Sprawozdanie z projektu: Sieci Neuronowe

Przewidywanie poziomu dochodów na podstawie danych demograficznych

Mateusz Hypta 280116 Mateusz Kwapisz 280107

22 stycznia 2026

Spis treści

1 Analiza problemu i wybór architektury

1.1 Definicja problemu

Celem projektu było zaprojektowanie i zaimplementowanie klasyfikatora opartego na sztucznej sieci neuronowej, który na podstawie cech demograficznych (takich jak wiek, wykształcenie, zawód, stan cywilny) przewidzi, czy roczny dochód danej osoby przekracza 50 tys. USD. Problem zdefiniowano jako klasyfikację binarną:

- **Klasa 0 ($<= 50K$):** Dochód niższy lub równy 50 tys. USD (klasa większościowa).
- **Klasa 1 ($> 50K$):** Dochód powyżej 50 tys. USD (klasa mniejszościowa).

1.2 Wyzwanie: Niezbalansowane klasy

Głównym wyzwaniem analitycznym była nierówna dystrybucja klas w zbiorze danych. Osoby zarabiające $<= 50K$ stanowią około 76% populacji, podczas gdy osoby zamożniejsze to jedynie 24%. Stwarzało to ryzyko wytrenowania klasyfikatora, który osiąga wysoką dokładność (Accuracy $\approx 76\%$) poprzez proste przypisywanie każdemu rekordu klasy większościowej, całkowicie ignorując mniejszość.

1.3 Uzasadnienie wyboru technologii

Do rozwiązania problemu wybrano architekturę **MLP (Multi-Layer Perceptron)**. Ze względu na tabelaryczny charakter danych oraz większą liczbę cech po zakodowaniu (One-Hot Encoding), sieć gęsta (Dense) jest optymalnym wyborem. Pozwala ona na modelowanie nieliniowych zależności między zmiennymi demograficznymi, co daje przewagę nad prostszymi modelami liniowymi.

2 Przygotowanie architektury programu i danych

2.1 Środowisko i biblioteki

Projekt zrealizowano w języku **Python** z wykorzystaniem bibliotek:

- **TensorFlow / Keras:** Budowa, trenowanie i ewaluacja modelu.
- **Pandas & NumPy:** Przetwarzanie danych i operacje na macierzach.
- **Scikit-learn:** Preprocessing (skalowanie, kodowanie), podział danych i metryki.
- **Matplotlib / Seaborn:** Wizualizacja wyników (macierze pomyłek, wykresy ROC).

2.2 Analiza i przygotowanie danych

Proces przygotowania danych przebiegał w następujących krokach:

1. **Czyszczenie:** Usunięto rekordy z brakującymi danymi oraz zredundowane kolumny (np. `fnlwgt` – waga statystyczna bez znaczenia predykcyjnego, `education` – informacja zawarta już w `education-num`).
2. **Kodowanie:**
    - Zmienna celu (`income`) została zmapowana na wartości binarne (0/1).

- Zmienne kategoryczne (np. `workclass`, `occupation`, `marital.status`) poddano kodowaniu **One-Hot Encoding**, co pozwoliło sieci neuronowej na interpretację danych tekstowych.

3. **Skalowanie:** Wszystkie dane wejściowe znormalizowano za pomocą **StandardScaler**.

2.3 Strategia podziału i balansowania

Zastosowano następujące podejście do przygotowania zbiorów, aby zapewnić rzetelną ewaluację:

- **Zbiór Treningowy (Oversampling):** Zastosowano technikę *Random Oversampling* tylko na danych treningowych. Klasa mniejszościowa została losowo powielona do momentu osiągnięcia proporcji 1:1. Dzięki temu sieć "widziała"obie klasy równie często i nie faworyzowała większości.
- **Zbiór Testowy (Undersampling 50/50):** Wydzielony zbiór testowy został sztucznie zbalansowany do proporcji 50%/50% poprzez odrzucenie nadmiaru próbek klasy większościowej. **Cel:** Uzyskanie "uczciwego"testu. Na takim zbiorze losowe zgadywanie daje wynik 50%, więc uzyskane wyniki są realnym dowodem skuteczności modelu.

