Case Study 3

1. What data did you collect? (Dataset Description)

For this case study, I used the Messidor Features dataset, originally derived from diabetic retinopathy screening examinations. Each row in the dataset represents one patient (one eye), and each column is a numeric feature automatically extracted from retinal fundus images.

- Number of instances: 1,151 patients
- Number of features: 19 predictors + 1 target variable
- Target variable: Class
    - 0 → No_DR (no diabetic retinopathy)
    - 1 → DR (presence of diabetic retinopathy)

The predictor variables include:

- Image quality indicators
- Pre-screening output
- Counts of microaneurysms at several confidence levels
- Exudate-related measurements
- Other image-derived features such as distances and area ratios

In my app, I renamed the predictors to X0–X18 for valid R names and kept Class as the response variable, converted to a factor with two levels: No_DR and DR.

2. Why is this topic interesting or important? (Motivation)

Diabetic retinopathy (DR) is one of the leading causes of preventable blindness among people with diabetes. Early detection and timely treatment are crucial to avoid vision loss, but manual grading of retinal images by specialists is time-consuming and resource-intensive.

Using machine learning to automatically predict whether a patient has DR from image-derived features can:

- Help screen large numbers of patients quickly

- Support ophthalmologists in decision making
- Prioritize high-risk patients for further examination

This topic is interesting to me because it combines healthcare and data science. Building a predictive model for DR classification shows how machine learning can be used for socially meaningful problems such as early disease detection and prevention of blindness.

3. How did you analyze the data? (Methodology & Data Science Life Cycle)

I followed the main steps of the data science life cycle:

3.1 Data Understanding and Preparation

- Loaded the messidor_features.arff file into R using the foreign package.
- Renamed the feature columns to valid R names (X0–X18) and converted Class into a factor with labels No_DR and DR.
- Performed basic exploratory analysis:
  - summary() to see ranges and distributions
  - head() to see the structure of the dataset
- There were no missing values, so no imputation was required.

3.2 Train/Test Split

- I randomly split the dataset into a training set and a testing set.
- The train proportion is controlled by a slider in the Shiny app (default 80% train, 20% test).
- This follows the idea of keeping a separate test set for unbiased performance evaluation.

3.3 Modeling – Logistic Regression (Classification)

I chose logistic regression as the classification algorithm. It models the probability that a patient has diabetic retinopathy (Class = DR) based on the predictor variables.

Let:

- $\mathrm{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})$be the features for patient i
- $y_i \in \{0,1\}$be the class label (0 = No_DR, 1 = DR)

The logistic regression model is:

$$P(y_i = 1 \mid \mathrm{x}_i) = \pi_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}}$$

Equivalently, the logit (log-odds) is linear in the predictors:

$$\log\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

The parameters $\beta_j$are estimated by maximum likelihood. After fitting the model on the training set, I used it to predict the probability $P(\mathrm{DR})$for each example in the test set.

3.4 Decision Threshold and Classification

The model outputs a probability between 0 and 1. To convert probabilities into class labels, I choose a classification threshold τ:

- If $P(\mathrm{DR}) \geq \tau \rightarrow$ predict DR
- If $P(\mathrm{DR}) < \tau \rightarrow$ predict No_DR

In the Shiny app, this threshold is controlled by a slider, so the user can interactively see how changing τaffects the confusion matrix and metrics like accuracy, sensitivity, and specificity.

3.5 Evaluation

On the test set, I compute:

- Confusion matrix
- Accuracy = $\frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$
- Sensitivity (True Positive Rate) = $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$
- Specificity (True Negative Rate) = $\frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$

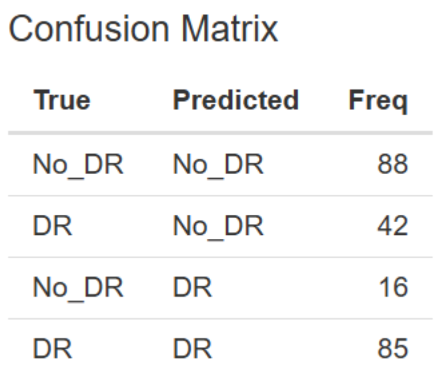
These results are displayed dynamically in the “Model Results” tab of the Shiny app.

