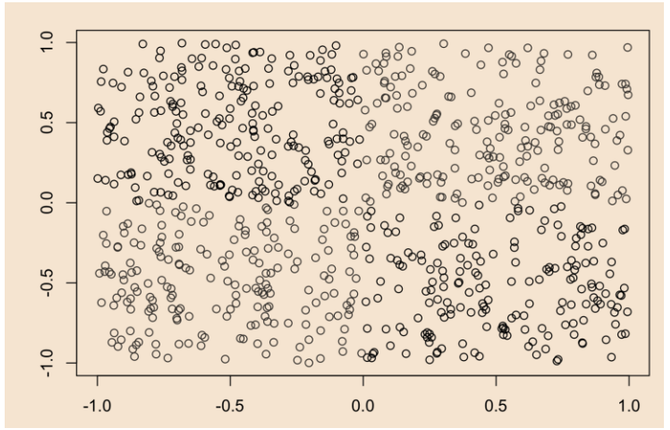
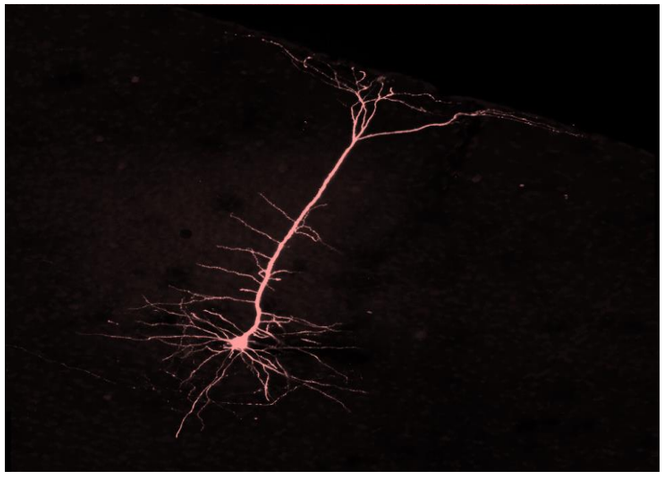
BINARY CLASSIFICATION

MACHINE LEARNING - MODULE 2



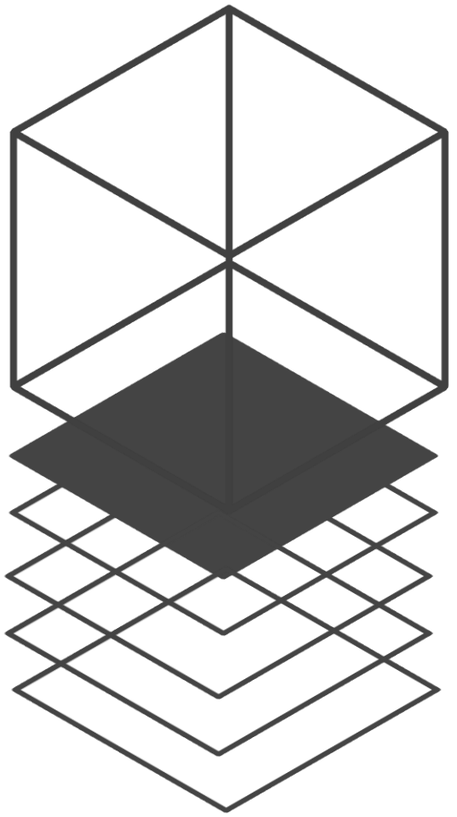


NINO PHILIP RAMONES | GITHUB
2020 - 05616

JULY 3, 2023

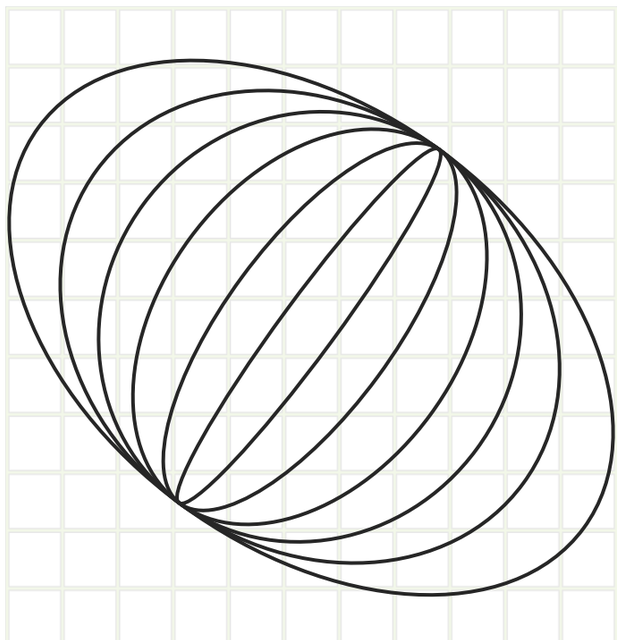
OBJECTIVES

- Explore the different binary classifiers such as perceptron and logistic regression on classifying or differentiating two datasets together
- Perform the basics of perceptron learning and logistic regression algorithms