3 Architektura sieci i proces trenowania

3.1 Ewolucja modelu i analiza odrzuconych rozwiązań

Ostateczny kształt sieci jest wynikiem serii eksperymentów. Poniżej przedstawiono analizę podejść, które zakończyły się niepowodzeniem:

1. Model bazowy

- *Konfiguracja:* Brak oversamplingu.
- *Wynik:* Accuracy $\approx 76\%$, ale Recall dla klasy $> 50K$ bliski 0%.
- *Wniosek:* Model wpadł w pułapkę "Leniwego Klasyfikatora", przewidując zawsze klasę 0. Odrzucony.

2. Modele "Zbyt Głębokie"(Deep Networks)

- *Konfiguracja:* Architektury typu [128, 64, 32] lub [256, 128], niski Dropout (0.1).
- *Wynik:* Szybkie przeuczenie (Overfitting). Błąd na zbiorze treningowym malał, ale na walidacyjnym rósł.
- *Wniosek:* Przy relatywnie małej liczbie cech (ok. 100 wejść), głębokie sieci "zapamiętywały"dane zamiast uczyć się reguł.

3.2 Ostateczna architektura

Na drodze eksperymentu wyłoniono optymalną architekturę:

- **Wejście:** Warstwa Input dopasowana do wymiarowości danych.
- **Warstwa Ukryta 1:** 64 neurony, aktywacja **ReLU**.

  - *BatchNormalization:* Stabilizacja procesu uczenia.
  - *Dropout (0.3):* Wyłączanie 30% neuronów w każdej epoce.
- **Warstwa Ukryta 2:** 32 neurony, aktywacja **ReLU**, BatchNormalization, Dropout (0.3).
- **Wyjście:** 1 neuron, aktywacja **Sigmoid**.

Wykorzystano optymalizator **Adam** z *Learning Rate* = 0.0005, który zapewnił stabilną zbieżność.

3.3 Metodologia eksperymentu

Przeprowadzono automatyczny dobór parametrów (**Grid Search**) połączony z mechanizmem **Pruning**.

- Przetestowano 12 kombinacji parametrów (Warstwy × Dropout × LR).
- Każdą konfigurację trenowano ponad 10 razy , aby wyeliminować wpływ losowej inicjalizacji wag.
- Zastosowano **Early Stopping** (zatrzymanie treningu, gdy brak poprawy przez 5 epok).

4 Analiza wyników

Ranking	Architektura (Layers)	Dropout	Learning Rate	AUC
1 (Best)	**[64, 32]**	**0.3**	**0.0005**	**0.9211**
2	[64, 32]	0.4	0.0005	0.9195
3	[128, 64]	0.3	0.0005	0.9182
4	[128, 64, 32]	0.3	0.0005	0.9150
5	[32, 16]	0.2	0.0010	0.9080

Tabela 1: Ranking 5 najlepszych konfiguracji modelu wyłonionych w procesie Grid Search. Zwycięska architektura [64, 32] zapewniła najlepszy balans między dokładnością a stabilnością, przewyższając bardziej złożone struktury.

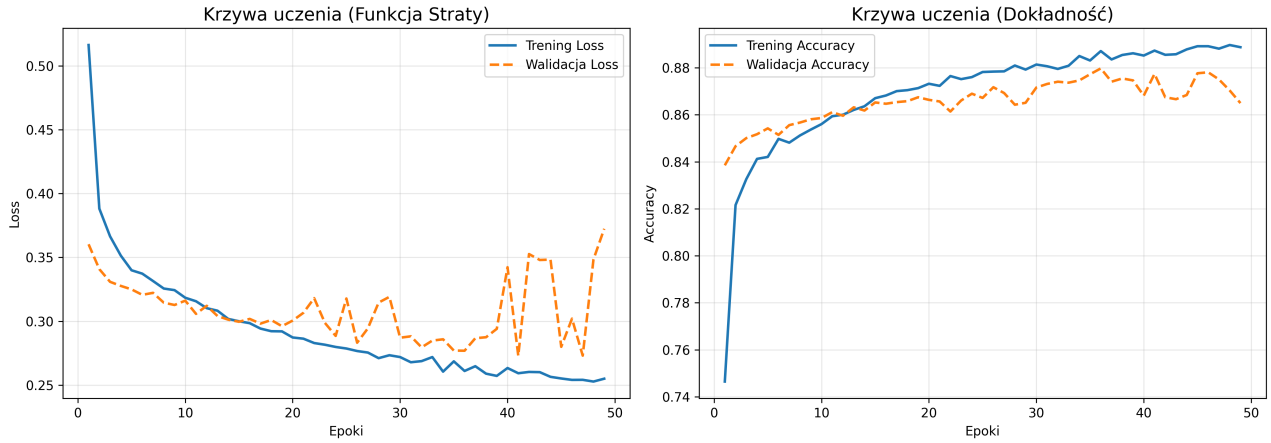
4.1 Osiągnięte rezultaty

Najlepszy model został zweryfikowany na niezależnym, w pełni zbalansowanym zbiorze testowym (1455 próbek klasy 0 i 1455 próbek klasy 1). Uzyskano następujące wyniki:

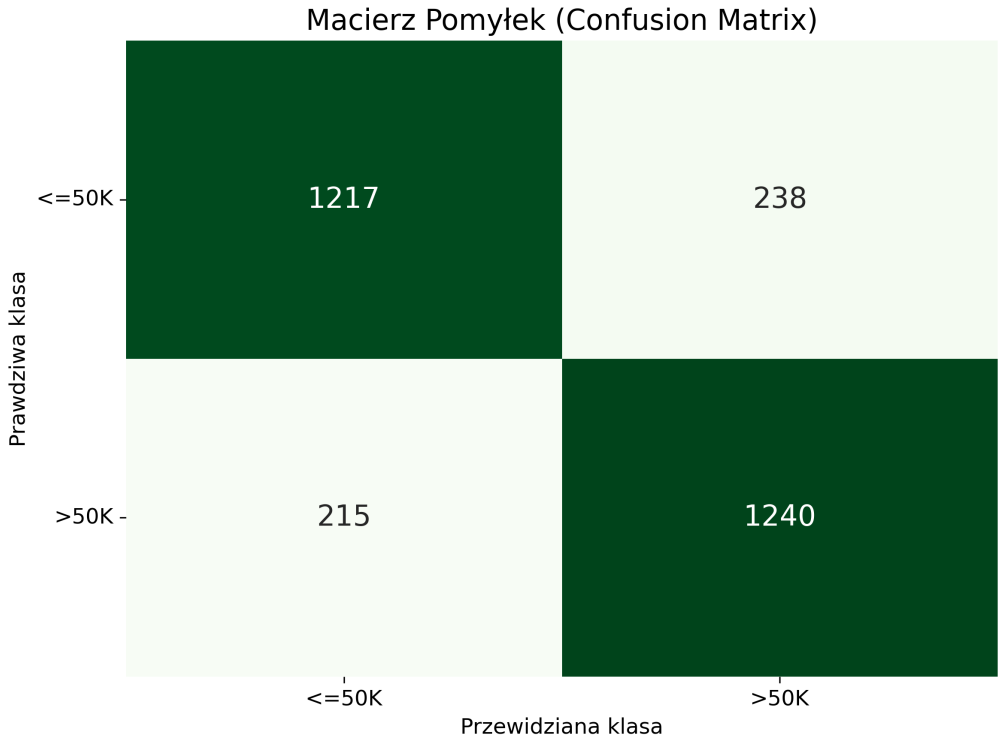
Metryka	Wartość
Accuracy (Dokładność)	84.43%
AUC (Area Under Curve)	0.9211
Precision (Klasa $> 50K$)	83.90%
Recall (Klasa $> 50K$)	85.22%
F1-Score	0.8456

