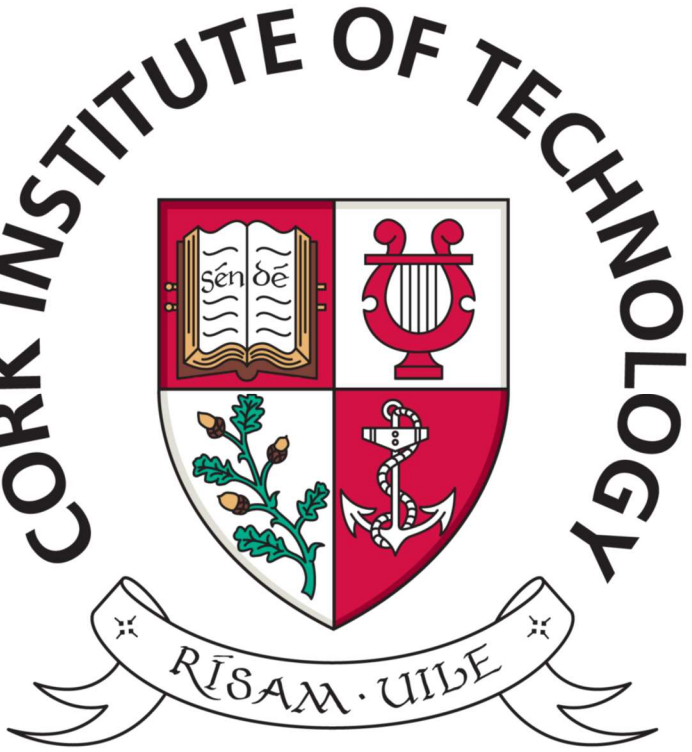


Image Classification using Machine Learning Techniques

For the module COMP9060 – Applied Machine Learning
Master of Science in Data Science and Analytics, Department of Mathematics, 21 May 2020

Name: Sumeet Shivgand
Student id: R00182850

Assignment 3

1. Introduction

In this assignment, we are going to implement image classification to classify images of Dogs and Cats which is consists of 1002 images. Also, we are going to apply some generalization and optimization techniques to improve the performance of the model.
The main objective of this assignment is to build a machine learning Cat/Dog image classifier. We are going to implement a system to classify whether images contain either a dog or a cat. This is easy for humans to detect dogs and cats, but it finds a bit more difficult for classifier. This is a binary classification problem.
The goal is to build a machine learning algorithm capable of detecting the correct animal (cat or dog) in new unseen images (test data).

2. Pre-processing Phase

In this phase, we are focusing on preparing the dataset by using techniques like normalization, resizing etc.

Normalization

It is used to rescale pixel values from the range of 0-255 to the range 0-1 preferred for models. Scaling data to the range of 0-1 is traditionally referred to as normalization. This can be achieved by setting the rescale argument to a ratio by which each pixel can be multiplied to achieve the desired range.

Resizing

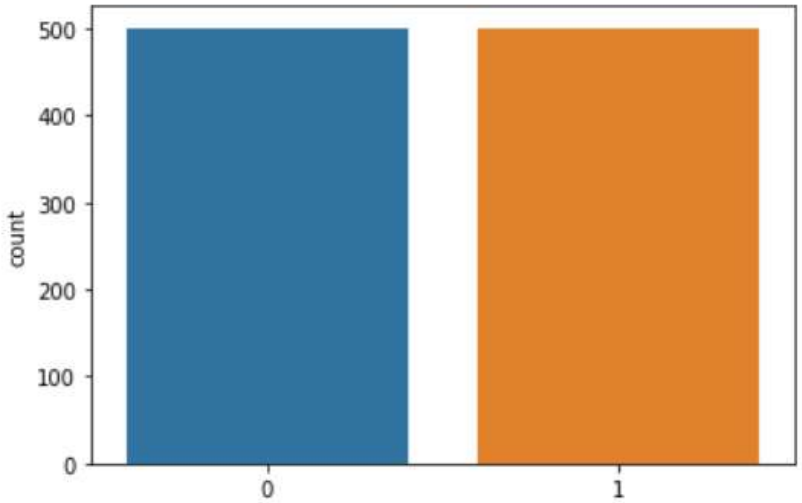
Resizing images to the same size without deforming patterns contained therein is a major challenge. In this, we are going to set all images of an equal size (350*350*2).

Firstly, we read the directory which contains images of dogs and cats. Secondly, we resize the images from directories into an equal size (350*350*3). Finally, the data has also been normalized.

```
100  image data retreived
200  image data retreived
300  image data retreived
400  image data retreived
500  image data retreived
600  image data retreived
700  image data retreived
800  image data retreived
900  image data retreived
1000  image data retreived
(1002, 12288)
(1002,)
```

Here, we use counter to get update and make sure that all images are retrieved. Later, we use ‘data.shape’ to know the rows and columns of the data. So, there are total ‘1002’ rows and ‘12288’ columns.

```
1    501
0    501
dtype: int64
```

