MNIST SIGN LANGUAGE
Direct replacement for MNIST for hand gesture recognition tasks (“sign-language-mnist”)
(June 2021)

Eng, Carlos Arbey Mejía
Student University Autónoma de Occidente, Cali
Carlos_arbey.mejia@uao.edu.co

Abstract—- This paper is intended to present the process and results obtained in the project of recognition and classification of letters of USA´s alphabet, using the models presented in the Machine Learning class of the University Autonoma de Occidente.

I. INTRODUCTION

The problem to be dealt with in this document is the generated a model for classification the Data Set raised on the website Kaggle [1]. This Data Set is a CSV format with information from American Sign Language hand gestures, which represents 24 kinds of letters.

MNIST, is a big data base of handwritten digits, which is commonly used for image training. This data base, also, it is widely used in the field of machine learning.

II. DESCRIPTION OF THE PROBLEM

A. *Why this problem?*

I have always been interested in being able to know and learn sign language, for to be able to have a conversation with a person with auditory disability.
In my country, Colombia, exist the INSCOR (Instituto Nacional para Sordos), which has as its main objective, “*Promote from the education sector, the development and implementation of public policy for the social inclusion of the deaf population*” [2].

In the world there are 360 million of people with auditory disability, that equals is the 5% of the world population, this information is obtained of the World Health Organization [3], of this 360 million, 32 million are children with disability, the WHO organized this population in two groups:

- Adults (15 or over year): Hearing loss of more than 40 decibels (dB) in the ear with which you hear best.
- Children (0 to 14 years): Hearing loss of more than 30 dB in the ear with the best hearing.

FUENTES DE SONIDO	DECIBELES
Umbral de audición	0
Susurro, respiración normal, pisadas suaves	10
Rumor de las hojas en el campo al aire libre	20
Murmullo, oleaje suave en la costa	30
Biblioteca, habitación en silencio	40
Tráfico ligero, conversación normal	50
Oficina grande en horario de trabajo	60
Conversación en voz muy alta, gritería, tráfico intenso de ciudad	70
Timbre, camión pesado moviéndose	80
Aspiradora funcionando, maquinaria de una fábrica trabajando	90
Banda de música rock	100
Claxon de un coche, explosión de petardos o cohetes empleados en pirotecnia	110
Umbral del dolor	120
Martillo neumático (de aire)	130
Avión de reacción durante el despegue	150
Motor de un cohete espacial durante el despegue	180

Fig. 1. Here is a table in decibel scale Spanish [4].

III. PROCESSING DIAGRAMS

A. *Alphabet to be evaluated in the experiment:*

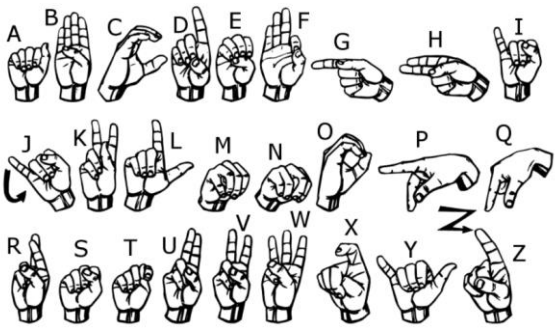

Fig. 2. Process of how the data set can be built [5]

B. *Information generation process:*

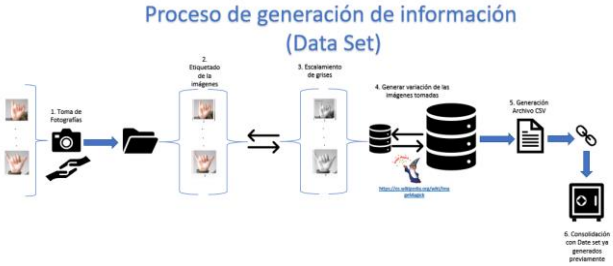

Fig. 3. Process of how the data set can be built.

The Data Set used in this paper was taken of the repository of Kaggle [1], therefore, the diagram above is an estimate.

