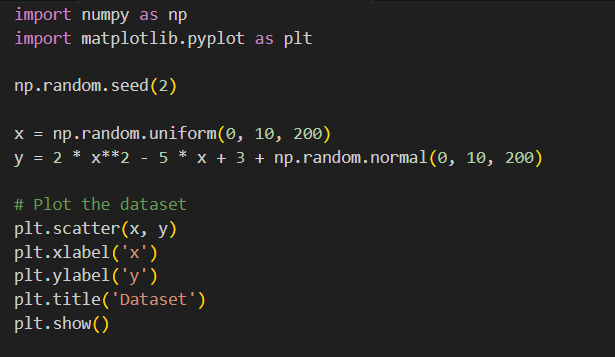
# Homework exercise week 6

# GitHub Instructions

1. Use the same repository as in week 2 and 4 (ds5\_assignment\_ group#), make sure all the group members and teacher are still collaborators.
2. Create one .ipynb (or .py) file that contains the answers to all the exercises! However, you can call other files in your main file.
3. Make sure that you push and commit all your changes to the file before the deadline. If changes are made after the deadline, it will not be taken into account for the bonus point.

# Exercise 1: Fitting a model

You have been given the following code, which generates data for x (independent variable) and y (dependent variable).



For this exercise, you need to fit a regression model to the data. To do so, perform the steps below. But before you start, make sure to divide exercises b-f among yourselves and work in parallel. Clearly describe in the main.ipynb who was responsible for which task.

1. Copy the following code to your own script
2. Split the data: Divide the dataset into a training set and a test set.
3. Plot the data (this code is already given) and look at the plot, determine which relationship exists between the variables and define one of the regression models learned in this course
4. Train the model: fit the model to the training data
5. Evaluated the trained model by the and the
6. Evaluate the model: Assess the performance of the trained model on the *test* data by the and the
7. Refine and iterate: If the model's performance is not satisfactory, you may need to choose a different model. Iterate this process until you are satisfied with the model's performance.

# Exercise 2: Predicting the quality of wine

For this exercise you need the ‘winequality-red.csv’ file. You will work with the winequality-red dataset to build a model that can predict the quality of red wine based on its chemical attributes. The winequality-red dataset contains various physicochemical attributes of red wine samples, along with their associated quality ratings. The dataset consists of 1599 samples and 12 columns, including attributes like fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality.

For this exercise you need to build a predictive model that can estimate the quality of red wine based on its chemical attributes. The quality rating ranges from 3 to 8, with higher values indicating better quality. To do so, perform the steps below. But before you start, make sure to divide the exercises among yourselves and work in parallel. Clearly describe in the main.ipynb who was responsible for which task.

1. Load the winequality-red dataset using any suitable tool or library for data manipulation (e.g., pandas).
2. Explore the dataset to understand its structure, column names, and data types.
3. Handle any missing values or outliers, if present, like you learned in week 4.
4. Visualize the distributions of the wine quality ratings and the relationships between the attributes and the quality.
5. Split the dataset into a training set and a test set. You can adjust the split ratio based on the dataset size and requirements.
6. Select an appropriate model to train on the training set based on step d).
7. Train the chosen model on the training data.
8. Evaluated the trained model by the
9. If your chosen model is not performing well, choose a different model and repeat step g and h. You only have to do this once, so that you will have two models in total.
10. Compare the performance of your two models and select the best performing model
11. Discuss the limitations of your models and any potential improvements or further experiments that can be done.

Note: You can use the material learned in class, or you can use the internet to try various machine learning libraries such as scikit-learn or TensorFlow to implement the models and perform the necessary steps. Feel free to explore different algorithms, feature engineering techniques, and evaluation metrics to enhance the model's performance (but this is not course material).

# Exercise 3: Malaria detection pipeline

*Note: this exercise is combines both pre-and post-processing techniques and the train/test method.*

But before you start working out the exercises below, make sure to create tasks and work in parallel. Clearly describe in the main.ipynb who was responsible for which task.