KEY TAKEAWAYS

- Binary classifiers can be applied to any linearly separable data
- Higher learning rate results to faster weight change at the risk of overshooting and not finding a good solution as the algorithm converges to a local minimum in the shortest time possible
- Lower learning rate means less risk of overshooting, but more iterations are needed to find a good value for convergence
- Different activation functions can be used to train the model



SOME PITFALLS

- Large number of datasets can yield to a longer runtime and training of the model as the algorithm is somewhat numerically sensitive to the learning rate value

BINARY CLASSIFIERS

On the interesting models of supervised learning of binary classifiers

Perceptron learning. The perceptron classification algorithm assumes that the data points are **linearly separable** wherein a hyperplane separates clusters of data. In principle, it is activated by a **step function**, i.e. it only returns 0 and 1 which will dictate to which class the object belongs to. The **weights** are chosen randomly at first and continually refined by iterating through the samples for a number of epochs.

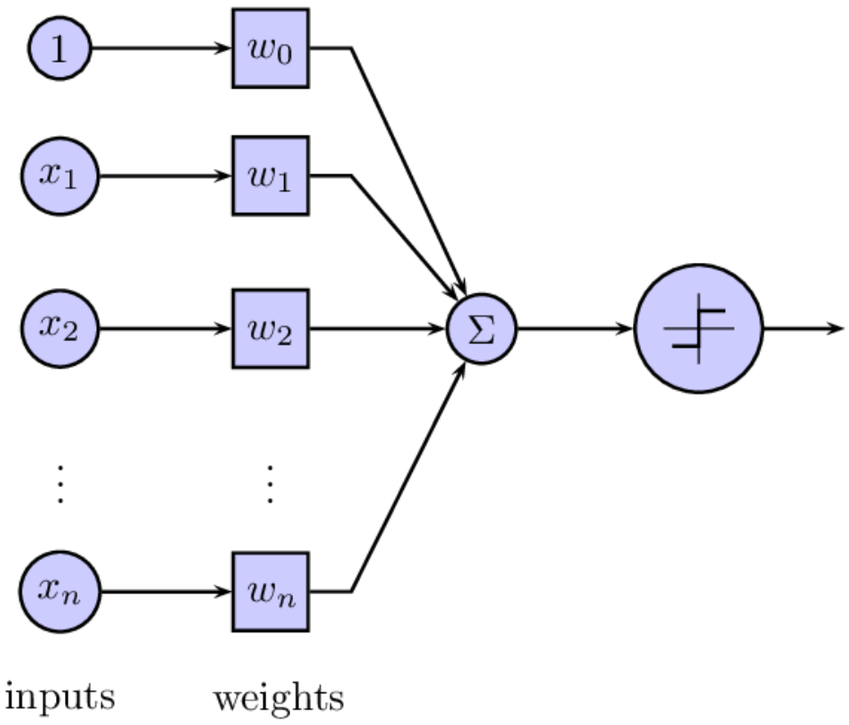


Figure 1. This Photo by Unknown Author is licensed under CC BY-SA.

Logistic regression. The logistic regression runs in the **same principle** as the perceptron except that instead of returning two discrete value of 0 and 1, the output ranges between 0 and 1 by modifying the activation function to be a **sigmoid**. The output is interpreted as the **probability** that a certain data point belongs to a class.

