TÉCNICO LISBOA

SmartLabDetection - Neural Network

Daniel Lopes

Department of Electrical Engineering,
Computer Engineering & Informatics

July 5, 2022

1 Objective

During the worse times of the pandemic, Técnico imposed a limit of 2 persons inside a certain lab. However, the students that use the lab had frequent deadlines, and often ignored the 2-person limit. Having this lab configured with many sensors, a good amount of data was collected over a period of time. The objectives of this project are to develop NN-based classifiers that, using this data, are able to:

- Detect when there are more than 2 persons inside the lab;
- Detect how many people are inside the lab.

For this purpose, **two** classifiers were created in regards to each of these objectives.

2 Data Preparation

Data preparation is a fundamental step to be taken while creating models that generalize a given function. Moreover, in the real world, *datasets* are not clean, contain missing values, contain outliers and in many cases are extremely unbalanced.

For this purpose, a set of data preparation techniques were applied that created multiple versions of the *dataset* for evaluating which of these techniques made a difference (increased significantly) to the results.

2.1 Missing Values

Given the nature and source of the data, it can be said with confidence that it didn't seem necessary to apply missing value imputation, and so a simple drop of all the rows that contained at least one missing value was probably enough.

```
data.dropna()
```

2.2 Outliers

Having in mind the occurrence of outliers in the data, there are many techniques that can handle this in order to obtain better results. For the purpose of this project, there was an advise by the teacher to not remove more than 10 outliers as the data itself probably contained only 4 real outliers, which could be easily seen by look at the plot of the features. Figure 1 shows the data before removing the outliers, and Figure 2 after removing the outliers.

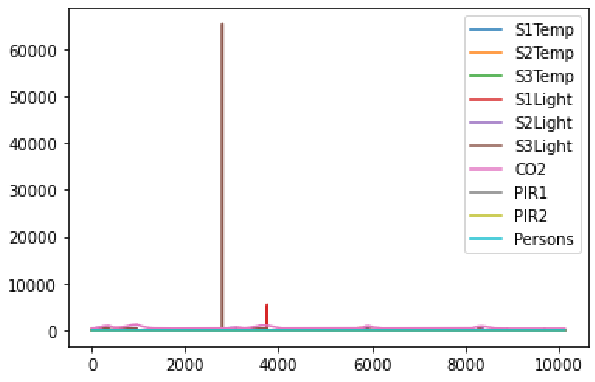

Figure 1: Dataset Plot With Outliers

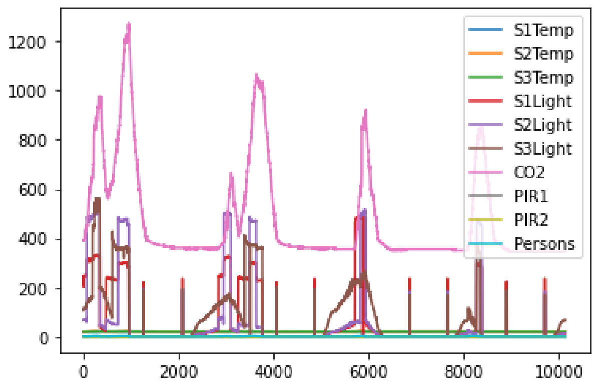

Figure 2: Dataset Plot after removal of Outliers

It's clear that with the removal of outliers, the *dataset* plot became stable and within a stable range. For the purpose of the removal of outliers, they could've been removed manually but to avoid manual operations and human error, the standard deviation technique was applied to all elements that differed $k * \sigma$ from the average (with k = 6), which was able to detect and remove 4 outliers.

There was also the option of filling these outliers with previous or posterior values but to avoid possibly filling wrong values or messing with the data, they were removed. For conscience relief, both strategies were ran against a baseline model and showed no difference whatsoever.

2.3 Date and Time

Usually when dealing with data, there's always the question of whether to include date and time. From a first perspective, it seemed obvious that the **time** could benefit the training of models because it is possible to interpolate that for a given hour, the likelihood of people being inside the lab is higher/lower (e.g. at night, it is less likely that more than 1 person will be inside the lab). From another perspective, the same could be said to dates, but only if the data contained representations for a whole year, for example, where one could interpolate seasonality and school holidays. That is not the case.