For automating the diagnoses of malaria parasites in blood cells, scientists want to make a model that detects and localises malaria in images. They have gathered 1328 images with multiple blood cells of which some of them are infected by the malaria parasite. An example can be found in the figure below.

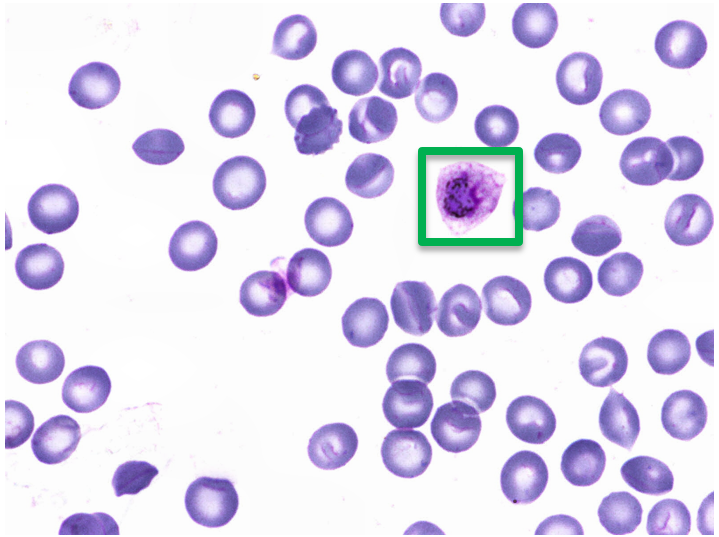


Figure 1: Example of blood cell images with an infected malaria image in the green 'bounding box'

In the above image, you can see that a green bounding box is placed on an infected malaria image. This bounding box is defined by the top left coordinate and the bottom right coordinate. An example is given in figure 2.

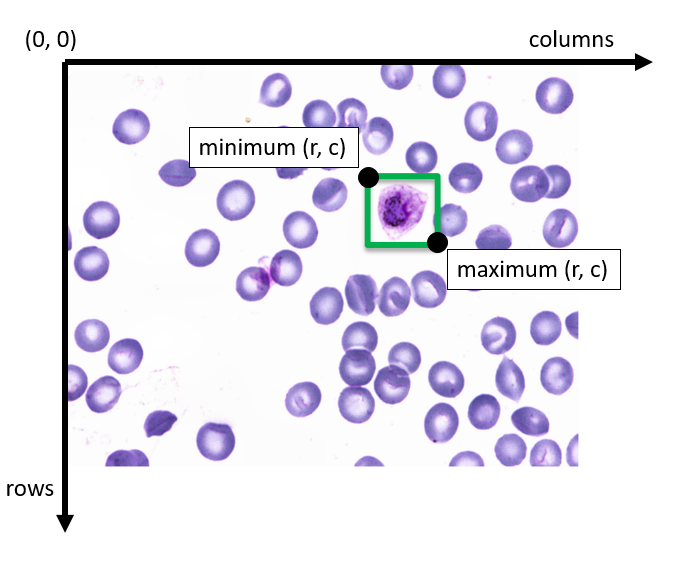


Figure 2: Example of bounding box definition. r = rows and c = columns

For every image in the training data set: ‘training.xlsx’, the coordinates of the bounding box are known. The data looks as follows

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| filename | category | min\_r | min\_c | max\_r | max\_c |
| example.png | 0 | 1050 | 1440 | 1150 | 1540 |

Here, min\_r, min\_c, max\_r, max\_c are the coordinates of the bounding box. For the category, we use 1 if the blood cell is infected with Ring and 0 otherwise.

## Exercise 3.1 Post-processing of predictions

We already used a model to make predictions about the coordinates. The file ‘predictions\_training.xlsx’ has a list of predictions that need to be evaluated.

To evaluate how well the model predicted whether blood cells were infected or not, we use the Intersection over Union (IOU). This is a measure of ‘overlap’ between the actual and predicted bounding box. For both infected and not infected class, we can take the average over all IOU’s to get the mean IOU. The higher the mean IOU, the greater the model. Figure 3 illustrates how to calculate the intersection over union for one pair of bounding boxes.

