Machine Learning: Perceptron and Logistic Regression

Submitted by Mary Franczine Tan

Objectives

1. Create a perceptron model that can classify fruits
2. Create a logistic regression that rates the freshness of a fruit

Results and Analysis

Perceptron

For the training of the perceptron model, I have **500 input data** with **2 features** x_1 and x_2. The **x_0** represents the bias input. The 500 data are split evenly between two fruits: 250 apples and 250 banana. **Apples are assigned 1, while bananas are assigned -1.**

The features used are the fruit images' **eccentricity and average value of the green channel**. Eccentricity differentiates the shape of the fruit. Apples are more circular and have lower eccentricity, while bananas are longer and have higher eccentricity. The average of the green channel differentiates the color. Apples are red so their green channel average will be lower. Bananas are yellow (formed by the combination of active red and green channels) so their green channel average will be higher.

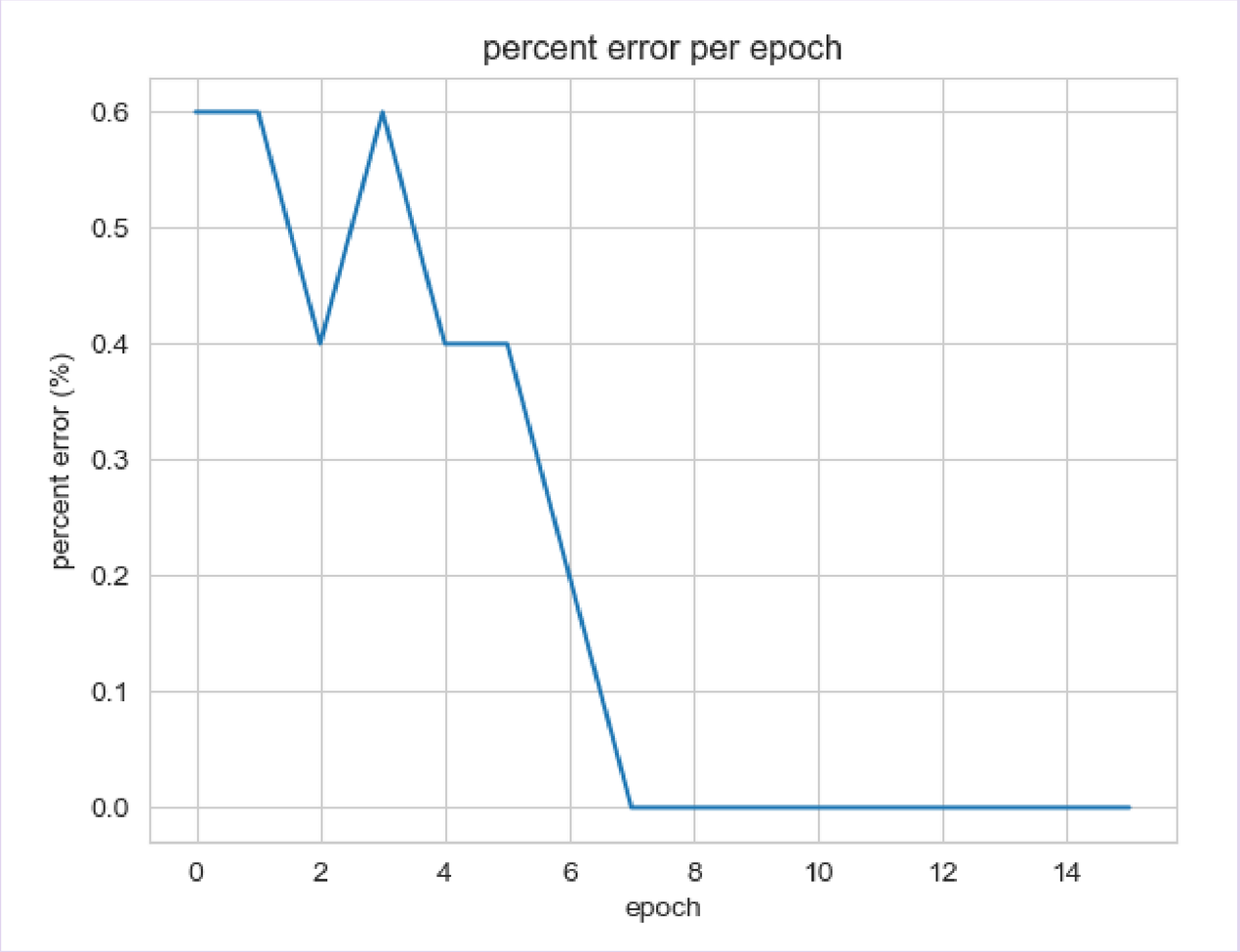
	fruit	x0	x1	x2	y
0	apple	1	0.515150	0.254916	1
1	apple	1	0.262903	0.225285	1
2	apple	1	0.258596	0.314737	1
3	apple	1	0.747035	0.446362	1
4	apple	1	0.542761	0.285845	1
...	...	...	...	...	...
495	banana	1	0.963050	0.536158	-1
496	banana	1	0.980396	0.878707	-1
497	banana	1	0.957705	0.863055	-1
498	banana	1	0.947108	0.793815	-1
499	banana	1	0.876745	0.560252	-1

