Second Project - Tackling Unsupervised Learning

Sasu Alexandru-Cristian

January 23, 2024

1 General Information

The unsupervised learning methods chosen for this project were the *K-means* and *DBSCAN* clustering algorithms. These methods were then trained and tested on the following dataset containing images of cats and dogs, developed by Microsoft and named Asirra: $https://huggingface.co/datasets/cats\_vs\_dogs$.

In the next sections it will be described details about the training and testing of these methods, as well as details about the visualisation and comparison of the results obtained.

The initial dataset contained approximately 25.000 images, each with label 0 or 1. For this task though, the total number of images used was reduced to 1600, so as to decrease the time required for handling the data and training the models. Moreover, the images initially had varying dimensions. In order for the algorithms to be able to process these images, each image was resized to a height and width of 200.

Label/Dataset	Train	Validation	Test
0	560	120	120
1	560	120	120

There will be used a lower bound and an upper bound in the comparison of each model's accuracy. The lower bound will be given by random chance, and the upper bound, by a supervised model.

2 Feature Extraction Method

The feature extraction method used consisted of the following: bringing each pixel of the image in the range $[0.0, 1.0]$ through *PyTorch's ToTensor* function, processing the image using the formula $output[channel] = (input[channel] - mean[channel])/std[channel]$ with the help of the *PyTorch* function *Normalize*, transforming the image from a 3D array into a 1D array of length $height \cdot width \cdot channels$.

3 K-means Results

In order to determine the right amount of clusters to be used for this task, each integer number from the interval $[2, 12]$ was used as the number of clusters utilized by the *K-means* model, for the calculation and plotting of the silhouette scores. After plotting these scores, the amount of 2 clusters was determined to be optimal, as the model with 2 clusters obtained the highest silhouette score:

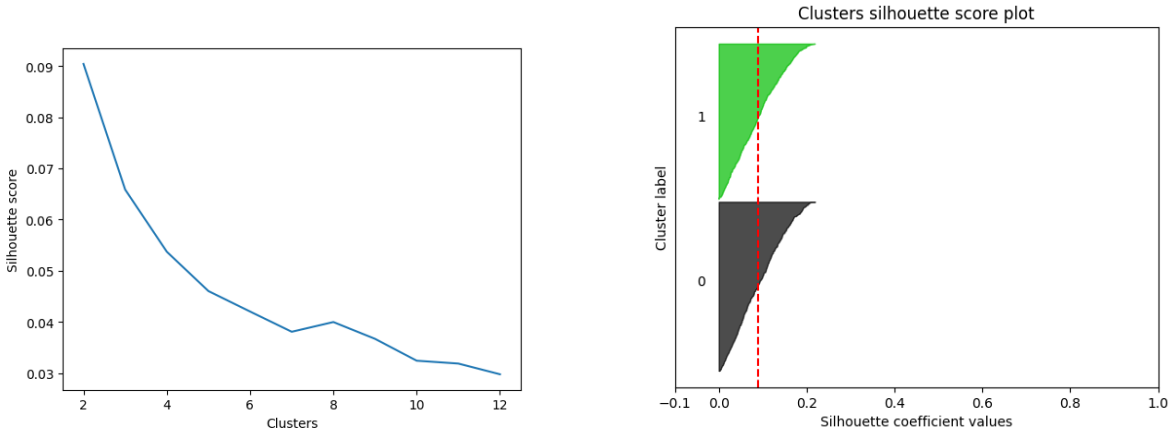


Figure 1: *K-means* cluster analysis

Based on the right plot, it can be further enforced that the number of clusters used is correct, as the silhouette score of each cluster is above the average silhouette score (the dotted red line).

Following the clustering process, in order to obtain the class labels, a 2×2 confusion matrix was built based on the true labels and the cluster labels. The values of the confusion matrix were then inversed (1 / $confusion\_matrix$), and based on the new values of the matrix, the Hungarian algorithm implemented in the function *linear_sum_assignment* from the *SciPy* library was applied in order to find an appropriate matching for the predicted class labels.