With this in mind, the Date and Time features were converted into more specialized numeric features that represent a value for days, hours, minutes and seconds.

Regardless of what the intuition tells us about which of these are relevant, it can be evaluated in a more objective way, by looking at the correlation matrix.

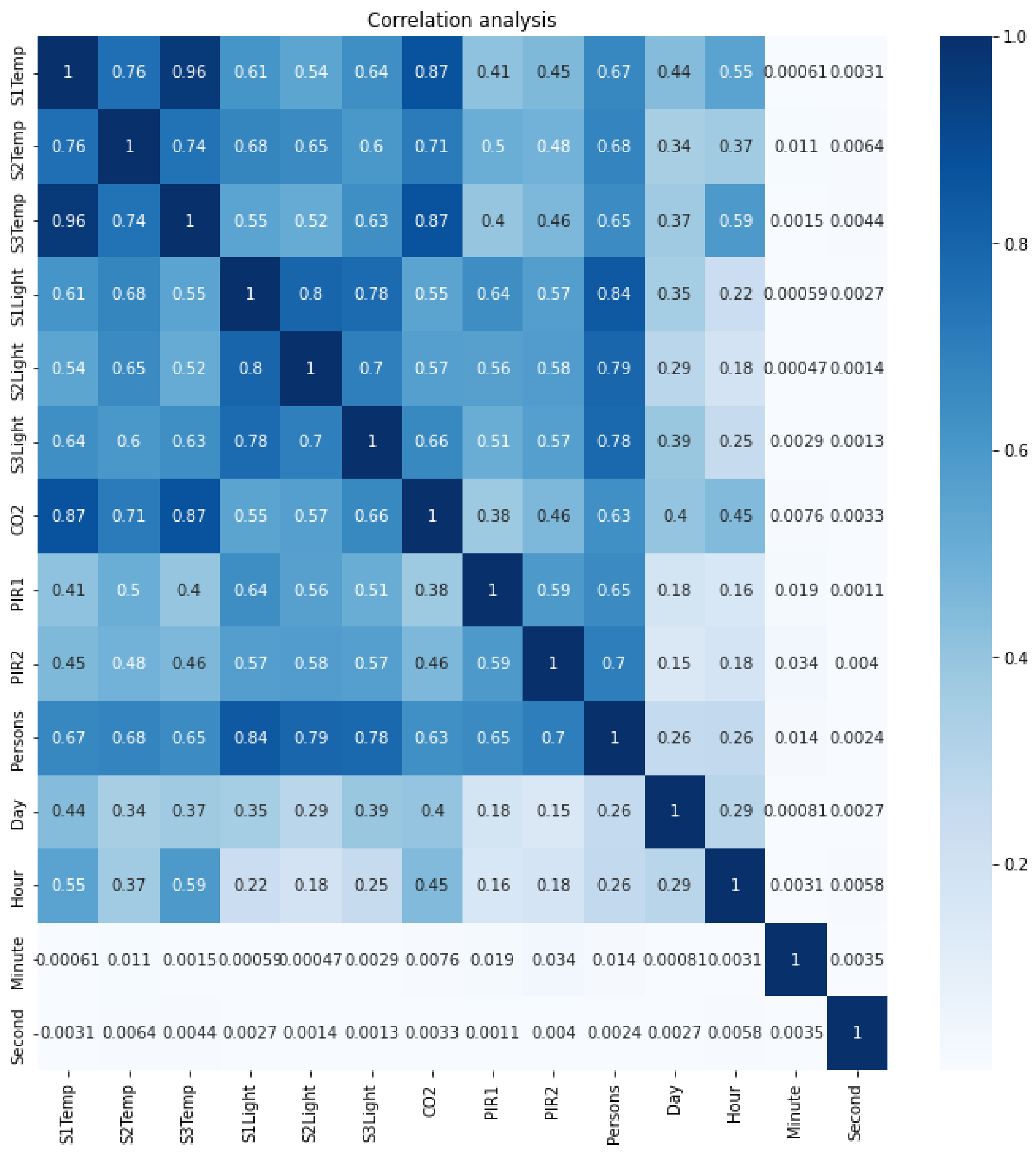

Figure 3: Correlation Matrix

With the matrix in Figure 3 it is clear that the features **minute** and **second** are not correlated with the Class (**Persons**), which essentially is due to the fact that the minutes and seconds are recurrent and occur equally in a given hour without much relevance.