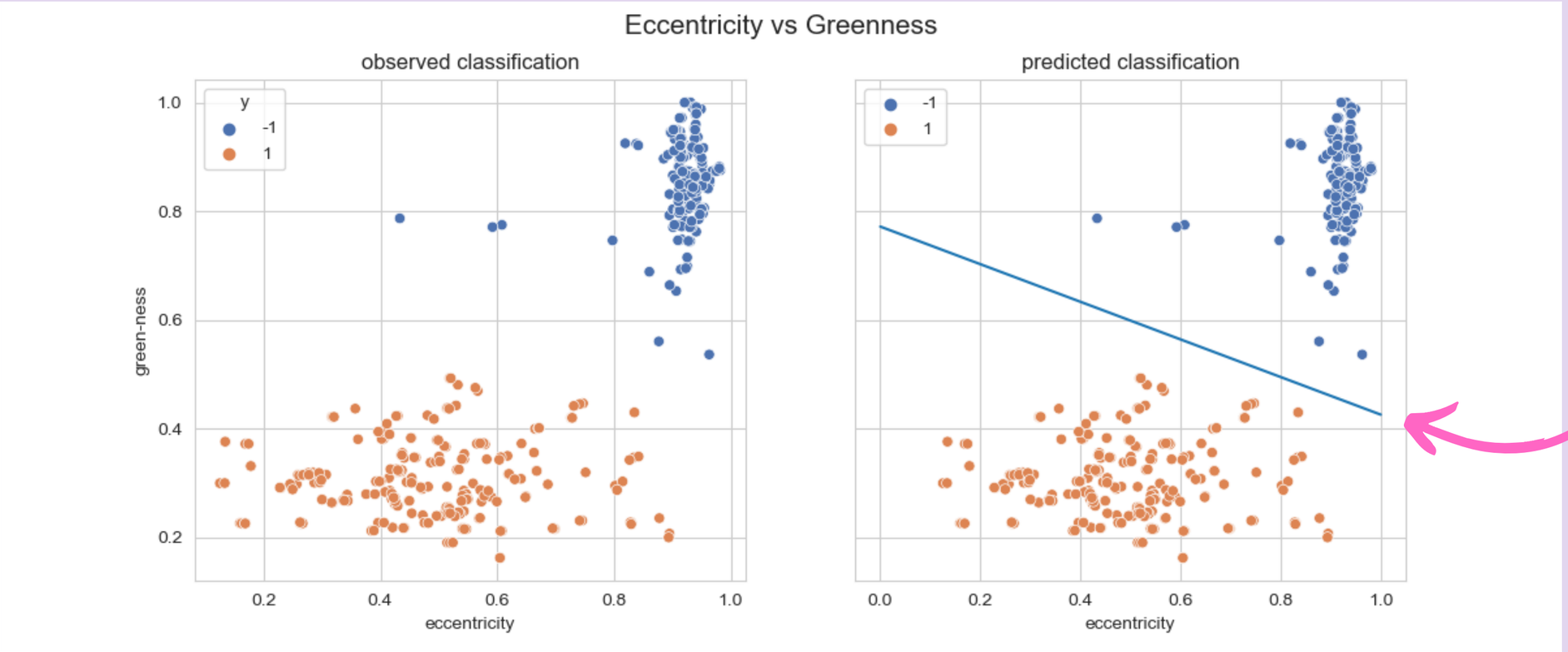
Model Peformance - Training

The figure to the right shows the **percent error per epoch.** We can see that **by epoch 7, the model has reached 0% error** on the training data. This means the model performed extremely well in classifying the fruits.

