Kwantowe sieci neuronowe jako modele ryzyka kredytowego

numer badania KAE/S25:1.25

Sebastian Zając
SGH Szkoła Główna Handlowa w Warszawie

Spis treści

Wstęp **5**

1 Motywacja badawcza i cel pracy **6**

2 Kwantowe Sieci Neuronowe (QNN) **7**
2.1 Klasyczne Sieci Neuronowe 7
2.2 Obliczenia Kwantowe w Erze NISQ 8
2.2.1 Matematyczne Sformułowanie Algorytmu Kwantowego . . 9
2.2.2 Parametryzowane Obwody Kwantowe (PQC) 10
2.3 Proces Kwantowego Uczenia Maszynowego (QML) 10
2.3.1 Strategie Kodowania Danych 11
2.3.2 Ansatz: Parametryzowany Obwód Wariacyjny 12
2.3.3 Pomiar i Wartość Oczekiwana (Obserwable) 12

3 Dane: Credit Card Fraud Detection **14**
3.1 Wstępne przetwarzanie danych 14
3.2 Analiza jakości zmiennych 15

4 Wyniki eksperymentów **17**
4.1 Modele Klasyczne jako Benchmark 17
4.1.1 Regresja Logistyczna 17
4.1.2 Las Losowy 17
4.1.3 XGBoost 18
4.1.4 Maszyna Wektorów Nośnych 18
4.1.5 Wielowarstwowy Perceptron (MLP) 19
4.2 Wyzwanie Niezrównoważonych Danych i Metodologia Oceny . . . 19
4.2.1 Wyniki klasyfikatorów uczenia maszynowego 19
4.2.2 Analiza Wniosków Wstępnych 20
4.3 Klasyczna Sieć Neuronowa: PyTorch: architektura i parametryzacja 22
4.4 Ocena Wydajności na Zbiorach Testowych 24
4.4.1 Macierz Pomyłek (Undersampled Test Set) 24
4.4.2 Wyniki na Pełnym i Zrównoważonym Zbiorze 24
4.5 Podsumowanie i wnioski 25

5 Implementacja Wariacyjnego Klasyfikatora Kwantowego (VQC) **25**
5.1 Przygotowanie Danych dla QNN 25
5.1.1 Skalowanie Danych do Zakresu Kątów 26
5.1.2 Wybór Cech 26
5.2 Model Bazowy: Jednokubitowy Klasyfikator (QNN1) 26
5.2.1 Architektura Obwodu 26
5.2.2 Trening i Optymalizacja 27
5.3 Wielokubitowe Klasyfikatory Kwantowe 28
5.3.1 Wariant 1: QNN2 (4 kubity, BasicEntanglerLayers) 28
5.3.2 Warianty 2-4: QNN3, QNN4, QNN5 (PCA, Różne Architektury) 30

5.3.3 Wariant 5: QNN6 (4 kubity, StronglyEntanglingLayers, 4 warstwy) 31
5.4 Podsumowanie Wyników Kwantowych 32

6 Końcowe Podsumowanie i Porównanie: Modele Kwantowe vs. Klasyczne 33
6.1 Kluczowe Wyniki na Zrównoważonym Zbiorze Testowym 33
6.2 Porównanie na Pełnym (Niezbalansowanym) Zbiorze Testowym . 34
6.2.1 Wniosek Naukowy: Zdolność do Generalizacji 34
6.3 Wnioski końcowe i perspektywy 35
6.4 Ograniczenia Metodologiczne i Kierunki Dalszych Badań 35

Bibliografia 37

Wstęp

Wykrywanie oszustw w transakcjach kartowych stanowi jeden z kluczowych obszarów bezpieczeństwa systemów płatniczych i bankowości elektronicznej. Oszustwo płatnicze definiuje się jako **nieautoryzowane lub oszukańcze użycie instrumentu płatniczego**, prowadzące do straty finansowej po stronie posiadacza karty lub instytucji finansowej. Skuteczne wykrywanie takich zdarzeń ma zasadnicze znaczenie zarówno dla ograniczenia strat, jak i dla utrzymania zaufania do bezgotówkowych systemów płatności.

Wraz z dynamicznym wzrostem liczby płatności bezgotówkowych oraz transakcji realizowanych zdalnie, skala i złożoność oszustw kartowych stale rosną. Ramy regulacyjne i standardy branżowe, takie jak **dyrektywa PSD2** oraz mechanizm **silnego uwierzytelniania klienta** (ang. *Strong Customer Authentication*, SCA) [1, 2], ugruntowały znaczenie automatycznych, działających w czasie rzeczywistym systemów monitorowania transakcji oraz ilościowych metod oceny ich wiarygodności.

Współczesne systemy wykrywania oszustw opierają się na zaawansowanych technikach statystycznych oraz **metodach uczenia maszynowego**, które pozwalają przetwarzać ogromne strumienie danych transakcyjnych, behawioralnych i kontekstowych. Zadanie sprowadza się do **klasyfikacji binarnej**: odróżnienia transakcji legalnych od oszukańczych. Do klasycznych metod stosowanych w tym zadaniu należą regresja logistyczna, drzewa decyzyjne, lasy losowe oraz głębokie sieci neuronowe.

Detekcja oszustw wyróżnia się jednak specyficznymi trudnościami, które odróżniają ją od typowych zadań klasyfikacyjnych. Należą do nich przede wszystkim **skrajne niezbalansowanie klas** (oszustwa stanowią zwykle ułamek procenta wszystkich transakcji), **silna asymetria kosztu błędu** (przeoczone oszustwo oznacza bezpośrednią stratę, natomiast fałszywy alarm — zablokowanie legalnej transakcji i pogorszenie doświadczenia klienta) oraz **wymóg podejmowania decyzji w czasie rzeczywistym**. Skuteczność stosowanych metod zależy przy tym w dużej mierze od jakości danych oraz od możliwości obliczeniowych, a rosnąca złożoność modeli prowadzi do **rosnących kosztów obliczeniowych i energetycznych**.

Dotychczasowy postęp w tym obszarze opierał się na zwiększaniu mocy obliczeniowej klasycznych systemów. Dalsze zwiększanie ich wydajności napotyka jednak **fundamentalne ograniczenia** wynikające z praw fizyki, ujawniające się w skali nanometrycznej. W tym kontekście **komputery kwantowe** wprowadzają nowy paradygmat obliczeniowy, wykorzystujący zjawiska **superpozycji** i **splątania kwantowego**. Umożliwiają one potencjalne osiągnięcie **kwantowej przewagi** (ang. *Quantum Advantage*) w wybranych typach obliczeń, a przede wszystkim **zwiększenie ekspresyjności** modeli przy ograniczonej liczbie parametrów.

Współczesne komputery kwantowe pozostają na etapie intensywnego rozwoju (era NISQ). Znajdują one eksperymentalne zastosowanie w problemach optymalizacyjnych oraz w zadaniach uczenia maszynowego i głębokiego, określanych jako **kwantowe uczenie maszynowe (*Quantum Machine Learning*, QML)**.

W niniejszej pracy przedstawiono **porównanie klasycznych i kwantowych sieci neuronowych** w zadaniu wykrywania oszustw w transakcjach kartowych. Zaprezentowano także podejście hybrydowe, łączące klasyczne algorytmy optymalizacji z parametryzacją obwodów kwantowych. Analiza ta ma wskazać na potencjał rozwiązań kwantowych w detekcji oszustw, zwłaszcza w obszarach wymagających przetwarzania danych o wysokim wymiarze, modelowania złożonych zależności nieliniowych oraz radzenia sobie ze skrajnie niezbalansowanymi zbiorami danych.

1 Motywacja badawcza i cel pracy

Dynamiczny rozwój metod uczenia maszynowego znacząco zwiększył skuteczność systemów wykrywania oszustw płatniczych. Pomimo tego klasyczne modele wciąż napotykają istotne ograniczenia — zarówno pod względem **efektywności obliczeniowej** przy dużych strumieniach transakcji, jak i **zdolności do uchwycenia złożonych, nieliniowych zależności** odróżniających nieliczne oszustwa od ogromnej większości transakcji legalnych. Trudność tę potęguje **skrajne niezbalansowanie danych**, które osłabia działanie standardowych klasyfikatorów.

Komputery kwantowe, wykorzystujące zjawiska superpozycji i splątania kwantowego, otwierają potencjalnie nowe możliwości przetwarzania informacji. W tym kontekście **kwantowe sieci neuronowe** (ang. *Quantum Neural Networks*, QNN) stanowią obiecującą gałąź badań QML. Ich teoretyczna zdolność do reprezentowania złożonych zależności przy użyciu mniejszej liczby parametrów może przyczynić się do usprawnienia detekcji oszustw, zwłaszcza tam, gdzie kluczowe znaczenie mają **dokładność, efektywność i szybkość decyzji**.

Celem niniejszej pracy jest **zbadanie możliwości zastosowania kwantowych sieci neuronowych** w wykrywaniu oszustw w transakcjach kartowych oraz porównanie ich efektywności z klasycznymi modelami uczenia maszynowego. W szczególności praca koncentruje się na trzech zagadnieniach:

- Opracowanie i przetestowanie modeli QNN w zadaniu klasyfikacji transakcji jako legalnych lub oszukańczych.
- Porównanie skuteczności i efektywności obliczeniowej modeli kwantowych i klasycznych, ze szczególnym uwzględnieniem skrajnie niezbalansowanych danych.
- Zbadanie możliwości **podejścia hybrydowego**, integrującego klasyczne techniki optymalizacji z parametryzacją obwodów kwantowych (VQC).

Uzyskane wyniki mają stanowić punkt wyjścia do dalszych badań nad zastosowaniem technologii kwantowych w sektorze finansowym oraz przyczynić się do lepszego zrozumienia potencjału i ograniczeń kwantowych metod uczenia w praktycznych zastosowaniach z zakresu bezpieczeństwa płatności.

2 Kwantowe Sieci Neuronowe (QNN)

Postęp technologiczny i powszechna cyfryzacja sprawiły, że dane stały się powszechnym i cennym zasobem [3]. Są one generowane i przetwarzane zarówno w sposób ustrukturyzowany, jak i nieustrukturyzowany. Strukturyzacja danych doprowadziła do rozwoju wielu modeli, które dziś ogólnie określamy jako **modele uczenia maszynowego** (ang. *machine learning*, ML). Natomiast przetwarzanie danych nieustrukturyzowanych, takich jak tekst, obrazy czy wideo, przyczyniło się do rozwoju **uczenia głębokiego** (ang. *deep learning*, DL). Oba te podejścia, często określane zbiorczo jako **sztuczna inteligencja** (ang. *artificial intelligence*, AI), zostały stworzone głównie do rozpoznawania wzorców, ale coraz częściej wykorzystywane są również do modelowania i generowania nowych danych.

Sieci neuronowe posiadają szereg zalet, które sprawiają, że są szeroko stosowane w analizie danych:

- **Zalety:**
  - Umiejętność uczenia się z danych i generalizacji wzorców.
  - Możliwość modelowania nieliniowych zależności pomiędzy zmiennymi.
  - Skalowalność — można dostosować liczbę warstw i neuronów do złożoności problemu.
- **Wyzwania/Ograniczenia:**
  - Wymagają dużych ilości danych treningowych.
  - Często trudne do interpretacji i wyjaśnienia decyzji (*black-box*).
  - Wysokie zapotrzebowanie na moc obliczeniową, szczególnie w przypadku głębokich sieci (DNN).