On the other hand, **hour** and **day** have the same correlation but, with a more intuitive analysis, it might be obvious that **days** are not very useful because new data will most likely be in different subsequent days.

Finally, it leaves us with **hours** which might be a useful feature that gives us time awareness for occurrences in a certain hour of the day.

With this, a baseline *dataset* was created with and without the **hour** feature.

3 Experimental Setup

On the previous steps, the *dataset* was prepared with techniques that extend well to any classifier that we could envision. Unfortunately, that's not enough.

Data is usually unscaled, unbalanced and a nightmare for models like MLP to work with when using the default data values.

For this reason, both classifiers had a specific data prep step where the necessary modifications to the target column were done and scaling/balancing was applied.

Of course that the setup comprised of creating many versions of the dataset and running them against a baseline MLP model, to infer which of these techniques of data preparation are useful and yield better results.

4 Classifier A

The first classifier that was trained was the classifier A, which in sum is a model that predicts whether there are more than 2 persons inside the lab for a given input.

The first thing that was done for the preparation of the dataset was generating a new boolean **Target Column**, i.e. since the objective was to obtain a model that decides on whether there are more than two persons, it is clear that the target column should be: True or False and therefore a transformation like this was applied:

y = y.apply(lambda x: x > 2)

After having a new target column the data was split into train, validation and test sets with the split percentage being:

train size = 0.7

test size = 0.1

validation size = 0.20

4.1 Scaling

As mentioned earlier, two new *datasets* were created, both had no outliers and one had the **hour** feature and the other didn't. At first, there was a doubt on whether to use z-score and min-max or only one of them. With a small controlled test, i.e. training an **MLP** on both *datasets* and testing it against the validation set, it was clear that z-score performed better which is good because it avoids possible problems that min-max could generate when account for new data with different ranges.

With that being said, z-score scaling was applied to both *datasets* and therefore creating two more *datasets* (total - 4).

4.2 Balancing

The last data preparation technique applied was balancing.

Having in mind that the data is limited at there's a huge unbalance between the two classes, the techniques that made the most sense were oversampling and SMOTE sampling.

Finally, these techniques were applied to both datasets (with and without hour feature) but **exclusively** in the **training subset** and generated 4 more datasets.

4.3 Baseline

As a result of the previous steps, it was time to put all of the 8 different *datasets* and train a base MLP to see which of these obtained better results.

The MLP model was evaluated with cross-validation of 5 folds with this basic parameters:

solver = lbfgs / hidden layer sizes = (5) / activation = relu

The results were as follows:

Dataset Strategy	Avg. Score (Prec, Rec, F1, Acc)
Default without hour	0.7786
Default with hour	0.7584
Zscore without hour	0.9553
Zscore with hour	0.9583
Zscore - Oversampling	0.9462
Zscore - SMOTE sampling	0.9575
Default with hour - Oversampling	0.8058
Default with hour - SMOTE	0.6484

Table 1: Analysis of MLP results for all datasets

The results show that the best dataset was the one with the hour feature and zscore scaling. Its performance was very close to the one with balancing, which means that for the sake of this simpler model, balancing will not be used since it apparently does not improve over a simple zscore scaling.

4.4 Hyper-parameter tuning

Having this baseline, the goal is to tune an MLP as much as possible so that it can outperform any of the baseline results.

To test many hyper-parameters, a list of combinations was created that contained the following parameters:

Parameter	Values
Hidden Layer Sizes	[2, 2] to [50, 50]
Activation Functions	tanh, relu and logistic
Solvers	lbfgs, sgd and adam
Alpha	0.001 and 0.01
Learning Rate	Adaptive

Table 2: MLP Hyper-parameters

Notice that only two hidden layers were used because some quick preliminary tests showed that it outperformed any other number of hidden layers and therefore it would be a waste to test more combinations than these.