Using the final weights on the input data, we get the following confusion matrix:

	1 (predicted)	**-1 (predicted)**
1 (observed)	250	0
-1 (observed)	0	250



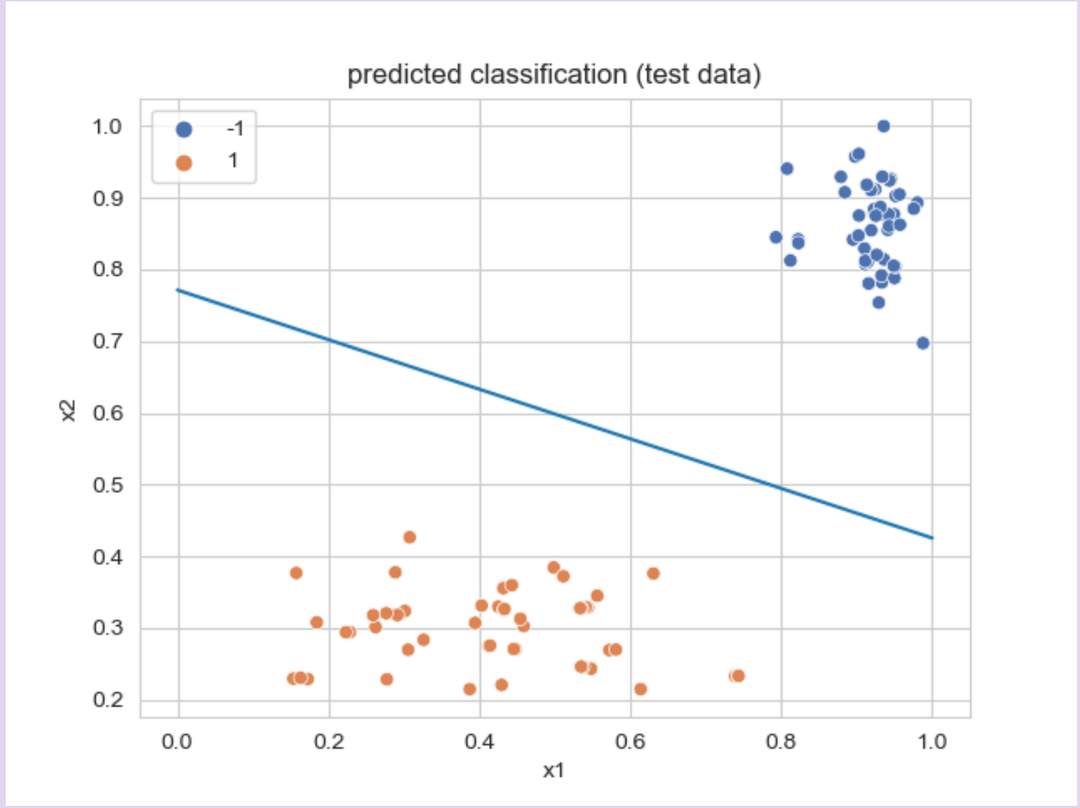


In the plot above, we can see that the fruits group together by classification and that the **observed and predicted classification perfectly match**. In the predicted classification, we also see the **decision line** which separates the two kinds of fruits.

For the test data set, the model once again successfully classified the fruits. Here, 100 test data were used (50 apples, 50 bananas).

For the test data set, the **accuracy score is 1.0**. The confusion matrix is also shown below.

	1 (predicted)	**-1 (predicted)**
1 (observed)	50	0
(observed)	0	50



Results and Analysis

Logistic Regression

For the training of the perceptron model, I have **500 input data** with **2 features** x_1 and x_2. The **x_0** represents the bias input. The 500 data are split evenly between two fruits: 250 ripe bananas and 250 rotten banana. **Ripe bananas are assigned 1, while rotten are assigned -1.**

