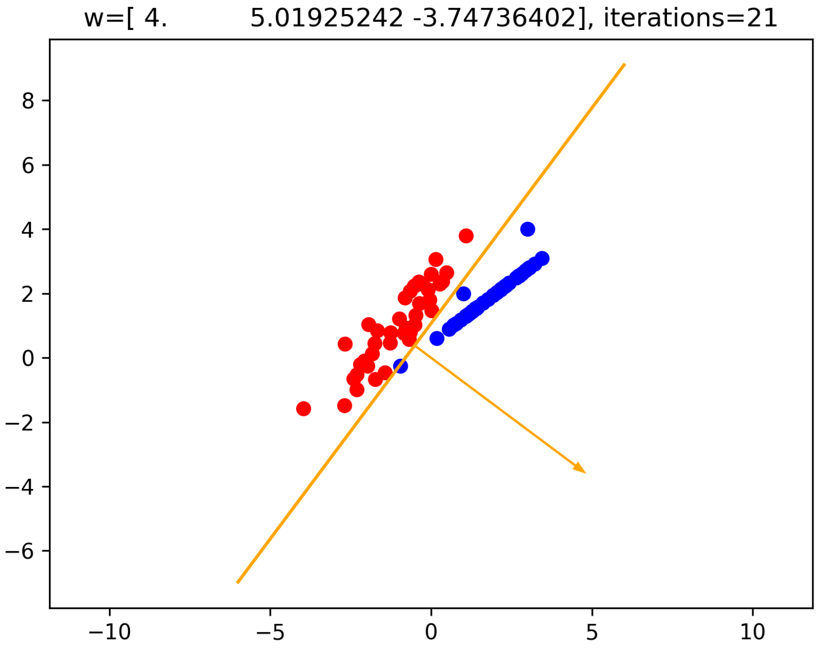
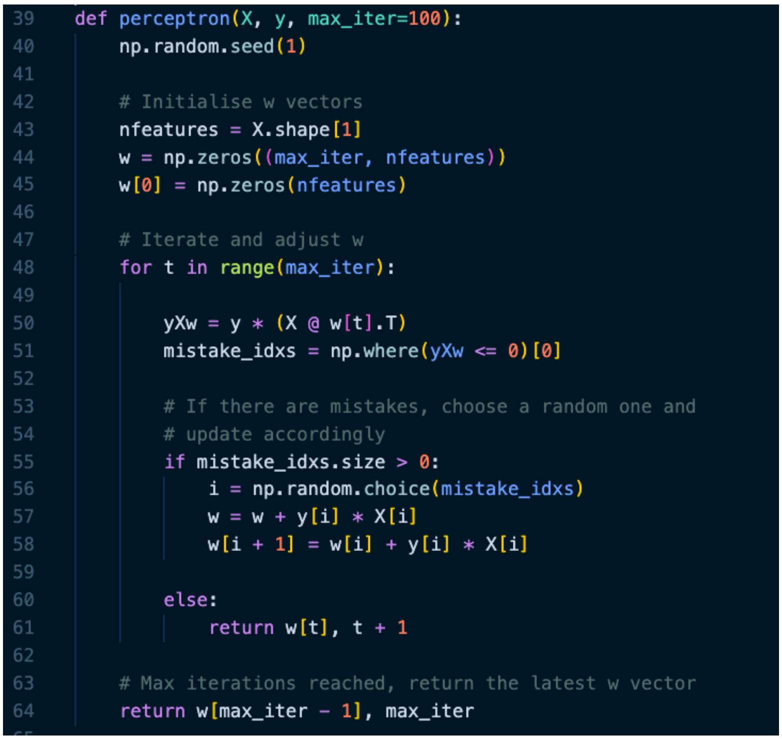
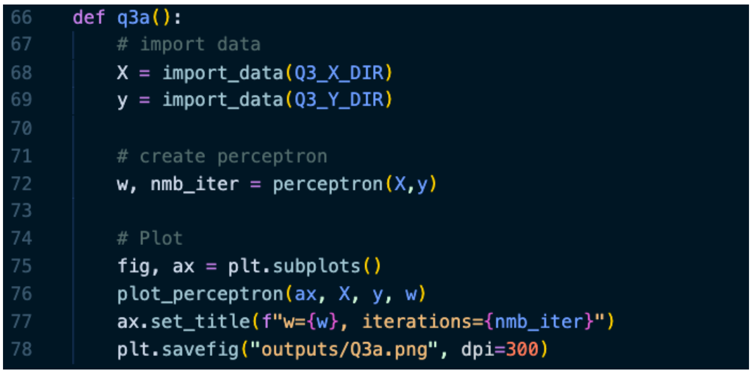
Question 3

