# Projekt

## Cześć II

Grupy tydzień nieparzysty (nr 1, 2, 4, 6):
- Wprowadzenie: 11 kwietnia 2025
- Termin wysłania Części II Projektu: ~~24 kwietnia 2025~~ 1 maja 2025
- Termin oddania: ~~25 kwietnia 2025~~ 2 maja 2025

Grypy tydzień parzysty (nr 3, 5, 7):
- Wprowadzenie: 16 maja 2025
- Termin wysłania Części I Projektu: 29 maja 2025
- Termin oddania: 30 maja 2025

Celem drugiej części projektu jest **opracowanie pierwszych modeli uczenia maszynowego**, które będą rozwiązywały wybrane przez Was zadanie. Skupiamy się na **implementacji**, testowaniu oraz analizie działania różnych metod. Zadania możesz realizować w Jupyter Notebooku lub zaimplementować osobne skrypty (np. train.py, test.py).

### Część implementacyjna

Zakres konieczny do zrealizowania na ocenę conajminej 3.0
1. Przygotuj pipeline przetwarzania danych. Wykorzystaj `sklearn.pipeline.Pipeline` i/lub `sklearn.compose.ColumnTransformer`
2. Sprawdź takie techniki jak `OneHotEncoding`, `SimpleImputer`. Sprawdź też `sklearn.preprocessing`
3. Wybierz **trzy różne modele** uczenia maszynowego i wytrenuj je na swoim zbiorze danych (np. regresja, drzewa decyzyjne, SVM, itd.)
4. Przeprowadź **trening** i **ewaluację** każdego modelu (w tej cześci nie musisz jeszcze martwić się o zabalansowanie danych).

Zakres konieczny do zrealizowania na ocenę conajmniej 4.0
Koszystając tylko z paczki `numpy`:
1. Samodzielnie zaimplementuj model **regresji liniowej** wykorzystując **zamkniętą formułę**:
    a. Jeśli Twoje zadanie to klasyfikacja, użyj innej **kolumny numerycznej** jako ground truth.
    b. Przeanalizuj **ograniczenia** zastosowania zamkniętej formuły. Dlaczego nie jest w praktyce wykorzysytwana?
2. Zaimplementuj regresję liniową lub logistyczną (zależnie od analizowanego problemu) korzystając z metody **gradientu prostego (gradient descent).**
    a. Pamietaj, że jest to proces iteracyjny, stąd odpowiednio podziel dane na tzw. batche

    b. Wybierz i zaimplementuj odpowiednią **funkcję kosztu** do analizowanego problemu, np. **MSE (Mean Squared Error)** sprawdzi się w problemie regresji, a **CE (Cross Entropy)** w problemie klasyfikacji
3. Porównaj wyniki własnych implementacji modeli z tymi zaimplementowanymi za pomocą `scikit-learn`
4. `Uwaga. Do przetwarzania danych nadal możesz koszystać z metod scikit-learn.`
5. `Uwaga 2. Implementuj swoje rozwiązanie wykonując obliczenia na macierzach, nie iteruj po elementach.`

Zakres konieczny do zrealizowania na ocenę 5.5

1. Zaimplementuj model regresji (**liniowej lub logistycznej**) w bibliotece `PyTorch`
2. Pipeline powinien wykorzystywać:
    a. `.backward()` do obliczenia gradientów
    b. iteracyjny proces treningu z podziałem na batche
3. Uruchom swój model na **GPU** korzystając z `PyTorch`
4. Możesz wykorzystać:
    - **Własny komputer** (jeśli posiadasz kartę graficzną Nvidia)
    - **Google Colab** UWAGA: dostęp do GPU może być ograniczony, więc zaplanuj zadanie z wyprzedzeniem
2. Porównaj czas treningu CPU a GPU.

**Część raportowa**

W części II projektu będziemy skupiać się na kwestiach implementacyjnych. W ramach części raportowej przygotuj tabelę, w której podasz wyniki ewaluacji na zbiorach treningowym, walidacyjnym oraz testowym.