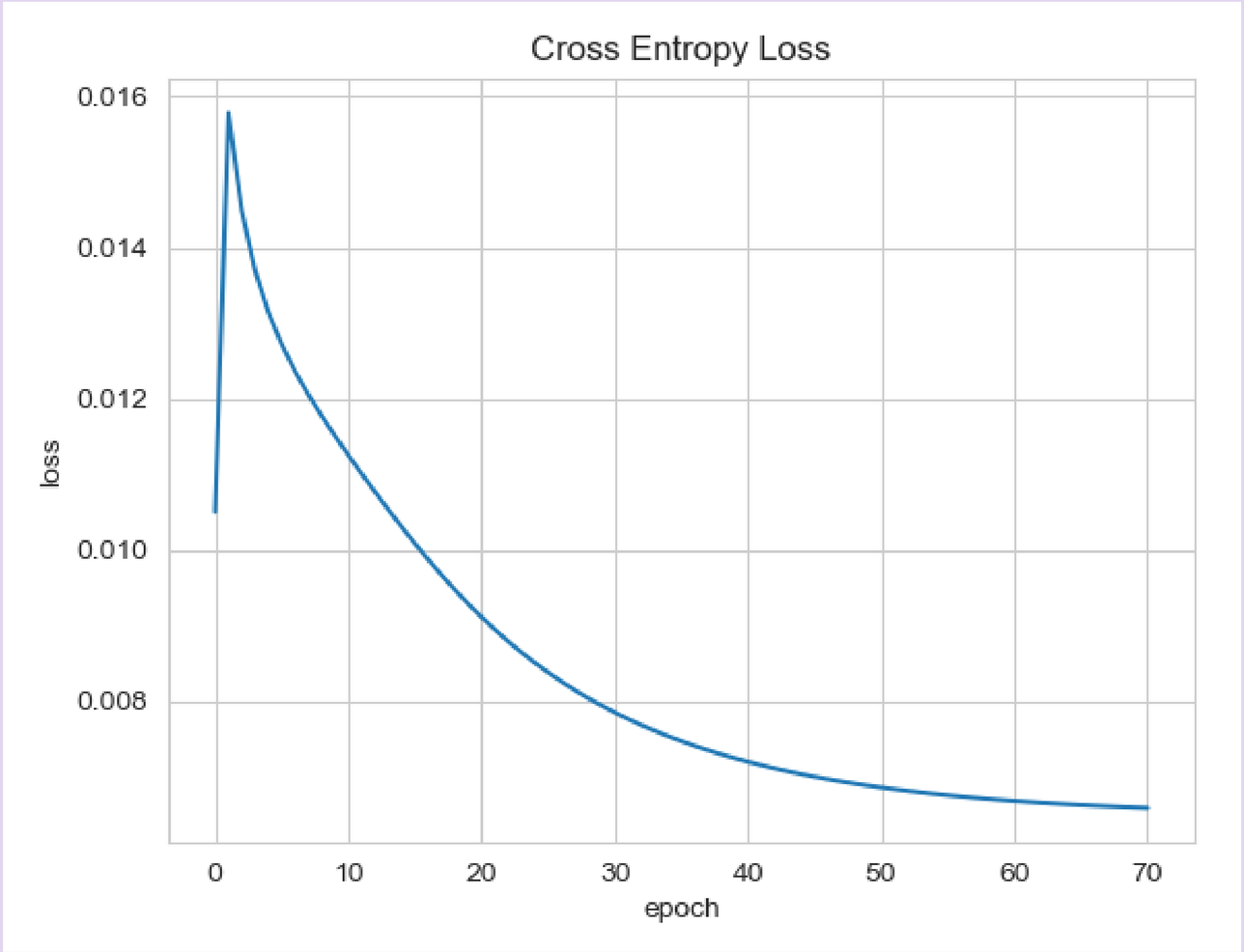
The features used are the fruit images' **average in their green and red channels**.

