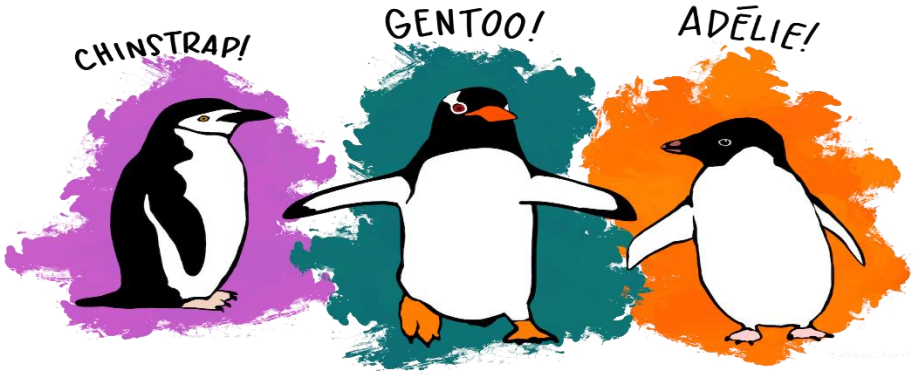
PALMER PENGUINS DATABASE MACHINE LEARNING IMPLEMENTATION WITH NEURAL NETWORKS IN PYTHON

BY: FRANCISCO OLVERA HERNÁNDEZ

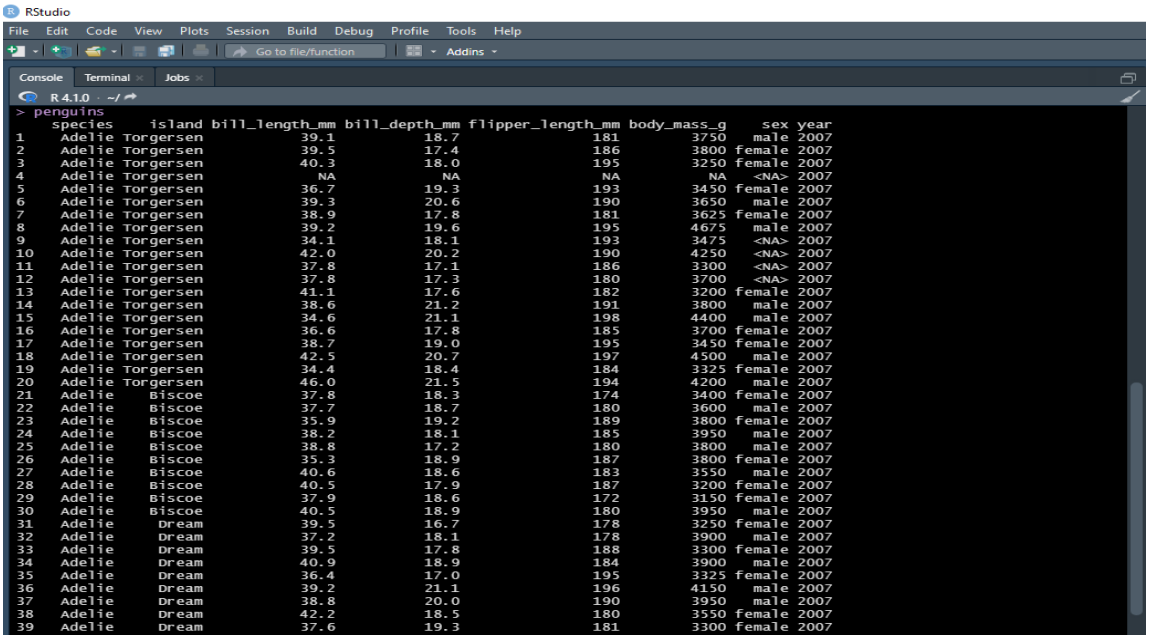


INTRODUCTION

The Palmer Penguins' dataset is a nice alternative to the Iris dataset commonly used in all machine learning examples. This dataset has data for 3 penguin species: Adélie, Chinstrap and Gentoo. By taking bill and flipper length measured in millimeters it is possible to classify a penguin. In this report I'll explain how my model works and the steps to clean, preprocess and train the dataset and finally predict and get precision percentage with train and test data.

