Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Advanced Machine Learning

Project 1

Michał Ciasnocha
Jerzy Kraszewski
Filip Kołodziejczyk

Contents

1 Introduction

This project is dedicated to implementing and comparatively analyzing various optimization algorithms for logistic regression. The central focus lies in developing and experimentally evaluating these algorithms, with balanced accuracy as the primary metric for assessing their performance.

Within the scope of this study, three optimization algorithms have been selected for in-depth examination: Iterative Reweighted Least Squares[1], Stochastic Gradient Descent[2], and Adaptive Moment Estimation[3]. Each offers a unique perspective on the optimization process, thus providing a comprehensive understanding of their respective efficacies.

Additionally, as part of the experimental process, appropriate datasets have been sourced and prepared to effectively compare logistic regression's classification performance with other established methods, including Linear and Quadratic Discriminant Analysis, Decision Trees, and Random Forest. This comparison aims to situate logistic regression within the broader spectrum of classification techniques. The project's overall goal is to elucidate the relative strengths and applicabilities of various optimization algorithms in logistic regression and to benchmark logistic regression's performance against other classification methods.

2 Methodology

In our project, apart from implementing the selected optimizers, we had to implement the compatible logistic regression and carefully choose and preprocess datasets. On top of that, the stopping rule was proposed to avoid the overfitting issue and speed up the learning process. A performance measure was proposed to compare these optimizers somewhat with other popular algorithms.

2.1 Datasets overview

Nine datasets were collected and preprocessed for experimentation. The datasets were sourced from the UCI Machine Learning Repository. The datasets and preprocessing details are described in Appendix A. The two most common operations on those datasets were removing collinear variables and encoding the nonnumerical variables to numerical ones (to simplify further experiments).

2.2 Stopping rule

The stopping rule, embedded within the training loop, is a crucial component of the optimization process. This project adopts a strategy where the stopping rule is based on observing changes in the binary cross-entropy loss between consecutive iterations. To guide the termination of the training process, a tolerance level has been set at 0.0001.

This approach serves a dual purpose. Firstly, it ensures that the training concludes when the change in loss is sufficiently small, indicating that the algorithm has converged to a stable solution. This point of minimal loss change is vital for determining the optimal stopping moment, ensuring both the accuracy and stability of the model. Secondly, the predetermined tolerance threshold plays a crucial role in enhancing the overall efficiency of the training process. By defining this threshold, the algorithm can preempt unnecessary computations and iterations, halting the process once the loss variation falls below this set value.

The choice of the tolerance value, in this case, 0.0001, is not arbitrary but grounded in empirical research and practical observations. It reflects a balance between precision and computational practicality, tailored to fit the specific requirements and convergence characteristics of the given problem and algorithm.

2.3 Performance measure

The balanced accuracy function is utilized as a pivotal metric for performance evaluation. This function operates by calculating the sensitivity and specificity for each class, which are averaged to derive the balanced accuracy. This methodology is especially beneficial in imbalanced datasets where class distribution is uneven. Traditional accuracy measures can be misleading in such contexts, often biased towards the majority class. In contrast, balanced accuracy offers a more accurate and equitable measure of model performance, considering the disparities in class sizes.

The versatility of the balanced accuracy metric extends to its applicability in both binary and multiclass classification tasks. This flexibility is crucial for comprehensively assessing different models across various classification problems. Furthermore, the metric is an effective tool for comparing the performance of different classifiers. By providing a uniform measurement standard, balanced accuracy enables a fair and consistent evaluation of classifier performance, ensuring that the assessment represents each class in the dataset.

3 Convergence analysis

After all the components mentioned earlier were implemented, we began our experiments with the convergence analysis. It is conducted to observe how the function's binary cross-entropy (we used than instead of plain log-likelihood function, as it provides a direct measure of what the training process aims to optimize), value changes with the number of iterations for each algorithm on the training data for every dataset . This analysis helps to understand how quickly each algorithm can converge and their stability during training on different datasets. The procedure was as in 3.1.

3.1 Computation steps

1. **Training Loop:** Each algorithm is trained on the training data for multiple epochs. The binary cross-entropy value is calculated at each iteration to monitor convergence.
2. **Binary Cross-Entropy Calculation:** A function is defined to compute the value based on the current model parameters and training data.
3. **Convergence Check:** The training loop continues until convergence criteria are met, i.e., when the change in BCE loss falls below a predefined tolerance level (stopping rule described in more details in 2.2.
4. **Visualisation:** The binary cross-entropy loss values are plotted against the number of iterations for each algorithm for each dataset to visualise their convergence behaviour.