FRUITS DATASET

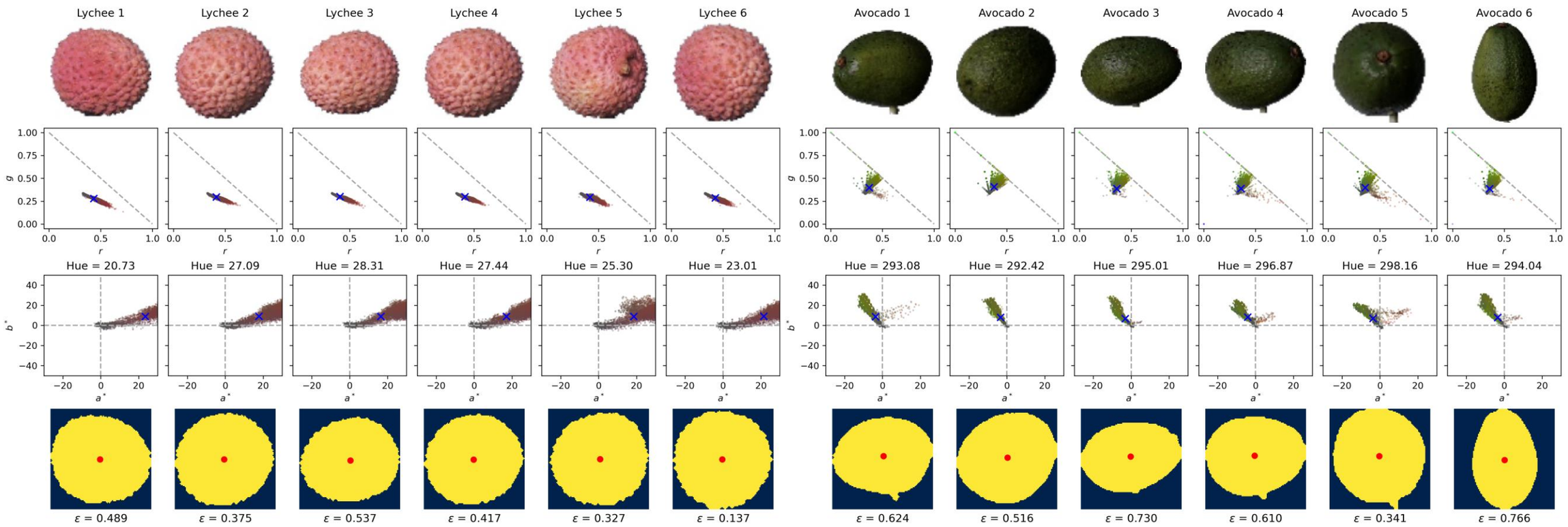
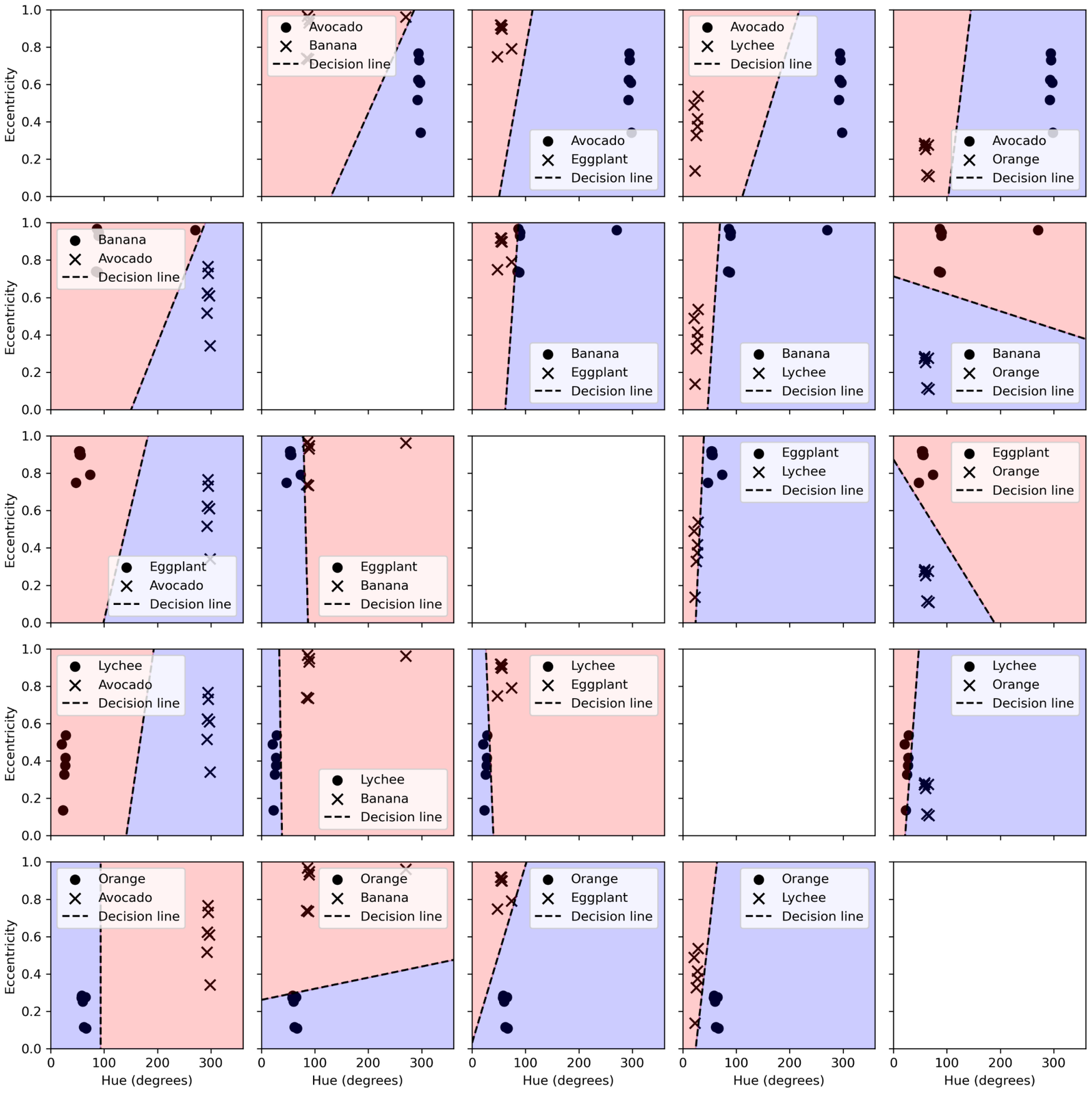


Figure 2. Feature extraction of some fruit images from Kaggle. The mean Lab from CIELab color space were deduced from the mean RGB of the fruits. Eccentricity data was obtained using regionprops.