DATA EXTRACTION FROM R FILE

I used RStudio to download the database in order to export it as a .CSV file.



	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
1	Adelie	Torgersen	39.1	18.7	181	3750	male	2007
2	Adelie	Torgersen	39.5	17.4	186	3800	female	2007
3	Adelie	Torgersen	40.3	18.0	195	3250	female	2007
4	Adelie	Torgersen	NA	NA	NA	NA	<NA>	2007
5	Adelie	Torgersen	36.7	19.3	193	3450	female	2007
6	Adelie	Torgersen	39.3	20.6	190	3650	male	2007
7	Adelie	Torgersen	38.9	17.8	181	3625	female	2007
8	Adelie	Torgersen	39.2	19.6	195	4675	male	2007
9	Adelie	Torgersen	34.1	18.1	193	3475	<NA>	2007
10	Adelie	Torgersen	42.0	20.2	190	4250	<NA>	2007
11	Adelie	Torgersen	37.8	17.1	186	3300	<NA>	2007
12	Adelie	Torgersen	37.8	17.3	180	3700	<NA>	2007
13	Adelie	Torgersen	41.1	17.6	182	3200	female	2007
14	Adelie	Torgersen	38.6	21.2	191	3800	male	2007
15	Adelie	Torgersen	34.6	21.1	198	4400	male	2007
16	Adelie	Torgersen	36.6	17.8	185	3700	female	2007
17	Adelie	Torgersen	38.7	19.0	195	3450	female	2007
18	Adelie	Torgersen	42.5	20.7	197	4500	male	2007
19	Adelie	Torgersen	34.4	18.4	184	3325	female	2007
20	Adelie	Torgersen	46.0	21.5	194	4200	male	2007
21	Adelie	Biscoe	37.8	18.3	174	3400	female	2007
22	Adelie	Biscoe	37.7	18.7	180	3600	male	2007
23	Adelie	Biscoe	35.9	19.2	189	3800	female	2007
24	Adelie	Biscoe	38.2	18.1	185	3950	male	2007
25	Adelie	Biscoe	38.8	17.2	180	3800	male	2007
26	Adelie	Biscoe	35.3	18.9	187	3800	female	2007
27	Adelie	Biscoe	40.6	18.6	183	3550	male	2007
28	Adelie	Biscoe	40.5	17.9	187	3200	female	2007
29	Adelie	Biscoe	37.9	18.6	172	3150	female	2007
30	Adelie	Biscoe	40.5	18.9	180	3950	male	2007
31	Adelie	Dream	39.5	16.7	178	3250	female	2007
32	Adelie	Dream	37.2	18.1	178	3900	male	2007
33	Adelie	Dream	39.5	17.8	188	3300	female	2007
34	Adelie	Dream	40.9	18.9	184	3900	male	2007
35	Adelie	Dream	36.4	17.0	195	3325	female	2007
36	Adelie	Dream	39.2	21.1	196	4150	male	2007
37	Adelie	Dream	38.8	20.0	190	3950	male	2007
38	Adelie	Dream	42.2	18.5	180	3550	female	2007
39	Adelie	Dream	37.6	19.3	181	3300	female	2007

Then looked for the .CSV path and opened it.

```
> path_to_file("penguins.csv")
[1] "D:/Documentos/R/win-library/4.1/palmerpenguins/extdata/penguins.csv"
> |
```

DATABASE ANALYSIS

Decided to check the dataset and noticed there were 8 columns with 344 rows. According to theory this is a small dataset and some parameters for the implemented neural network should be changed. But first I created a Jupyter Notebook for my Python program.