3.2 Convergence results

The procedure was applied to all three methods for each prepared dataset so that the optimizers could be compared next to each other in different scenarios. Results are plotted in the Fig.1. IWSL converges much faster than the rest, by a scale, stopping after a few-likelier than a dozen iterations. ADAM usually needed hundreds of iterations to stop. It has not converged for two datasets, which could be expected considering that those datasets had a small number of samples to learn from. However, the worst convergence could be noted for an SGD, which has not converged within a hard limit of 500 epochs for any dataset. The reason could be a choice of a small learning rate (0.001), which was set to be equal for ADAM, to direct comparison. However, for SGD, it is often recommended to use a more significant default learning rate (0.01).

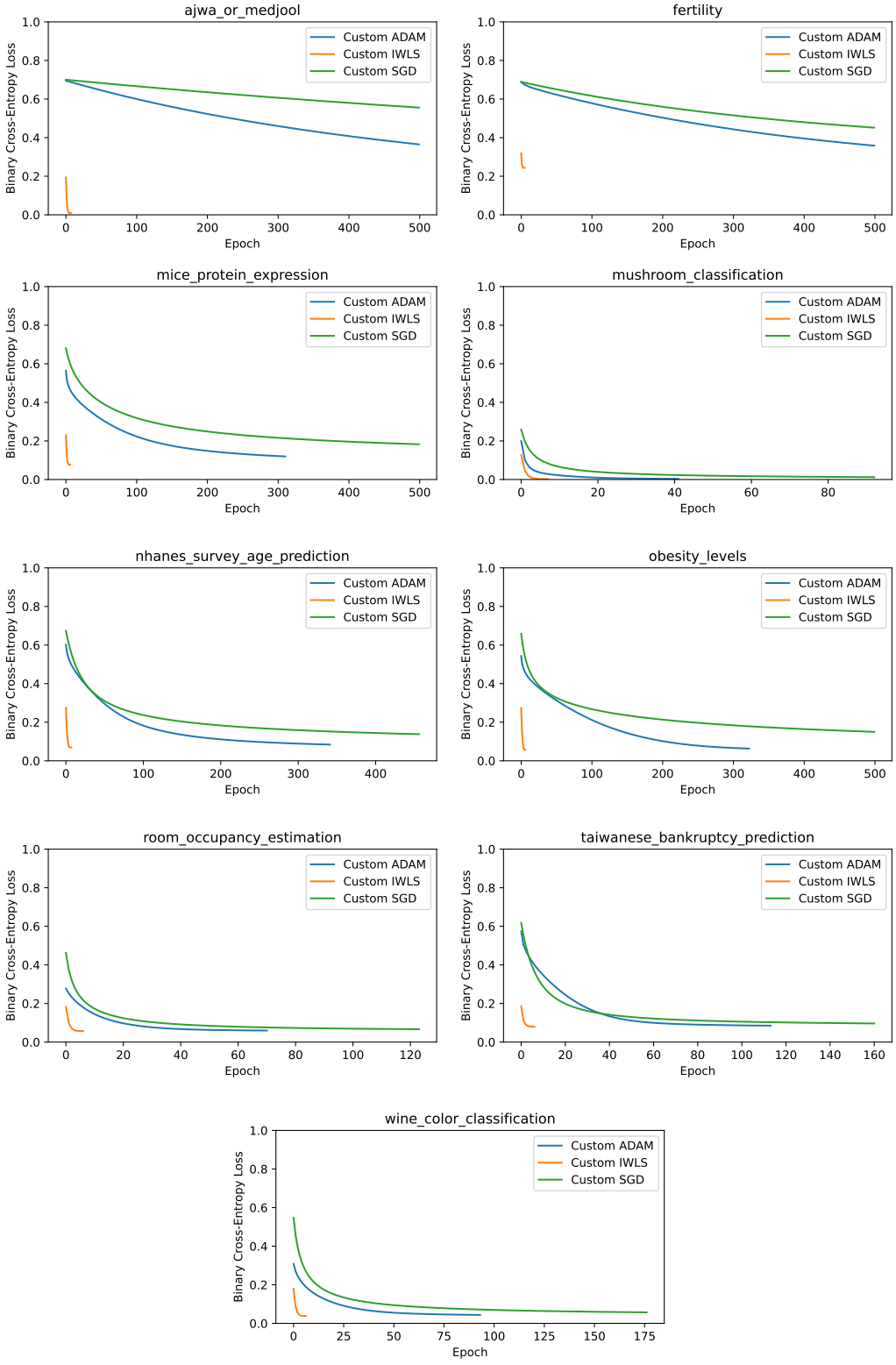


Figure 1: Convergence results

4 Comparison of classification performance

While convergence measures the time efficiency pretty well, it does not provide information about the quality of the model at all. Therefore, using the method described in 2.3, we used different optimizers to check how well the logistic regression classifies data correctly. To embed it within other popular machine learning methods, we also used predefined models (from scikit-learn library) for comparison. Those were:

- LDA (Linear Discriminant analysis),
- QDA (Quadratic Discriminant Analysis),
- Decision tree and
- Random Forest.

4.1 Accuracy results

Once again, experiments were conducted on different datasets to collect results in various scenarios. To derivate the stable results, each dataset was measured on the same dataset 5 times, using a different train-test split (but the splits were the same among all the models). The results are shown in Fig. 2. Unfortunately, there were datasets accessible to be trained on, such that all the models provided perfect or almost perfect accuracy (mushroom classification, wine color classification, nhanes survey age prediction, and room occupancy estimation). In the cases where the dataset was more challenging, we could see that ADAM and IWSL happened to score significantly lower for a specific dataset. However IWSL also outperformed rest twice, too. Focusing on our implementations of optimizers, there seems that IWSL is the best one. SGD and ADAM performed similarly in general. We used default parameters for each predefined model, and even minor hyperparameter tuning could offer significantly different results. The overall conclusion could be that each problem should be considered individually, and the appropriate model carefully selected for it, as there is no perfect solution.