4. What did you find in the data? (Results & Findings)

From the logistic regression model, I observed the following:

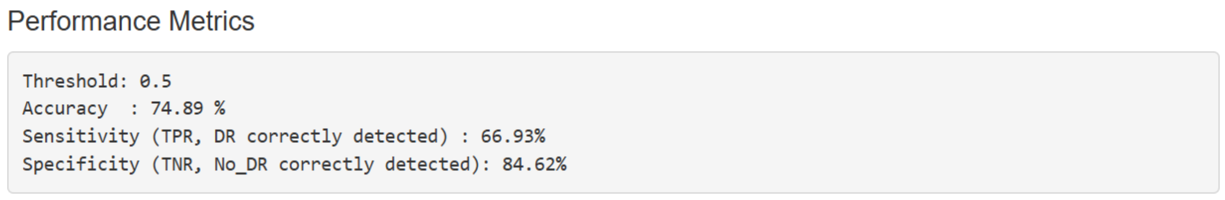
- ❖ **Overall performance**
    - o On a typical split with 80% train and 20% test and a threshold of 0.5, the model achieved:

- **Accuracy:** 74.89 %
- **Sensitivity:** 66.93% (correctly detecting DR cases)
- **Specificity:** 84.62% (correctly detecting No_DR cases)



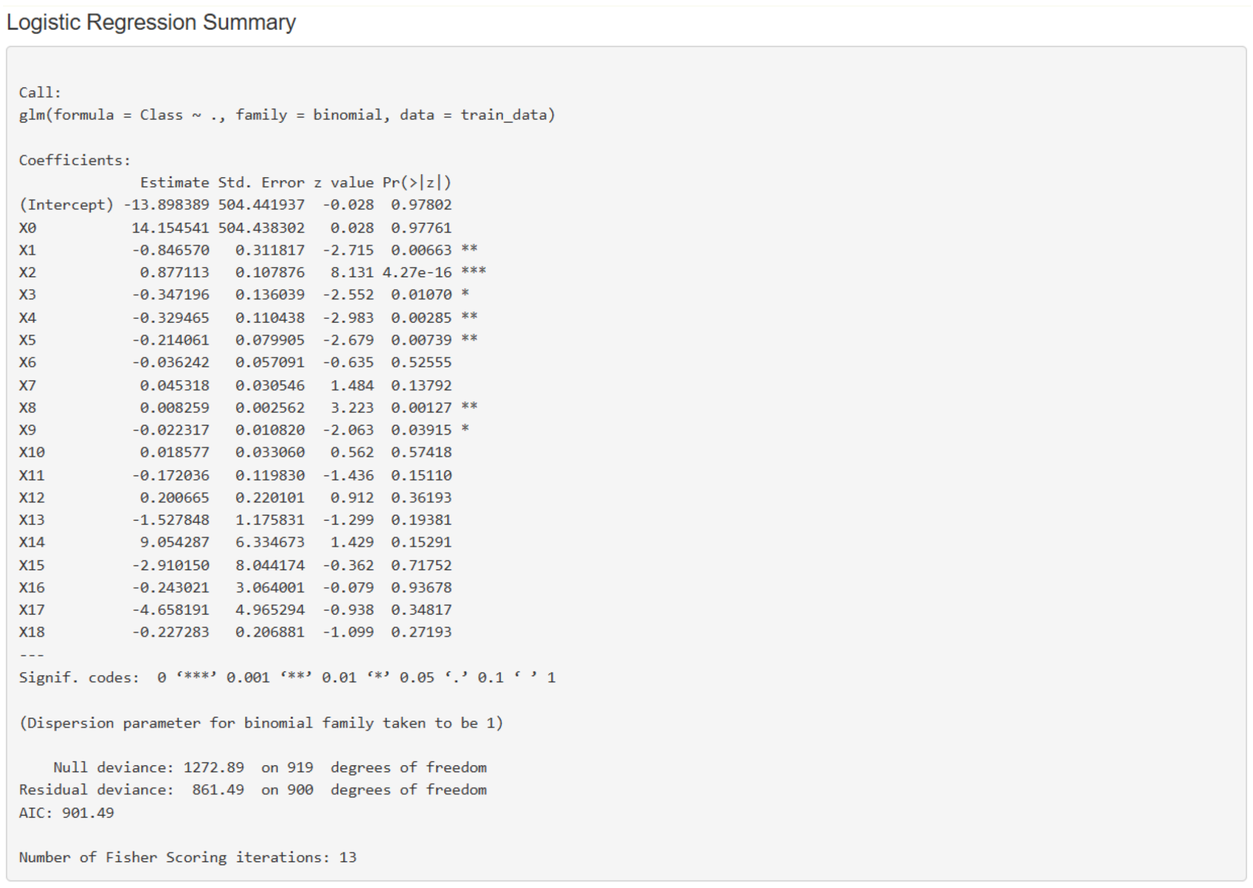
Confusion Matrix

True	Predicted	Freq
No_DR	No_DR	88
DR	No_DR	42
No_DR	DR	16
DR	DR	85



Performance Metrics

```
Threshold: 0.5
Accuracy  : 74.89 %
Sensitivity (TPR, DR correctly detected) : 66.93%
Specificity (TNR, No_DR correctly detected): 84.62%
```