IMPLEMENTED MODEL

Neural networks are without a doubt one of the best models from Machine Learning to predict data so I needed to use *sklearn* library and a few others like *pandas* and *numpy*.

The neural network has 3 hidden layers with 15 neurons in each one with two X1 and X2 input values (bill length and flipper length).

TRAINING DATA AND TEST DATA

Data was cleaned because it had NaN values in some rows and got 333 rows with valid data as my 100%. I could have taken 60% or 80% but I finally took 70% as my training data which has 233 rows.

The dataset was split into two: **data** and **backup**. 'data' is 70% of the original dataframe and 'backup' 100%. Before splitting I shuffled the rows to get a better extraction of data and not mostly Adélie and Gentoo rows, that will help a lot to get a better trained model.

I highly recommend checking the program comments to get a better understanding of every single step to clean, split and backup the dataframe but here's a list of functions I used to make this work.

- pd.isnull(data["bill_length_mm])
- data.dropna()
- sk.utils.shuffle()
- data.reset_index(inplace = True, drop =True)
- data.copy()

After that I needed to set my X and Y values for input and expected predictions so I took the species column and used a LabelEncoder() to assign numerical values instead of words, this gave me a set of 3 ID's for species: 0-Adélie, 1-Chinstrap, 2-Gentoo stored in sp column.

Both bill length and flipper length already have float values, so those rows won't be changed. Encoding was applied for backup and data because I needed that column for both datasets and get my precision percentage at the end of my program.

MULTI-LAYER PERCEPTRON SETTING

Train and test data are stored in Xtrain and Xtest, sp column (which has species id's) is stored in Ytrain and Ytest. Preprocessing that data was important so I used a StandardScaler to fit Xtrain and Xtest to get better predictions.

My neural network was created with MLPClassifier with a few modified parameters (check program for more details) and then trained the model.

PREDICTIONS WITH 70% TRAINED DATA

After training with only 70% original data, I stored all predictions for every pair of values in predictions, then checked if first and last element predictions were correct, and finally a prediction with random values and the model predicted it perfectly, it was a Gentoo.

```
[0 1]
[2]
[0 2 1 1 1 2 0 0 0 2 2 2 2 0 1 2 0 2 2 1 2 1 1 0 0 1 2 0 1 2 0 0 2 2 2 2 2
 2 2 0 2 1 0 0 1 2 2 2 0 2 1 2 1 0 2 0 0 0 2 2 2 2 2 0 1 2 0 0 1 0 2 0 0 0
 2 0 2 2 2 1 1 2 0 2 1 2 0 1 2 0 0 1 0 0 2 1 0 0 2 2 2 0 2 0 2 0 2 0 0 0 1
 0 2 0 0 0 2 2 1 0 1 2 0 2 1 0 2 1 2 2 1 0 0 0 0 2 2 2 0 2 0 2 2 0 0 1 0 0
 2 0 1 2 0 1 0 2 2 2 2 0 0 2 0 0 1 2 0 1 2 1 1 2 0 0 0 1 2 2 0 0 1 0 2 1 0
 0 2 2 0 1 1 1 1 0 0 0 0 0 2 0 2 0 2 0 0 2 0 0 0 2 0 1 2 1 1 0 0 1 0 0 1 0
 1 2 2 0 0 2 1 1 0 0 1]
```

MODEL PRECISION WITH 70% DATA

To get my precision values for the model I used a classification_report with Ytrain and my predictions list.