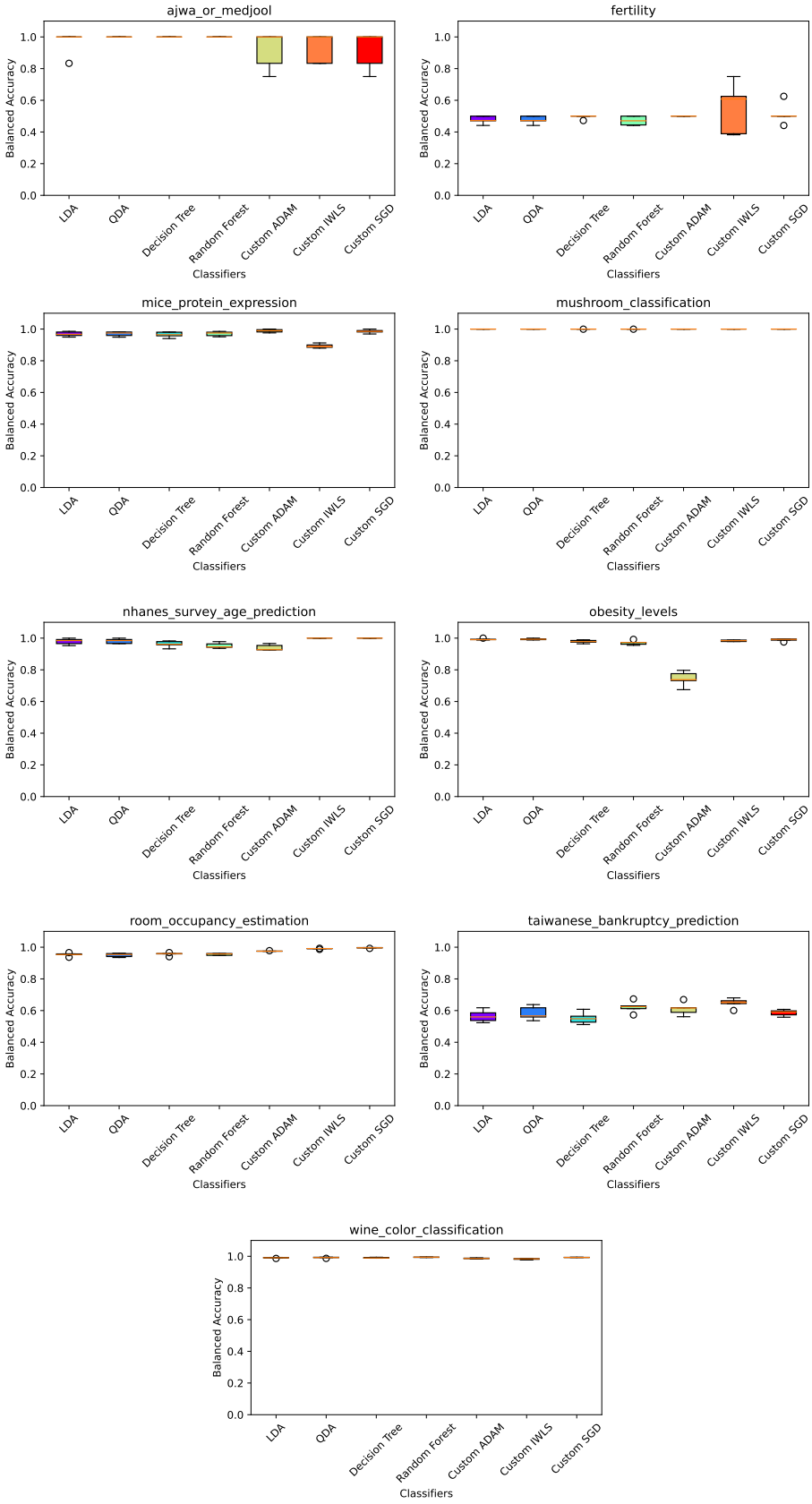


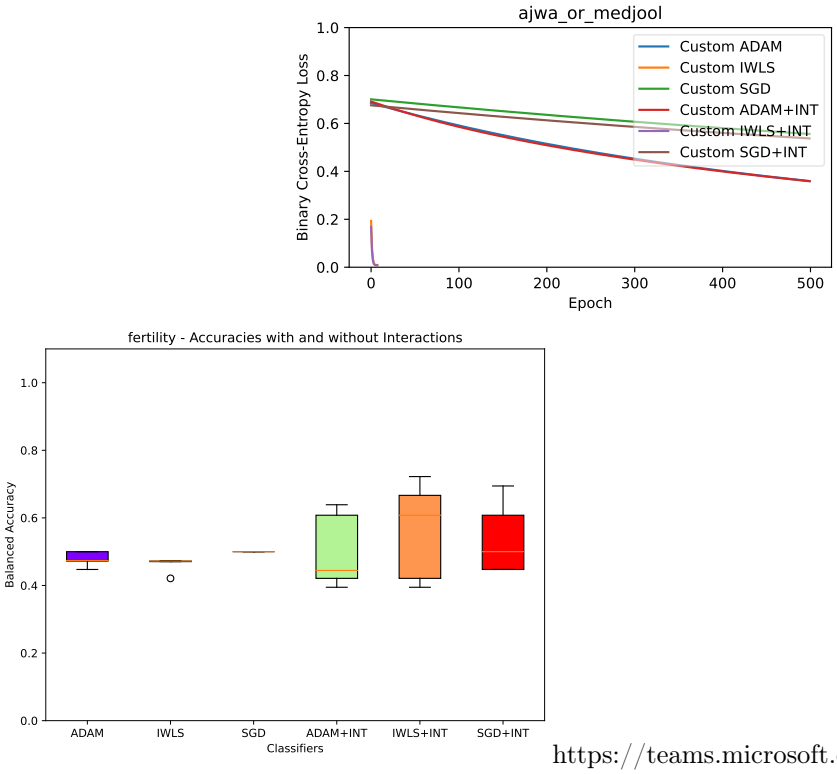
Figure 2: Accuracy results

5 Interaction influence on convergence

As a final experiment, we checked how introducing interactions to the training process influences our optimizers in terms of convergence and performance. In this context, interactions are the features that are a product of pairs of features. It is based on the assumption that the other variable could influence the effect of one feature on the target variable. Therefore, to consider those, we can introduce such interactions. We could reuse the previous scenarios with the modified dataset to perform this experiment, where we generated the interaction features.