Logistic Regression Summary

```
Call:
glm(formula = Class ~ ., family = binomial, data = train_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -13.898389 504.441937  -0.028  0.97802
X0           14.154541 504.438302   0.028  0.97761
X1           -0.846570   0.311817  -2.715  0.00663 **
X2            0.877113   0.107876   8.131 4.27e-16 ***
X3           -0.347196   0.136039  -2.552  0.01070 *
X4           -0.329465   0.110438  -2.983  0.00285 **
X5           -0.214061   0.079905  -2.679  0.00739 **
X6           -0.036242   0.057091  -0.635  0.52555
X7            0.045318   0.030546   1.484  0.13792
X8            0.008259   0.002562   3.223  0.00127 **
X9           -0.022317   0.010820  -2.063  0.03915 *
X10           0.018577   0.033060   0.562  0.57418
X11          -0.172036   0.119830  -1.436  0.15110
X12           0.200665   0.220101   0.912  0.36193
X13          -1.527848   1.175831  -1.299  0.19381
X14           9.054287   6.334673   1.429  0.15291
X15          -2.910150   8.044174  -0.362  0.71752
X16          -0.243021   3.064001  -0.079  0.93678
X17          -4.658191   4.965294  -0.938  0.34817
X18          -0.227283   0.206881  -1.099  0.27193
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1272.89  on 919  degrees of freedom
Residual deviance:  861.49  on 900  degrees of freedom
AIC: 901.49

Number of Fisher Scoring iterations: 13
```

- ❖ **Effect of the classification threshold**

When I lowered the threshold (e.g., from 0.5 to 0.3), the model predicted more patients as DR:

- ✓ Sensitivity increased (fewer missed DR cases),
- ✓ Specificity decreased (more false positives).

When I raised the threshold (e.g., from 0.5 to 0.7):

- ✓ Sensitivity decreased (more missed DR cases),
- ✓ Specificity increased (fewer false alarms).

This shows the typical trade-off between catching as many diseased patients as possible and avoiding unnecessary alarms.

5. Feature effects (logistic regression coefficients)

- ❖ The logistic regression summary in the app shows the estimated coefficients and their significance.
- ❖ Positive coefficients correspond to features that increase the log-odds of DR when they increase, while negative coefficients decrease it.
- ❖ Some features related to microaneurysms and exudates showed stronger association with DR, which matches clinical expectations.

6. Shiny Application Design (Parameter Change & Explanation)

The Shiny app has three main tabs:

1. **Introduction**
   - o Describes the dataset, the prediction task (DR vs No_DR), and the logistic regression methodology in simple terms.
   - o Explains that the app visualizes how the classification threshold affects model performance, following the data science life cycle (data → model → evaluation → interpretation).
2. **Data Overview**
   - o Shows summary() statistics for all variables.
   - o Displays the first 10 rows of the dataset to give the user an idea of the features.

3. **Model Results**
    - Allows the user to:
        - Adjust the train/test split via a slider.
        - Adjust the classification threshold via another slider.
    - A “Train / Update Model” button refits the logistic regression model and updates:
        - The confusion matrix,
        - Accuracy, sensitivity, specificity,
        - The full logistic regression summary (coefficients and p-values).

By changing the threshold, the user can immediately see how classification results change, which satisfies the requirement to “give end user options to change parameters and see how they affect the results.”

Link to Shiny Application

Shiny app URL:
https://arnaroy.shinyapps.io/assignment/

This web application implements a classification model for predicting diabetic retinopathy using the Messidor features dataset and allows interactive exploration of model parameters and performance.

7. GitHub and Reproducibility (optional text if you use GitHub)

I also uploaded the app code and dataset to GitHub in a public repository:

- Repository: https://github.com/Roy1707018/messidor_shiny_logistic
- Files included:
    - app.R
    - messidor_features.arff

The app can be run directly from GitHub using:

```
shiny::runGitHub("messidor_shiny_logistic", "Roy1707018")
```

This ensures that the code and data for the project are version-controlled and reproducible.