This is the output:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	97
1	0.98	0.98	0.98	50
2	1.00	1.00	1.00	86
micro avg	0.99	0.99	0.99	233
macro avg	0.99	0.99	0.99	233
weighted avg	0.99	0.99	0.99	233

PREDICTIONS WITH 100% TEST DATA

I stored my final predictions with test data (Xtest and Ytest) in final predictions

This is the output (remember this is for 333 rows):

```
[2 2 2 0 1 2 0 0 2 2 0 1 1 0 2 1 2 1 0 2 2 2 1 1 2 0 0 0 1 2 0 0 2 1 0 0 2
 1 2 0 1 1 0 0 0 2 2 2 0 2 0 0 2 2 0 2 0 0 1 0 2 1 0 0 0 2 1 2 2 2 1 2 0 1
 2 2 0 2 2 0 1 2 0 2 2 2 1 1 2 0 0 0 2 2 0 1 1 0 0 0 0 2 2 2 0 1 0 2 2 0 0
 0 1 0 2 0 1 2 2 0 2 0 2 0 2 2 2 0 0 0 1 0 0 0 2 0 0 2 1 2 0 2 1 0 0 2 2 2
 1 1 0 0 0 1 2 2 0 1 0 2 0 2 2 0 1 0 2 0 2 0 0 2 0 2 0 2 0 0 0 0 2 2 0 0 1
 1 0 2 2 0 2 0 0 0 0 0 1 1 2 2 1 0 0 2 0 0 0 1 2 2 2 1 1 1 0 2 2 0 2 0 0 2
 2 0 0 1 0 1 1 0 2 0 2 0 2 2 0 2 0 2 1 2 0 2 2 0 2 0 0 0 1 0 0 2 0 0 0 0 2
 1 0 2 1 0 2 2 0 2 2 2 0 1 2 1 0 0 2 0 0 1 0 2 1 1 0 0 2 0 2 1 1 2 2 2 2 0
 2 0 0 2 0 0 0 0 2 1 1 2 0 0 2 0 1 0 2 1 1 0 1 0 0 0 0 2 0 0 1 0 1 2 0 1 2]
```

MODEL PRECISION WITH 100% TEST DATA

My classification report showed that precision was slightly changed but it was without a doubt a good percentage.

	precision	recall	f1-score	support
0	0.97	0.98	0.97	146
1	0.94	0.93	0.93	68
2	0.99	0.98	0.99	119
micro avg	0.97	0.97	0.97	333
macro avg	0.97	0.96	0.96	333
weighted avg	0.97	0.97	0.97	333