To train and evaluate all of this combinations against the previously chosen dataset, a technique called **Cross-validation** was used with 5 folds that contained on average 1800 entries each to better approximate to the real performance of a given MLP model, which is line with the suggested workflow in Figure 4.

The datasets used for this training and evaluation were the train and validation sets.

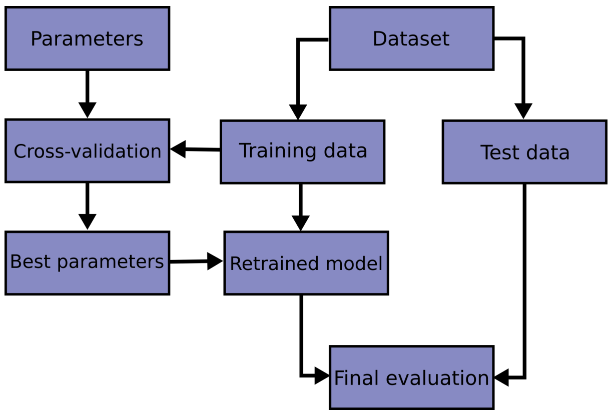

Figure 4: Hyper-parameter tuning with Cross-validation

4.5 Results

After running all of those iterations the following results were obtained.

Best model parameters:

Parameter	Values
Hidden Layer Sizes	(44, 26)
Activation Functions	relu
Solvers	lbfgs
Alpha	0.001
Learning Rate	Adaptive

Table 3: MLP Hyper-parameters

Results for validation set:

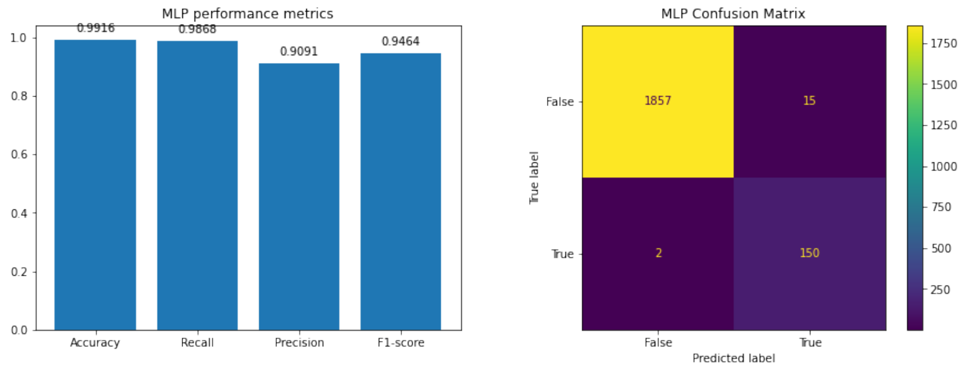

Figure 5: Results on Validation Set with Classifier A

Results for test set:

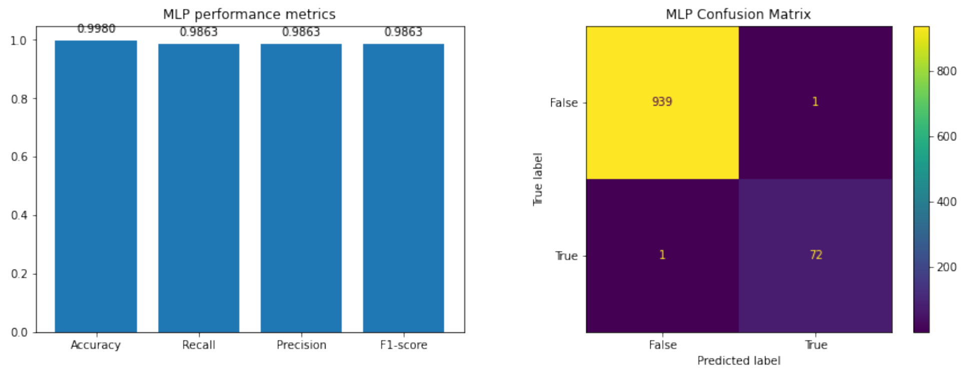

Figure 6: Results on Test Set with Classifier A

These results show a significant improvement over the baseline model and show very good results when running the holdout set (test set).