We revisit some of the previous algorithms used for **feature extraction** in the previous activities. By converting the **RGB color space to CIELab color space**, the hue of the fruits were obtained using

$$\text{Hue} = \tan^{-1} \frac{b^*}{a^*}$$

where a and b denote the **chroma** of the fruit from green to red along the a-axis, and blue to yellow for the b-axis. The **eccentricities** were calculated upon thresholding and labelling using **regionprops**.



A sort of **adjacency matrix** that demonstrates the classification between two fruits were done through the **perceptron learning algorithm**. In this activity, five different fruits were tested, and their **eccentricities and hues** were used as the classifying parameter to distinguish two fruits together. Since the data are mostly **linearly separable**, the perceptron was trained with **optimal convergence** for around 1000 epochs and a learning rate of 0.01.

Figure 3. The decision lines generated by the perceptron algorithm that distinguishes two fruits together.

STAR CLASSIFICATIONS

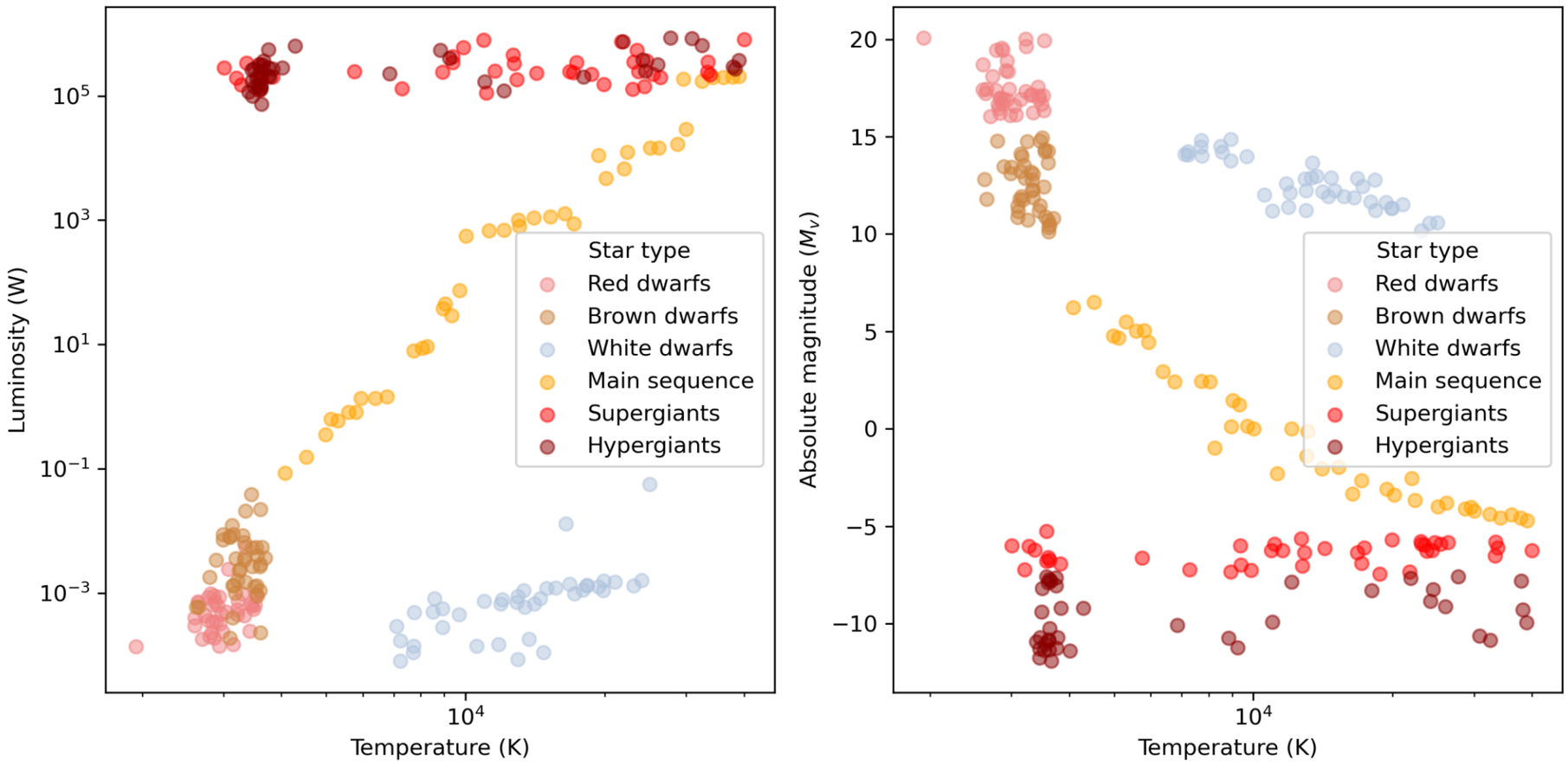


Figure 4. Star dataset to predict star types from Kaggle. The average luminosity and absolute temperature of different stars were plotted to resemble the Hertzsprung-Russell diagram that shows the relationship between the two mentioned quantities.