DECLARING A LIST WITH INPUT VALUES AND ORIGINAL SPECIES

Just wrote a bit of code to store backup's bill length, flipper length and original species values in _list[] and then just printed it.

```
[[42.7, 196.0, 0], [41.6, 192.0, 0], [39.3, 190.0, 0], [53.4, 219.0, 2], [37.0, 185.0, 0], [42.1, 195.0, 0], [50.7, 203.0, 1],
[37.6, 194.0, 0], [43.6, 217.0, 2], [48.5, 219.0, 2], [37.5, 199.0, 0], [50.2, 202.0, 1], [52.0, 197.0, 1], [46.8, 215.0, 2],
[39.2, 195.0, 0], [48.7, 210.0, 2], [49.5, 224.0, 2], [39.5, 188.0, 0], [39.8, 184.0, 0], [36.2, 187.0, 0], [42.0, 200.0, 0],
[40.8, 208.0, 0], [35.3, 187.0, 0], [40.2, 193.0, 0], [35.0, 192.0, 0], [42.5, 197.0, 0], [37.9, 193.0, 0], [41.5, 201.0, 0],
[45.2, 215.0, 2], [50.5, 216.0, 2], [42.8, 209.0, 2], [48.6, 230.0, 2], [39.5, 186.0, 0], [46.0, 195.0, 1], [55.8, 207.0, 1],
[36.5, 182.0, 0], [38.7, 195.0, 0], [51.4, 201.0, 1], [45.6, 191.0, 0], [58.0, 181.0, 1], [38.8, 180.0, 0], [49.3, 203.0, 1],
[39.0, 191.0, 0], [51.9, 206.0, 1], [40.6, 199.0, 0], [47.6, 215.0, 2], [45.2, 191.0, 1], [50.0, 230.0, 2], [40.6, 183.0, 0],
[45.4, 188.0, 1], [36.6, 184.0, 0], [38.3, 189.0, 0], [38.6, 191.0, 0], [38.9, 181.0, 0], [38.2, 185.0, 0], [43.2, 197.0, 0],
[45.7, 193.0, 1], [40.5, 187.0, 0], [50.7, 223.0, 2], [46.5, 217.0, 2], [51.1, 220.0, 2], [38.8, 190.0, 0], [40.3, 195.0, 0],
[42.6, 213.0, 2], [46.8, 189.0, 1], [50.4, 222.0, 2], [46.9, 192.0, 1], [35.5, 190.0, 0], [50.0, 220.0, 2], [59.6, 230.0, 2],
[42.8, 195.0, 0], [35.7, 202.0, 0], [45.1, 215.0, 2], [37.8, 193.0, 0], [39.2, 196.0, 0], [40.7, 190.0, 0], [37.3, 192.0, 0],
[51.5, 187.0, 1], [41.0, 203.0, 0], [51.1, 225.0, 2], [47.5, 212.0, 2], [45.4, 211.0, 2], [36.2, 187.0, 0], [46.4, 190.0, 1],
[45.5, 210.0, 2], [43.8, 208.0, 2], [41.1, 189.0, 0], [45.0, 220.0, 2], [52.0, 210.0, 1], [46.8, 215.0, 2], [50.0, 196.0, 1],
[46.0, 194.0, 0], [42.7, 208.0, 2], [54.2, 201.0, 1], [52.8, 205.0, 1], [47.5, 199.0, 1], [41.1, 192.0, 0], [42.5, 187.0, 1],
[47.2, 215.0, 2], [49.6, 193.0, 1], [46.4, 221.0, 2], [43.5, 213.0, 2], [49.5, 200.0, 1], [45.8, 210.0, 2], [48.1, 209.0, 2],
[50.4, 224.0, 2], [48.8, 222.0, 2], [34.6, 198.0, 0], [47.6, 195.0, 1], [36.0, 187.0, 0], [37.8, 174.0, 0], [37.9, 172.0, 0],
[38.1, 190.0, 0], [39.6, 191.0, 0], [46.4, 216.0, 2], [37.2, 178.0, 0], [42.5, 187.0, 1], [46.2, 217.0, 2], [51.3, 218.0, 2],