These steps estimated are:

1. Taking a photograph of the right and left hand referencing the alphabetic letters (by both hands), an image of 28 x 28 pixels is generated.
2. Generate labeling of images.
3. Grayscale is generated.
4. Generate variation of the images taken, position, width, height, etc.
5. Generate a CSV file with the information of the pixels, validating the labeling generated in the images stage.
6. Consolidate a new Data Set with the Data Sets that have been collected

C. *Validation process and problem modeling*

This problem is considered a classification problem, do the following models planted in the Machine learning class will be used.

1) *KNN-neighbors classification*
2) *Logistic Regression.*
3) *Random Forest*

IV. DATA

For the project's data was used the file "DataSet Unificado.csv". This file is the merger of the data set of files "sign_mnist_test" and "sign mnist_train", of original project "MNIST"[1].

The data set has the following characteristics:

- 24 numeric labels, which represent a letter of American's alphabet:

```
######################################################
Etiqueta Objetivo: [ 0  1  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
######################################################
```

Fig. 4. Labels represented in the project's Colab.

- 784 features, which represent a pixel of letter´s picture:

```
Caracteristicas:
 Index(['pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6', 'pixel7',
        'pixel8', 'pixel9', 'pixel10',
        ...
        'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779', 'pixel780',
        'pixel781', 'pixel782', 'pixel783', 'pixel784'],
       dtype='object', length=784)
```

Fig. 5. Features represented in the project´s Colab.

- The data set unified of mint repository has 34.627 samples.

The data set was split in 3 groups, training, validation and test, either one with 70%, 20% and 10% respectively:

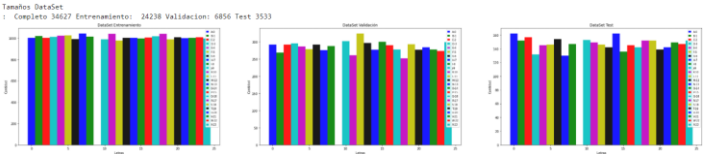

Fig. 6. Data Set Distribution in training, validation and test in project´s Colab.

V. DATA PROCESSING

As an initial phase of project, processing was do using the sklearn´s libraries:

- **StandardScaler:** Standardize features by removing the mean and scaling to unit variance [7].
- **Normalizer:** Normalize samples individually to unit norm [7]. Has as parameter the function to normalize with L1.
- **MinMaxScaler:** Transform features by scaling each feature to a given range [7].

Given that the data set has very features (784), has been made a reduction of dimension of dataset using PCA, but previously we used the model lasso for generated the number of features that really contribute information to the model, to be able to use it as a hyperparameter of the PCA model. Also, we used the parameter of features for PCA variance, with of value of 95% accumulative variance.

With the methods of pre-processing of datasets that we used; new datasets were generated that were evaluated on the models that were taken into account to solve our problem. These methods were combined as follows for all models:

- The training, validation and test datasets were transformed in each of the methods indicated above.
- A model was generated using PCA indicating the number of components found with lasso.
- A model was generated using PCA indicating the expected distribution percentage of 95% of variance.

With the steps indicated above, new training, validation and test datasets were created for each indicated methodology (StandarScaler, Normalizer and MinMaxScaler).

1. **Lasso**: Training was carried out with the lasso model to find the number of characteristics used from our training dataset, which gave us a total of 288 characteristics:

```
##################################################
Número de características usadas: 288
##################################################
```

Fig. 7. The number of features with weight after we use the lasso model.

2. **StandardScaler**: The transformation of the data was carried out, generating two groups of dataset of training, validation and test, these were with the following characteristics:

```
transformado shape PCA con lasso : (24238, 288)
transformado shape PCA con 95% : (24238, 116)
```

Fig. 8. The number of features generated applying PCA with the number of features of lasso and 95% of variance.

In the case of data transformed with PCA using lasso, 288 characteristics will be used, unlike with PCA with a distribution of 95% of variance, 116 characteristics will be used.

3. **Normalizer**: The transformation of the data was carried out, generating two groups of dataset of training, validation and test, these were with the following characteristics:

```
transformado shape PCA con lasso : (24238, 288)
transformado shape PCA con 95% : (24238, 149)
```

Fig. 9. The number of features generated applying PCA with the number of features of lasso and 95% of variance.

