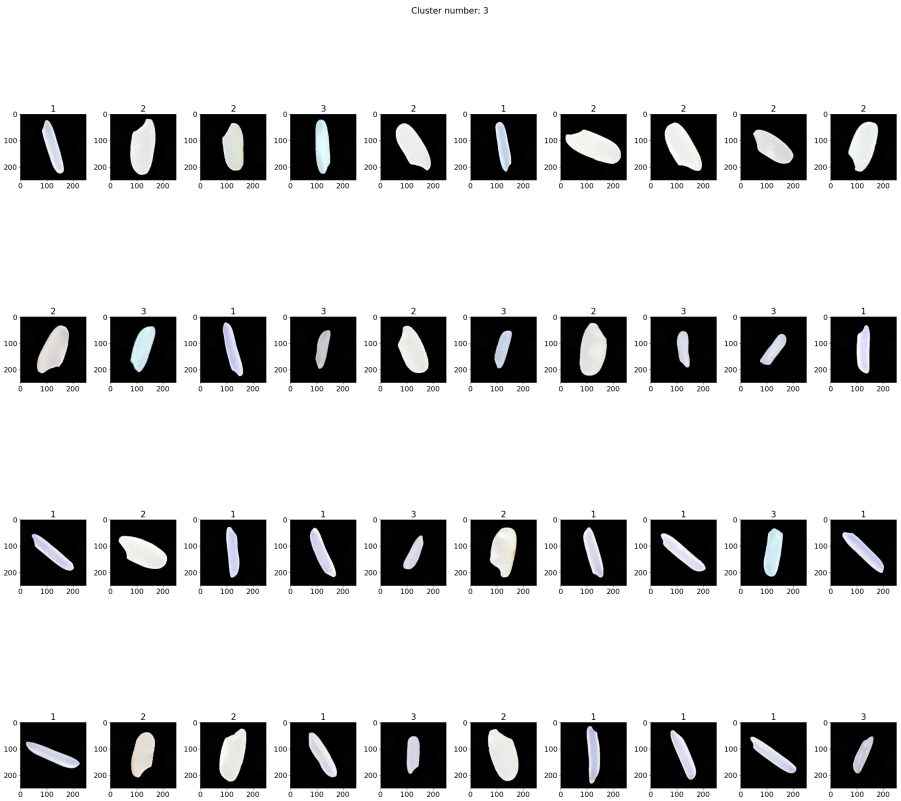


Figure 6.9: Prediction samples from Cluster 4 of Kmeans with InceptionResnet50

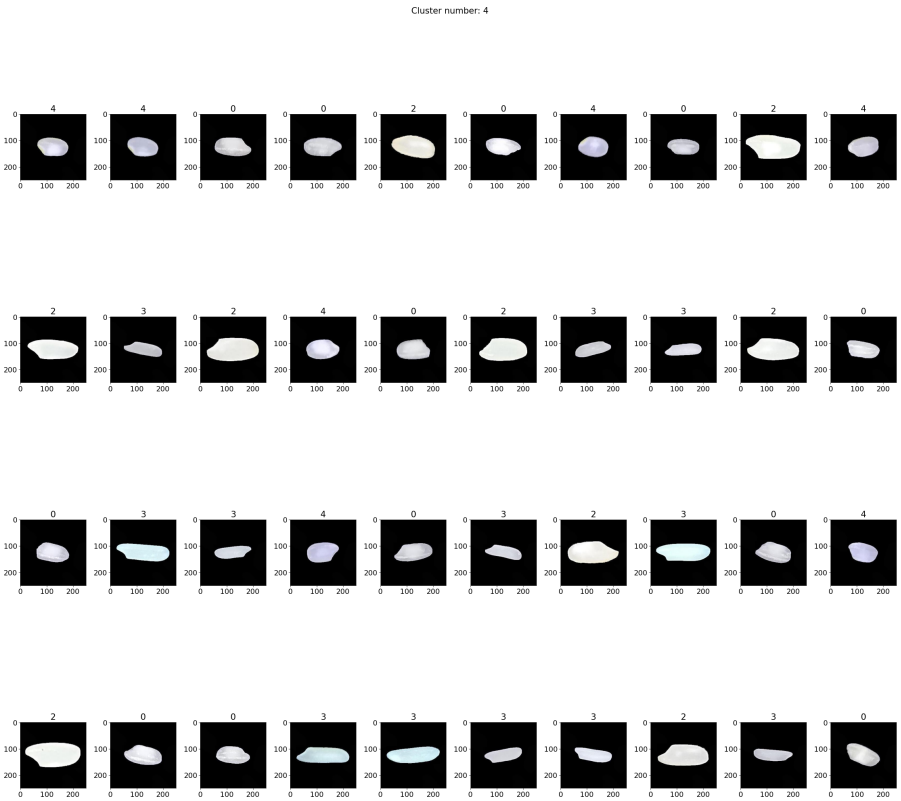
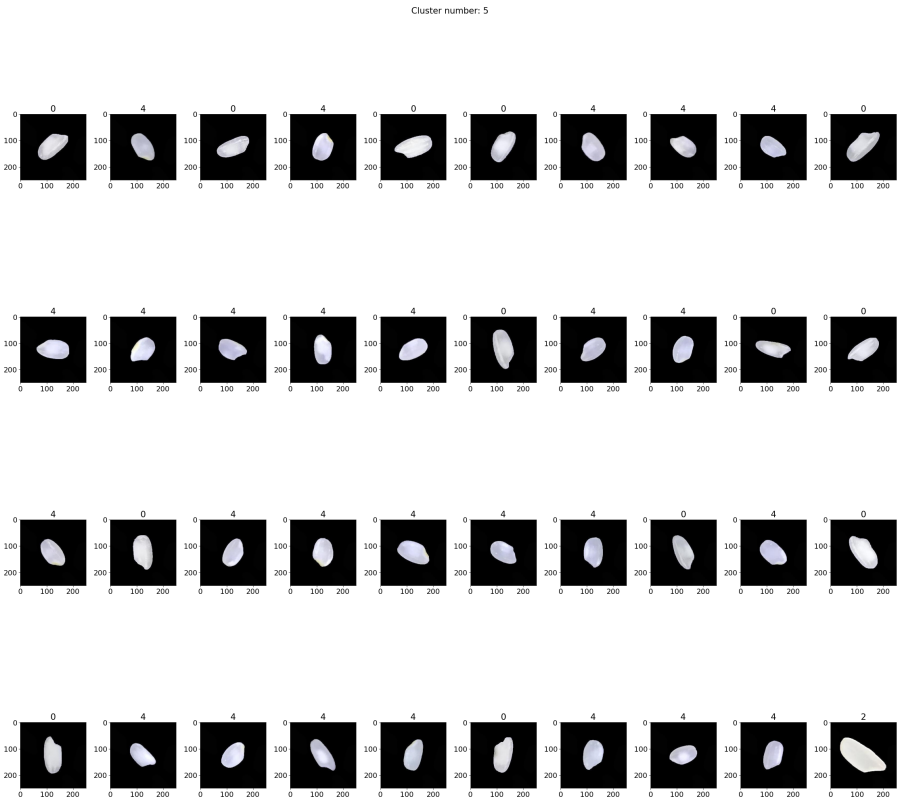