[40.9, 187.0, 1], [39.1, 181.0, 0], [40.1, 188.0, 0], [40.6, 187.0, 0], [50.1, 225.0, 2], [39.6, 186.0, 0], [47.4, 212.0, 2],
[40.2, 176.0, 0], [41.7, 210.0, 2], [50.3, 197.0, 1], [46.5, 213.0, 2], [45.5, 214.0, 2], [48.4, 203.0, 2], [35.5, 195.0, 0],
[37.3, 199.0, 0], [44.5, 214.0, 2], [45.7, 195.0, 1], [40.6, 193.0, 0], [45.1, 207.0, 2], [50.0, 224.0, 2], [45.2, 223.0, 2],
[51.0, 203.0, 1], [48.4, 220.0, 2], [45.8, 219.0, 2], [46.5, 217.0, 2], [40.8, 195.0, 0], [52.7, 197.0, 1], [45.3, 208.0, 2],
[34.5, 187.0, 0], [41.3, 194.0, 0], [40.9, 184.0, 0], [36.7, 193.0, 0], [37.6, 181.0, 0], [46.7, 219.0, 2], [35.9, 190.0, 0],
[40.2, 200.0, 0], [38.5, 190.0, 0], [47.0, 185.0, 1], [35.0, 190.0, 0], [45.2, 215.0, 2], [46.5, 210.0, 2], [36.8, 193.0, 0],
[49.0, 210.0, 1], [38.6, 199.0, 0], [53.5, 205.0, 1], [48.7, 222.0, 2], [48.2, 221.0, 2], [37.6, 185.0, 0], [46.3, 215.0, 2],
[51.7, 194.0, 1], [39.0, 186.0, 0], [50.8, 201.0, 1], [41.1, 205.0, 0], [38.9, 190.0, 0], [41.8, 198.0, 0], [50.2, 218.0, 2],
[43.3, 208.0, 2], [35.7, 189.0, 0], [38.2, 190.0, 0], [39.7, 193.0, 0], [44.0, 208.0, 2], [50.8, 226.0, 2], [39.7, 184.0, 0],
[36.2, 187.0, 0], [39.7, 193.0, 0], [44.9, 213.0, 2], [43.4, 218.0, 2], [55.1, 230.0, 2], [47.2, 214.0, 2], [50.5, 201.0, 1],
[42.9, 196.0, 0], [43.5, 220.0, 2], [40.5, 180.0, 0], [37.2, 184.0, 0], [39.6, 186.0, 0], [45.8, 197.0, 1], [36.7, 187.0, 0],
[46.2, 221.0, 2], [50.1, 190.0, 1], [47.5, 209.0, 2], [45.5, 212.0, 2], [49.8, 229.0, 2], [39.6, 190.0, 0], [42.2, 197.0, 0],
[50.0, 218.0, 2], [52.1, 230.0, 2], [43.2, 192.0, 0], [39.2, 190.0, 0], [45.2, 198.0, 1], [45.9, 190.0, 1], [55.9, 228.0, 2],
[38.1, 187.0, 0], [49.5, 229.0, 2], [49.1, 220.0, 2], [47.3, 222.0, 2], [54.3, 231.0, 2], [44.1, 196.0, 0], [38.1, 198.0, 0],
[52.2, 228.0, 2], [45.7, 214.0, 2], [36.5, 181.0, 0], [37.0, 185.0, 0], [49.1, 228.0, 2], [37.3, 191.0, 0], [38.8, 191.0, 0],
[39.7, 190.0, 0], [48.5, 191.0, 1], [45.5, 220.0, 2], [46.8, 215.0, 2], [49.2, 221.0, 2], [36.0, 195.0, 0], [41.1, 182.0, 0],
[36.4, 195.0, 0], [33.5, 190.0, 0], [49.8, 198.0, 1], [34.0, 185.0, 0], [47.7, 216.0, 2], [49.3, 217.0, 2], [45.1, 210.0, 2],
[52.2, 197.0, 1], [39.6, 196.0, 0], [35.1, 193.0, 0], [42.3, 191.0, 0], [46.2, 187.0, 1], [45.3, 210.0, 2], [46.1, 211.0, 2],
```