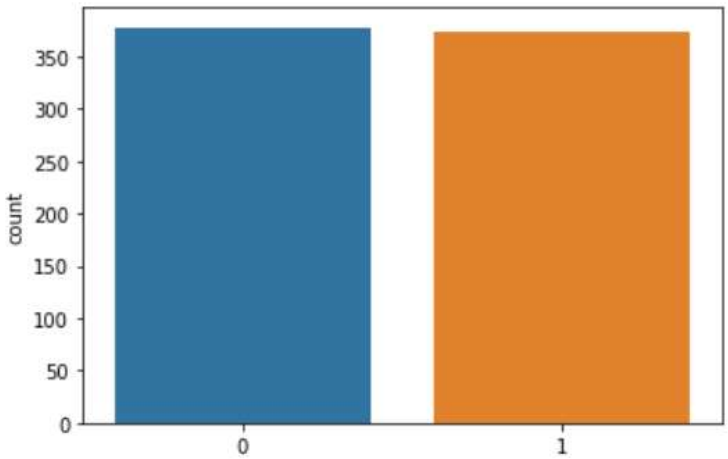


From above plot, we can see that there are equal number of cat and dog images i.e. '501' each.

3. Training Phase

Here, we are going to split the data into train and test in the ratio of 75:25. Below is the graphical representation of train and split data.

```
0    378
1    373
dtype: int64
```



Now, we are going to train different machine learning algorithms such as K-Nearest Neighbor (KNN), Support Vector Machine (SVM) and Decision tree using the pre-processed dataset comprised of the images and their corresponding labels split in train.

K-Nearest Neighbor (KNN):

```
Accuracy Score:
 0.545816733067729
              precision    recall  f1-score   support

           0       0.52      0.85      0.65       123
           1       0.64      0.25      0.36       128

    accuracy                           0.55       251
   macro avg       0.58      0.55      0.50       251
weighted avg       0.58      0.55      0.50       251
```

	Predicted 0	Predicted 1
Actual 0	105	18
Actual 1	96	32

Here, we train KNN classifier and get the accuracy 54.58 %.
From classification report, we can see the precision 52% and recall 85%.
From the above confusion matrix, we see that model misclassify 96 as 'cat' instead of 'dog' whereas 18 misclassify as 'dog' instead of 'cat'. There are two types of error i.e. Type I and Type II. Type I error generated by the model is 18 and Type II error generated by the model is 96.

Support Vector Machine (SVM):

```
Accuracy Score:
 0.6254980079681275
              precision    recall  f1-score   support

           0       0.63      0.59      0.61       123
           1       0.62      0.66      0.64       128

    accuracy                           0.63       251
   macro avg       0.63      0.62      0.62       251
weighted avg       0.63      0.63      0.62       251
```

	Predicted 0	Predicted 1
Actual 0	72	51
Actual 1	43	85

Here, we train SVM classifier and get the accuracy 62.54 %.
From classification report, we can see the precision 63% and recall 59%.
From the above confusion matrix, we see that model misclassify 43 as 'cat' instead of 'dog' whereas 51 misclassify as 'dog' instead of 'cat'. There are two types of error i.e. Type I and Type II. Type I error generated by the model is 51 and Type II error generated by the model is 43.

Decision Tree:

```
Accuracy Score:
 0.5258964143426295
              precision    recall  f1-score   support

           0       0.51      0.58      0.54       123
           1       0.54      0.48      0.51       128

    accuracy                           0.53       251
   macro avg       0.53      0.53      0.53       251
weighted avg       0.53      0.53      0.52       251
```

	Predicted 0	Predicted 1
Actual 0	71	52
Actual 1	67	61

Here, we train Decision tree classifier and get the accuracy 52.58 %.
From classification report, we can see the precision 51% and recall 58%.
From the above confusion matrix, we see that model misclassify 67 as 'cat' instead of 'dog' whereas 52 misclassify as 'dog' instead of 'cat'. There are two types of error i.e. Type I and Type II. Type I error generated by the model is 52 and Type II error generated by the model is 67.

From above three model, we get the highest accuracy for SVM classifier. So, we are going to select SVM as our best model and then perform the hyper-parameter tuning on it.

4. Optimization Phase

From above classifier, we select Support Vector Machine (SVM) as our best model. Now, we are going to perform hyper-parameter tuning on SVM using GridSearchCV. Below is the GridSearchCV approach for SVM.

```
GridSearchCV(cv=None, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'gamma': [1.0, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
Accuracy Score:
 0.6135458167330677
              precision    recall  f1-score   support

           0       0.62      0.56      0.59       123
           1       0.61      0.66      0.64       128

    accuracy                           0.61       251
   macro avg       0.61      0.61      0.61       251
weighted avg       0.61      0.61      0.61       251
```

	Predicted 0	Predicted 1
Actual 0	69	54
Actual 1	43	85

Here, we tune SVM classifier and get the accuracy 61.35 %.
From classification report, we can see the precision 62% and recall 56%.
From the above confusion matrix, we see that model misclassify 43 as 'cat' instead of 'dog' whereas 54 misclassify as 'dog' instead of 'cat'. There are two types of error i.e. Type I and Type II. Type I error generated by the model is 54 and Type II error generated by the model is 43.

5. Prediction on Test Data

```
20  image data retreived
40  image data retreived
60  image data retreived
80  image data retreived
100  image data retreived
(100, 12288)
```

Here, we had performed the same process as performed on train data and then the 'test_data1.shape' to know the number of rows and columns of the data. So, there are total '100' rows and '12288' columns.

After applying the model on test data then we get the following output.

```
[0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0
 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 1 0
 0 0 0 0 0 0 1 0 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0]
```