In the case of data transformed with PCA using lasso, 288 characteristics will be used, unlike with PCA with a distribution of 95% of variance, 149 characteristics will be used.

4. **MinMaxScaler**: The transformation of the data was carried out, generating two groups of dataset of training, validation and test, these were with the following characteristics:

```
transformado shape PCA con lasso : (24238, 288)
transformado shape PCA con 95% : (24238, 115)
```

Fig. 10. The number of features generated applying PCA with the number of features of lasso and 95% of variance.

In the case of data transformed with PCA using lasso, 288 characteristics will be used, unlike with PCA with a distribution of 95% of variance, 115 characteristics will be used.

The distribution graph of the data obtained with the training of the two PCA models is as follows:

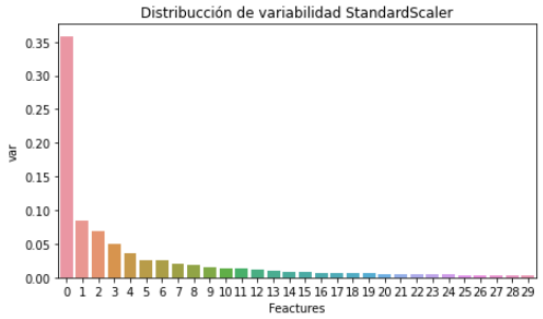

Fig. 11. Variance distribution using model PCA.

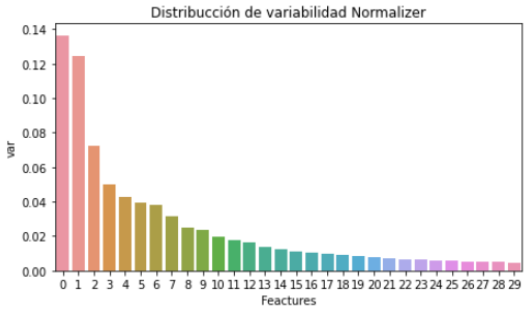

Fig. 12. Variance distribution using model PCA.

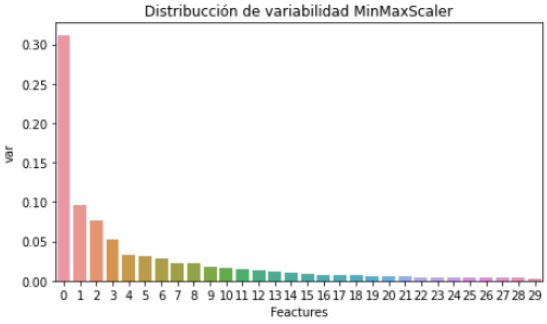

Fig. 13. Variance distribution using model PCA.

VI. MODELS RESULTS

Next in the presentation of our results, the PCA training with 288 characteristics will be called as **PCA_1** and the training with 95% of the variance as **PCA_2**:

A. ***KNN-neighbors classification:*** We perform our complexity analysis using PCA_1, obtaining the following:

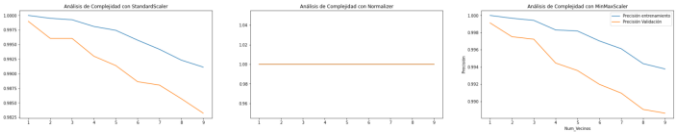

Fig. 14. The complexity analysis was done with a list 10 neighbors, begin since 1, 2, ... until 10

In these results it was possible to show that in the three data processing models the best hyperparameter is 1 neighbor, which yielded the following scores in the validation data set:

- StandardScaler of 99.898%
- Normalizer of 100%
- MinMaxScaler of 99.912%

The metrics of the models were validated using Precision, recall and accuracy for the three. Validating the models on test dataset, the following results were generated:

	PRECISION	RECALL	ACCURACY
MÉTRICAS KNN-neighbour with StandarScaler	0.998893	0.998885	0.998868
MÉTRICAS KNN-neighbour with Normalizer	1.000000	1.000000	1.000000
MÉTRICAS KNN-neighbour with MinMaxScaler	0.999421	0.999421	0.999434

Fig. 15. Results obtained when validating the models built in the Project´s Colab.