5.1 Interaction influence results

The results are presented in Fig. 3. In all cases, with interaction features, as expected, convergence was the same or better than without them (since the dataset was more extensive, thus more complex model). However, the time complexity increases. In exchange, in all cases, the interactions improved the models' performance (except the one where accuracy was already near perfect). Therefore, this method proves to be suitable for increasing the model performance, but the cost is due to the complexity of computation. However, as mentioned earlier, it all depends on the dataset. In case of independent features (which still is rather unlikely in real dataset), it probably won't increase the accuracy, while increasing the computational requirements.



https://teams.microsoft.com/l/message/19:c52a72ffd7504688a1bfa0f4

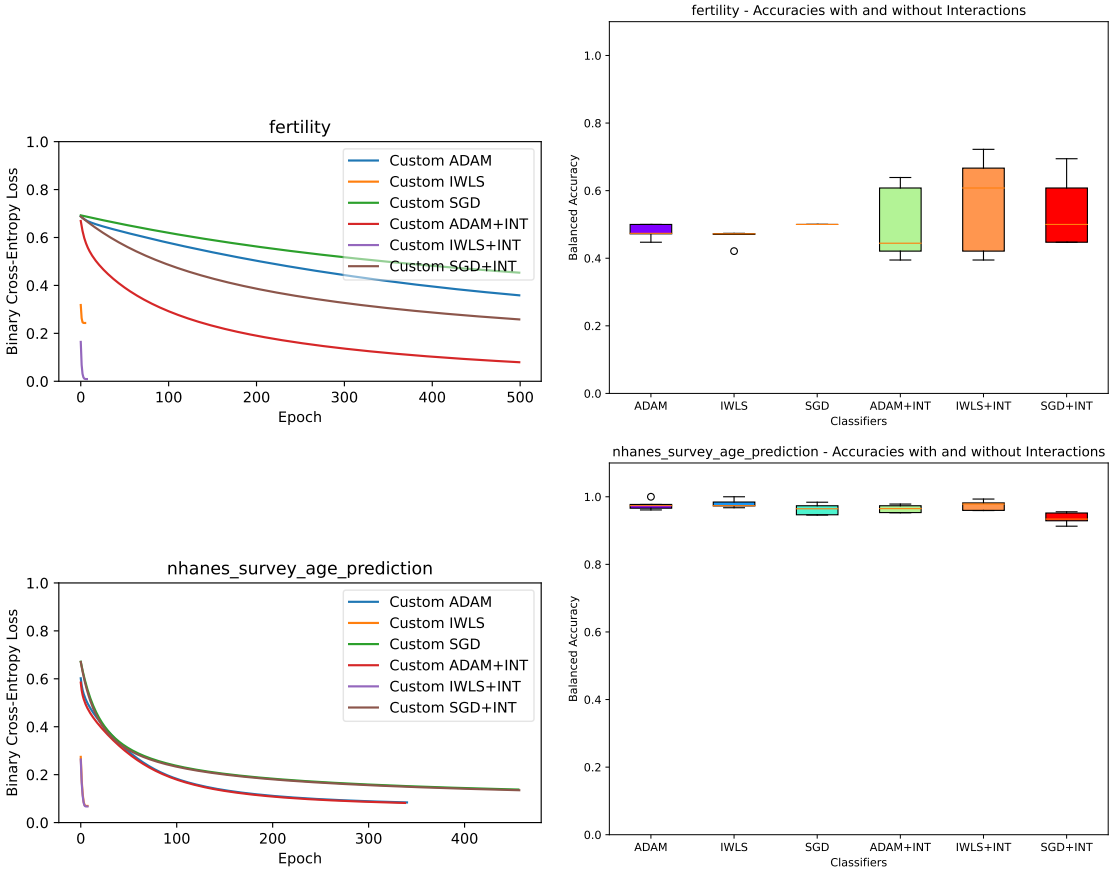


Figure 3: Interaction influence results

References

[1] Raymond F. Muzic Jr. and Bradley T. Christian. Evaluation of objective functions for estimation of kinetic parameters. *Medical Physics*, 33(2):342–353, 2006.

[2] Jincheng Sun Xiaoqin Shen Huaqiang Zhang, Fusheng Yu and Kun Li. Deep learning for sea cucumber detection using stochastic gradient descent algorithm. *European Journal of Remote Sensing*, 53(sup1):53–62, 2020.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

A Datasets

Summary of datasets chosen for the project. In total 9 datasets are presented, including 3 small ones (at most 10 features) and 6 large ones (more than 10 features). Each entry includes link where one can find the dataset, number of features and instances after the preprocessing, and the details how a particular dataset was preprocessed. Code used for preprocessing can be found in *preprocessing.ipynb* notebook.