After setting the number of clusters to 2 and trying through a grid search-like approach a combination of initialization methods (k-means++, random) and algorithms for *K-means* (lloyd, elkan), the model obtained roughly the same accuracies for every combination. As such, it was considered that the tuning of parameters does not bring significant results.

Results on the test dataset:

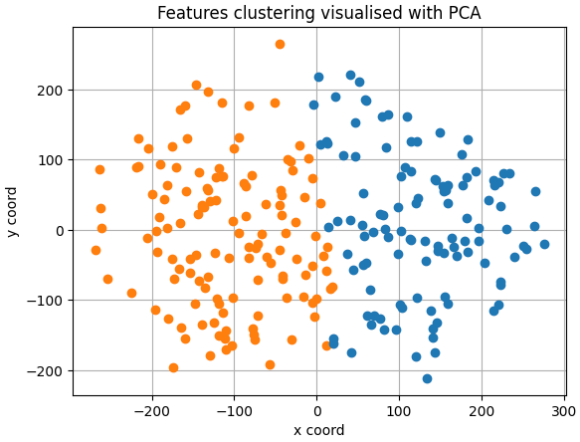
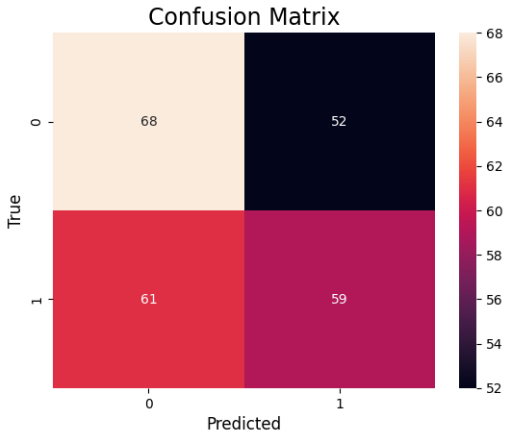


Figure 2: *K-means* clustering on the test dataset



	precision	recall	f1-score	support
0	0.53	0.57	0.55	120
1	0.53	0.49	0.51	120
accuracy	-	-	0.53	240
macro avg	0.53	0.53	0.53	240
weighted avg	0.53	0.53	0.53	240

Table 1: Confusion matrix and metrics computed on the results of *K-means* on the test dataset

Given that the data is split into two classes: class with label 0 resembling a cat, and class with label 1 resembling a dog; and there is an equal amount of data from each class found in every dataset, the random chance of predicting a correct label is 50% , as calculated through a *DummyClassifier* from the *scikit-learn* library. This percentage will be the lower bound used to measure the performance of the model.

In order to obtain the upper bound used to measure the performance of the model, the *Random Forest* algorithm obtained the best results out of all the tried supervised methods, with an accuracy on the test dataset of 63%.

Method	**Accuracy**
Random	0.50
K-means feature extraction	0.53
Random Forest	0.63

4 DBSCAN Results

As there is not an official implementation for a function that predicts cluster labels for a *DBSCAN* model, without also fitting the model, an attempt was made to create such a function. This attempt did not see success though, as there were technical difficulties encountered, and as such the results that are about to be shown were achieved on the train data.

DBSCAN is often unable to assign every data point to a cluster and thus generates outliers. Because of this, the features used for this task also resulted in the generation of outliers, and thus metrics couldn't be computed on all of the data points.

In order to determine an appropriate set of parameters for the model, a grid search-like approach was employed, similar to the one used for the *K-means*. As such, a combination of values for epsilon and the minimum number of samples was tested, where epsilon belonged to the set $\{0.1, 0.2, \ldots, 1.0\}$, and the minimum number of samples took values from the set $\{2, 3, \ldots, 10\}$. Also, the metric used for calculating distances between data points was the cosine metric, as this was considered to be one of the better suited metrics for working with images. After trying these parameters, the combination where $eps = 0.4$ and $min\_samples = 9$ yielded one of the highest silhouette scores, with 2 clusters, as well as one of the highest accuracies, and as a result these were the chosen parameters for working with *DBSCAN*.