Figure 6.10: Prediction samples from Cluster 5 of Kmeans with InceptionResnet50



Bibliography

[1] *(99.9%) Rice Image Classification using CNN in TF*. en. URL: https://kaggle.com/code/karladriandeguzman/99-9-rice-image-classification-using-cnn-in-tf (visited on 01/14/2023).

[2] Gabe Flomo. *How to cluster images based on visual similarity*. en. Oct. 2020. URL: https://towardsdatascience.com/how-to-cluster-images-based-on-visual-similarity-cd6e7209fe34 (visited on 01/21/2023).

[3] S. Joel Franklin. *K-Means Clustering for Image Classification*. en. Jan. 2020. URL: https://medium.com/@joel_34096/k-means-clustering-for-image-classification-a648f28bdc47 (visited on 01/14/2023).

[4] franky. *Using Keras' Pre-trained Models for Feature Extraction in Image Clustering*. en. Aug. 2022. URL: https://franky07724-57962.medium.com/using-keras-pre-trained-models-for-feature-extraction-in-image-clustering-a142c6cdf5b1 (visited on 01/21/2023).

[5] *How to Plot K-Means Clusters with Python? - AskPython*. en-US. Section: Python Programming Examples. Oct. 2020. URL: https://www.askpython.com/python/examples/plot-k-means-clusters-python (visited on 01/21/2023).

[6] Sujeewa Kumaratunga PhD. *KMeans Hyper-parameters Explained with Examples*. en. May 2020. URL: https://towardsdatascience.com/kmeans-hyper-parameters-explained-with-examples-c93505820cd3 (visited on 01/21/2023).

[7] Kelvin Salton do Prado. *How DBSCAN works and why should we use it?* en. June 2019. URL: https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80 (visited on 01/22/2023).

[8] *Rice Classification — 99% Accuracy*. en. URL: https://kaggle.com/code/ayessa/rice-classification-99-accuracy (visited on 01/14/2023).

[9] *Rice Type Classification˜99% Accuracy + W&B*. en. URL: https://www.kaggle.com/code/padmanabhanporaiyar/rice-type-classification-99-accuracy-w-b (visited on 01/14/2023).

[10] *RiceDetection4*. en. URL: https://kaggle.com/code/temporary2427/ricedetection4 (visited on 01/14/2023).

[11] *sklearn.cluster.DBSCAN*. en. URL: https://scikit-learn/stable/modules/generated/sklearn.cluster.DBSCAN.html (visited on 01/21/2023).

[12] *sklearn.cluster.KMeans*. en. URL: https://scikit-learn/stable/modules/generated/sklearn.cluster.KMeans.html (visited on 01/16/2023).

[13] Erdogan Taskesen. *A step-by-step guide for clustering images*. en. Sept. 2022. URL: https://towardsdatascience.com/a-step-by-step-guide-for-clustering-images-4b45f9906128 (visited on 01/14/2023).