From the outsourced data of **different stars** in Kaggle, the **absolute temperature and luminosity of stars** were observed. The **Hertzsprung-Russell diagram** shows the distribution of such stars across the **celestial space** based on their temperatures and luminosities. We can then train the **perceptron** to classify the data and see how the decision lines compare for a data that are **not linearly separable** in some cases!

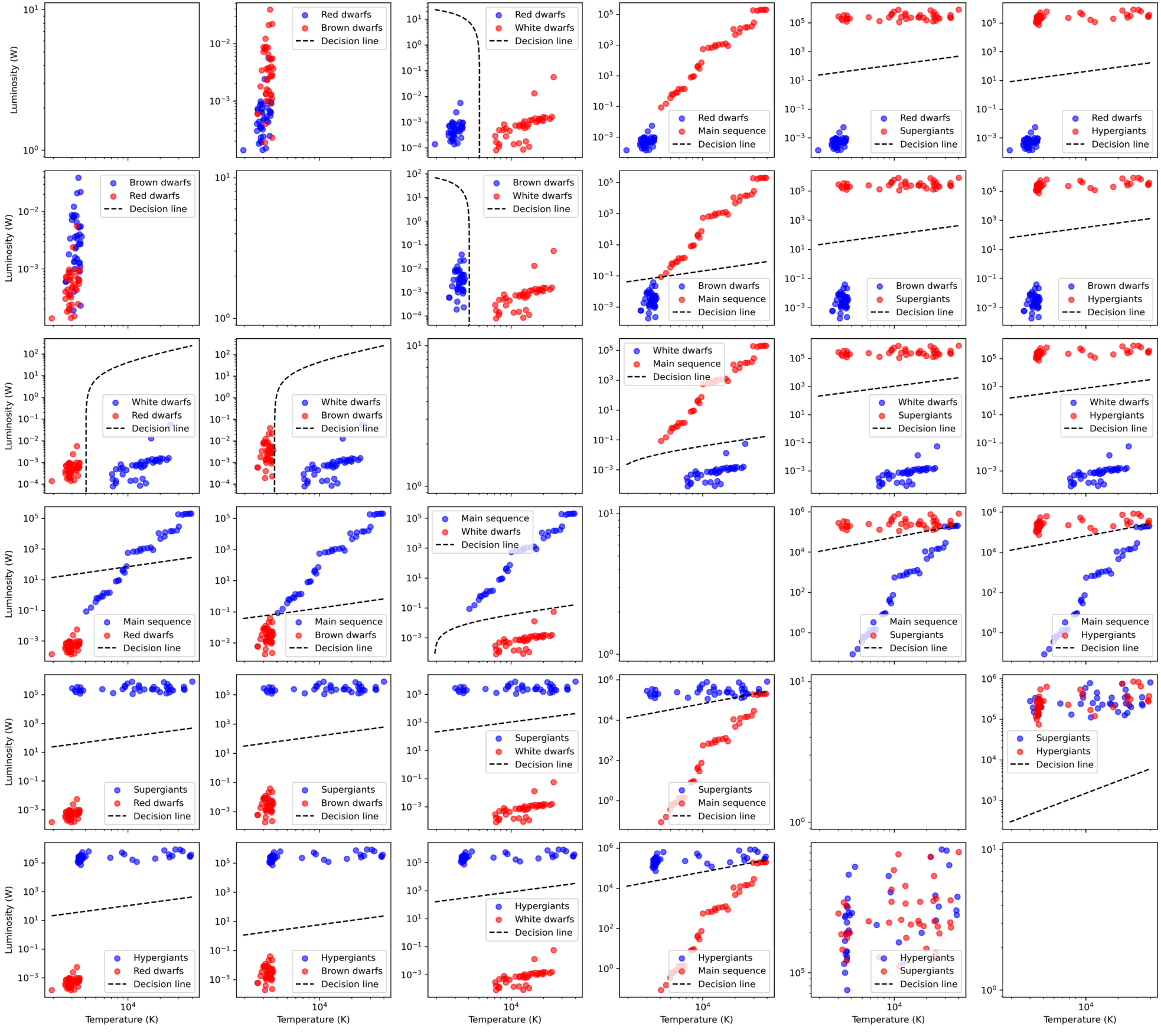


Figure 5. The decision lines generated by the perceptron algorithm that distinguishes two star types together.

Training the perceptron to classify the stars generated a decision line to most of the star pairs. Although, **convergence were not guaranteed** on some cases where the data are **not linearly separable** such that of the brown dwarf vs red dwarf comparison. Since the dataset is generally huge, **higher iteration counts** were needed and a slight **adjustment to the learning rate** – slightly higher lower than 0.01 – were made.

RIPENESS PREDICTION

By modifying the **activation function** from the perceptron learning to a **sigmoid**, we can then obtain a **probabilistic output** that determines the likelihood of an object or feature to belong in a class. One of the most interesting characteristic of **logistic regression** is the ability to train a model to determining the **ripeness of a fruit** based on its color. The model was trained by feeding the **mean RGB** of ripe and unripe tomatoes and iterating throughout the training samples until convergence is achieved and the desired weight is drawn!

