Sieci neuronowe

Przewidywanie poziomu dochodów na podstawie danych demograficznych

- **Autorzy:** Mateusz Hypta 280116, Mateusz Kwapisz 280107

- **Technologie:** Python, TensorFlow/Keras, Scikit-learn

Cel projektu i zbiór danych

Co chcemy osiągnąć?

Problem: Klasyfikacja, czy dana osoba zarabia powyżej 50 tys. USD rocznie (>50K) czy poniżej (<=50K).

Dane: Zbiór "Adult" zawierający ~32 tys rekordów.

Cechy:

- Demograficzne: wiek, rasa, płeć.
- Społeczne: edukacja, stan cywilny, relacje rodzinne.
- Zawodowe: profesja, branża, liczba godzin pracy tygodniowo.

Wyzwanie: Niezbalansowane klasy

Problem "Class Imbalance”

Dystrybucja danych: Osoby zarabiające <=50K stanowią ok. 75% zbioru, a >50K tylko 25%.

Zagrożenie: Model może nauczyć się "zgadywać" klasę większościową, ignorując mniejszość.

Zastosowane rozwiązania:

1. **Oversampling:** To rozwiązanie fizycznie zmienia strukturę zbioru treningowego przed uruchomieniem sieci.
- **O ile:** Zwiększamy liczebność klasy mniejszościowej tak, aby **zrównała się 1:1** z klasą większościową.
    - **Przed:** ok. 22 600 rekordów (<=50K) vs ok. 7 500 rekordów (>50K).
    - **Po:** ok. 22 600 rekordów (<=50K) vs **22 600 rekordów** (>50K).
- **Mechanizm (Random Oversampling):** Algorytm losuje istniejące już przykłady osób zarabiających >50K i powiela je (duplikuje), aż obie grupy będą miały identyczny rozmiar.
- **Efekt:** Model widzi oba scenariusze (bogaty/biedny) z taką samą częstotliwością, co eliminuje tendencję do ignorowania mniejszości.

Architektura Sieci Neuronowej

Budowa modelu (MLP, 2 warstwy ukryte)

64 neurony: Wystarczająco dużo, by zrozumieć wszystkie cechy wejściowe, ale nie za dużo, by nie "zapchać" pamięci.
32 neurony: Kondensuje wiedzę – odrzuca szum, zostawia tylko fakty decydujące o zarobkach.
Dropout (0.3 - 0.4): Klucz do sukcesu. Bez niego model uczy się danych na pamięć (overfitting) i zawodzi w rzeczywistości.

Parametr	Najlepsza Wartość	Dlaczego ta wygrała?
Architektura	**[64, 32]**	**Najlepszy balans między precyzją a szybkością.**
Dropout	**0.4**	**Wyższy dropout lepiej chroni przed błędami na nowych danych.**
Learning Rate	**0.001**	**Standardowa szybkość "Adam", która trafia w optimum.**
Metryka AUC	**~0.92**	**Oznacza, że model dobrze oddziela obie grupy.**

Metodologia: Eksperyment 200 prób

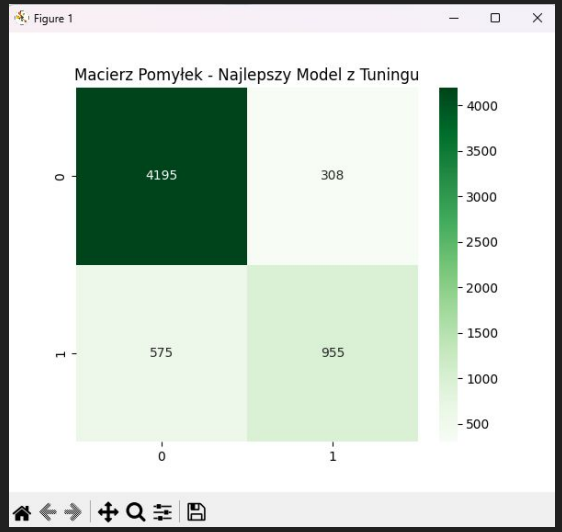
Optymalizacja poprzez iterację

Dlaczego powtarzamy trening?

- **Losowość startowa:** Sieci neuronowe zaczynają z losowymi wagami. Różny "punkt startu" może prowadzić do różnych wyników końcowych.
- **Szukanie Optimum:** 200 prób pozwala uniknąć "pułapek" (minimów lokalnych) i znaleźć najlepsze możliwe parametry dla tego problemu.
- **Statystyczna pewność:** Wybieramy model, który nie jest dziełem przypadku, ale stabilnym wynikiem udowodnionym wielokrotnym testem.

Analiza Wyników

Skuteczność najlepszego modelu



Accuracy: ~85% (ogólna poprawność).

Precyzja vs Recall:

- Model świetnie radzi sobie z klasą <=50K.
- Dzięki wagom/oversamplingowi, **Recall dla klasy >50K** (wykrywalność bogatszych osób) wzrósł do wysokiego poziomu (~65%).
- Precyzja dla klasy >50K: **~76%**

Krzywa ROC-AUC: Wysoka wartość pola pod krzywą świadczy o dobrej separacji klas przez model.

Wnioski i praktyczne zastosowanie

Główne Wnioski

- **Skuteczność modelu:** Uzyskana ogólna poprawność na poziomie **~85%** potwierdza, że sieci neuronowe świetnie radzą sobie z danymi demograficznymi.
-
- **Klucz do sukcesu – Balans:** Dzięki technikom **Oversampling** i **Class Weights** model przestał ignorować mniejszość i zyskał realną zdolność wykrywania wysokich dochodów.
-
- **Model:** Proces tuningu (200 prób) pozwolił na uzyskanie wysokiej precyzji (**~76%**). Model rzadko się myli, wskazując osoby z grupy >50