a



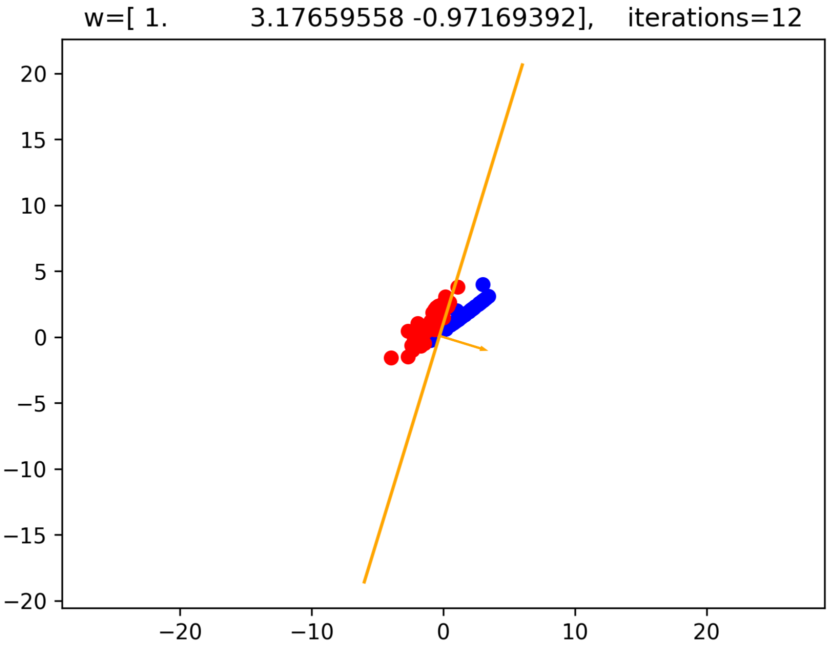


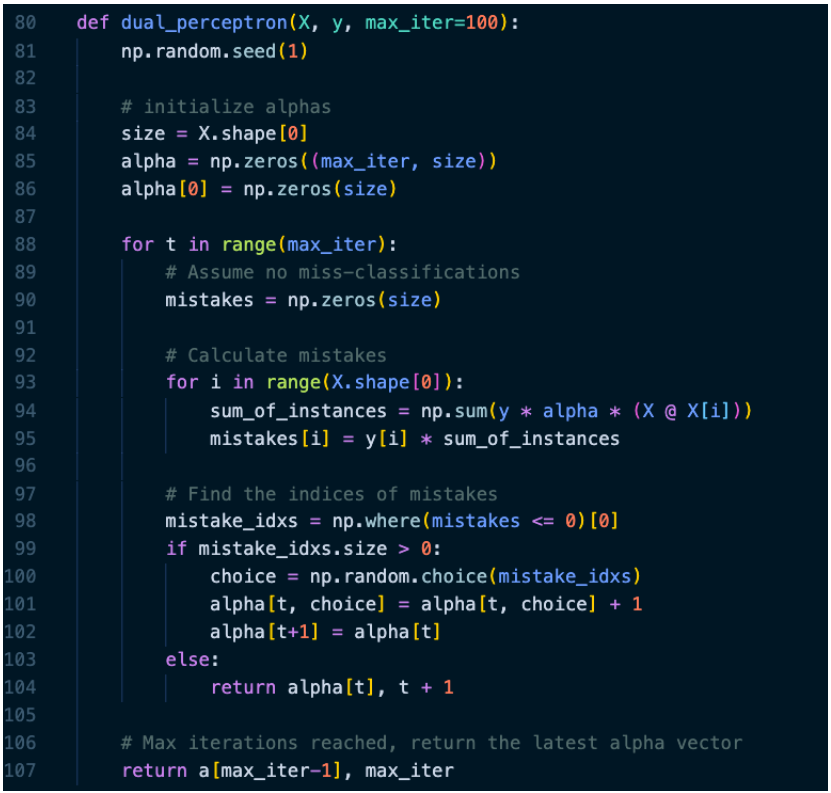
```
def perceptron(X, y, max_iter=100):
    np.random.seed(1)

    # Initialise w vectors
    nfeatures = X.shape[1]
    w = np.zeros((max_iter, nfeatures))
    w[0] = np.zeros(nfeatures)

    # Iterate and adjust w
    for t in range(max_iter):

        yXw = y * (X @ w[t].T)
        mistake_idxs = np.where(yXw <= 0)[0]

        # If there are mistakes, choose a random one and
        # update accordingly
        if mistake_idxs.size > 0:
            i = np.random.choice(mistake_idxs)
            w = w + y[i] * X[i]
            w[i + 1] = w[i] + y[i] * X[i]

        else:
            return w[t], t + 1

    # Max iterations reached, return the latest w vector
    return w[max_iter - 1], max_iter
```