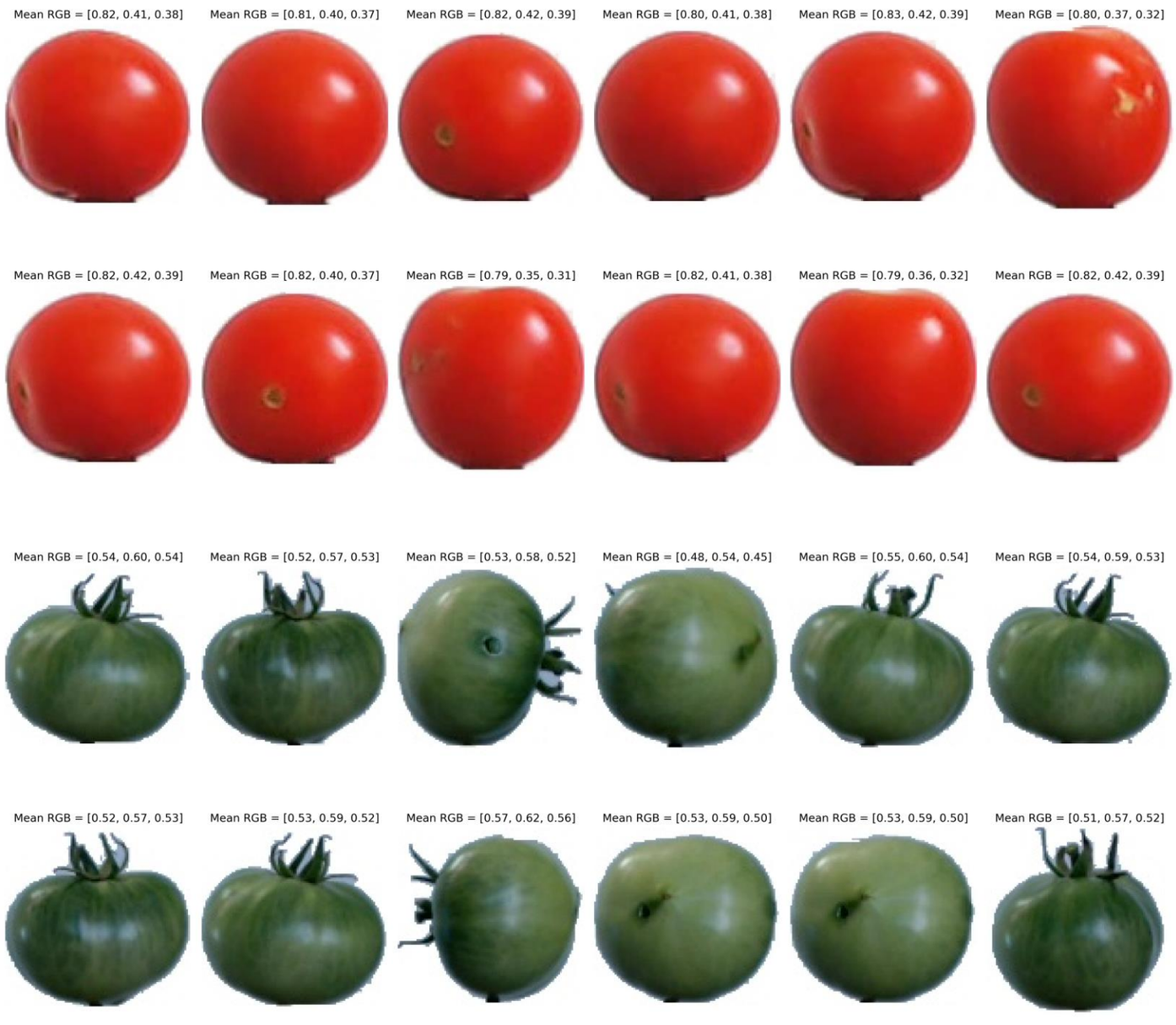


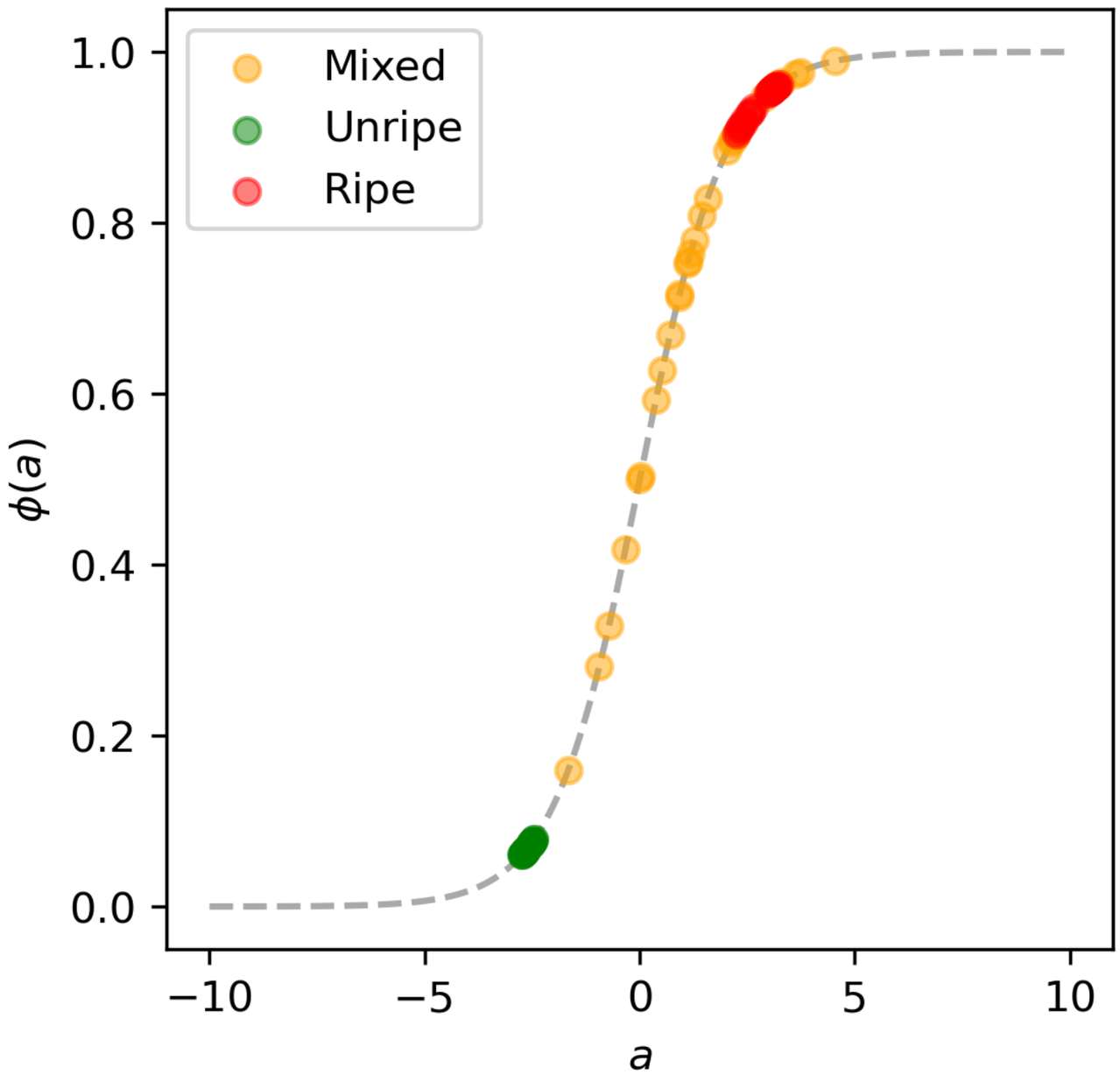
Figure 6. Images of ripe and unripe tomatoes from Kaggle. The sample images were used as training data in determining the ripeness of a fruit of varying shades between green (unripe) and red (ripe).

RIPENESS

Using different images of tomatoes of varying colors or degree of ripeness, the output (ripeness percent) was obtained by feeding the final weight from the training data and the mean RGB of every image of the fruit to an algorithm, i.e. the dot product of the padded feature vector and weight vector is the probability that the fruit is riped or it belongs to the riped class!

Figure 7. Images of different tomatoes of varying color shades to determining the degree of ripeness. The model was trained at around 2000 epochs at a learning rate of 0.01.

SIGMOID OUTPUTS



Plotting the **activation function** and the corresponding outputs after the **weight change** from the trained model, it can be inferred that the mixed images of tomatoes falls in between the **unripe and ripe thresholds**. Some **outliers** beyond the ripe training data were observed, which can be attributed to the **presence of other colors** in the image as the training images were isolated compared to the actual prediction data samples.

Figure 8. The input *a* and sigmoid output $\phi(a)$ plotted against each other for the training and prediction dataset.