5 Classifier B

This classifier aims to predict exactly how many people are in the lab. With regards to classifier A, a lot of the choices here were very similar with few exceptions. The first exception is that now there's not the need to create a new **Target Column** because the default format of the labels is the required one.

5.1 Scaling

Following the usual data prep and baseline steps there's Scaling which was applied in the same way as the previous classifier, generating 2 datasets.

5.2 Balancing

Balancing is a different case because now there's unbalance between classes, where some classes have more representation in the dataset than others as show in Figure7.

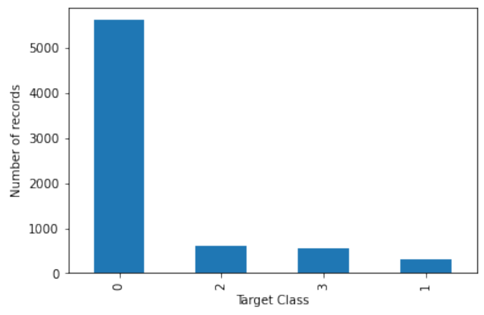

Figure 7: Classes distribution on the unbalanced dataset

To circumvent this issue the only balancing technique that seemed to make sense for this problem was SMOTE because of its nature and because there was the need to balance all of the classes simultaneously to reach the majority class (class with more values - 0) which is already done by the SMOTE function.

With this being said, Figure 8 show the full balanced **training** dataset (other subsets should never be applied balancing because they are used for validation and testing).

5.3 Baseline

Once again, as a result of the previous steps, it was time to put all of the different *datasets* and train a base MLP to see which of these obtained better results.

The MLP model was also evaluated with cross-validation of 5 folds with this basic parameters:

solver = lbfgs / hidden layer sizes = (5) / activation = relu

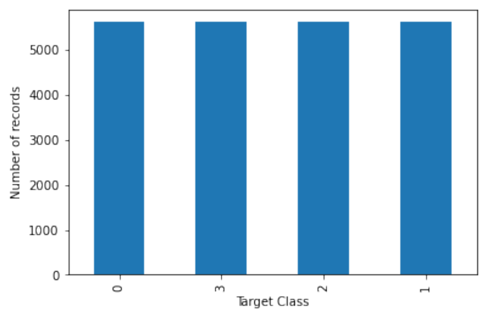

Figure 8: Classes distribution on the balanced dataset

Dataset Strategy	Avg. Score (Prec, Rec, F1, Acc)
Default without hour	0.4698
Default with hour	0.4698
Zscore without hour	0.9780
Zscore with hour	0.9801
Zscore - SMOTE	0.9754
Zscore with hour - SMOTE	0.9789
Default with hour - SMOTE	0.2242

Table 4: Analysis of MLP results for all datasets

The results were as follows:

The results show the same story to the ones from classifier A but in this case the balancing set makes a lot more sense because it is relevant for the model given the multiclass nature of the problem and since it is an advantage to have better representations of each class, or else the model could just predict the majority class and still yield good accuracy and precision.

This means that the hyper-parameter tuning was done with a balanced dataset.

5.4 Hyper-parameter tuning

Once again, the goal is to tune an MLP as much as possible so that it can outperform any of the baseline results.

Parameter	Values
Hidden Layer Sizes	[3, 3] to [15, 15]
Activation Functions	relu and logistic
Solvers	lbfgs
Alpha	0.001 and 0.01
Learning Rate	Adaptive

Table 5: MLP Hyper-parameters

Notice that hidden layers only go up to (15, 15) because more neurons would yield very marginal results, would add complexity to the model, and possibly could **overfit** the model. After some experimentation it was also clear that tanh activation function and adam/sgd solvers performed poorly for this dataset and therefore were not included in the combinations of parameters.

To train and evaluate all of this combinations against the previously chosen dataset, a technique called **Cross-validation** was used with 5 folds that contained on average 1800 entries each to better approximate to the real performance of a given MLP model.

The datasets used for this training and evaluation were the train and validation sets.

5.5 Results

After running all of those iterations the following results were obtained.