2.1 Klasyczne Sieci Neuronowe

Klasyczny model sztucznej inteligencji można wyrazić jako funkcję $f(\mathbf{x}, \boldsymbol{\theta})$, która zależy od danych reprezentowanych przez macierz $\mathbf{x}$, oraz parametrów $\boldsymbol{\theta}$, których wartości zostaną ustalone w procesie uczenia.

Funkcja ta stanowi złożenie wielu prostych przekształceń nieliniowych, realizowanych przez poszczególne warstwy sieci. Każda warstwa dokonuje liniowego przekształcenia danych wejściowych, a następnie stosuje funkcję aktywacji wprowadzającą nieliniowość, co umożliwia aproksymację złożonych zależności.

Głębokie Sieci Neuronowe (ang. *Deep Neural Networks*, DNN) stanowią rozwinięcie klasycznych modeli perceptronowych poprzez zwiększenie liczby warstw ukrytych. Większa głębokość architektury pozwala modelowi reprezentować bardziej złożone struktury danych i hierarchiczne wzorce zależności. W sektorze płatności takie modele umożliwiają lepsze uchwycenie nieliniowych relacji między cechami transakcji, wzorcami behawioralnymi a prawdopodobieństwem, że dana transakcja ma charakter oszukańczy.

W kontekście wykrywania oszustw płatniczych, głębokie sieci neuronowe znajdują zastosowanie w takich zadaniach jak:

- Klasyfikacja transakcji (ang. *fraud detection*) – odróżnianie transakcji legalnych od oszukańczych na podstawie cech transakcyjnych.
- Wykrywanie anomalii – identyfikacja nietypowych wzorców odbiegających od typowego zachowania posiadacza karty.
- Analiza sekwencji transakcji – modelowanie czasowych wzorców w historii płatności, np. w celu wykrycia nagłej zmiany aktywności.
- Profilowanie behawioralne – budowanie profili normalnego zachowania klientów i sygnalizowanie odstępstw od nich.

Do najczęściej stosowanych architektur należą sieci w pełni połączone (ang. *Fully Connected Networks*, FCN) oraz modele rekurencyjne (ang. *Recurrent Neural Networks*, RNN) wykorzystywane do analizy danych sekwencyjnych, takich jak historia płatności. Coraz częściej w literaturze pojawiają się również podejścia oparte na sieciach konwolucyjnych (CNN).

Pomimo wysokiej skuteczności predykcyjnej, głębokie modele obarczone są istotnymi ograniczeniami praktycznymi – wymagają dużych zasobów obliczeniowych, znacznych ilości danych treningowych oraz charakteryzują się ograniczoną interpretowalnością wyników. Z tego względu w zastosowaniach regulacyjnych często stosuje się podejścia hybrydowe, łączące klasyczne modele statystyczne z komponentami opartymi na uczeniu głębokim.

2.2 Obliczenia Kwantowe w Erze NISQ

Współczesne **komputery kwantowe** nie są jeszcze maszynami do powszechnego wykorzystania. Obecnie dostępne urządzenia umożliwiają wykonywanie algorytmów na ograniczonej liczbie kubitów (zazwyczaj nieprzekraczającej kilkuset) i wciąż nie dysponują pełnymi mechanizmami korekcji błędów kwantowych. Dodatkowo, działanie bramek kwantowych wymaga, aby ich czas operacyjny był znacznie krótszy niż czas dekoherencji kubitów. Ograniczenie to uniemożliwia realizację długich sekwencji bramek w złożonych algorytmach.

Z tego względu obecny etap rozwoju technologii kwantowych określany jest mianem **ery NISQ** (ang. *Noisy Intermediate-Scale Quantum*) [4].

W tej fazie rozwoju systemy kwantowe mają **potencjał** wykazania tzw. **kwantowej przewagi** (ang. *Quantum Advantage*) w problemach o istotnym znaczeniu praktycznym. Natomiast **kwantowa supremacja** została eksperymentalnie wykazana w ściśle określonym problemie próbkowania z obwodu losowego, nie mającym bezpośredniego przełożenia na zastosowania w finansach.

Istotną klasą rozwiązań opracowywanych w ramach tej ery są **parametryzowane obwody kwantowe** (ang. *Parameterized Quantum Circuits*, PQC). Obwody te, z wykorzystaniem klasycznych optymalizatorów, mogą być trenowane w celu znalezienia wartości minimalizujących określoną funkcję kosztu. Podejście to, łączące klasyczne techniki optymalizacyjne z kwantowym przetwarzaniem informacji, określane jest mianem **uczenia hybrydowego**.

PQC realizowane są poprzez obwody zawierające bramki o ustalonej strukturze (np. bramki CNOT) oraz bramki parametryzowane, umożliwiając generowanie

złożonych, nieliniowych przekształceń przestrzeni stanów [5, 6]. Na ich bazie rozwijane jest **kwantowe uczenie maszynowe** (ang. *Quantum Machine Learning*, QML), które obejmuje grupę **wariacyjnych algorytmów kwantowych** (ang. *Variational Quantum Algorithms*, VQA).

Do najbardziej znanych reprezentantów tej klasy należą: *Variational Quantum Eigensolver* (VQE) [7], *Variational Quantum Solvers* (VQS) [8], *Variational Quantum Classifier* (VQC) [9], *Quantum Support Vector Classifier* (QSVC) [10], *Quantum Neural Networks* (QNN) [11], *Quantum Autoencoder* [12] oraz *Quantum Approximate Optimization Algorithm* (QAOA) stosowany w zadaniach optymalizacyjnych typu QUBO [13].

Wszystkie wymienione typy modeli mogą być implementowane w środowiskach programistycznych Python, takich jak IBM Qiskit, PennyLane czy Cirq, z wykorzystaniem zarówno symulatorów, jak i rzeczywistych komputerów kwantowych.

2.2.1 Matematyczne Sformułowanie Algorytmu Kwantowego

Podstawowym nośnikiem informacji jest **kubit** (ang. *Quantum Bit*, Qubit), którego stan $|\psi\rangle$ opisujemy jako wektor w przestrzeni Hilberta, z wykorzystaniem notacji Diraca (bra $\langle .|$ i ket $|.\rangle$). Stan kubitu wyrażany jest za pomocą **superpozycji** wektorów bazowych $|0\rangle$ i $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{1}$$

gdzie $\alpha, \beta \in \mathbb{C}$ są amplitudami prawdopodobieństwa, spełniającymi warunek normalizacyjny $|\alpha|^2 + |\beta|^2 = 1$.

Stan kubitu można również zareprezentować na **sferze Blocha** (Rysunek 1):

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2}$$

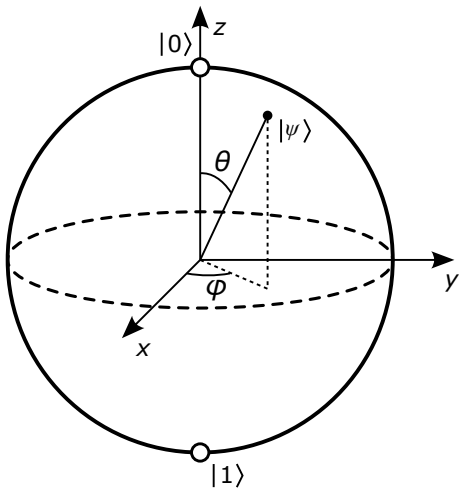
gdzie $\theta, \phi \in \mathbb{R}$ są kątami sferycznymi.

Wektory $\langle\psi|$ (bra) są tworzone poprzez sprzężenie hermitowskie (†) wektorów $|\psi\rangle$ (ket), tj. $\langle\psi| = |\psi\rangle^\dagger$. Notacja ta ułatwia zapis iloczynu skalarnego $\langle\psi|\phi\rangle$.

Stan układu wielu n kubitów reprezentowany jest przez iloczyn tensorowy, co oznacza, że przestrzeń stanów rośnie **wykładniczo** (2^n wymiarów) wraz z liczbą kubitów.

W obliczeniach kwantowych operacje na stanach realizowane są przez **operatory liniowe**, reprezentujące **bramki kwantowe**. Bramki te muszą być macierzami unitarnymi ($U^\dagger U = I$), co zapewnia zachowanie prawdopodobieństwa i odwracalność operacji. Przykłady bramek nieparametrycznych (macierze pauliego i hadamarda):

$$\begin{aligned} H &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \\ Y &= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}, \quad Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}. \end{aligned} \tag{3}$$



Rysunek 1: Sfera Blocha. [14]

2.2.2 Parametryzowane Obwody Kwantowe (PQC)

PQC stanowią podstawę hybrydowych algorytmów kwantowych. Każda bramka w obwodzie może być zależna od zestawu parametrów $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, optymalizowanych klasycznymi metodami.

Formalnie, stan początkowy $|\psi_0\rangle$ jest przekształcany przez unitarny operator $U(\boldsymbol{\theta})$ tworząc nowy stan:

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\psi_0\rangle\,, \tag{4}$$

gdzie $U(\boldsymbol{\theta})$ jest złożeniem parametryzowanych bramek (np. R_X, R_Y, R_Z) oraz bramek stałych (np. CNOT). Przykłady parametryzowanych bramek obrotów:

$$R_X(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \tag{5}$$

$$R_Y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \tag{6}$$

$$R_Z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}.$$

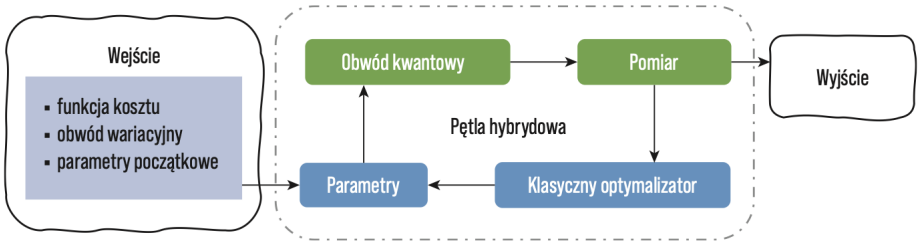
Obwody te są wielokrotnie modyfikowane w celu minimalizacji funkcji kosztu:

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|\,H\,|\psi(\boldsymbol{\theta})\rangle\,, \tag{7}$$

gdzie H jest operatorem (operatorem Hamiltona) odpowiadającym zadaniu optymalizacyjnemu.

2.3 Proces Kwantowego Uczenia Maszynowego (QML)

Kwantowe modele uczenia maszynowego są realizowane jako proces w architekturze hybrydowej, składający się z trzech głównych etapów (Rysunek 2):



Rysunek 2: Schemat wariacyjnego algorytmu kwantowego (VQA).

1. **Przygotowanie Danych (Mapowanie Cecha/Feature Map):** Klasyczny wektor danych $\mathbf{x}$ jest ładowany do obwodu kwantowego poprzez odwzorowanie $\phi(\mathbf{x})$, wyrażone jako operator $U_{\phi(\mathbf{x})}\,|0\rangle$. Procedura ta jest analogiczna do stosowania zanurzeń (*embeddings*) w klasycznych sieciach neuronowych. Wymaga ona wcześniejszej eksploracji, czyszczenia i standaryzacji danych.