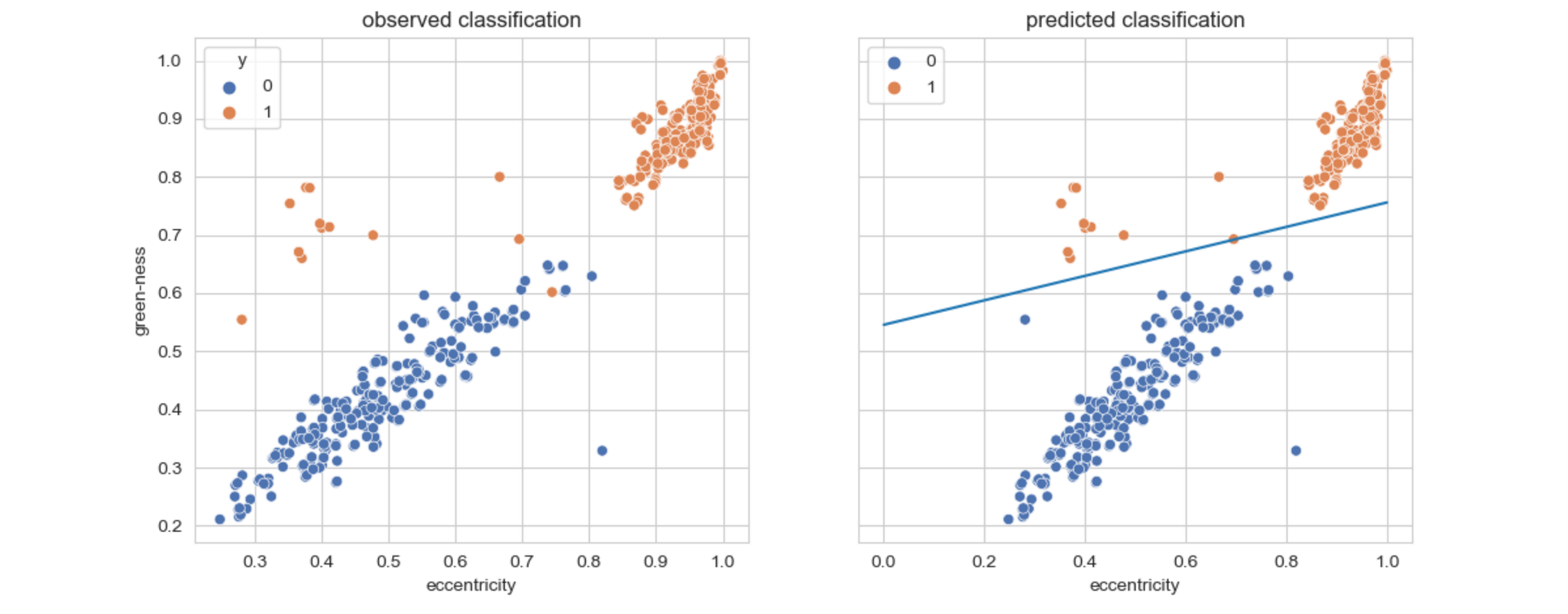
	type	x0	x1	x2	y
0	ripe	1	0.975462	0.943187	1
1	ripe	1	0.955542	0.921663	1
2	ripe	1	0.951425	0.881859	1
3	ripe	1	0.995674	0.998909	1
4	ripe	1	0.954293	0.851984	1
...	...	...	...	...	...
495	rotten	1	0.531817	0.451352	0
496	rotten	1	0.477332	0.425340	0
497	rotten	1	0.542857	0.464335	0
498	rotten	1	0.550458	0.549445	0
499	rotten	1	0.531512	0.522295	0

The figure to the right shows the **cross entropy loss per epoch.** We can see that **the loss decreases as the epoch increases** on the training data. This means the model performed extremely well in classifying the fruits.

The model has an **accuracy of 0.996** and the following confusion matrix:

	1 (predicted)	**0 (predicted)**
1 (observed)	248	2
0 **(observed)**	0	250



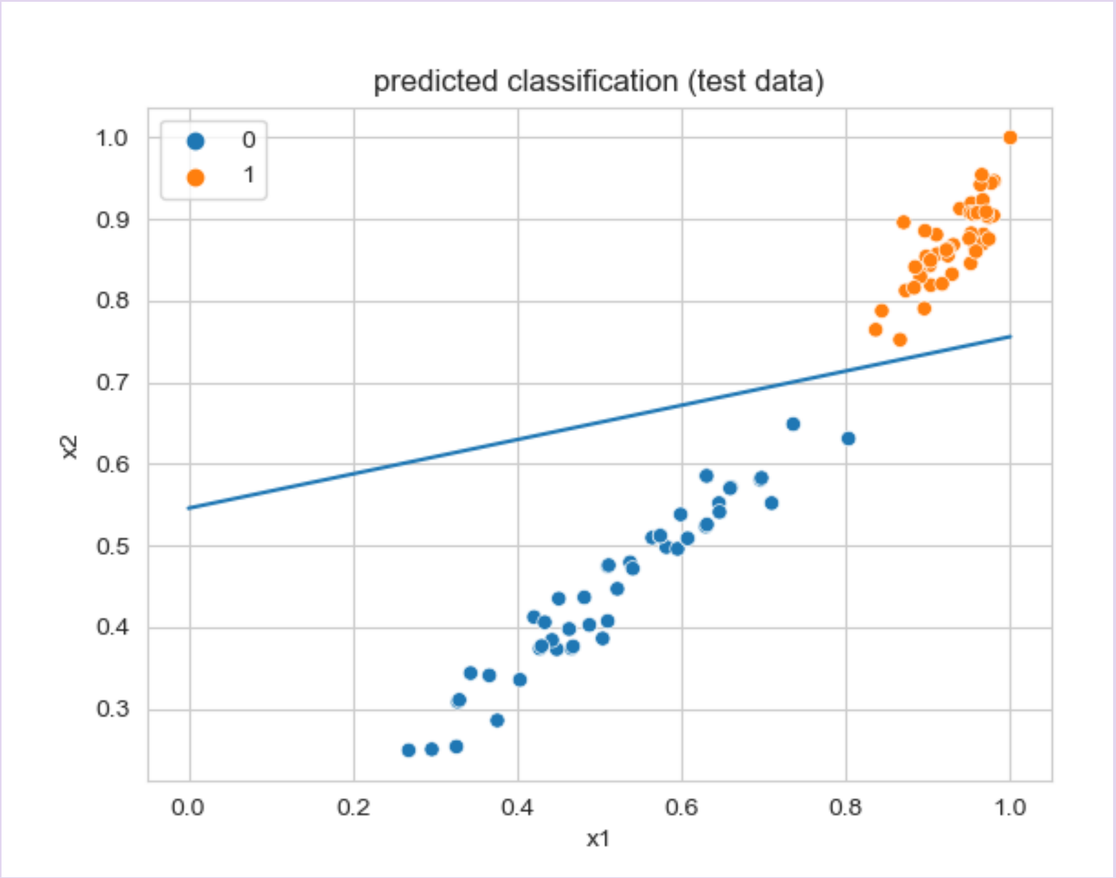


In the plot above, we can see that the fruits group together by classification and that the **observed and predicted classification closely match**. In the predicted classification, we also see the **decision line** which separates the two ripe and rotten fruits.

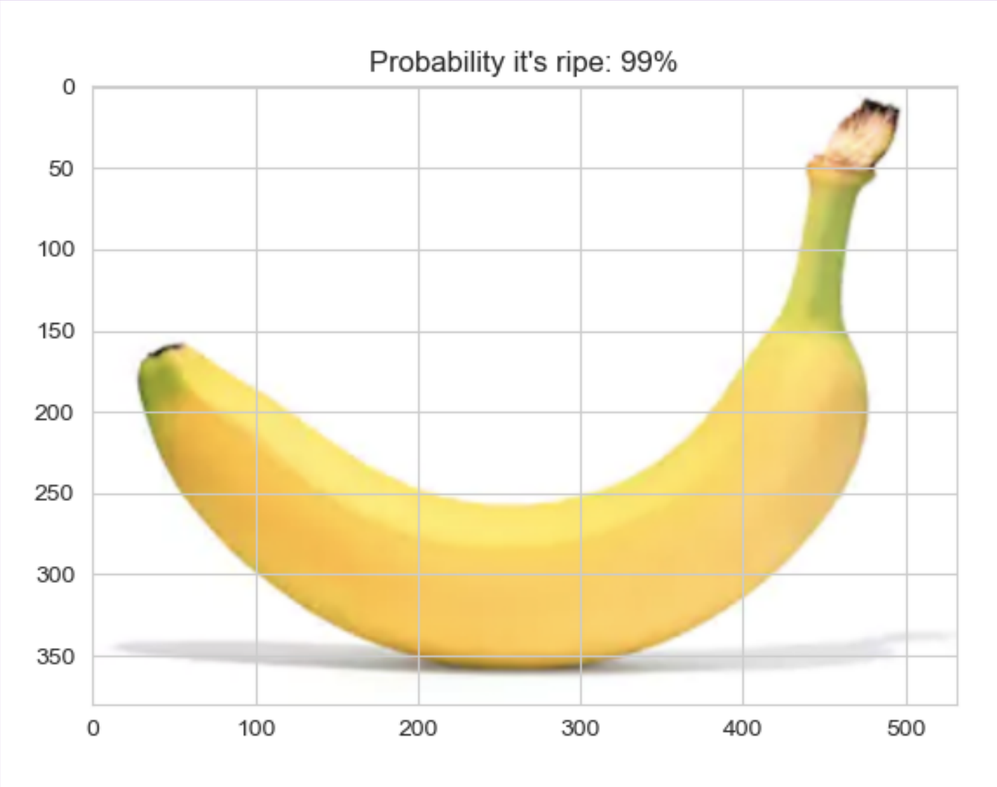
For the test data set, the model once again was very successful in classifying the fruits. Here, 100 test data were used (50 ripe, 50 rotten).

For the test data set, the **accuracy score is 1.0**. The confusion matrix is also shown below.