DECLARING A LIST WITH INPUT VALUES AND PREDICTED SPECIES

Did the same but with predicted species for each pair of values.

```
[[42.7, 196.0, 0], [41.6, 192.0, 0], [39.3, 190.0, 0], [53.4, 219.0, 2], [37.0, 185.0, 0], [42.1, 195.0, 0], [50.7, 203.0, 1],
[37.6, 194.0, 0], [43.6, 217.0, 2], [48.5, 219.0, 2], [37.5, 199.0, 0], [50.2, 202.0, 1], [52.0, 197.0, 1], [46.8, 215.0, 2],
[39.2, 195.0, 0], [48.7, 210.0, 1], [49.5, 224.0, 2], [39.5, 188.0, 0], [39.8, 184.0, 0], [36.2, 187.0, 0], [42.0, 200.0, 0],
[40.8, 208.0, 0], [35.3, 187.0, 0], [40.2, 193.0, 0], [35.0, 192.0, 0], [42.5, 197.0, 0], [37.9, 193.0, 0], [41.5, 201.0, 0],
[45.2, 215.0, 2], [50.5, 216.0, 2], [42.8, 209.0, 2], [48.6, 230.0, 2], [39.5, 186.0, 0], [46.0, 195.0, 1], [55.8, 207.0, 1],
[36.5, 182.0, 0], [38.7, 195.0, 0], [51.4, 201.0, 1], [45.6, 191.0, 0], [58.0, 181.0, 1], [38.8, 180.0, 0], [49.3, 203.0, 1],
[39.0, 191.0, 0], [51.9, 206.0, 1], [40.6, 199.0, 0], [47.6, 215.0, 2], [45.2, 191.0, 1], [50.0, 230.0, 2], [40.6, 183.0, 0],
[45.4, 188.0, 1], [36.6, 184.0, 0], [38.3, 189.0, 0], [38.6, 191.0, 0], [38.9, 181.0, 0], [38.2, 185.0, 0], [43.2, 197.0, 0],
[45.7, 193.0, 1], [40.5, 187.0, 0], [50.7, 223.0, 2], [46.5, 217.0, 2], [51.1, 220.0, 2], [38.8, 190.0, 0], [40.3, 195.0, 0],
[42.6, 213.0, 2], [46.8, 189.0, 1], [50.4, 222.0, 2], [46.9, 192.0, 1], [35.5, 190.0, 0], [50.0, 220.0, 2], [59.6, 230.0, 2],
[42.8, 195.0, 0], [35.7, 202.0, 0], [45.1, 215.0, 2], [37.8, 193.0, 0], [39.2, 196.0, 0], [40.7, 190.0, 0], [37.3, 192.0, 0],
[51.5, 187.0, 1], [41.0, 203.0, 0], [51.1, 225.0, 2], [47.5, 212.0, 2], [45.4, 211.0, 2], [36.2, 187.0, 0], [46.4, 190.0, 1],
[45.5, 210.0, 2], [43.8, 208.0, 2], [41.1, 189.0, 0], [45.0, 220.0, 2], [52.0, 210.0, 1], [46.8, 215.0, 2], [50.0, 196.0, 1],
[46.0, 194.0, 1], [42.7, 208.0, 2], [54.2, 201.0, 1], [52.8, 205.0, 1], [47.5, 199.0, 1], [41.1, 192.0, 0], [42.5, 187.0, 1],
[47.2, 215.0, 2], [49.6, 193.0, 1], [46.4, 221.0, 2], [43.5, 213.0, 2], [49.5, 200.0, 1], [45.8, 210.0, 2], [48.1, 209.0, 2],
[50.4, 224.0, 2], [48.8, 222.0, 2], [34.6, 198.0, 0], [47.6, 195.0, 1], [36.0, 187.0, 0], [37.8, 174.0, 0], [37.9, 172.0, 0],
[38.1, 190.0, 0], [39.6, 191.0, 0], [46.4, 216.0, 2], [37.2, 178.0, 0], [42.5, 187.0, 1], [46.2, 217.0, 2], [51.3, 218.0, 2],
[40.9, 187.0, 1], [39.1, 181.0, 0], [40.1, 188.0, 0], [40.6, 187.0, 0], [50.1, 225.0, 2], [39.6, 186.0, 0], [47.4, 212.0, 2],
[40.2, 176.0, 0], [41.7, 210.0, 2], [50.3, 197.0, 1], [46.5, 213.0, 2], [45.5, 214.0, 2], [48.4, 203.0, 2], [35.5, 195.0, 0],
[37.3, 199.0, 0], [44.5, 214.0, 2], [45.7, 195.0, 1], [40.6, 193.0, 0], [45.1, 207.0, 2], [50.0, 224.0, 2], [45.2, 223.0, 2],
[51.0, 203.0, 1], [48.4, 220.0, 2], [45.8, 219.0, 2], [46.5, 217.0, 2], [40.8, 195.0, 0], [52.7, 197.0, 1], [45.3, 208.0, 2],