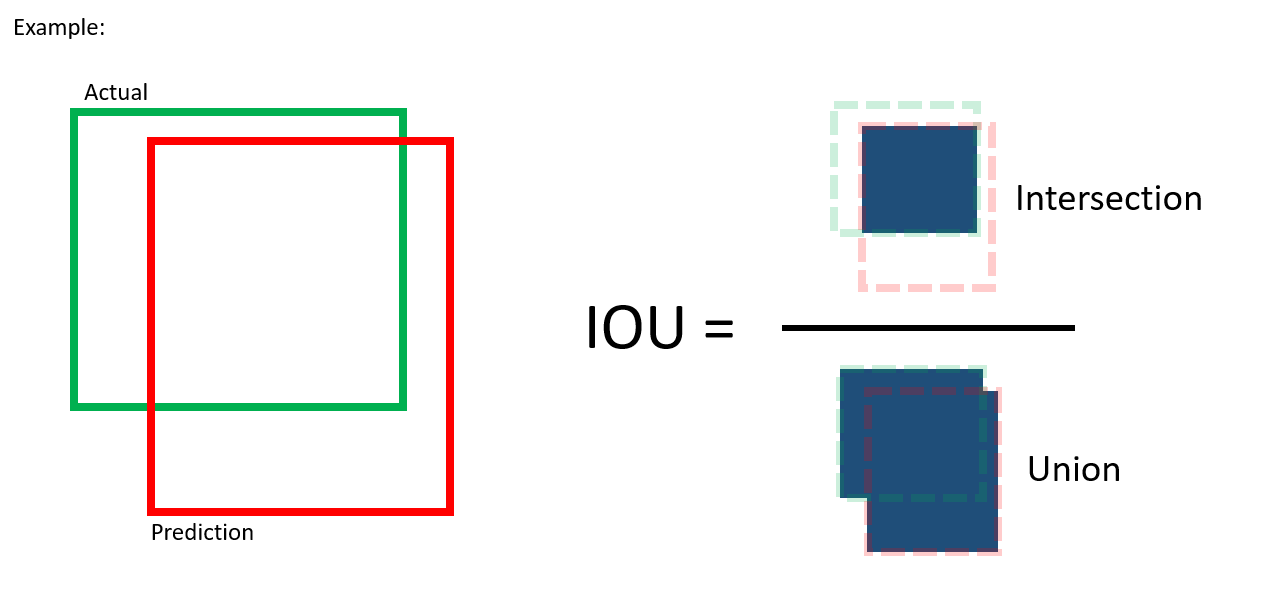


Figure 3: Example intersection over union

For this exercise, you need to calculate the mean IOU of our predictions for the **training data**. To do so, perform the steps below.

1. Create a function that returns the IOU for each box
2. Create a function that calculates the mean of the IOU’s of all the boxes

You may use the following algorithm:

**IOU calculation algorithm**

= original image with dimensions

= empty image (zeros matrix) with same dimensions as

= list of the true bounding boxes

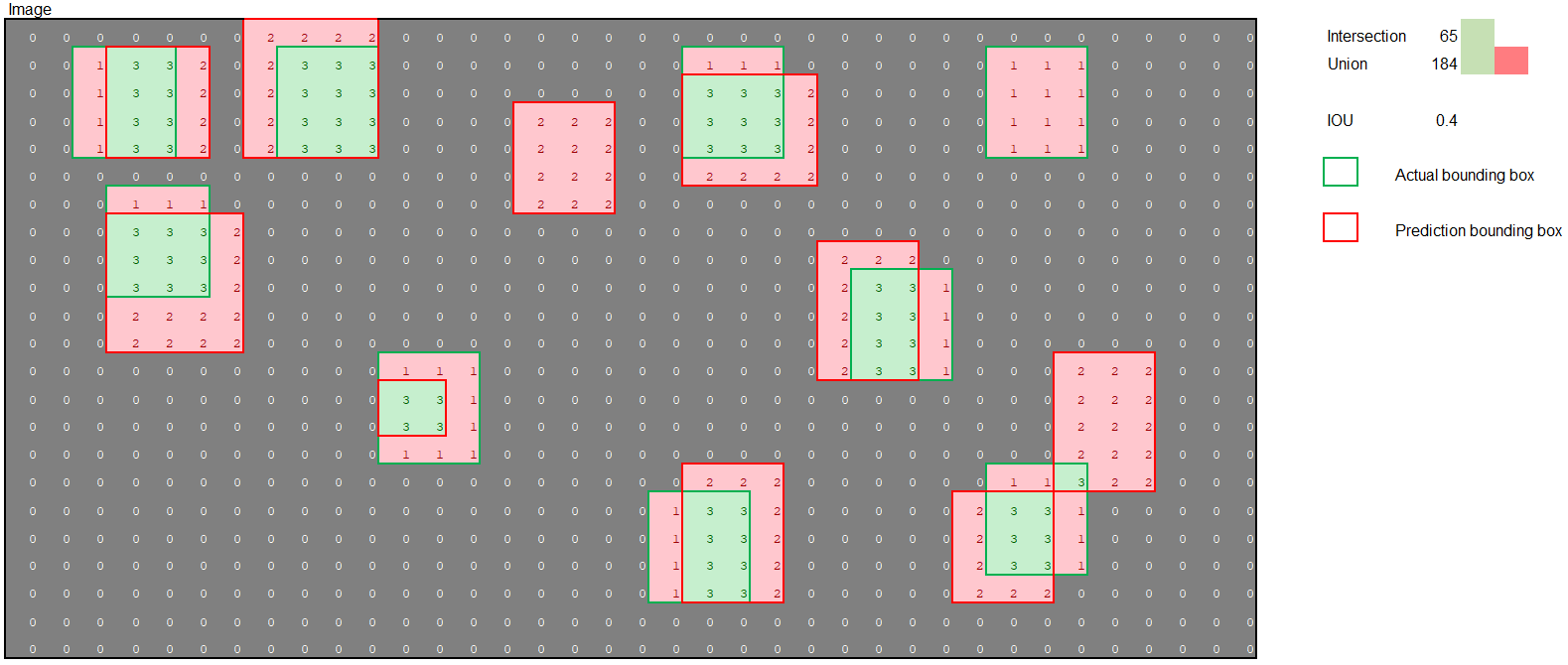
= list of the predicted bounding boxes

**Procedure:**

1. ***For all in :***
2. Fill the region of with
3. **End**
4. **For all in :**
5. Add 2 to the region of with
6. **End**
7. intersection = the number of pixels in with value equal to 3
8. union = the number of pixels in with value 1, 2, or 3

*Note: this is not the actual IOU algorithm but a simplified version. Normally, you would have to match a predicted box to an actual box. But we’ll ignore that for this exercise.*

An illustration might look like this:



In this illustration, the intersections between the actual and predicted bounding box have a green background and all the non-overlapping regions have a red background. The IOU is the number of cells with a green background divided by the number of cells with a green or red background.

## Exercise 3.2 Fitting a model on the IOU

In the previous exercise, you have calculated the IOU for each bounding box to get the mean IOU. Now, we want to fit a model on the IOU, so that we can make predictions of future values. Our independent variable will be the vector with *all the IOU’s for each bounding box* and our dependent variable will be the *category* of the box, which is 1 if the blood cell is infected with Ring and 0 otherwise (see 3.1). In this exercise, we want to fit a model on the **training** data. As linear regression models will not work, we will use a logistic regression model. As you probably have not seen this model before, you can use the following syntax:

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X, y)

For this exercise, fit a **logistic** regression model on the training data and answer the following questions:

1. What is the value of the R squared?
2. Unfortunately, no predictions of the test set are available. Why is it important to test the model using the test set?