With these results it is concluded that the best model used in the preprocessing of the data was Normalizer.

Next, we proceed to show the results obtained with PCA_2. A complexity analysis was carried out, obtaining:

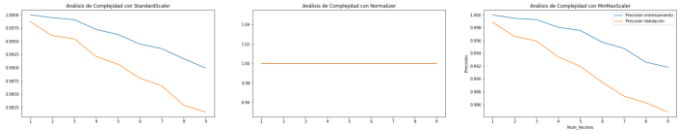

Fig. 16. The complexity analysis was done with a list 10 neighbors, begin since 1, 2, ... until 10

In these results it was possible to show that in the three data processing models the best hyperparameter is 1 neighbor, which yielded the following scores in the validation data set:

- StandardScaler of 99.869%
- Normalizer of 100%
- MinMaxScaler of 99.883%

The metrics of the models were validated using Precision, recall and accuracy for the three. Validating the models on test dataset, the following results were generated:

	PRECISION	RECALL	ACCURACY
MÉTRICAS KNN-neighbour with StandarScaler	0.998893	0.998885	0.998868
MÉTRICAS KNN-neighbour with Normalizer	1.000000	1.000000	1.000000
MÉTRICAS KNN-neighbour with MinMaxScaler	0.999149	0.999151	0.999151

Fig. 17. Results obtained when validating the models built in the Project´s Colab.

Here we can also see that the Normalizer preprocessing model generated the best result.

B. ***Logistic Regression***: We perform our complexity analysis using PCA_1, obtaining the following:

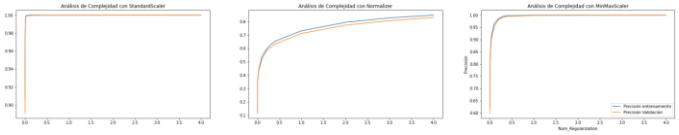

Fig. 18. The complexity analysis was done with a list of number C equals to [0.001, 0.002, 0.003, 0.004, 0.01, 0.02, 0.03, 0.04, 0.1, 0.2, 0.3, 0.4, 1.0, 2.0, 3.0, 4.0]

In these results it can be seen that in the three data processing models the best hyperparameter is C = 1, which yielded the following scores in the validation dataset:

- StandardScaler of 100%
- Normalizer of 100%
- MinMaxScaler of 100%

The metrics of the models were validated using Precision, recall and accuracy for the three. Validating the models on test dataset, the following results were generated:

	PRECISION	RECALL	ACCURACY
MÉTRICAS Logistic Regression with StandarScaler	1.000000	1.000000	1.000000
MÉTRICAS Logistic Regression with Normalizer	0.721802	0.721021	0.722615
MÉTRICAS Logistic Regression with MinMaxScaler	0.999179	0.999138	0.999151

Fig. 19. Results obtained when validating the models built in the Project´s Colab

In this model, unlike in KNN, the preprocessing with StandardScaler generates a better result in the test dataset and the Normalizer presents a considerable decrease in the results.

Next, we proceed to show the results obtained with PCA_2. A complexity analysis was carried out, obtaining:

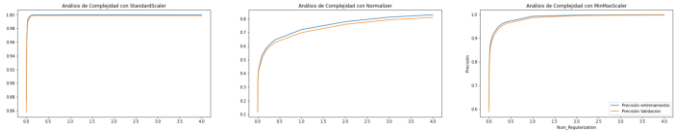

Fig. 20. The complexity analysis was done with a list of number C equals to [0.001, 0.002, 0.003, 0.004, 0.01, 0.02, 0.03, 0.04, 0.1, 0.2, 0.3, 0.4, 1.0, 2.0, 3.0, 4.0]

In these results it can be seen that in the three data processing models the best hyperparameter is C = 1, which yielded the following scores in the validation dataset:

- StandardScaler of 99.869%
- Normalizer of 100%
- MinMaxScaler of 99.883%

The metrics of the models were validated using Precision, recall and accuracy for the three. Validating the models on test dataset, the following results were generated:

	PRECISION	RECALL	ACCURACY
MÉTRICAS Logistic Regression with StandarScaler	0.999696	0.999729	0.999717
MÉTRICAS Logistic Regression with Normalizer	0.709653	0.709238	0.711010
MÉTRICAS Logistic Regression with MinMaxScaler	0.989615	0.989358	0.989527

Fig. 21. Results obtained when validating the models built in the Project´s Colab

In this model, a reduction of results could be seen in the three models, this could be due to the fact that the dataset is being scaled to fewer characteristics, generating a loss in data information.

C. ***Random Forest***: In this model, it began with the validation of the original dataset, that is, without performing any preprocessing of the data. The complexity analysis was carried out validating as an option the limit of trees of the ensemble which is 100 and the following complexity results were obtained:

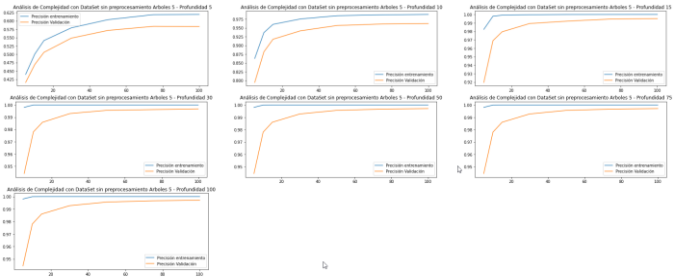

Fig. 22. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

In this experiment the best score in the validation dataset was 99.708% obtained with the hyperparameter num_tree of 100 and a depth of 50.

The metrics were validated using Precision, recall and accuracy for the model with the hyperparameters provided:

	PRECISION	RECALL	ACCURACY
MÉTRICAS Randome Forest with Original DataSet	0.995713	0.995805	0.995754

Fig. 17. Results obtained when validating the models built in the Project´s Colab with the dataset original

The complexity analysis was also carried out using PCA_1 and the DataSet obtained with preprocessing, obtaining the following:

Standard:

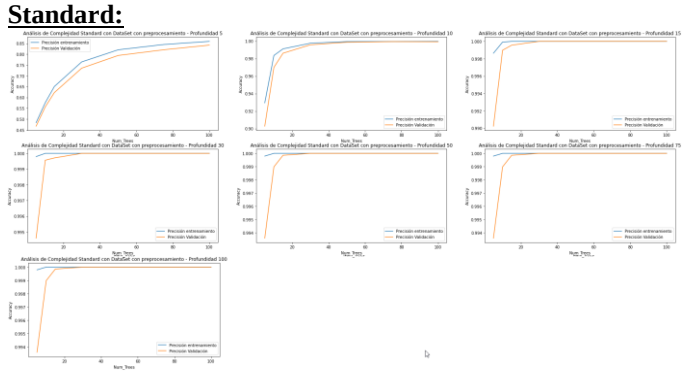

Fig. 23. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

Normalizer:

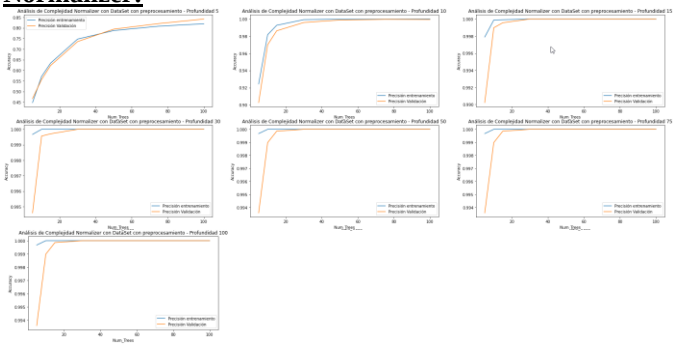

Fig. 24. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

MinMaxScaler:

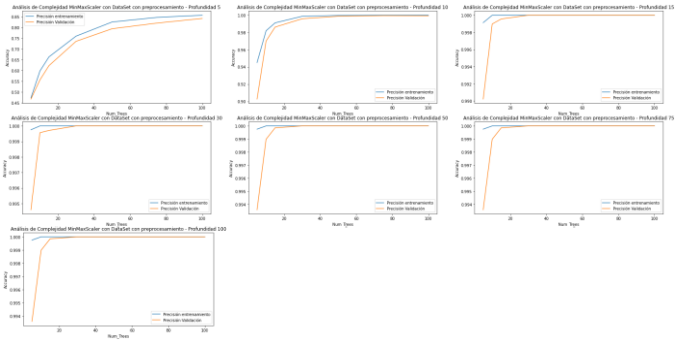

Fig. 25. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

With these results it was possible to show that in the three data processing models the best hyperparameters in the validation DataSet were:

- StandardScaler number of tree 30 and depth 15 score of 100%
- Normalizer number of tree 30 and depth 15 score of 100%
- MinMaxScaler number of tree 75 and depth 10 score of 100%

The module metrics were validated using Precision, recall and accuracy for the three models, validating the test DateSet, generating the following results:

	PRECISION	RECALL	ACCURACY
MÉTRICAS Randome Forest with Standard	1.000000	1.000000	1.000000
MÉTRICAS Randome Forest with Normalizer	1.000000	1.000000	1.000000
MÉTRICAS Randome Forest with MinMaxScaler	0.999152	0.999138	0.999151

Fig. 26. Results obtained when validating the models built in the Project´s Colab with the pre-processing dataset.

Now, we proceed to show the results obtained with PCA_2. A complexity analysis was carried out, obtaining:

Standard:

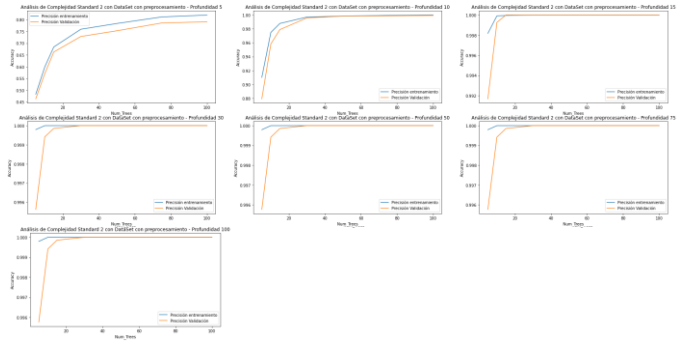

Fig. 27. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

Normalizer:

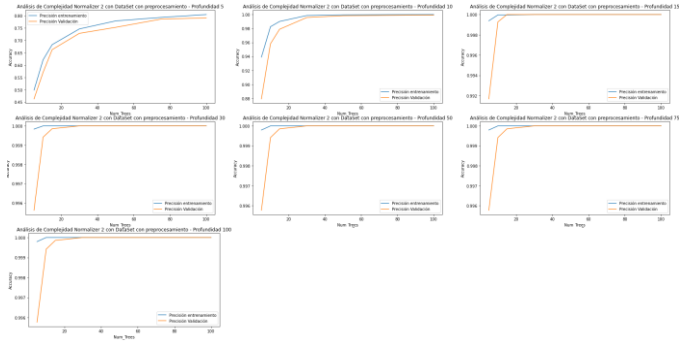

Fig. 28. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

MinMaxScaler:

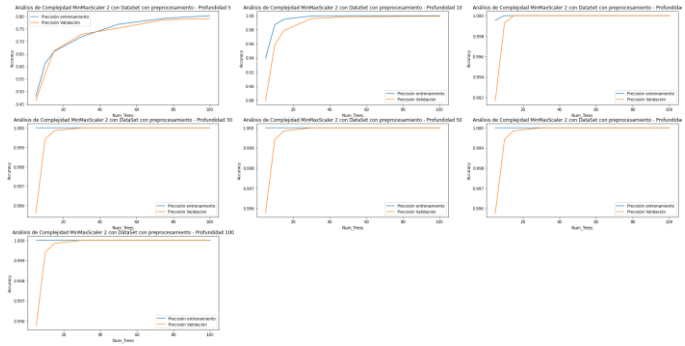

Fig. 29. The complexity analysis with the hyperparameters **n_estimators** using [5, 10, 15, 30, 50, 75, 100] and **max_depth** using [5, 10, 15, 30, 50, 75, 100]

With these results it was possible to show that in the three data processing models the best hyperparameters in the validation DataSet were:

- StandardScaler number of tree 15 and depth 15 score of 100%.
- Normalizer number of tree 15 nd depth 15 score of 100%.
- MinMaxScaler number of tree 30 and depth 15 score of 100%.