Tabela 2: Wyniki końcowe najlepszego modelu na zbiorze testowym

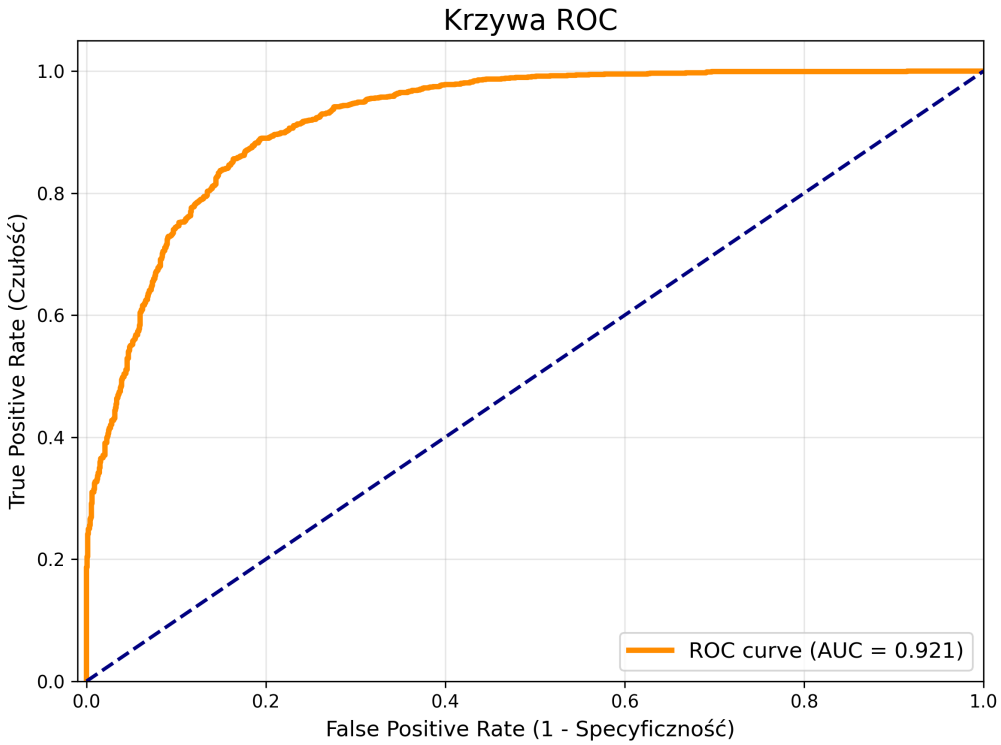
4.2 Wizualizacja i interpretacja



Rysunek 1: **Proces uczenia się sieci.** Wykres po lewej (Loss) obrazuje spadek funkcji straty. Widać wyraźnie, że od około 20. epoki błąd walidacyjny (linia przerywana) przestaje spadać i zaczyna fluktuować, co jest sygnałem do zatrzymania treningu (Early Stopping), aby uniknąć przeuczenia. Wyższa dokładność na walidacji (wykres prawy) względem ostatecznego testu wynika z oversamplingu w procesie treningowym.



Rysunek 2: **Macierz Pomyłek**. Model wykazuje symetrię błędów: 238 Fałszywie Pozytywnych vs 215 Fałszywie Negatywnych. Oznacza to, że model nie faworyzuje żadnej z klas.



Rysunek 3: **Krzywa ROC**. Wysoka wartość AUC (0.921) świadczy o wysokiej zdolności separacji klas.

4.3 Sukcesy i porażki

Sukcesy:

- **Wysoki Recall (85.22%):** Dzięki zastosowaniu Oversamplingu model skutecznie wykrywa osoby zamożne. Jest to znacząca poprawa względem modelu bazowego, który miał tendencję do ignorowania tej grupy.
- **Stabilność (F1-Score 0.8456):** Wynik F1 świadczy o doskonałym balansie między Precyzją a Czułością.
- Uzyskanie AUC > 0.92 na danych demograficznych jest wynikiem bardzo dobrym, wskazującym na skuteczną ekstrakcję cech przez warstwy ukryte.

Problemy i rozwiązania:

- *Ryzyko przeuczenia:* Analiza krzywej uczenia (Rys. 3) wykazała, że trening powyżej 40 epok prowadzi do destabilizacji błędu walidacyjnego. Rozwiązaniem było zastosowanie agresywnego mechanizmu **Early Stopping** oraz warstw **Dropout (0.3)**.
- *Niezbalansowane klasy:* Początkowe eksperymenty pokazały, że sieć dąży do maksymalizacji Accuracy poprzez ignorowanie klasy mniejszościowej. Zastosowanie **Random Oversampling** w zbiorze treningowym wyeliminowało to zjawisko.

5 Podsumowanie

Realizacja projektu zakończyła się sukcesem, a założone cele klasyfikacji zostały osiągnięte. Przeprowadzone badania dowiodły, że w przypadku danych tabelarycznych o ograniczonej liczbie cech, kluczem do skuteczności nie jest głębokość sieci, lecz odpowiedni dobór parametrów i preprocessing danych.

Architektura [64, 32] okazała się optymalnym rozwiązaniem – zapewniła wysoką zdolność generalizacji, unikając jednocześnie overfittingu , które obserwowano przy głębszych strukturach.

Najważniejszym osiągnięciem projektu jest skuteczne rozwiązanie problemu niezbalansowanych klas. Dzięki zastosowaniu techniki *Random Oversampling*, model osiągnął współczynnik Recall na poziomie ponad 85% dla klasy mniejszościowej. Oznacza to, że system z wysoką skutecznością identyfikuje osoby o wysokich dochodach, co czyni go użytecznym narzędziem. Wysoka wartość AUC (0.92) oraz symetryczna macierz pomyłek potwierdzają, że model jest stabilny i rzetelny.