2. **Parametryzowany Kwantowy Obwód Wariacyjny (Anzatz):** Reprezentowany przez zależny od parametrów $\boldsymbol{\theta}$ operator $W_{\boldsymbol{\theta}}$. Obwód ten składa się z sekwencji bramek parametryzowanych oraz bramek splątania (np. CNOT). Dobór struktury obwodu jest analogiczny do wyboru architektury (liczby warstw i neuronów) w klasycznych sieciach neuronowych.

3. **Pomiar i Funkcja Kosztu:** Wynik klasyfikacji uzyskuje się poprzez estymację wartości oczekiwanej $\langle H\rangle$ na jednym lub wielu kubitach, gdzie H jest operatorem pomiarowym. Najczęściej $H = \sigma_Z$ i wartość oczekiwana przyjmuje wtedy wartości z przedziału $[-1, 1]$.

Cały model QML można wyrazić jako:

$$f(\boldsymbol{\theta}, \mathbf{x}) = \langle 0|\, U^{\dagger}_{\phi(\mathbf{x})} W^{\dagger}_{\boldsymbol{\theta}} \sigma_Z W_{\boldsymbol{\theta}} U_{\phi(\mathbf{x})}\,|0\rangle \tag{8}$$

2.3.1 Strategie Kodowania Danych

Ponieważ chcemy załadować klasyczne wartości zmiennych do kubitów, musimy zdecydować się na odpowiednie odwzorowanie. W przypadku dużej ilości zmiennych konieczna jest selekcja lub redukcja wymiaru (np. za pomocą PCA lub autoencodera).

- **Kodowanie Bazowe (Basis Encoding):** Polega na zamianie wektora danych i wyrażeniu go w postaci bitowej. Wymagałoby to dużej liczby kubitów, aby zachować odpowiednią precyzję dla danych transakcyjnych, dlatego jest to podejście mało skalowalne.

- **Kodowanie Amplitudowe (Amplitude Encoding):** Polega na normalizacji wektora wejściowego $\mathbf{x}$ i zakodowaniu jego elementów x_k jako amplitud stanu kwantowego. Np. wektor $x = (x_0, x_1, x_2, x_3)$ może być przekształcony na stan $|x\rangle = x_0 |00\rangle + x_1 |01\rangle + x_2 |10\rangle + x_3 |11\rangle$. Umożliwia to wykładniczą kompresję danych.
- **Kodowanie Kątowe (Angle Encoding):** Wykorzystuje macierze obrotów $R_X(\theta)$, $R_Y(\theta)$, $R_Z(\theta)$ do zakodowania danych poprzez kąty obrotu. Kodowanie to przekształca zmienne na iloczyny i sumy funkcji sinus i cosinus. W niniejszym opracowaniu wykorzystano to podejście. Przed kodowaniem dane muszą być przeskalowane do wartości z przedziału $[0, \pi]$ (lub $[-\pi, \pi]$).

2.3.2 Ansatz: Parametryzowany Obwód Wariacyjny

Obwód wariacyjny (*Ansatz*), reprezentowany przez operator $W_{\boldsymbol{\theta}}$ w równaniu (4), stanowi serce kwantowej sieci neuronowej. Jest to parametryzowana sekwencja bramek, której celem jest modelowanie skomplikowanej nieliniowej funkcji. Odpowiada to za architekturę warstw ukrytych w klasycznej sieci neuronowej (DNN).

Projektowanie Ansatzu wymaga kompromisu pomiędzy **zdolnością do wyrażania** (ang. *expressibility*) a **trudnością w optymalizacji**. Zbyt skomplikowany Ansatz, choć teoretycznie zdolny do lepszej aproksymacji, może prowadzić do zjawiska tzw. **płaskich krajobrazów kosztu** (ang. *Barren Plateaus*), gdzie gradient funkcji kosztu bliski jest zeru, co uniemożliwia efektywny trening [15].

Najczęściej stosowane Ansätze wykorzystują następujące bloki:

- **Bramki Rotacji (Parametryzowane):** $R_X(\theta), R_Y(\theta), R_Z(\theta)$ – odpowiedzialne za wprowadzanie parametrów optymalizacyjnych $\boldsymbol{\theta}$.
- **Bramki Splątania (Entangling Gates):** Najczęściej CNOT lub CZ – kluczowe dla generowania **splątania** między kubitami. Splątanie umożliwia modelowanie złożonych korelacji między cechami, co stanowi główną potencjalną przewagę QNN nad klasycznymi sieciami.

Struktura Ansatzu może być prosta (np. sekwencja rotacji i jedno splątanie) lub głęboka (wielokrotne powtórzenia bloku Rotacja-Splątanie). W niniejszej pracy zastosowano Ansätze o różnej głębokości, aby zweryfikować, w jakim stopniu stopień splątania wpływa na skuteczność klasyfikacji.

2.3.3 Pomiar i Wartość Oczekiwana (Obserwable)

Ostatnim etapem algorytmu kwantowego jest pomiar, który przekształca stan kwantowy $|\psi(\boldsymbol{\theta})\rangle$ z powrotem na klasyczną wartość numeryczną, używaną do obliczenia funkcji kosztu $C(\boldsymbol{\theta})$ lub wyniku predykcji. Proces ten wymaga wyboru **operatora pomiarowego** (ang. *Observable*), oznaczonego jako H (lub σ_Z w przypadku jednego kubitu). Wynikiem pomiaru jest **wartość oczekiwana** (ang. *expectation value*):

$$\langle H \rangle = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle$$

W kontekście klasyfikacji binarnej, najczęściej stosowanymi opcjami są:

- **Pomiar na Pojedynczym Kubicie (σ_Z):** Wartość oczekiwana jest obliczana na wybranym kubicie (np. ostatnim), przy użyciu operatora Pauliego $\sigma_Z = Z$. Operator ten przyjmuje wartości własne ± 1. Wartość $\langle \sigma_Z \rangle \in [-1, 1]$ jest następnie mapowana na prawdopodobieństwo $[0, 1]$ (np. za pomocą funkcji aktywacji) i interpretowana jako wynik klasyfikacji. W niniejszej pracy zastosowano tę metodę, mierząc wartość oczekiwaną operatora σ_Z na pierwszym kubicie.
- **Pomiar na Wszystkich Kubitach (Global Observable):** Stosuje się iloczyn tensorowy operatorów Pauliego (np. $\sigma_Z \otimes \sigma_Z \otimes \ldots$) na wszystkich kubitach. Taki pomiar jest teoretycznie bardziej informacyjny, ale jest trudniejszy do estymacji i bardziej podatny na szumy w sprzęcie NISQ.
- **Pomiar Operatora Gęstości:** W bardziej złożonych scenariuszach (np. reguły decyzji), H może być operatorem odpowiadającym określonej kombinacji stanów lub reprezentować stan mieszany poprzez operator gęstości, którego użycie jest bardziej skomplikowane w obliczeniach.

Wariacyjne algorytmy kwantowe (VQA), leżące u podstaw QNN, oferują projektantom modeli trzy główne punkty modyfikacji, które mają bezpośredni wpływ na wydajność modelu:

1. **Mapowanie Cech ($\phi(\mathbf{x})$)**: Decyzja o tym, jak klasyczne dane są kodowane w przestrzeń stanów kwantowych (np. kodowanie kątowe, amplitudowe).
2. **Ansatz ($W_{\boldsymbol{\theta}}$)**: Wybór architektury obwodu wariacyjnego, w tym głębokości i typu splątania, co definiuje zdolność modelu do wyrażania złożonych relacji.
3. **Pomiar (H)**: Określenie sposobu, w jaki wynik kwantowy jest konwertowany z powrotem na klasyczną wartość predykcyjną.

Elastyczność w tych trzech obszarach czyni QNN potężnym, choć wciąż eksperymentalnym, narzędziem, pozwalającym na eksplorację nowych nieliniowych relacji w danych, które mogą być trudno dostępne dla klasycznych architektur uczenia maszynowego.

3 Dane: Credit Card Fraud Detection

Na potrzeby eksperymentu wykorzystano publicznie dostępny zbiór danych **Credit Card Fraud Detection**, opublikowany na platformie Kaggle. Zbiór powstał we współpracy przedsiębiorstwa Worldline oraz grupy badawczej *Machine Learning Group* Uniwersytetu Wolnego w Brukseli (ULB) [...]. Zawiera on rzeczywiste transakcje dokonane kartami kredytowymi przez klientów europejskich w ciągu dwóch dni we wrześniu 2013 roku. Spośród 284807 transakcji jedynie 492 przypadki zostały oznaczone jako oszustwa finansowe (klasa 1), co stanowi zaledwie 0.172% wszystkich obserwacji. Zbiór jest zatem silnie niezrównoważony, co stanowi typowe wyzwanie w zadaniach klasyfikacji nadużyć finansowych.

Każda obserwacja opisuje pojedynczą transakcję i obejmuje 30 zmiennych objaśniających oraz zmienną celu:

- **V1–V28** – zanonimizowane cechy powstałe w wyniku transformacji głównych składowych (PCA),
- **Time** – liczba sekund, które upłynęły od pierwszej zarejestrowanej transakcji w zbiorze,
- **Amount** – wartość transakcji w jednostkach pieniężnych,
- **Class** – zmienna celu, oznaczająca przypadki oszustwa (1) oraz transakcje poprawne (0).

3.1 Wstępne przetwarzanie danych

Proces przetwarzania danych obejmował kilka etapów, mających na celu przygotowanie zbioru do dalszego modelowania klasycznego oraz kwantowego. Pierwszym krokiem było wczytanie pełnego zbioru danych *Credit Card Fraud Detection* oraz wydzielenie cech (X) i etykiet klas (y).

Dane zostały następnie podzielone na zbiór treningowy i testowy w proporcji 80:20, wraz z wykorzystaniem parametru `stratify=y`. Umożliwia on zachowanie oryginalnych proporcji liczby transakcji fraudowych i niefraudowych w obu zbiorach, co jest szczególnie istotne przy silnej nierównowadze klas. Dzięki temu zarówno zbiór treningowy, jak i testowy odzwierciedlają rzeczywistą strukturę danych, co pozwala na bardziej wiarygodną ocenę jakości klasyfikacji.

W kolejnym etapie dokonano przetworzenia dwóch zmiennych: `Time` oraz `Amount`. Zmienna `Time` opisuje moment wystąpienia transakcji (liczba sekund od pierwszej rejestracji w zbiorze), natomiast `Amount` określa jej wartość pieniężną. Obie te zmienne mogą zawierać wartości odstające, dlatego zostały poddane transformacji z wykorzystaniem metody *RobustScaler*. Metoda ta skaluje dane w oparciu o medianę i kwartyle, co pozwala ograniczyć wpływ obserwacji skrajnych. Transformacja została przeprowadzona oddzielnie dla każdej zmiennej, przy czym dopasowanie skalerów (`fit`) wykonano wyłącznie na zbiorze treningowym, a następnie zastosowano je do obu zbiorów (`transform`), co zapobiega tzw. *data leakage*.

Po przekształceniu zmiennych `Time` i `Amount`, dane zostały uzupełnione o ich zeskalowane odpowiedniki (`scaled_time`, `scaled_amount`), natomiast oryginalne kolumny zostały usunięte.

