A logo for college computing

Description automatically generated

|  |  |
| --- | --- |
| *Student Name:* ***Giorgio Zoppi*** |  |
| *Student Number:* ***sba2301*** |  |
| *Module Name:* ***Programming for AI (HDip in Artificial Intelligence - Feb 2024 - HCI cohort)*** |  |
| *Assignment Title:* ***CA 2*** |  |
| *Assessment Due Date: 21****t May 2024*** |  |
| *Date of Submission 21****th May 2024*** |  |

**Declaration**

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

## Task 1

## Introduction

In the context of this assignment, we’ve created a classifier about type of glasses and materials. As any data environment, the first step in this process has been what it is called descriptive statistics and data exploration to understand the nature of the dataset. The results are alleged in a Jupiter Notebook, and published at <https://github.com/giorgiozoppi/ProgrammingAICA2> for cross reference, also the system that used for training is a laptop with AMD Ryzen and GeForce RTX with two thousand CUDA core.

## Exploratory Data Analysis.

We’ve computed:

* Medium, variance, mode, median of the dataset to understand how the values behaves and their ranges and if anomalies are present.
* The types of the object in the data frame loaded from the csv to understand whether there are numerical or categorical data. No categorical data has been found so there is no need of techniques like one shot encoding.
* We’ve plotted each feature distribution as you can see in figure 1 to understand how much the dataset was balanced or not.

Immagine che contiene edificio, grattacielo, diagramma, testo

Descrizione generata automaticamente

Figure Feature Distribution

We have noticed that the target variable is not balanced, and we have different types of glasses but:

* Glass type 4 is missing in the data set.
* Glass type 1 and 2 have a huge number of items whereas Glass type 6 is not so common.

Immagine che contiene testo, schermata, diagramma, Diagramma

Descrizione generata automaticamente

Figure Unbalanced dataset.

This makes the data set unbalanced. The most common technique to balance a data set is using data augmentation. The most common algorithm using data augmentation to balance a dataset is called **SMOTE** or the Synthetic Minority Over-sampling Technique.

SMOTE is a data augmentation technique that increases the size of the training set by generating many realistic variants of each training instance. In case of an unbalanced dataset, it is used to generate more samples of the less frequent classes.

As part of the data cleaning process before using SMOTE we want to remove any outliers. An outlier is a data point that significantly deviates from the rest of the data. It can be either much higher or much lower than the other data points, and its presence can have a significant impact on the results of machine learning algorithms. They can be caused by measurement or execution errors. The analysis of outlier data is referred to as outlier analysis or outlier mining. To remove the outliers, we’ve used an algorithm called Isolation Forests.

Immagine che contiene schermata

Descrizione generata automaticamente

Figure Outlier Detection

The algorithm builds a random forest in which each decision tree is grown randomly: at each node, it pickes a feature randomly, then it picks a random threshold value to split the dataset in two. This process is repeated until all instances are isolated. Anomalies are usually far from other instances,

so on average they tend to get isolated in fewer steps then normal instances. Once we've identified the outliners we select and drop them. It's not the best approach since we risk underfit doing but the number is not meaningful in our case, we can drop. Alternative strategy could be replaced with the mean.

At this point we’ve applied the SMOTE algorithm and computed the correlation heatmap before and after the data augmentation to see the impact. We’ve noticed the with balanced data the impact is minimal.

Immagine che contiene testo, schermata, quadrato

Descrizione generata automaticamente

Now the data is clean we wanted to be sure that the range of the data was correct, so no data of different ranges. We split the training set and test set. We noticed that we’ve a problem of scaling, the data is not well centred around zero, so a standard scaling is required. This is important in neural networks especially for consistent initialization. Neural networks rely on random weight initialization, if we don’t scale the initial weights might be in the correct range. With the scaling this task is of data cleaning is terminated.

## Dense Neural Network Classifier.

Now that the data looks good, we had to create and implement a scalable Dense Neural Network that will output a classification based on the 'Type of glass' feature (class attribute).

After the creation we tested this model and aim to enhance its performance by experimenting with various configurations of neurons, layers, loss functions, and activation functions.

As base neural network we’ve decided to use is a Multi-Layer Perception classifier and we’ve used the Kera framework. Kera is a high level deep learning network framework to develop neural networks.  
Immagine che contiene diagramma, schermata, linea

Descrizione generata automaticamente

We know that the activation function of hidden layers function will be ReLu (rectified linear unit function), which has the advantage to be fast to be computed and has no maximum output value so we can use a hyperparameters any number of neurons to achieve the best accuracy. Other popular activation function we’ve tried but with no success in term of increase of accuracy are tanh and sigmod.

We’ve performed hypermeter tuning for detecting the right number of layers and neurons, using both GridSearchCV and keras-tuner package. GridSearchCV used an early stopping mechanism, keras-tuner is faster so there was no need. We can think to early stopping as a termination function that allows to terminate you hyperparameters search if it is not reach the goal after a given amount of time or condition is met. Part of this condition is the function objective; in our case we’ve preferred loss to accuracy.

The reason of this choice is that loss function quantifies how certain the model is about a prediction (basically having a value close to 1 in the right class and close to 0 in the other classes). The accuracy merely account for the number of correct predictions.

After creating a model, we had to compile it, during the compilation we had to choose the loss function and the optimizer:

* *Loss function*: we’ve chosen **sparse\_categorical\_crossentropy** because we’ve sparse labels (i.e. for each instance, there is just a target class) and the classes are exclusive.
* *Optimizer*. Here we had two options:
  + Adam (Adaptive Moment Estimation) that is a optimization technique for training neural networks. on an average, it is the best optimizer. It works with momentums of first and second order. The intuition behind the Adam is that we don’t want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search.
  + SGD, also called Gradient descent (with momentum) optimizer. The training will be performed using stochastic gradient descent, so Kera will perform the backpropagation algorithm. To use SDG properly, we had to tune the learning rate. It is important for SDG because when it is too high, it will converge quickly to the optimum of the loss function, but it never stabilizes itself. When it is too low, a learning rate that is too low will result in very slow convergence, making the training process inefficient and time-consuming.

Once we’ve compiled the model, we fit with training data, leaving 20% as validation data. Once we’ve trained and captured his history in Figure 4.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

Figure Learning curves: the mean training loss and accuracy over the epochs

The above figure is important to understand, you can see that both the training accuracy and the validation accuracy increase during training, starting from 0.2 to 0.8 while the training loss decrease. It means that more we train more we get accuracy up to a point, choosing the number of epochs to 50 is good. After the around thirty epochs the validation curves begin to see close enough so there is not so much overfitting.

We’re not satisfied because at the beginning the curves are instable and too low, also the maximum is 80%, we can get more. So, we tried the same process with keras-tuner with a new model creation function. This time we didn’t fix the optimization function to sgd but let keras tuner to discover it.

We select a RandomSearch tuner and as result of the tuning, we’ve reached over 90% of accuracy. Success!

The neural network became:

* Input Layer: 256 neurons, relu activation function.
* Middle Layer: 256 neurons, relu activation function
* Output Layer: 9 neurons, softmax.

Now the curves are better: loss is always monotonically decreasing, and accuracy is monotonically increasing. The change of the optimization function in compiling and the number of neurons helped a lot.

Immagine che contiene testo, linea, Diagramma, schermata

Descrizione generata automaticamente

# Conclusion.

We’ve built a classifier that it can predict with accuracy over 90 per cent the type of glass using our dataset and performed all the three tasks mentioned in the assignment.