Introduction to Logistic Regression: Takeaways

Syntax

- Gradient descent using the log-loss cost function:

```
def gradient_descent(init, x, y, iterations = 1000, learning_rate = 0.0001,
                             stopping_threshold = 1e-6):

    # Set the previous cost and parameters
    previous_cost = None
    beta0 = init[0]
    beta1 = init[1]

    # Perform the gradient descent
    for i in range(iterations):

        # Calculate the predicted outcome based on the predictor
        hz = 1 / (1 + np.exp(-1 * (beta0 + beta1 * x)))

        # Calculate the log-loss cost based on current parameters
        costs = y * (-np.log(hz)) + (1 - y) * (-np.log(1 - hz))
        current_cost = sum(costs)

        # Check if we've met the conditions for breaking the loop
        if previous_cost and abs(previous_cost - current_cost) <= stopping_threshold:
            break

        previous_cost = current_cost

        beta0_derivative = np.mean(hz - y)
        beta1_derivative = np.mean(x * (hz - y))
```

```
        beta0 = beta0 - learning_rate * beta0_derivative
        beta1 = beta1 - learning_rate * beta1_derivative

    return np.array([beta0, beta1])
```

Concepts

- Classification is a supervised learning task wherein we try to predict the value of a categorical variable.
- Logistic regression is the classification equivalent of linear regression.
- Probability refers to the probability that the outcome will be `1`.
- Odds are the ratio of the probability of an event happening over the event not happening. It ranges between the infinities, which makes it easier to relate to a linear model
- Log-odds are the logarithm of the odds, and they are directly related to changes in the predictors.
- The predictors are related to the binary outcome via the sigmoid function:
    - $Z = \beta_0 + \beta_1 X$
    - $EY = h(Z)$
    - $EY = h(\beta_0 + \beta_1 X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$