Ze względu na bardzo silną nierównowagę klas w zbiorze, zastosowano metodę *RandomUnderSampler*. Polega ona na losowym usunięciu części obserwacji klasy dominującej (transakcji prawidłowych), w celu zrównoważenia liczby przykładów obu klas. Wybór tej metody podyktowany był zarówno względami obliczeniowymi — mniejszy zbiór danych pozwala na efektywniejsze uczenie modeli kwantowych — jak i metodologicznymi, ponieważ redukcja liczby przykładów nie prowadzi do sztucznego generowania danych, jak ma to miejsce w przypadku metod nadpróbkowania (ang. *oversampling*).

Ostateczny zbiór treningowy zawierał po 394 przypadki klasy 0 i 1 (30 zmiennych i 788 wierszy). Zbiór testowy natomiast po 98 przypadków każdej klasy (30 zmiennych i 196 wierszy).

Zachowano również pełny, niezbalansowany zbiór testowy, liczący 56962 wiersze (z czego 98 przypadków to zdarzenia fraudowe). Zostanie on wykorzystany do oceny, jak dobrze model radzi sobie z uogólnieniem w warunkach zbliżonych do rzeczywistych. Ponieważ pełny zbiór testowy zawiera te same 98 przypadków fraudowych co zbiór zbalansowany, czułość modeli na klasie oszustw (ang. *recall*) pozostaje niezmieniona; tym, co ujawnia pełny zbiór, jest natomiast odsetek fałszywych alarmów wśród ogromnej większości transakcji prawidłowych. Pozwala on zatem zweryfikować, jak często oraz które transakcje legalne modele błędnie klasyfikują jako fraudowe.

3.2 Analiza jakości zmiennych

W celu wstępnej oceny zdolności dyskryminacyjnej poszczególnych cech przeprowadzono eksploracyjny, jednozmiennowy przegląd zmiennych według następującej procedury:

1. wybierz jedną zmienną,
2. przeskaluj ją operacją `MinMaxScaler()`,
3. zbuduj model regresji logistycznej na podstawie tej jednej zmiennej,
4. oblicz statystykę Gini dla modelu.

Otrzymaną listę zmiennych uszeregowano malejąco według współczynnika Gini. Poniżej przedstawiono kilka najlepszych zmiennych:

Powyższy ranking traktowany jest wyłącznie jako analiza pomocnicza. Selekcja oparta na modelu jednozmiennowym zakłada bowiem liniową (w sensie logitu) zależność między cechą a klasą i nie uwzględnia zależności między samymi cechami. Dodatkowo różnice między czołowymi zmiennymi są niewielkie — V12 (0.9094) i V14 (0.9086) są praktycznie nierozróżnialne.

Z tego względu wiążącego wyboru pojedynczej zmiennej dokonano niezależną metodą, opartą na modelu drzewiastym. Wygenerowano model XGBoost

Tabela 1: Współczynniki istotności cech (Gini) dla kluczowych zmiennych — analiza eksploracyjna

Zmienna	**Współczynnik Gini**	**Ranking**
V12	0.9094	1
V14	0.9086	2
V11	0.8802	3
V4	0.8485	4
V3	0.8248	5
V10	0.8183	6
V16	0.7020	7
V7	0.6975	8
V17	0.6738	9
V9	0.6626	10

o maksymalnej głębokości 1 i jednym estymatorze, trenowany wyłącznie na zbiorze treningowym. Taki „pieniek decyzyjny"wybiera cechę o największym zysku informacyjnym pojedynczego podziału, niezależnie od założenia o liniowości. Kryterium to wskazało zmienną $V14$ (próg podziału ≈ -3.616), która została wykorzystana jako cecha wejściowa jednokubitowego modelu kwantowego.

4 Wyniki eksperymentów

4.1 Modele Klasyczne jako Benchmark

W ramach wstępnej weryfikacji potencjału różnych podejść do modelowania ryzyka kredytowego, zastosowano szereg klasycznych algorytmów **uczenia nadzorowanego (Supervised Learning)**. Celem tego etapu było ustalenie bazowego poziomu wydajności (benchmarku), jaki można osiągnąć za pomocą powszechnie stosowanych metod, stanowiących punkt odniesienia dla kolejnych eksperymentów z kwantowymi sieciami neuronowymi.

Problem zaklasyfikowano jako **klasyfikację binarną**, w której algorytm musi znaleźć optymalną funkcję $f : X \to Y$ zdolną do uchwycenia ukrytych zależności w danych wejściowych X i najtrafniejszego przewidywania wartości zmiennej celu Y (np. 0 – brak ryzyka, 1 – ryzyko defaultu).

Zastosowano pięć klasyfikatorów z biblioteki Scikit-learn: **Regresja Logistyczna**, **Las Losowy**, **XGBoost**, **SVM** z jądrem RBF, oraz **Wielowarstwowy Perceptron (MLP)**. Przed modelowaniem, dane wejściowe zostały ustandaryzowane za pomocą metody `StandardScaler`. Podział na zbiór treningowy i testowy zrealizowany został w stosunku 80:20.

Wybrano zróżnicowany zestaw modeli klasyfikacyjnych, a ich hiperparametry zostały dostosowane w celu optymalizacji wydajności, zwłaszcza w kontekście danych niezbalansowanych i potencjalnie złożonych zależności.

4.1.1 Regresja Logistyczna

Model ten został skonfigurowany następująco:

- *max_iter=1000*: Ustawia maksymalną liczbę iteracji dla algorytmu optymalizacyjnego. Zwiększono do 1000, aby zapewnić zbieżność modelu.
- *class_weight=balanced*: Kluczowy parametr mający na celu automatyczne dostosowanie wag proporcjonalnie do odwrotności częstości występowania klas w danych treningowych. Pomaga to zrównoważyć wpływ mniejszościowej klasy w przypadku niezbalansowanych danych.
- *solver=lbfgs*: Określa algorytm optymalizacji do użycia. "lbfgs"jest domyślnym i często skutecznym algorytmem.

4.1.2 Las Losowy

- *n_estimators=300*: Liczba drzew w lesie. Wyższa wartość (300 zamiast domyślnej 100) zwiększa stabilność i dokładność kosztem czasu obliczeń, pomagając zredukować wariancję.
- *_depth=None*: Drzewa są rozwijane w pełni, dopóki wszystkie liście nie będą czyste lub dopóki liść nie będzie miał mniej niż *min_samples_split*. Zwiększa potencjał dopasowania, ale może wymagać uwagi pod kątem przetrenowania.

- *class_weight=balanced*: Podobnie jak w Regresji Logistycznej, zrównoważenie wag klas jest stosowane w procesie budowania drzew, minimalizując wpływ niezbalansowania.
- *n_jobs=-1*: Użycie wszystkich dostępnych rdzeni procesora do równoległego budowania drzew, co przyspiesza trening.
- *random_state=42*: Ziarno losowości, zapewniające replikowalność wyników.

4.1.3 XGBoost

- *n_estimators=400*: Liczba etapów wzmacniania (liczba drzew). Wartość 400 stanowi kompromis między dokładnością a ryzykiem przetrenowania.
- *max_depth=5*: Maksymalna głębokość każdego drzewa. Ograniczenie to pomaga kontrolować złożoność i zapobiegać przetrenowaniu.
- *learning_rate=0.05*: Wpływ każdego pojedynczego drzewa na wynik. Niższa wartość (0.05) wymaga więcej estymatorów, ale zazwyczaj prowadzi do bardziej solidnego i uogólnialnego modelu.
- *subsample=0.9*: Kontrola nad losowym próbkowaniem danych (wiersze i kolumny) na etapie budowy drzewa, co zwiększa różnorodność modelu i pomaga zapobiegać przetrenowaniu.
- *objective=binary:logistic, eval_metric=logloss*: Określenie zadania klasyfikacji binarnej i użycie Log Loss jako miary oceny.
- *n_jobs=-1, random_state=42, use_label_encoder=False*: Ustawienia techniczne dla przyspieszenia i replikowalności.

4.1.4 Maszyna Wektorów Nośnych

- *kernel=rbf*:Użycie promieniowego jądra bazowego (Radial Basis Function), umożliwiającego tworzenie nieliniowej granicy decyzyjnej w przestrzeni cech.
- *probability=True*:Umożliwienie modelu zwracania oszacowań prawdopodobieństwa (wymagane do obliczania metryk opartych na prawdopodobieństwie, takich jak ROC-AUC).
- *class_weight=balanced*: Ponowne zastosowanie automatycznego ważenia klas, aby skompensować niezbalansowanie danych.
- *C=1.0*:Parametr regularyzacji. Wartość 1.0 jest domyślna i stanowi kompromis między dużą marżą a małą liczbą błędów klasyfikacji.

4.1.5 Wielowarstwowy Perceptron (MLP)

Prosta sieć neuronowa z dwiema ukrytymi warstwami:

- *hidden_layer_sizes=(64, 32)*: Architektura sieci: dwie warstwy ukryte o wielkości odpowiednio 64 i 32 neurony.
- *activation=relu*: Funkcja aktywacji Rectified Linear Unit (ReLU) jest używana w warstwach ukrytych ze względu na jej efektywność.
- *solver=adam*: Algorytm optymalizacji (Adam) do aktualizacji wag sieci, znany ze swojej efektywności w praktyce.
- *max_iter=500*: Maksymalna liczba epok treningowych.
- *early_stopping=True*: Kluczowy mechanizm zapobiegający przetrenowaniu. Trening zatrzyma się, jeśli wynik na zbiorze walidacyjnym nie poprawi się przez pewną liczbę epok.
- *random_state=42*: Ziarno losowości, zapewniające replikowalność inicjalizacji wag i podziału na zbiory walidacyjne (dla *early_stopping*).

4.2 Wyzwanie Niezrównoważonych Danych i Metodologia Oceny

Istotnym wyzwaniem w modelowaniu ryzyka kredytowego jest **silna nierównowaga klas** (ang. *class imbalance*). W związku z tym, każdy model został poddany ocenie w czterech scenariuszach, różniących się zbiorem treningowym i testowym:

1. Trening na zbiorze **pełnym (full)** → Test na zbiorze pełnym (full)
2. Trening na zbiorze **pełnym (full)** → Test na zbiorze zredukowanym (undersampled)
3. Trening na zbiorze **zredukowanym (undersampled)** → Test na zbiorze pełnym (full)
4. Trening na zbiorze **zredukowanym (undersampled)** → Test na zbiorze zredukowanym (undersampled)

W ocenie wydajności pominięto szczegóły hiperparametrów i skupiono się na kluczowych metrykach dla problemów z nierównowagą klas: **Precyzja**, **Czułość**, oraz **miara F1**.

4.2.1 Wyniki klasyfikatorów uczenia maszynowego

Dla zachowania czytelności i poprawy składu dokumentu, pierwotna tabela została podzielona na dwie mniejsze. Tabela 2 prezentuje wyniki dla modeli liniowych i ensemble (Regresja Logistyczna, Las Losowy i XGBoost), natomiast Tabela 3 przedstawia wyniki dla klasyfikatorów nieliniowych (SVM i MLP).