Best model parameters:

Parameter	Values
Hidden Layer Sizes	(13, 13)
Activation Functions	relu
Solvers	lbfgs
Alpha	0.001
Learning Rate	Adaptive

Table 6: MLP Hyper-parameters

Results for validation set:

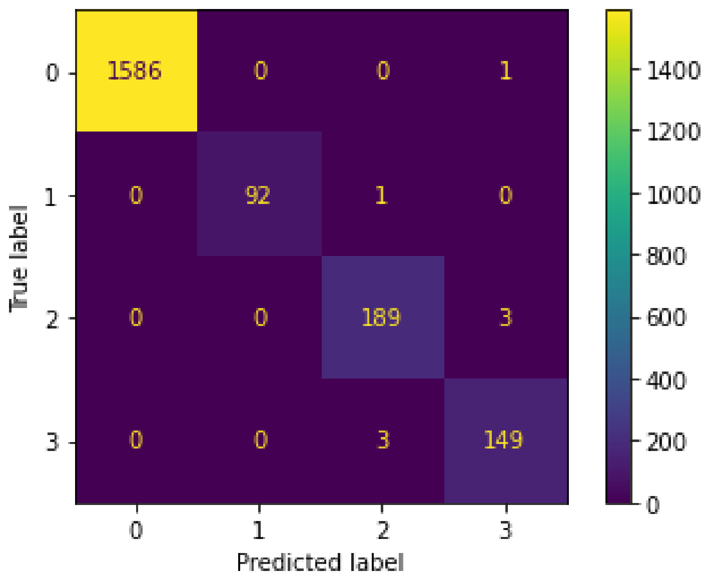

Figure 9: Classifier B results for Validation Set

With the given macro metrics/scores:

Metric	Value
Macro Precision	0.9883
Macro Recall	0.9882
Macro F-measure	0.9882

Table 7: MLP Macro Metrics for Validation Set

And scores for each class:

Class	Precision	Recall	F1-score
Class 0	1.0000	0.9994	0.9997
Class 1	1.0000	0.9892	0.9946
Class 2	0.9793	0.9844	0.9818
Class 3	0.9739	0.9803	0.9770

Table 8: MLP Metrics for each class on Validation Set

Results for Test set:

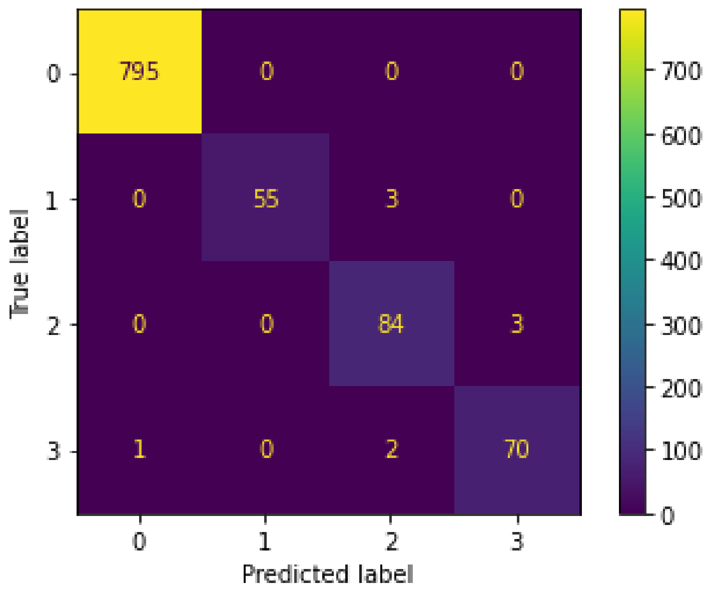

Figure 10: Classifier B results for Test Set

With the given macro metrics/scores:

Metric	Value
Macro Precision	0.9681
Macro Recall	0.9753
Macro F-measure	0.9715

Table 9: MLP Macro Metrics for Test Set

And scores for each class:

Class	Precision	Recall	F1-score
Class 0	0.9987	1.0000	0.9994
Class 1	1.0000	0.9483	0.9735
Class 2	0.9438	0.9655	0.9545
Class 3	0.9589	0.9589	0.9589

Table 10: MLP Metrics for each class on Test Set