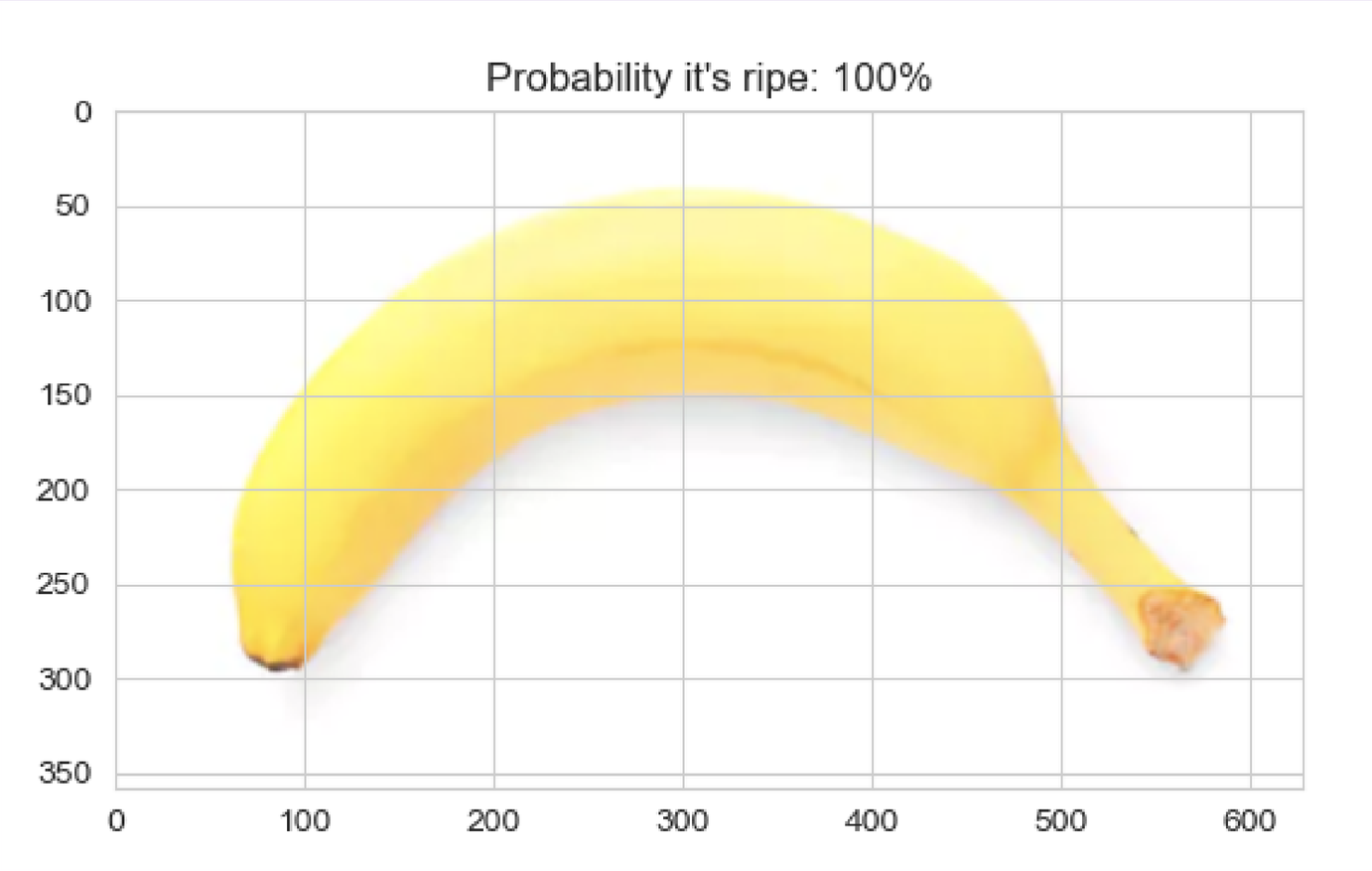
	1 (predicted)	**-1 (predicted)**
1 (observed)	50	0
(observed)	0	50