Tabela 2: Wyniki klasyfikatorów klasycznych: Modele Liniowe i Ensemble

Model	Trening	Test	Acc.	Prec.	Rec.	F1	ROC-AUC
Logistic Regression							
LogisticReg	full	full	0.976	0.061	0.918	0.114	0.972
LogisticReg	full	under	0.949	0.978	0.918	0.947	0.972
LogisticReg	under	full	0.971	0.052	0.918	0.098	0.975
LogisticReg	under	under	0.949	0.978	0.918	0.947	0.977
Random Forest							
RandomFr	full	full	1.000	0.961	0.755	0.846	0.957
RandomFr	full	under	0.878	1.000	0.755	0.860	0.957
RandomFr	under	full	0.965	0.043	0.918	0.082	0.977
RandomFr	under	under	0.949	0.978	0.918	0.947	0.977
XGBoost							
XGBoost	full	full	0.999	0.856	0.847	0.851	0.979
XGBoost	full	under	0.923	1.000	0.847	0.917	0.976
XGBoost	under	full	0.909	0.017	0.929	0.034	0.983
XGBoost	under	under	0.913	0.901	0.929	0.915	0.983

4.2.2 Analiza Wniosków Wstępnych

Analizując Tabela 2 i Tabela 3 można zaobserwować kilka kluczowych wzorców, typowych dla danych z silną nierównowagą klas:

- **Czułości (Recall):** Modele trenowane na pełnym zbiorze i testowane na pełnym zbiorze (np. Logistic Regression, pierwszy wiersz) osiągają bardzo wysoką Czułość (Rec. = 0.918), ale ekstremalnie niską Precyzję (Prec. = 0.061). Jest to wynikiem tendencji modelu do klasyfikowania większości obserwacji jako należące do klasy mniejszościowej w celu uniknięcia fałszywych negatywów, co sztucznie zawyża metrykę Acc. (0.976).
- **Efekt Undersamplingu: Undersampling** jest efektywny w zmuszeniu modeli do uczenia się istotnych cech klasy mniejszościowej. Modele trenowane i testowane na zbiorach zredukowanych (np. Logistic Regression, czwarty wiersz) osiągają bardzo wysokie F1 (0.947), ponieważ na zrównoważonym zbiorze testowym **Precyzja** i **Czułość** są zbalansowane (odpowiednio 0.978 i 0.918).
- **Najlepsza Wydajność (F1):** Najwyższą metrykę F1 dla realistycznego scenariusza (`full` → `undersampled`) osiągnął model **XGBoost** (F1 = 0.917). W scenariuszu optymalnym (`undersampled` → `undersampled`) najlepszą wydajność osiąga **MLP** (F1 = 0.941). Modele ensemble i neuronowe wydają się najlepiej radzić sobie z charakterystyką tych danych.

Tabela 3: Wyniki klasyfikatorów klasycznych: Modele Nieliniowe (SVM i MLP)

Model	Trening	Test	Acc.	Prec.	Rec.	F1	ROC-AUC
SVM (RBF)							
SVM	full	full	0.999	0.662	0.480	0.556	0.973
SVM	full	under	0.740	1.000	0.480	0.648	0.976
SVM	under	full	0.957	0.035	0.918	0.068	0.973
SVM	under	under	0.949	0.978	0.918	0.947	0.974
MLP							
MLP	full	full	0.999	0.909	0.714	0.800	0.977
MLP	full	under	0.857	1.000	0.714	0.833	0.977
MLP	under	full	0.969	0.048	0.898	0.091	0.969
MLP	under	under	0.944	0.989	0.898	0.941	0.972

Wnioski te stanowią solidną bazę do porównania z wynikami kwantowych sieci neuronowych, pozwalając ocenić, czy złożoność obliczeń kwantowych przynosi wymierną poprawę metryk F1 lub Precyzji w tym wymagającym zadaniu klasyfikacji.

Generowanie i uczenie kwantowych sieci neuronowych odbywać się będzie tylko i wyłącznie na zbiore treningowym (under) po zastosowaniu undersamplingu. Warto sprawdzić jak radziały sobie modele ML w tym zakresie 4:

Statystyka F1 oraz Precision jasno wskazują, że żaden model na małym, zbalansowanym zbiorze treningowym nie radzi sobie z przeuczeniem i problemem nadmiernego przypisywania fraudów.

Tabela 4: Wyniki klasyfikatorów klasycznych trenowanych na zbiorze zredukowanym (under)

Model	Trening	Test	Acc.	Prec.	Rec.	F1	ROC-AUC
Logistic Regression							
LogisticReg	under	full	0.971	0.052	0.918	0.098	0.975
LogisticReg	under	under	0.949	0.978	0.918	0.947	0.977
Random Forest							
RandomFr	under	full	0.965	0.043	0.918	0.082	0.977
RandomFr	under	under	0.949	0.978	0.918	0.947	0.977
XGBoost							
XGBoost	under	full	0.909	0.017	0.929	0.034	0.983
XGBoost	under	under	0.913	0.901	0.929	0.915	0.983
SVM (RBF)							
SVM	under	full	0.957	0.035	0.918	0.068	0.973
SVM	under	under	0.949	0.978	0.918	0.947	0.974
MLP							
MLP	under	full	0.969	0.048	0.898	0.091	0.969
MLP	under	under	0.944	0.989	0.898	0.941	0.972

4.3 Klasyczna Sieć Neuronowa: PyTorch: architektura i parametryzacja

Model klasycznej sieci neuronowej, zrealizowany w środowisku PyTorch, został wybrany jako dodatkowa klasyczna możliwosc. Związane jest to z dość łatwą obsługą Pennylane (pakietu do kwantowych obwodów) i PyTorch, co daje unikalną możliwość szybkiej budowy tzw. sieci hybrydowych, w których poszczególne warstwy mogą być realizowane jako klasyczne bądź kwantowe.

Dla celów porównanwczych przygotowano jeszcze jeden model klasycznej sieci neuronowej.

Zaprojektowana architektura obejmowała cztery warstwy liniowe o następującej konfiguracji neuronów:

- Warstwa Wwejściowa: 30 neuronów (liczba cech wejściowych).
- Warstwy ukryte: dwie warstwy o 8 i 4 neuronach.
- Warstwa wyjściowa: pojedynczy neuron, zwracający prawdopodobieństwo w przedziale $[0, 1]$ po zastosowaniu funkcji aktywacji sigmoidalnej.

Pomiędzy warstwami wykorzystano funkcję aktywacji **ReLU (Rectified Linear Unit)**, która wprowadza nieliniowość i pozwala na efektywne przyspie-

szenie procesu uczenia. Całkowita liczba parametrów do optymalizacji w tym modelu wynosiła **289**.

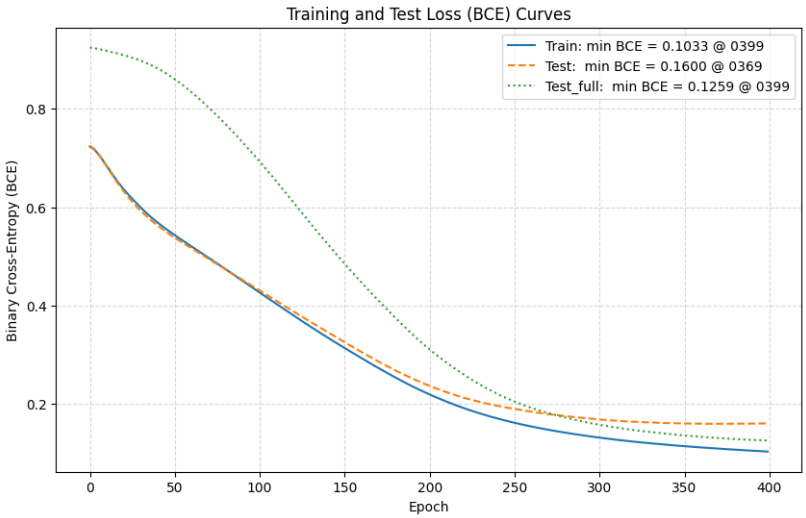
Jako funkcję straty wybrano **Entropię Krzyżową dla klasyfikacji binarnej** (ang. *Binary Cross Entropy*, BCE), definiowaną równaniem 9, a optymalizację przeprowadzono z wykorzystaniem algorytmu **Adam**.

$$\mathrm{BCE}(y, \hat{y}) = -\big[\, y \log(\hat{y}) + (1 - y) \log\big(1 - \hat{y}\big)\big], \tag{9}$$

gdzie y to rzeczywista etykieta klasy, a $\hat{y}$ to predykowane prawdopodobieństwo.

Proces uczenia przeprowadzono przez 400 epok, uzyskując następujące końcowe wartości funkcji straty:

- Zbiór treningowy: $\mathrm{BCE}_{\mathrm{train}} = 0.1033$,
- Zbiór walidacyjny (undersampled): $\mathrm{BCE}_{\mathrm{test\_us}} = 0.1600$,
- Pełny zbiór testowy: $\mathrm{BCE}_{\mathrm{test\_full}} = 0.1259$.



Rysunek 3: Ilustracja funkcji straty w procesie uczenia klasycznej sieci neuronowej.

4.4 Ocena Wydajności na Zbiorach Testowych

Po zakończeniu procesu uczenia przeprowadzono ocenę jakości klasyfikatora na dwóch niezależnych zbiorach testowych: zrównoważonym (po undersamplingu) oraz pełnym (silnie niezbalansowanym).

4.4.1 Macierz Pomyłek (Undersampled Test Set)

Poniżej przedstawiono macierz pomyłek dla zrównoważonego podzbioru testowego.

Tabela 5: Macierz pomyłek dla MLP na zrównoważonym zbiorze testowym

	Pred: Pozytywna (1)	**Pred: Negatywna (0)**
Real: Pozytywna (1)	96 (TP)	2 (FN)
Real: Negatywna (0)	9 (FP)	89 (TN)

Na podstawie Tabela 5 obliczono następujące miary skuteczności:

- **Accuracy (Dokładność):** 0.9439
- **Precision (Precyzja):** 0.9780 (Wśród przewidywanych pozytywnych, 97.8% było poprawnych.)
- **Recall (Czułość):** 0.9082 (Model wykrył 90.8% rzeczywistych przypadków pozytywnych.)
- **F1-score:** 0.9418

4.4.2 Wyniki na Pełnym i Zrównoważonym Zbiorze

W Tabeli 7 zestawiono miary skuteczności uzyskane na obu zbiorach testowych. Macierz pomyłek dla pełnego zbioru testowego (≈ 57000 obserwacji) wskazuje na silne dominowanie klasy negatywnej (TN $\gg$ TP):

Tabela 6: Macierz pomyłek dla MLP na pełnym, niezbalansowanym zbiorze testowym

	Pred: Pozytywna (1)	**Pred: Negatywna (0)**
Real: Pozytywna (1)	54820	2044
Real: Negatywna (0)	9	89

Tabela 7: Porównanie metryk skuteczności klasyfikatora MLP na zbiorze zrównoważonym i pełnym

Zbiór Testowy	Accuracy	Precision	Recall	F1-score
Zrównoważony (Under)	0.9439	0.9780	0.9082	0.9418
Pełny Zbiór	0.9640	0.0417	0.9082	0.0798

4.5 Podsumowanie i wnioski

Model MLP, trenowany na zrównoważonym podzbiorze i oceniany na nim, uzyskał bardzo zrównoważone wyniki klasyfikacji (F1 ≈ 0.94), potwierdzając, że zaproponowana, uproszczona architektura sieci neuronowej skutecznie uogólnia wzorce. Wysoka **Precyzja** na zrównoważonym zbiorze (0.9780) oznacza, że model generuje minimalną liczbę fałszywych alarmów, podczas gdy wysoka **Czułość** (0.9082) świadczy o dobrej skuteczności w identyfikacji rzeczywistych przypadków klasy rzadkiej.