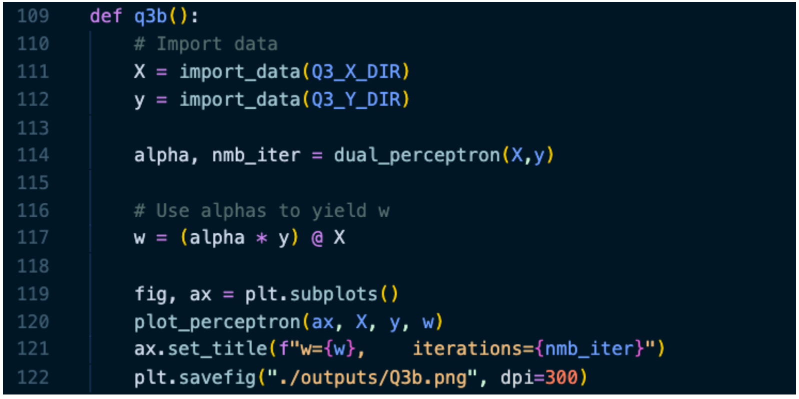
```
def q3a():
    # import data
    X = import_data(Q3_X_DIR)
    y = import_data(Q3_Y_DIR)

    # create perceptron
    w, nmb_iter = perceptron(X,y)

    # Plot
    fig, ax = plt.subplots()
    plot_perceptron(ax, X, y, w)
    ax.set_title(f"w={w}, iterations={nmb_iter}")
    plt.savefig("outputs/Q3a.png", dpi=300)
```