A.1 Small Datasets

1. National Health and Nutrition Health Survey 2013-2014 (NHANES) Age Prediction Subset
   - **Features:** 6
   - **Instances:** 2278
   - **Objective:** Predict age group (senior/non-senior) based on medical features.
   - **Preprocessing applied:**
     - drop SEQN serial number
     - drop collinear variables:
       * LBXIN (0.55 correlation with BMXBMI)
       * LBXGLU (0.69 correlation with LBXGLT)
     - transform age group to binary target (senior/non-senior):
       * 0: Adult
       * 1: Senior
2. Ajwa or Medjool
   - **Features:** 3
   - **Instances:** 20
   - **Objective:** Predict date fruit species (Ajwa/Medjool) based on physical dimensions, weight, and calories.
   - **Preprocessing applied:**
     - drop columns:
       * color (it's 1-to-1 with target)
       * calories (highly correlated with date weight and date length)
       * date length (highly correlated with dat weight and pit length)
     - transform species to binary target (Ajwa/Medjool):
       * 0: Ajwa
       * 1: Medjool
3. Fertility Dataset
   - **Features:** 12

- **Instances:** 100
- **Objective:** Predict fertility based on behavioral factors (discrete numerical values based on subject's response).
- **Preprocessing applied:**
  - one-hot encode the season feature
- transform fertility to binary target (normal/altered):
  - 0: Normal
  - 1: Altered
- Note that the dataset is small and imbalanced, with only 12 instances of altered fertility.

A.2 Large Datasets

1. Mice Protein Expression
   - **Features:** 72
   - **Instances:** 1077 (has missing values)
   - **Objective:** Predict whether mice have Down syndrome based on some proteins in the brain.

   Preprocessing applied:
   - Originally, 49 columns had missing values:
     - Dropped 3 instances with lots of missing columns, afterwards only 9 columns had missing values
     - Dropped 5 columns with more than 10
     - For the last 4 columns, filled missing values with the overall mean of the column as there was no significant difference between means of the two classes
   - Dropped categorical features other than Genotype
   - Dropped highly correlated columns ($r > 0.8$)
   - Transformed Genotype binary target (Control vs. Ts65Dn):
     - 0: Control
     - 1: Ts65Dn
2. Mushroom Dataset
   - **Features:** 117
   - **Instances:** 8124
   - **Objective:** Real data on mushrooms, predict whether a mushroom is edible or poisonous.

   Preprocessing applied:
   - One-hot encode all features

- Transform class to binary target (edible/poisonous):
  - 0: Edible
  - 1: Poisonous

3. Room Occupancy Estimation
   - **Features:** 11
   - **Instances:** 10129
   - **Objective:** Estimate room occupancy (0, 1, 2, 3 people) based on sensor data.

   Preprocessing applied:
   - Drop date and time columns
   - Drop highly correlated columns ($r > 0.8$)
   - Transform occupancy to binary target (none or one person/two or three people):
     - 0: Zero or one person
     - 1: Two or three people
4. Taiwanese Bankruptcy Prediction
   - **Features:** 70
   - **Instances:** 6819
   - **Objective:** Predict whether a company will go bankrupt based on business-related data.

   Preprocessing applied:
   - Drop highly correlated columns ($r > 0.8$)
   - Transform bankruptcy to binary target (non-bankrupt/bankrupt):
     - 0: Non-bankrupt
     - 1: Bankrupt

   Note that the dataset is imbalanced, with only 220/6819 instances of bankrupt companies.
5. Estimation of Obesity Levels
   - **Features:** 27
   - **Instances:** 2111 (preprocessing required)
   - **Objective:** Predict obesity level based on patient's survey. Data conversion required for binary meaning (e.g., normal+overweight vs. obese).

   Preprocessing applied:
   - One-hot encode following features: Gender, MTRANS, CAEC, CALC
   - Replace 'yes'/'no' with 1/0
   - Transform obesity level to binary target (underweight+normal+overweight/obese):
     - 0: Insufficient_Weight, Normal_Weight, Overweight_Level_I, Overweight_Level_II

- 1: Obesity_Type_I, Obesity_Type_II, Obesity_Type_III

6. Wine Color

   - **Features:** 12
   - **Instances:** 6497
   - **Objective:** Predict wine color (red/white) based on chemical properties.

   Preprocessing applied:

   - Introduce binary target (red/white):
     - 0: Red
     - 1: White