[34.5, 187.0, 0], [41.3, 194.0, 0], [40.9, 184.0, 1], [36.7, 193.0, 0], [37.6, 181.0, 0], [46.7, 219.0, 2], [35.9, 190.0, 0],
[40.2, 200.0, 0], [38.5, 190.0, 0], [47.0, 185.0, 1], [35.0, 190.0, 0], [45.2, 215.0, 2], [46.5, 210.0, 2], [36.8, 193.0, 0],
[49.0, 210.0, 1], [38.6, 199.0, 0], [53.5, 205.0, 1], [48.7, 222.0, 2], [48.2, 221.0, 2], [37.6, 185.0, 0], [46.3, 215.0, 2],
[51.7, 194.0, 1], [39.0, 186.0, 0], [50.8, 201.0, 1], [41.1, 205.0, 0], [38.9, 190.0, 0], [41.8, 198.0, 0], [50.2, 218.0, 2],
[43.3, 208.0, 2], [35.7, 189.0, 0], [38.2, 190.0, 0], [39.7, 193.0, 0], [44.0, 208.0, 2], [50.8, 226.0, 2], [39.7, 184.0, 0],
[36.2, 187.0, 0], [39.7, 193.0, 0], [44.9, 213.0, 2], [43.4, 218.0, 2], [55.1, 230.0, 2], [47.2, 214.0, 2], [50.5, 201.0, 1],
[42.9, 196.0, 0], [43.5, 220.0, 2], [40.5, 180.0, 0], [37.2, 184.0, 0], [39.6, 186.0, 0], [45.8, 197.0, 1], [36.7, 187.0, 0],
[46.2, 221.0, 2], [50.1, 190.0, 1], [47.5, 209.0, 2], [45.5, 212.0, 2], [49.8, 229.0, 2], [39.6, 190.0, 0], [42.2, 197.0, 0],
[50.0, 218.0, 2], [52.1, 230.0, 2], [43.2, 192.0, 0], [39.2, 190.0, 0], [45.2, 198.0, 1], [45.9, 190.0, 1], [55.9, 228.0, 2],
[38.1, 187.0, 0], [49.5, 229.0, 2], [49.1, 220.0, 2], [47.3, 222.0, 2], [54.3, 231.0, 2], [44.1, 196.0, 0], [38.1, 198.0, 0],
[52.2, 228.0, 2], [45.7, 214.0, 2], [36.5, 181.0, 0], [37.0, 185.0, 0], [49.1, 228.0, 2], [37.3, 191.0, 0], [38.8, 191.0, 0],
[39.7, 190.0, 0], [48.5, 191.0, 1], [45.5, 220.0, 2], [46.8, 215.0, 2], [49.2, 221.0, 2], [36.0, 195.0, 0], [41.1, 182.0, 0],
[36.4, 195.0, 0], [33.5, 190.0, 0], [49.8, 198.0, 1], [34.0, 185.0, 0], [47.7, 216.0, 2], [49.3, 217.0, 2], [45.1, 210.0, 2],
[52.2, 197.0, 1], [39.6, 196.0, 0], [35.1, 193.0, 0], [42.3, 191.0, 0], [46.2, 187.0, 1], [45.3, 210.0, 2], [46.1, 211.0, 2],
[43.5, 202.0, 1], [49.0, 212.0, 2], [46.7, 195.0, 1], [45.5, 196.0, 1], [50.9, 196.0, 1], [46.1, 178.0, 1], [36.6, 185.0, 0],
```

PRECISION PERCENTAGE

And finally I compared both final_predictions list (which has all predicted species from the original dataset) and species_backup (which has only the species numerical value from the original dataset) to get my final precision percentage.

```
percentage = len(set(final_predictions) & set(species_backup)) / float(len(set(final_predictions) | set(species_backup))) * 100
print("Precision percentage amongst lists: ",percentage)

Precision percentage amongst lists:  100.0
```

All penguins from the original dataset (100%) were perfectly classified according to the 70% trained data.

Precision percentage is equal to 100%.

CONCLUSION

Working with neural networks in Python to classify individuals is a great way to predict future data. With this code it is possible to process a bigger dataset and get good results. It is possible to change the neural networks architecture but with 3 hidden layers and less than 1000 max iterations works perfectly fine.