b



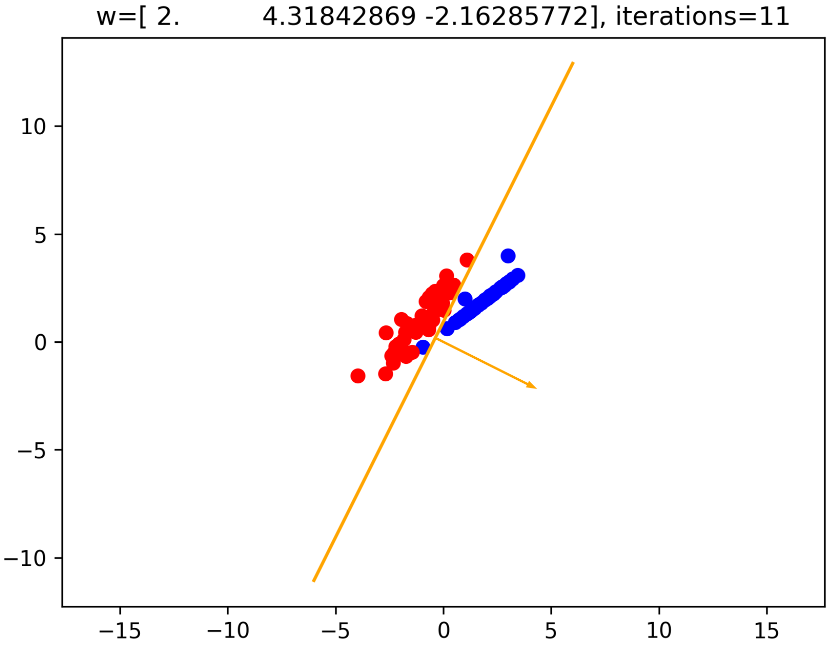


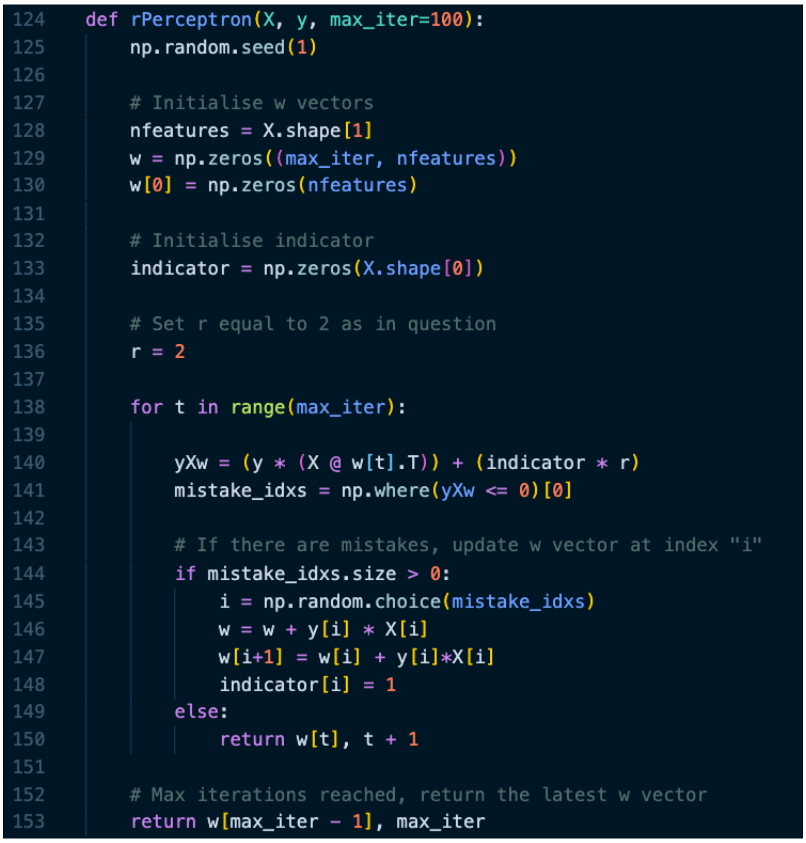
```
def dual_perceptron(X, y, max_iter=100):
    np.random.seed(1)

    # initialize alphas
    size = X.shape[0]
    alpha = np.zeros((max_iter, size))
    alpha[0] = np.zeros(size)

    for t in range(max_iter):
        # Assume no miss-classifications
        mistakes = np.zeros(size)

        # Calculate mistakes
        for i in range(X.shape[0]):
            sum_of_instances = np.sum(y * alpha * (X @ X[i]))
            mistakes[i] = y[i] * sum_of_instances

        # Find the indices of mistakes
        mistake_idxs = np.where(mistakes <= 0)[0]
        if mistake_idxs.size > 0:
            choice = np.random.choice(mistake_idxs)
            alpha[t, choice] = alpha[t, choice] + 1
            alpha[t+1] = alpha[t]
        else:
            return alpha[t], t + 1

    # Max iterations reached, return the latest alpha vector
    return a[max_iter-1], max_iter
```



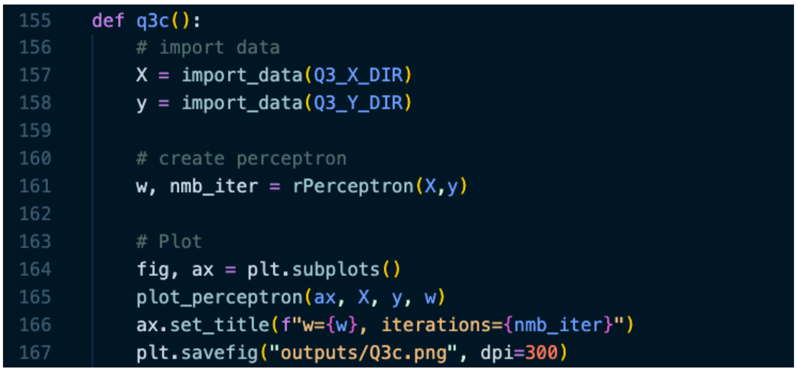
```
def q3b():
    # Import data
    X = import_data(Q3_X_DIR)
    y = import_data(Q3_Y_DIR)

    alpha, nmb_iter = dual_perceptron(X,y)

    # Use alphas to yield w
    w = (alpha * y) @ X

    fig, ax = plt.subplots()
    plot_perceptron(ax, X, y, w)
    ax.set_title(f"w={w},    iterations={nmb_iter}")
    plt.savefig("./outputs/Q3b.png", dpi=300)
```

c





```python
def rPerceptron(X, y, max_iter=100):
    np.random.seed(1)

    # Initialise w vectors
    nfeatures = X.shape[1]
    w = np.zeros((max_iter, nfeatures))
    w[0] = np.zeros(nfeatures)

    # Initialise indicator
    indicator = np.zeros(X.shape[0])

    # Set r equal to 2 as in question
    r = 2

    for t in range(max_iter):

        yXw = (y * (X @ w[t].T)) + (indicator * r)
        mistake_idxs = np.where(yXw <= 0)[0]

        # If there are mistakes, update w vector at index "i"
        if mistake_idxs.size > 0:
            i = np.random.choice(mistake_idxs)
            w = w + y[i] * X[i]
            w[i+1] = w[i] + y[i]*X[i]
            indicator[i] = 1
        else:
            return w[t], t + 1

    # Max iterations reached, return the latest w vector
    return w[max_iter - 1], max_iter
```



```python
def q3c():
    # import data
    X = import_data(Q3_X_DIR)
    y = import_data(Q3_Y_DIR)

    # create perceptron
    w, nmb_iter = rPerceptron(X,y)

    # Plot
    fig, ax = plt.subplots()
    plot_perceptron(ax, X, y, w)
    ax.set_title(f"w={w}, iterations={nmb_iter}")
    plt.savefig("outputs/Q3c.png", dpi=300)
```