WILDFIRE PREDICTION

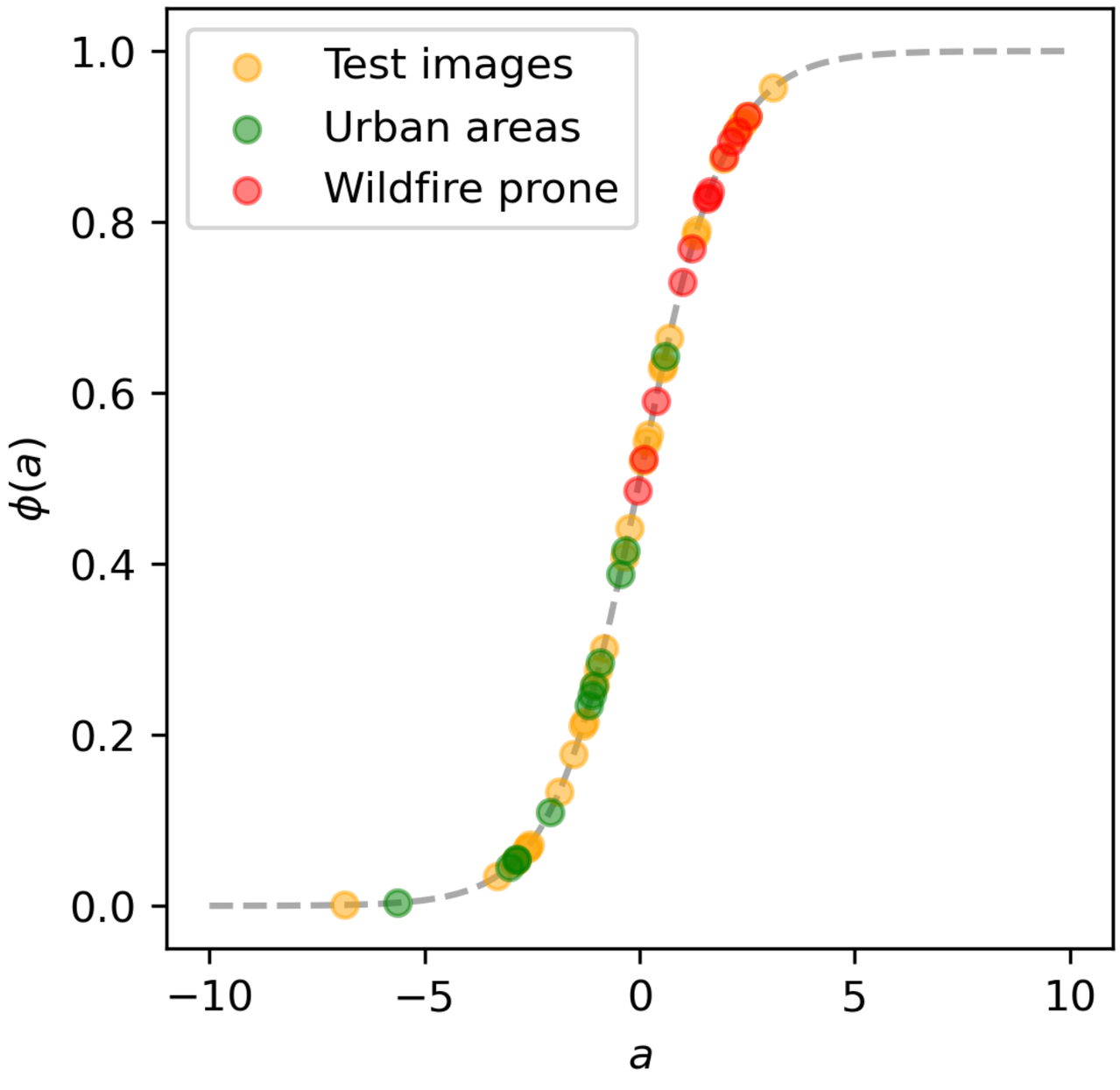
Interestingly, we also examined some **satellite images of urban and rural area** captures from Kaggle around some areas in Canada that have or may have not experienced a wildfire before. Extracting the **mean RGB** of the individual images, the model was trained to determine whether an area is at **risk to a wildfire** based on a **likelihood basis**. Since rural areas tend to be more dominated by lush greens compared to the pavement-dominated urban areas, the mean RGB can be a **reasonable feature** to consider although the accuracy of the training data can be quite subpar as the presence of greenery is also seen in urban areas as shown.

Figure 9. Satellite images of urban and rural areas from Kaggle. The sample images were used as training data in determining the likelihood of an area to experience wildfire using the mean RGB value of the scenes.

Images were randomly chosen from a **prediction dataset**, and the resulting likelihood values were reasonable enough to check whether an area or scene is **at risk to a wildfire**. The regression algorithm was still able to yield a sound likelihood although **neural networks** are much more designed in feeding images like these and decide which areas are at risk. Perhaps, using the mean RGB of the images could work to train the model, but not the most ideal sort of!

Figure 10. Likelihood of different satellite captures to catch a wildfire using logistic regression. Model was trained at ~2000 epochs at a learning rate of 0.01.

SIGMOID OUTPUTS



Examining the sigmoid function and prediction data outputs, the test image points were quite **scattered** across the **urban and wildfire prone area thresholds**, which is indicative that the **mean RGB** of the training data were **not the most ideal data** to use. The results were reasonable, but these can be further improved by increasing the **number of training data images** or increasing the number of iterations at the expense of a much lower learning rate.

Figure 11. The input a and sigmoid output $\phi(a)$ plotted against each other for the training and prediction dataset.

REFLECTION

I was able to enjoy the activity as it served as an introductory course and test our understanding on machine learning in general. The application of the binary classification algorithms to the real-world datasets were fulfilling as the concept of training a model through perceptron learning and regression made more sense. Overall, I would give myself a score of **110/100** !

REFERENCES | GITHUB

1. M. Soriano, Applied Physics 157 – Machine Learning, 2023.
2. The CIELAB L*a*b* System – the Method to Quantify Colors of Coatings - Prospector Knowledge Center (ulprospector.com)
3. Fruits 360 | Kaggle
4. Star dataset to predict star types | Kaggle
5. Wildfire Prediction Dataset (Satellite Images) | Kaggle