The module metrics were validated using Precision, recall and accuracy for the three models, validating the test DateSet, generating the following results:

	PRECISION	RECALL	ACCURACY
MÉTRICAS Randome Forest with Standard	1.000000	1.000000	1.000000
MÉTRICAS Randome Forest with Normalizer	0.999709	0.999707	0.999717
MÉTRICAS Randome Forest with MinMaxScaler	1.000000	1.000000	1.000000

Fig. 30. Results obtained when validating the models built in the Project´s Colab with the pre-processing dataset

In this model, a great improvement could be evidenced by preprocessing the data Vs validating the complete dataset, both in the result of precision, recall and accuracy, as well as in the hyperparameters used for the training of this.

VII. CONCLUSION

Validating the results of the best models of the problem using preprocessing, it was possible to show as good options for our problem, to use the models:

- ***KNN-neighbors classification:*** Using a 1-neighbor hyperparameter and data preprocessing using PCA_2 transformed with Normalizer.
- ***Logistic Regression:*** Using a hyperparameter C = 1 and a data preprocessing using PCA_1 transformed with StandardScaler.
- ***Random Forest:*** Using the hyperparameters num_tree = 15 and depth 15 for StandardScaler, num_tree = 15 and depth 15 for Normalizer and num_tree = 30 and depth 15 for MinMaxScaler and for all a process using PCA_2.

	PRECISION	RECALL	ACCURACY
MÉTRICAS KNN-neighbour with StandarScaler	0.998893	0.998885	0.998868
MÉTRICAS KNN-neighbour with Normalizer	1.000000	1.000000	1.000000
MÉTRICAS KNN-neighbour with MinMaxScaler	0.999421	0.999421	0.999434

Fig. 31. The best results obtained with KNN previously

	PRECISION	RECALL	ACCURACY
MÉTRICAS Logistic Regression with StandarScaler	1.000000	1.000000	1.000000
MÉTRICAS Logistic Regression with Normalizer	0.721802	0.721021	0.722615
MÉTRICAS Logistic Regression with MinMaxScaler	0.999179	0.999138	0.999151

Fig. 32. The best results obtained with Logistic Regression previously

	PRECISION	RECALL	ACCURACY
MÉTRICAS Randome Forest with Standard	1.000000	1.000000	1.000000
MÉTRICAS Randome Forest with Normalizer	0.999709	0.999707	0.999717
MÉTRICAS Randome Forest with MinMaxScaler	1.000000	1.000000	1.000000

Fig. 33. The best results obtained with Random Forest previously

RECOGNITION

I am grateful to the university for having this excellent specialization program and to Professor Victor Romero Cano, PhD, for transferring his knowledge and experiences in class.

REFERENCES

[1] Sign-language-mnist, website. [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist
[2] Instituto Nacional Para Sordos, website. [Online]. Available: https://www.insor.gov.co/home/entidad/objetivos-y-funciones/
[3] World Health Organization, website. [Online]. Available: https://www.who.int/features/factfiles/deafness/facts/es/
[4] Intensidad del sonido en desibelios, website. [Online]. Available: https://www.dbelectronics.es/intensidad-del-sonido-en-decibelios/
[5] Sign-language-mnist, American sign language, website. [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist?select=american_sign_language.PNG
[6] Model IEE Paper, website. [Online]: http://www.unisecmexico.com/archivosPDF/Formato_IEEE.pdf http://normas-apa.com/descargar-plantilla-formato-ieee-word/
[7] Scikit-learn, website. [Online]: https://scikit-learn.org/