An example of the model returning the probability that a fruit is ripe

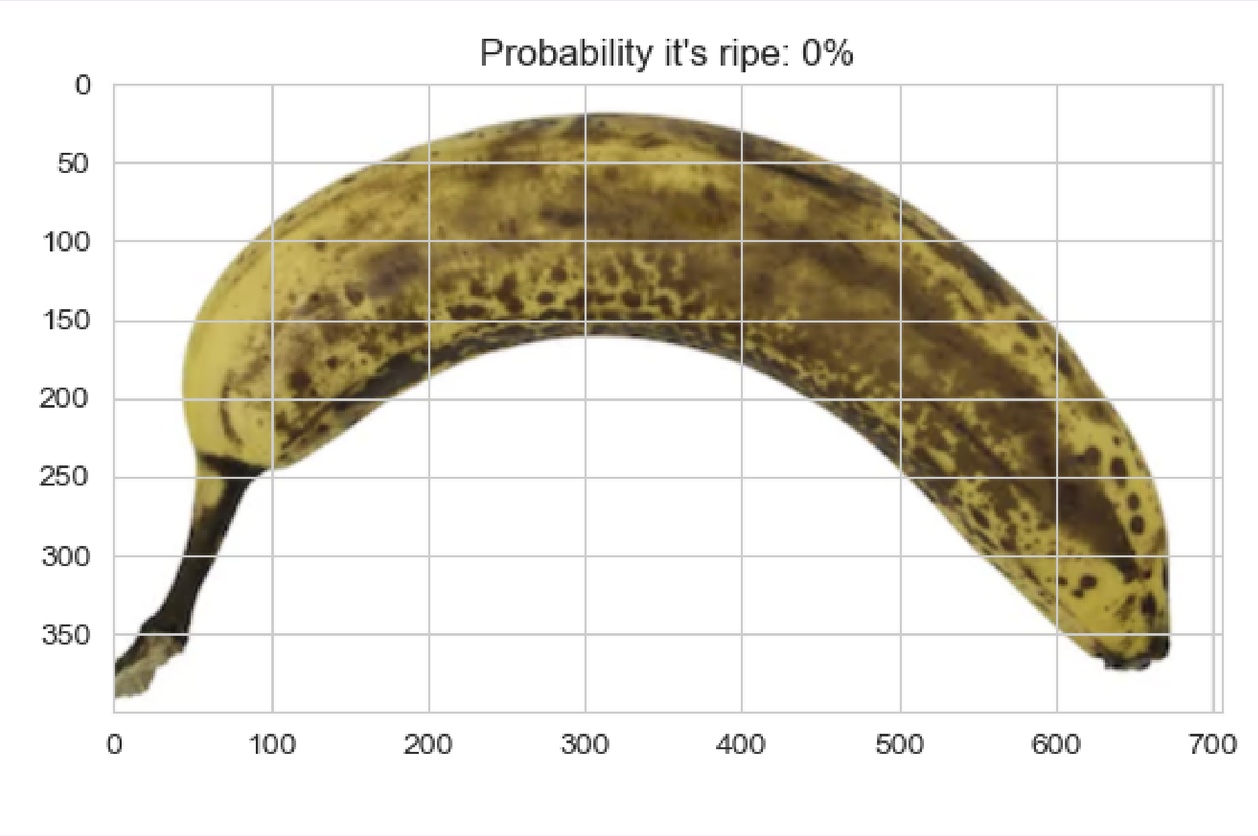


An example of the model returning the probability that a fruit is ripe



Model Peformance - Probability

An example of the model returning the probability that a fruit is ripe

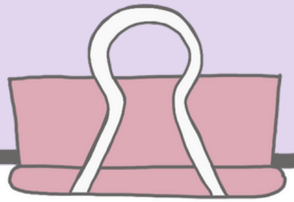


Reflection

The results of this activity are as expected, as shown by the fact that **we have successfully classified the fruits as apples or bananas and as ripe or rotten**. The machine learning models performed very well, with very high accuracy for the training and testing data.

It was very difficult for me to start the activity, mostly because I didn't really know how to begin with the model itself. But it was very satisfying doing the image pre-processing. It was a very smooth process, and I felt like it was a culmination of all the things we've learned so far in this class. I really have to give a big credit and thanks to the tutorials by Simplified Python because it really helped me understand the process of perceptron better, as well as how to code the weight changes and the cross entropy loss.

Over-all, I think I did a really good job in this activity. It was very nice to see the model perform well. And that's why I decided to give myself a perfect score.



Self-grading

- Technical Correctness = 35
- Quality of Presentation = 35
- Self-reflecton = 30
- Initiative = 0

Total: 100

References

I would also like to cite and thank the following sources that I used in this activity!

[1] Soriano, Maricor. (2020). ML2 - Perceptron

[2] Soriano, Maricor. (2020). ML2 - Logistic Regression

[3] Python Simplified. (2021). Perceptron algorithm. Link here.

[4] Python Simplified. (2021). Gradient Descent algorithm. Link here.

[5] Python Simplified. (2021). Cross Entropy Loss. Link here.

[6] Saraswat, Manish. (n.d). Practical Guide to Logistic Regression. Link here.

[7] Fruit images are retrived from kaggle. Link here.