Jednakże, wyniki na pełnym, niezbalansowanym zbiorze testowym (0.9640 Acc.; 0.0798 F1) jasno pokazują pułapkę metryki Accuracy w przypadku niezbalansowanych danych. Mimo wysokiego Recall (0.9082), ekstremalnie niska **Precyzja** (0.0417) na pełnym zbiorze oznacza, że model, w celu zachowania wysokiej czułości, generuje dużą liczbę Fałszywych Pozytywów (FP), co dyskwalifikuje ten wynik w praktycznym zastosowaniu.

Dalsze prace powinny koncentrować się na potencjalnym dostosowaniu progu decyzyjnego lub wprowadzeniu wag klas, aby ograniczyć liczbę fałszywych alarmów (FP) przy zachowaniu wysokiego Recall. W kontekście niniejszej pracy, model ten stanowi **klarowny benchmark** (F1 = 0.9418 na zbiorze zrównoważonym), z którym porównywane będą osiągi kwantowych sieci neuronowych.

5 Implementacja Wariacyjnego Klasyfikatora Kwantowego (VQC)

W oparciu o teoretyczne ramy wariacyjnych algorytmów kwantowych (VQA), zdefiniowanych w Sekcji 2.2 (przyjęte założenie), zaprojektowano i zaimplementowano serię kwantowych sieci neuronowych (QNN) do klasyfikacji ryzyka kredytowego. Modele te zostały skonstruowane w architekturze hybrydowej z wykorzystaniem bibliotek `PennyLane` oraz `PyTorch` do optymalizacji.

5.1 Przygotowanie Danych dla QNN

Przed implementacją kwantowych modeli, konieczne było dostosowanie wektorów cech $\mathbf{x}$ do wymogów **kodowania kątowego**, tj. przekształcenia ich na parametry bramek obrotu.

5.1.1 Skalowanie Danych do Zakresu Kątów

Dane wejściowe muszą być przeskalowane do zakresu $[0, \pi]$, ponieważ π radiany (180°) odpowiadają połowie obrotu na Sferze Blocha i umożliwiają pełną eksplorację stanu pojedynczego kubitu. Zastosowano w tym celu transformację **Min-Max Scaler**:

$$\tilde{x}_i = \pi \frac{x_i - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})}$$

gdzie $\tilde{x}_i \in [0, \pi]$. Przetworzone wartości $\tilde{x}_i$ są używane jako kąty rotacji w bramkach kodujących dane.

5.1.2 Wybór Cech

Warianty modeli QNN zostały zaimplementowane z różną liczbą kubitów.

- **Model Jednokubitowy (QNN1):** Wymagał wyboru pojedynczej, najbardziej informacyjnej zmiennej. Na podstawie analizy klasycznych modeli regresji logistycznej i XGBoost, wybrano zmienną **V14**, która wykazała najwyższą istotność predykcyjną (najwyższy współczynnik Giniego).
- **Modele Wielokubitowe (QNN2, QNN6):** Wykorzystano zestaw czterech cech: **V14**, **V10**, **V12**, **V3**, które również charakteryzowały się wysoką istotnością.
- **Modele z PCA (QNN3, QNN4, QNN5):** Zaimplementowano alternatywne podejście polegające na redukcji wymiarowości do pięciu głównych składowych (PCA), które wyjaśniały 90% wariancji, a następnie przeskalowaniu ich do zakresu kątów.

5.2 Model Bazowy: Jednokubitowy Klasyfikator (QNN1)

Zaprojektowany model QNN1 jest minimalną strukturą VQC, mającą służyć jako punkt odniesienia (benchmark). Działa na pojedynczym kubicie z trzema parametrami wariacyjnymi i jednym klasycznym obciążeniem (bias).

5.2.1 Architektura Obwodu

Model QNN1 definiują trzy kluczowe etapy (Rysunek 4):

1. **Kodowanie Danych (Feature Map):** Wejściowa zmienna $\tilde{x} = \mathbf{V14}$ jest ładowana do kubitu poprzez bramkę obrotu $\mathbf{R_X}(\tilde{x})$.
2. **Ansatz (Parametry Wariacyjne):** Obwód wariacyjny $W_{\boldsymbol{\theta}}$ składa się z pojedynczej, parametryzowanej bramki obrotu $\mathbf{Rot}(\theta_1, \theta_2, \theta_3)$. Wprowadza ona **trzy optymalizowalne parametry kwantowe** $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$.
3. **Pomiar (Observable):** Wartość oczekiwana $\langle H \rangle$ jest mierzona przy użyciu **operatora Pauliego** σ_Z na kubicie zerowym: $H = \mathbf{Z}_0$. Wynik pomiaru należy do przedziału $[-1, 1]$.

Rysunek 4: Schemat logiczny jednokubitowego kwantowego klasyfikatora QNN1.

Pełny obwód kwantowy jest zdefiniowany funkcją `quantum_net` i ma postać:

$$\mathbf{f}(\boldsymbol{\theta}, \tilde{x}) = \langle 0 | \, \mathbf{Rot}^{\dagger}(\boldsymbol{\theta}) \mathbf{R_X}^{\dagger}(\tilde{x}) \mathbf{Z}_0 \mathbf{R_X}(\tilde{x}) \mathbf{Rot}(\boldsymbol{\theta}) \, | 0 \rangle \tag{10}$$

Ponieważ model kwantowy zwraca wartość oczekiwaną $\in [-1, 1]$, do predykcji końcowej dodano klasyczny parametr obciążenia b:

$$\text{Predykcja} = \mathbf{f}(\boldsymbol{\theta}, \tilde{x}) + b$$

Proces uczenia polegał na jednoczesnej optymalizacji czterech parametrów: trzech kwantowych ($\boldsymbol{\theta}$) i jednego klasycznego (b).

5.2.2 Trening i Optymalizacja

Do treningu wykorzystano zbiór danych zrównoważony metodą undersamplingu.

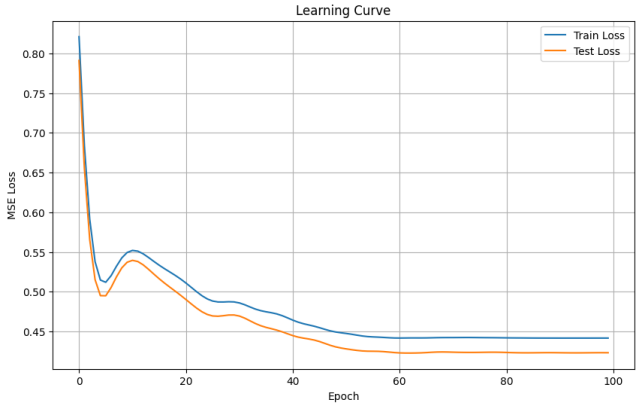
- **Funkcja Kosztu:** Zastosowano **kwadratową funkcję straty** (Mean Squared Error, MSE), optymalną dla wartości oczekiwanych $\in [-1, 1]$.
- **Optymalizator:** Algorytm **Adam** z krokiem uczenia $\alpha = 0.1$.
- **Epoki:** Trening prowadzono przez 100 epok.

Wyniki metryk skuteczności dla QNN1 są przedstawione w Tabeli 8, a macierze pomyłek w Tabelach 9 i 10.

Tabela 8: Porównanie metryk skuteczności jednokubitowego klasyfikatora kwantowego (QNN1) na zbiorze zrównoważonym i pełnym

Zbiór Testowy	**Accuracy**	**Precision**	**Recall**	**F1-score**
Zrównoważony (Under)	0.9336	1.0000	0.8673	0.9289
Pełny Zbiór	0.9936	0.1954	0.8673	0.3189

Jednokubitowy klasyfikator kwantowy QNN1, wykorzystując tylko jedną zmienną (**V14**) i minimalną liczbę parametrów (4, w tym bias), osiągnął wynik **F1-score (**0.9289**)** bardzo zbliżony do klasycznej sieci trenowanej na wszystkich



Rysunek 5: Ilustracja funkcji straty w procesie uczenia kwantowej sieci neuronowej QNN1.

Tabela 9: Macierz pomyłek dla QNN1 na pełnym, niezbalansowanym zbiorze testowym

	Pred: Negatywna (0)	**Pred: Pozytywna (1)**
Real: Negatywna (0)	56514	350
Real: Pozytywna (1)	13	85

30 zmiennych. Wskaźnik **Precision** (1.0000) na zrównoważonym zbiorze oznacza brak Fałszywych Pozytywów (FP), co wskazuje na silną separowalność cechy **V14** w przestrzeni kwantowej.

5.3 Wielokubitowe Klasyfikatory Kwantowe

Zaimplementowano serię modeli wielokubitowych (QNN2 - QNN6) w celu zbadania wpływu liczby kubitów, wyboru cech oraz głębokości i ekspresyjności obwodu na wydajność.

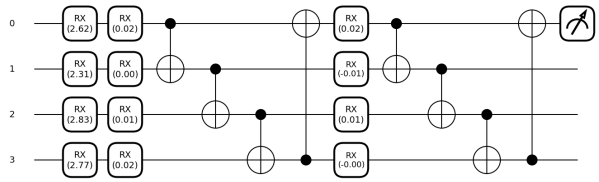
5.3.1 Wariant 1: QNN2 (4 kubity, BasicEntanglerLayers)

Model wykorzystujący cztery cechy (**V14**, **V10**, **V12**, **V3**) oraz architekturę opartą na `BasicEntanglerLayers` z dwiema warstwami. Obwód ten charak-

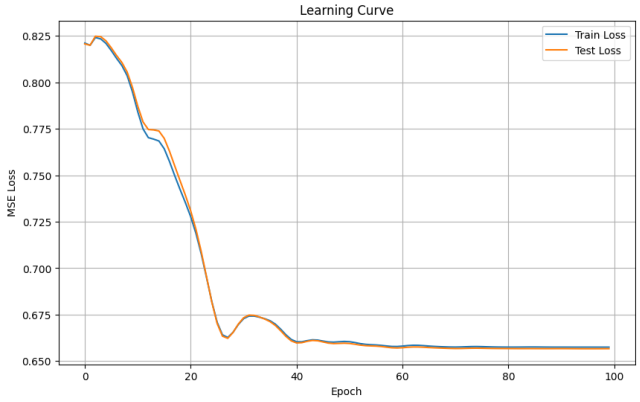
Tabela 10: Macierz pomyłek dla QNN1 na zbalansowanym zbiorze testowym

	Pred: Negatywna (0)	**Pred: Pozytywna (1)**
Real: Negatywna (0)	98	0
Real: Pozytywna (1)	13	85

teryzuje się parametrycznymi bramkami obrotu z jednym kątem i bramkami splątującymi CNOT (Rysunek 6). Całkowita liczba parametrów optymalizowanych wynosiła 9 (8 kwantowych + 1 klasyczny bias). Pomiar wykonano na pierwszym kubicie ($\mathbf{Z}_0$).



Rysunek 6: Schemat logiczny czterokubitowego kwantowego klasyfikatora QNN2.