Results on the train dataset:

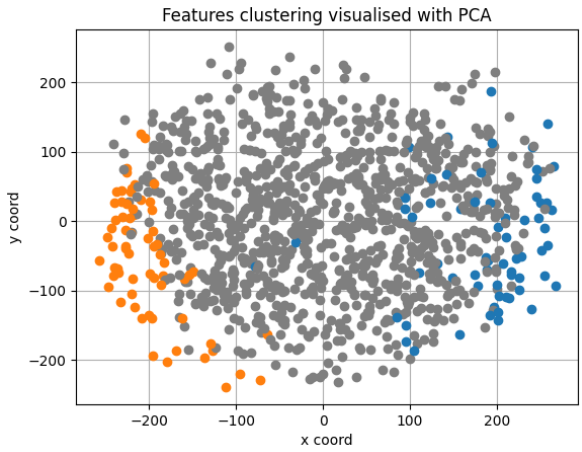
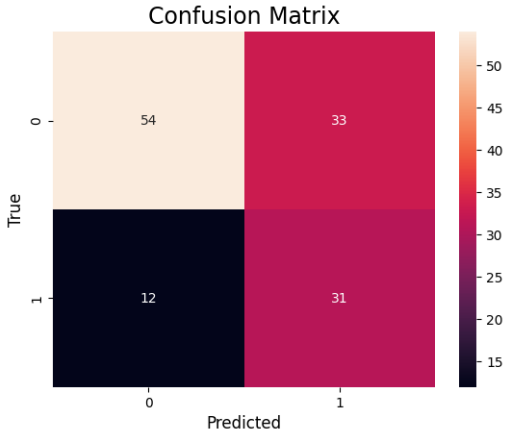


Figure 3: *DBSCAN* clustering on the train dataset



	precision	recall	f1-score	support
0	0.82	0.62	0.71	87
1	0.48	0.72	0.58	43
accuracy	-	-	0.65	130
macro avg	0.65	0.67	0.64	130
weighted avg	0.71	0.65	0.66	130

Table 2: Confusion matrix and metrics computed on the results of *DBSCAN* on a subset of the train dataset

The silhouette scores of the clusters were the following:

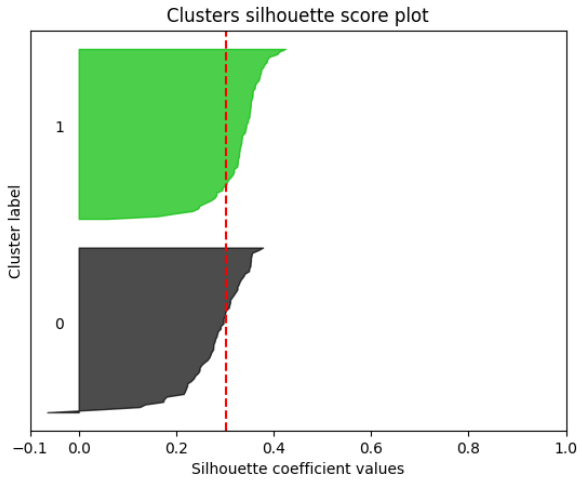


Figure 4: *DBSCAN* cluster analysis

Based on this plot, it can be theorized that the number of clusters used is correct, as the silhouette score of each cluster is above the average silhouette score (the dotted red line).

The lower bound used to measure the model's performance was again determined using the *DummyClassifier* from *scikit-learn*. Through its use, the random chance was determined to be 67%.

The upper bound used to measure the model's performance was again also determined through the *Random Forest* model, but given that the metrics could only be computed on the train data, this upper bound was of course a 100%.

Method	**Accuracy**
Random	0.67
DBSCAN feature extraction	0.65
Random Forest	1.0

5 Conclusions

Both the *K-means* and the *DBSCAN* models performed poorly on the given features. Other features were also tested, such as edge extraction, calculating the mean of each channel for every pixel, and using *PCA* to reduce the dimensionality of data, but each approach obtained results similar to the ones presented in this documentation. This may mean that the data is too difficult for the models to cluster, and further research about feature extraction methods and parameters fine tuning must be done, or the number of train samples must be increased.