DLSS 2025 - Assignment: Regression Task with Multilayer Perceptrons

Giordano De Marzo

24.04.2025

To be handed in by 14.5.2025 23:59 at https://github.com/orgs/DLSS-2025

In this assignment, you will use a **Multilayer Perceptron (MLP)** to solve a regression task. Your objective is to predict **salary** from a cleaned dataset of job vacancies. The dataset includes various job-related features such as required skills, education level, job location, and more. This is a dataset provided by the International Labour Organization and it was collected from one of the main Uruguayan job portals. You can download it from my website at URL.

The goal is to build, train, and evaluate different MLP models and discuss their performance.

The maximum number of points is **30**. You may receive **bonus points** for completing optional tasks.

Tasks

1. Data Import and Exploration (4 points)

- Load the provided CSV file (`filtered_vacancies_dataset_uruguay.csv`).
- Print the shape and column names.
- Visualize the distribution of the target variable (`Sueldo`).
- Perform EDA.
- Check for missing values and outliers.

2. Data Preprocessing (5 points)

- Apply preprocessing steps:
    - Standardize all numerical features.
    - One-hot encode categorical features.
    - Handle missing values.
    - Extract the log of the salary as the target variable. This is a good practice when dealing with large numbers that span a big range of values. You can try also without taking the log to see if performances degrade.ince the distribution of the variable is
    - Split the data into train/validation/test sets.

3. MLP Model Building and Training (10 points)

- Build and train **at least 3 MLP variants** with:
    - Different number of layers/neurons

    - You can also vary the learning rate, the batch size, activation functions etc.
- Use an appropriate activation for the output neuron and a Loss suitable for regression.
- Track learning curves (train vs validation loss).

4. Model Evaluation and Selection (3 points)

- Evaluate the models on the validation set.
- Select the best model and justify your choice

5. Final Testing (3 points)

- Report the performances on the test set.
- Compare it to a simple Linear Regression baseline.

6. Report Writing (5 points)

- Summarize:
    - Dataset exploration
    - Description of preprocessing steps
    - Architecture and training details of each model
    - Results and evaluation
    - Learning curves
- Max 3 pages. Include plots and tables where needed.
- The report should be structured like a scientific document. For instance a possible structure is the following
    - Title, Name and Surname
    - Intro: Introduce the problem and the approach you are gonna follow
    - Results:
        - Data: Describe the data and detail any preprocessing you did
        - Models: The models you used, their architectures
        - Training: Plots of the training curves, discussion about the curves
        - Evaluation: Evaluation of the model on the test set, tables or plots showing the performances
    - Conclusions and discussion: Final remarks

Bonus Tasks

You can use up to an additional half page in the report for each bonus task you perform. The maximum grade is 30, if you collect more than 30 points, the score will be cut to 30.

Bonus Task 1: Standard Machine Learning (1 bonus points)

- Train a **Random Forest Regressor** and compare its performance with the MLP.

Bonus Task 2: Model Ensemble (1 bonus points)

- Create an **ensemble** of at least 3 MLPs.
- Compare ensemble performance with individual models.

Bonus Task 3: Classification (2 bonus points)

- Divide the salaries in two classes, above and below 30k.
- Train a MLP on this binary classification task.
- Compare its performances with a standard machine learning classifier

---

Submission Guidelines

- Submit your work on the course GitHub as a **Colab notebook (.ipynb)** and a **PDF report**.
- Make sure your notebook is clean, well-documented, and reproducible.
- Include a `README.md` if needed.

Data Imputation

This dataset contains a lot of nan values, so dropping rows (samples) containing a nan would result in most of the dataset being dropped. As a consequence we need to do some form of data imputation, that is we need to fill the missing holes in the dataset. A standard approach is:

- fill missing numerical values with the median
- fill missing categorical values with the mode

This below is an example section of code performing a simple imputation

```
# Numerical columns to impute with median
numerical_cols = ['year', 'EdadDesde', 'EdadHasta', 'num_words',
'CantidadPuestosVacantes']

# Categorical columns to impute with most frequent value
categorical_cols = [col for col in df.columns if col not in
numerical_cols + ['Sueldo']]

# Fill numerical missing values with median
for col in numerical_cols:
    if df[col].isnull().any():
        median_value = df[col].median()
        df[col] = df[col].fillna(median_value)

# Fill categorical missing values with mode
for col in categorical_cols:
```

```
if df[col].isnull().any():
    mode_value = df[col].mode()[0]
    print(col, mode_value)
    df[col] = df[col].fillna(mode_value)
```