Rysunek 7: Ilustracja funkcji straty w procesie uczenia kwantowej sieci neuronowej QNN2.

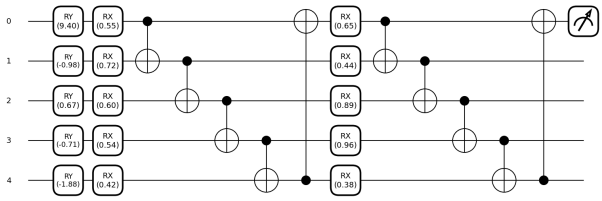
Tabela 11: Porównanie metryk skuteczności QNN2 (4 kubity) na zbiorze zrównoważonym i pełnym

Zbiór Testowy	**Accuracy**	**Precision**	**Recall**	**F1-score**
Zrównoważony (Under)	0.9285	0.9883	0.8673	0.9239
Pełny Zbiór	0.9843	0.0880	0.8673	0.1599

5.3.2 Warianty 2-4: QNN3, QNN4, QNN5 (PCA, Różne Architektury)

W wariantach QNN3, QNN4 i QNN5 wykorzystano pięć głównych składowych PCA (wyjaśniających 90% wariancji) jako wejścia na pięć kubitów.

- **QNN3 (5 kubitów, BasicEntanglerLayers, 2 warstwy):** Posiadał 11 parametrów. Osiągnięte wyniki były niezadowalające, co sugeruje, że wektory cech z transformacji PCA nie są optymalne dla tego typu architektury kwantowej (Tabela 12).
- **QNN4 (5 kubitów, StronglyEntanglingLayers, 2 warstwy):** Wprowadzono bardziej ekspresyjne bramki obrotu (z trzema kątami), co zwiększyło liczbę parametrów do 31. Wyniki uległy nieznacznej poprawie, ale nadal były słabsze niż QNN1 (Tabela 13).
- **QNN5 (5 kubitów, StronglyEntanglingLayers, 4 warstwy):** Zwiększenie głębokości obwodu do 4 warstw spowodowało wzrost liczby parametrów do 61. Poprawa była marginalna (Tabela 14).



Rysunek 8: Schemat logiczny pięciokubitowego klasyfikatora QNN3/QNN4/QNN5 (wizualizacja ogólna).

Tabela 12: Porównanie metryk skuteczności QNN3 (PCA, 5 kubitów)

Zbiór Testowy	**Accuracy**	**Precision**	**Recall**	**F1-score**
Zrównoważony (Under)	0.1734	0.2538	0.3367	0.2894
Pełny Zbiór	0.009	0.0005	0.3367	0.001

Tabela 13: Porównanie metryk skuteczności QNN4 (PCA, Strongly, 2 warstwy)

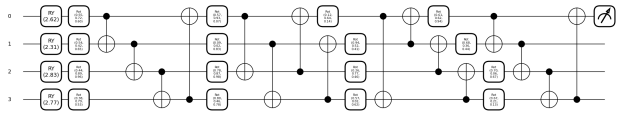
Zbiór Testowy	**Accuracy**	**Precision**	**Recall**	**F1-score**
Zrównoważony (Under)	0.6683	0.6299	0.8163	0.7111
Pełny Zbiór	0.5157	0.0028	0.8163	0.0057

Tabela 14: Porównanie metryk skuteczności QNN5 (PCA, Strongly, 4 warstwy)

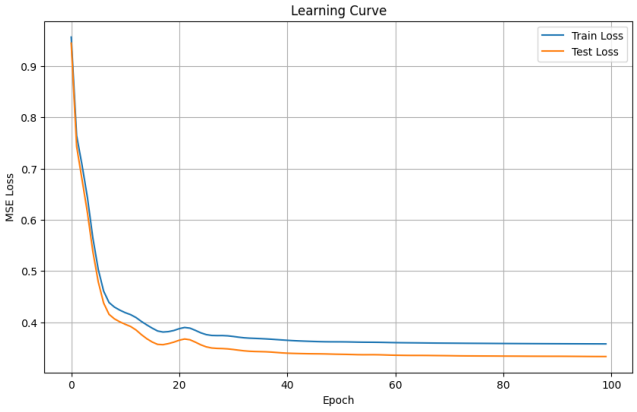
Zbiór Testowy	Accuracy	Precision	Recall	F1-score
Zrównoważony (Under)	0.6734	0.8541	0.4183	0.5616
Pełny Zbiór	0.9342	0.010	0.4183	0.0214

5.3.3 Wariant 5: QNN6 (4 kubity, StronglyEntanglingLayers, 4 warstwy)

Wariant ten powrócił do selekcji cech (**V14**, **V10**, **V12**, **V3**), zwiększając jednocześnie ekspresyjność ansatzu poprzez użycie `StronglyEntanglingLayers` oraz zwiększając głębokość do 4 warstw. Model optymalizował 49 parametrów (48 kwantowych + 1 bias).



Rysunek 9: Schemat logiczny czterokubitowego kwantowego klasyfikatora QNN6.



Rysunek 10: Ilustracja funkcji straty w procesie uczenia kwantowej sieci neuronowej QNN6.

QNN6 osiągnął najlepsze metryki spośród wszystkich testowanych modeli kwantowych. Jego **F1-score** (0.9347) na zbiorze zrównoważonym jest najwyższy, a **Precision** (1.0000) na tym samym zbiorze oznacza perfekcyjne uniknięcie Fałszywych Pozytywów.

Tabela 15: Porównanie metryk skuteczności QNN6 (4 kubity, 4 warstwy) na zbiorze zrównoważonym i pełnym

Zbiór Testowy	**Accuracy**	**Precision**	**Recall**	**F1-score**
Zrównoważony (Under)	0.9387	1.0000	0.8775	0.9347
Pełny Zbiór	0.9979	0.4574	0.8775	0.6013

Tabela 16: Macierz pomyłek dla QNN6 na pełnym, niezbalansowanym zbiorze testowym

	Pred: Negatywna (0)	**Pred: Pozytywna (1)**
Real: Negatywna (0)	56762	102
Real: Pozytywna (1)	12	86

5.4 Podsumowanie Wyników Kwantowych

Przeprowadzona analiza wykazała, że architektura kwantowa, odpowiednio dobrana do danych, może osiągnąć wysoką skuteczność w zadaniu klasyfikacji ryzyka kredytowego, szczególnie w kontekście niezrównoważenia klas.

- **Efektywność Minimalistyczna (QNN1):** Model jednokubitowy, wykorzystujący tylko jedną, krytyczną cechę (**V14**), osiągnął $\mathbf{F1-score} = 0.9289$, demonstrując, że odpowiednie kwantowe kodowanie danych (kodowanie kątowe) może wydobyć silną separowalność nawet z minimalnego wektora cech.
- **Optymalna Architektura (QNN6):** Najlepszą wydajność osiągnął model QNN6 (4 kubity, selekcja cech, StronglyEntanglingLayers, 4 warstwy), uzyskując $\mathbf{F1-score} = 0.9347$ i utrzymując perfekcyjną precyzję ($\mathbf{Precision} = 1.0000$) na zbiorze zrównoważonym. Zwiększenie ekspresyjności i głębokości obwodu w połączeniu z odpowiednią selekcją cech okazało się kluczowe dla maksymalizacji wydajności.
- **Nieefektywność PCA:** Modele oparte na cechach uzyskanych z transformacji PCA (QNN3, QNN4, QNN5) konsekwentnie dawały gorsze rezultaty, co sugeruje, że liniowa transformacja danych może usuwać nieliniowe zależności, które są kluczowe dla kwantowego ansatzu.

Tabela 17: Macierz pomyłek dla QNN6 na zbalansowanym zbiorze testowym

	Pred: Negatywna (0)	**Pred: Pozytywna (1)**
Real: Negatywna (0)	98	0
Real: Pozytywna (1)	12	86

Ostatecznie, model QNN6 wykazał **najwyższą wartość F1-score** oraz **najlepszą Precyzję** na pełnym, niezbalansowanym zbiorze testowym (**0.4574**), co stanowiło znaczącą poprawę w porównaniu do QNN1 (**0.1954**). Wskazuje to na lepszą zdolność generalizacji modelu QNN6.

6 Końcowe Podsumowanie i Porównanie: Modele Kwantowe vs. Klasyczne

Przeprowadzone eksperymenty umożliwiły porównanie wydajności wariacyjnych kwantowych klasyfikatorów (VQC) oraz szeregu zaawansowanych algorytmów klasycznych (MLP, XGBoost, Random Forest) w złożonym zadaniu klasyfikacji ryzyka kredytowego z silną nierównowagą klas. Wszystkie modele kwantowe zostały wytrenowane na zrównoważonym (undersampled) zbiorze treningowym, tak jak ich klasyczne odpowiedniki użyte do porównania.

6.1 Kluczowe Wyniki na Zrównoważonym Zbiorze Testowym

Najbardziej miarodajnym wskaźnikiem porównawczym, uwzględniającym zbalansowaną Czułość i Precyzję, jest miara F1-score na zbiorze zrównoważonym (undersampled test set).

Tabela 18: Porównanie najlepszych wyników klasycznych i kwantowych na zrównoważonym zbiorze testowym (**under** $\rightarrow$ **under**)

Model	# Cech	Accuracy	Precision	Recall	F1-score
Najlepsze Modele Klasyczne					
MLP (PyTorch)	30	0.9439	0.9780	0.9082	**0.9418**
Logistic Regression	30	0.9490	0.9780	0.9180	0.9470
Random Forest	30	0.9490	0.9780	0.9180	0.9470
Najlepsze Modele Kwantowe (VQC)					
QNN6 (4 q, Strong)	4 (**V14**, **V10**, **V12**, **V3**)	0.9387	**1.0000**	0.8775	0.9347
QNN1 (1 q, Rot)	1 (**V14**)	0.9336	**1.0000**	0.8673	0.9289

Analiza Tabeli 18 prowadzi do następujących wniosków:

1. **Konkurencyjność F1-score:** Kwantowy klasyfikator QNN6, używający jedynie 4 cech wejściowych, osiągnął $\mathbf{F1-score} = 0.9347$, co jest wynikiem minimalnie niższym (o około 1.3-1.4%) niż najlepsze modele klasyczne (Logistic Regression, Random Forest, F1 = 0.9470) trenowane na pełnym zestawie 30 cech.

2. **Perfekcyjna Precyzja Kwantowa:** Modele kwantowe QNN1 i QNN6 osiągnęły $\mathbf{Precision} = \mathbf{1.0000}$ na zrównoważonym zbiorze testowym. To

oznacza **zerową liczbę Fałszywych Pozytywów (FP)**. Każda predykcja przypadku ryzykownego (klasa 1) przez te modele była poprawna. W praktycznym kontekście finansowym, ta cecha jest niezwykle cenna, ponieważ minimalizuje koszty fałszywych alarmów.

3. **Efektywność Informacyjna QNN:** Model QNN1, wykorzystujący zaledwie **1** cechę (**V14**), uzyskał F1 = 0.9289. Osiągnięcie tak wysokiego F1-score przy tak drastycznej redukcji wymiarowości wejściowej (z 30 do 1) jest silnym dowodem na to, że kwantowe kodowanie kątowe w połączeniu z odpowiednim ansatzem jest zdolne do wydobycia bardzo silnie separującej informacji z pojedynczej zmiennej.

6.2 Porównanie na Pełnym (Niezbalansowanym) Zbiorze Testowym

Ocena modeli na pełnym, niezbalansowanym zbiorze testowym (**under** → **full**) jest kluczowa dla weryfikacji praktycznej użyteczności. W tym scenariuszu, miara Precision oraz F1-score są decydujące.

Tabela 19: Porównanie wyników klasycznych i kwantowych na pełnym, niezbalansowanym zbiorze testowym (**under** → **full**)

Model	# Cech	Accuracy	Precision	Recall	F1-score
Modele Klasyczne					
Logistic Regression	30	0.9710	0.0520	0.9180	0.0980
MLP (PyTorch)	30	0.9640	0.0417	0.9082	0.0798
XGBoost	30	0.9090	0.0170	0.9290	0.0340
Modele Kwantowe (VQC)					
QNN6 (4 q, Strong)	4	0.9979	**0.4574**	0.8775	**0.6013**
QNN1 (1 q, Rot)	1	0.9936	0.1954	0.8673	0.3189

6.2.1 Wniosek Naukowy: Zdolność do Generalizacji

W przeciwieństwie do zbioru zrównoważonego, na pełnym zbiorze testowym, gdzie stosunek klas jest ekstremalny, modele kwantowe wykazały **zdecydowaną przewagę w miarach istotnych praktycznie**:

- **Precyzja:** Najlepszy model kwantowy, **QNN6**, osiągnął **Precision = 0.4574**, co jest wynikiem **ponad 8-krotnie wyższym** niż najlepszy klasyczny model w tym scenariuszu (Logistic Regression, Prec. = 0.0520). Oznacza to, że model QNN6 jest znacznie mniej skłonny do generowania Fałszywych Pozytywów na realistycznym zbiorze produkcyjnym.

- **F1-score:** Model **QNN6** uzyskał $\mathbf{F1-score} = \mathbf{0.6013}$, podczas gdy najwyższy F1 wśród modeli klasycznych wynosił jedynie 0.0980. Jest to **6-krotnie wyższa** wydajność syntetyczna, świadcząca o znacznie lepszej zdolności kwantowej architektury do utrzymania równowagi między Precision a Recall w warunkach silnej nierównowagi.

6.3 Wnioski końcowe i perspektywy

1. **Zaleta Kwantowa w niezbalansowaniu danych**: Kwantowe sieci neuronowe, zwłaszcza **QNN6** (4 kubity, 4 cechy, silny ansatz), okazały się **znacznie bardziej odporne** na problem silnej nierównowagi klas. Osiągnięcie wysokiego Precision i F1-score na pełnym zbiorze testowym, przy jednoczesnym użyciu znacznie mniejszej liczby cech wejściowych (4 zamiast 30), sugeruje, że kwantowy model może efektywniej mapować dane do przestrzeni Hilberta, prowadząc do lepiej separowalnej granicy decyzyjnej (patrz zastrzeżenia w sekcji 6.4 poniżej).

2. **Ekonomia Kwantowa**: Model **QNN1** pokazał, że VQC może osiągnąć bardzo wysoki F1-score (0.9289) przy użyciu **jednej cechy** i **jednego kubitu**. To otwiera perspektywy dla przyszłych implementacji na sprzęcie kwantowym (NISQ devices), gdzie zasoby obliczeniowe są ekstremalnie ograniczone.

3. **Optymalny Benchmark**: Ostatecznie, w ramach przeprowadzonego porównania modeli o ograniczonej złożoności (patrz sekcja 6.4), model **QNN6** ustanawia najwyższy uzyskany w tej pracy wynik dla tego zadania, oferując najlepszą równowagę między F1-score a Precyzją na niezbalansowanym zbiorze danych.

6.4 Ograniczenia Metodologiczne i Kierunki Dalszych Badań

Powyższe wnioski należy czytać w świetle kilku istotnych ograniczeń przyjętej metodyki.

1. **Cel badawczy jako test wykonalności.** Celem tego badania było sprawdzenie, czy kwantowe sieci neuronowe są w ogóle w stanie wydobyć użyteczny sygnał z tego zbioru danych, a nie ustalenie, czy przewyższają najlepsze dostępne metody klasyczne. Modele klasyczne użyte do porównania (mała sieć MLP, regresja logistyczna, Random Forest, XGBoost) nie były agresywnie strojone pod kątem tego zadania (dobór hiperparametrów, ważenie klas, inżynieria cech) — porównanie dotyczy modeli o zbliżonej, ograniczonej złożoności, nie absolutnego stanu wiedzy (SOTA) dla detekcji oszustw. Silniej dostrojony model klasyczny (np. gradient boosting z odpowiednim `scale_pos_weight` lub funkcją straty typu focal loss) mógłby zawęzić lub odwrócić przewagę widoczną w Tabeli 19.

2. **Pojedynczy seed i pojedynczy podział danych.** Wszystkie wyniki pochodzą z jednego treningu na ustalonym ziarnie losowości i jednym podziale trening/test, bez walidacji krzyżowej ani uśredniania po wielu powtórzeniach. Niezależna reimplementacja architektury QNN6 (te same cztery cechy $V14, V10, V12, V3$ i ten sam ansatz `StronglyEntanglingLayers`, ale zrealizowana innym torem technicznym — PyTorch + `TorchLayer` zamiast czystego PennyLane/autograd, inny optymalizator, o jedną warstwę więcej i bez klasycznego biasu) dała wynik F1 = 0.7125 na zbiorze zrównoważonym i Precision = 0.0192 na zbiorze pełnym — wyraźnie słabszy niż oficjalny QNN6 (F1 = 0.9347, Precision = 0.4574). Wskazuje to, że raportowany wynik QNN6 może być wrażliwy na konkretną implementację, inicjalizację i dobór hiperparametrów, a nie jest niezmienniczą właściwością samej architektury obwodu. Różnice rzędu 1–2 punktów procentowych F1 pomiędzy poszczególnymi wariantami QNN (np. QNN1 vs. QNN6 na zbiorze under, liczącym jedynie 196 próbek) nie powinny więc być traktowane jako rozstrzygające bez potwierdzenia na wielu ziarnach losowości.

3. **Nieefektywność PCA jako artefakt redukcji wymiarowości, nie ansatzu.** Słabsze wyniki modeli opartych na PCA (QNN3–QNN5) najprawdopodobniej wynikają z natury samej transformacji PCA, która maksymalizuje wariancję, a nie separowalność klas — podobnego efektu należałoby oczekiwać również dla klasyfikatora klasycznego trenowanego na tych samych pięciu składowych głównych. Nie jest to zatem odkrycie specyficzne dla kwantowego kodowania danych.

4. **Symulacja bez szumu sprzętowego.** Wszystkie obwody symulowano analitycznie (`shots=None`), bez modelowania szumu, dekoherencji ani ograniczeń rzeczywistego sprzętu NISQ. Wnioski dotyczą wyłącznie matematycznej ekspresywności użytych ansatzów w warunkach idealnych, nie realnej wykonalności na obecnie dostępnym sprzęcie kwantowym.

5. **Undersampling jako założony punkt startowy.** Cała analiza zakłada undersampling zbioru treningowego jako punkt wyjścia. Nie zbadano alternatywnych strategii radzenia sobie z niezrównoważeniem klas (np. trening na pełnym zbiorze z ważeniem klas, SMOTE, czy funkcje straty wrażliwe na koszt błędu) — pozostaje to otwartym pytaniem dla dalszych badań.

6. **Brak uzasadnienia biznesowego przy obecnym stanie technologii.** Trening obwodów kwantowych w symulacji trwa rzędu minut do godzin nawet dla kilku kubitów, podczas gdy klasyczny model tej skali liczy się na milionach wierszy w ułamku sekundy. Wartość tej pracy ma charakter poznawczy — odpowiedź na pytanie, czy podejście kwantowe w ogóle działa na tym zadaniu — a nie inżynierski postulat gotowości wdrożeniowej.

Bibliografia

[1] European Parliament and Council of the European Union. *Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market, amending Directives 2002/65/EC, 2009/110/EC and 2013/36/EU and Regulation (EU) No 1093/2010, and repealing Directive 2007/64/EC (PSD2)*. Official Journal of the European Union, L 337, 23.12.2015, p. 35–127. CELEX:32015L2366. 2015. URL: https://eur-lex.europa.eu/eli/dir/2015/2366/oj.

[2] European Commission. *Commission Delegated Regulation (EU) 2018/389 of 27 November 2017 supplementing Directive (EU) 2015/2366 of the European Parliament and of the Council with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication*. Official Journal of the European Union, L 69, 13.3.2018. CELEX:32018R0389. 2018. URL: https://eur-lex.europa.eu/eli/reg_del/2018/389/oj.

[3] Sebastian Zajac. „Modelowanie dla biznesu. Analityka w czasie rzeczywistym. Narzędzia informatyczne i biznesowe". W: *Oficyna Wydawnicza SGH* (2022).

[4] John Preskill. „Quantum Computing in the NISQ era and beyond". W: *Quantum* 2 (2018), s. 79. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[5] A. P. Lund, M. J. Bremner i T. C. Ralph. „Quantum Sampling Problems, Boson Sampling and Quantum Supremacy". W: *NPJ Quantum Information* 3 (2017).

[6] A. W. Harrow i A. Montanaro. „Quantum Computational Supremacy". W: *Nature* 549 (2017), s. 203–209.

[7] Alberto Peruzzo i in. „A Variational Eigenvalue Solver on a Photonic Quantum Processor". W: *Nature Communications* 5.1 (2014), s. 4213.

[8] M. Cerezo i in. „Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits". W: *Nature Communications* 12 (2021), s. 1791.

[9] Vojtěch Havlíček, M. Mohseni i S. Lloyd. „Supervised Learning with Quantum-Enhanced Feature Spaces". W: *Nature* 567.7747 (2019), s. 209––212.

[10] Chih-Wei Hsu i T. D. Ladd. „Quantum Support Vector Classification". W: *Quantum* 4 (2020), s. 328.

[11] M. Benedetti i in. „Parameterized quantum circuits as machine learning models". W: *Quantum Science and Technology* 4.4 (2019), s. 043001. DOI: 10.1088/2058-9565/ab4eb5. URL: https://doi.org/10.1088/2058-9565/ab4eb5.

[12] J. L. Cybulski i S. Zajac. „Design Considerations for Denoising Quantum Time Series Autoencoder". W: *Computational Science – ICCS 2024*. Oprac. Leonardo Franco i in. T. 14045. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2024, s. 252–267. ISBN: 978-3-031-63778-0.

[13] Edward Farhi, Jeffrey Goldstone i Sam Gutmann. „A Quantum Approximate Optimization Algorithm". W: *MIT-CTP/4610* (listopad 2014). eprint: 1411.4028. URL: https://arxiv.org/pdf/1411.4028.pdf.

[14] Wikipedia contributors. „Sfera Blocha". W: (2025). Accessed 18 Nov 2025. URL: https://pl.wikipedia.org/wiki/Sfera_Blocha.

[15] Jarrod R. McClean i in. „Barren Plateaus in Quantum Neural Networks". W: *Nature Communications* 9 (1 2018), s. 4812. DOI: 10.1038/s41467-018-07090-4.