Prognoza ceny dolara amerykańskiego w stosunku do polskiego złotego.

Tim Schopinski, Kamil Kelsz 27.05.2023

1. Opis Problemu

W niniejszym sprawozdaniu przedstawiono prognozowanie ceny dolara amerykańskiego w stosunku do polskiego złotego. Celem pracy było zaimplementowanie dwóch rodzajów modeli regresji: liniowej oraz segmentowej. Następnie porównano wyniki tych modeli z wbudowanymi rozwiązaniami dostępnymi w bibliotece TensorFlow.

Dodatkowo przeprowadzono porównanie procesu uczenia z wykorzystaniem zaimplementowanej sieci neuronowej FeedForward i wbudowanej implementacji tej sieci w TensorFlow.

Implementacja została zrealizowana w formie aplikacji wiersza poleceń (CLI), co umożliwiło modyfikację parametrów takich jak interwał danych, przedział czasowy, liczba danych treningowych i testowych, a także ilość epok w przypadku sieci neuronowej.

W dalszej części sprawozdania przedstawiono szczegóły implementacji oraz omówienie wyników uzyskanych dla poszczególnych modeli. Analiza obejmuje porównanie predykcji oraz ocenę procesu uczenia w kontekście zaimplementowanych rozwiązań oraz wbudowanych funkcji dostępnych w bibliotece TensorFlow.

Celem pracy jest zbadanie skuteczności modeli regresji liniowej i segmentowej w prognozowaniu ceny dolara w stosunku do złotówki oraz porównanie tych wyników z modelem sieci neuronowej FeedForward dostępnym w TensorFlow. Analiza tych modeli pozwoli na ocenę, który z nich najlepiej odwzorowuje rzeczywiste trendy cenowe oraz który model posiada lepsze zdolności prognostyczne.

Cały projekt jest dostępny na https://github.com/timschopinski/dolar-price-prediction

2. Teoretyczny opis użytej metody/metod

1. Metryki oceny modelu: MSE, MAE, RMSE

W celu oceny jakości modelu regresji liniowej oraz segmentowej, wykorzystano trzy powszechnie stosowane metryki: Mean Squared Error (MSE), Mean Absolute Error (MAE) oraz Root Mean Squared Error (RMSE). Metryki te pozwalają nam dokładnie zrozumieć, jak dobrze model dopasowuje się do rzeczywistych danych i jakie są różnice między prognozowanymi a rzeczywistymi wartościami.

1. Mean Squared Error (MSE): MSE oblicza średnią wartość kwadratu różnicy między prognozowanymi a rzeczywistymi wartościami. Jest to popularna metryka oceny błędu, która penalizuje większe różnice między wartościami prognozowanymi a rzeczywistymi. Wyższe wartości MSE oznaczają większy błąd modelu. W przypadku MSE, im bliżej wynik jest zeru, tym lepszy model.

2. Mean Absolute Error (MAE): MAE oblicza średnią wartość bezwzględną różnicy między prognozowanymi a rzeczywistymi wartościami. Jest to metryka, która mierzy średnią wielkość błędu bez uwzględniania kierunku różnicy. MAE jest bardziej odporny na wartości odstające niż MSE. Podobnie jak w przypadku MSE, niższe wartości MAE wskazują na lepszą jakość modelu.

3. Root Mean Squared Error (RMSE): RMSE oblicza pierwiastek kwadratowy z MSE i daje wynik w tych samych jednostkach co zmienna zależna. Jest to często stosowana metryka, ponieważ jest intuicyjna i łatwa do interpretacji. RMSE jest szczególnie przydatne, gdy chcemy mieć miarę błędu w oryginalnych jednostkach danych. Im niższa wartość RMSE, tym lepszy model.

2. Regresja Liniowa

Regresja liniowa jest jedną z najprostszych technik wyznaczania przybliżonego wzoru funkcji f($\vec{x}$) na podstawie posiadanego zbioru obserwacji. W modelu tym zakładamy, że naszą cechę wynikową (zwaną zmienną zależną) można wyliczyć na podstawie liniowej kombinacji cech wejściowych (zwanych zmiennymi niezależnymi):

$$\widehat{y} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + nx_n$$

gdzie $\widehat{y}$ jest naszą prognozowaną wartością cechy wyjściowej,
n określa liczbę cech wejściowych modelu,
i to wartość i-tej cechy,
a Θ_j to j-ty parametr modelu, gdzie θ0 stanowi wyraz wolny, zwany punktem obciążenia (bias term).
Równanie to możemy zapisać także w formie wektorowej:

$\hat{y} = \vec{\theta} \circ \vec{x}$

gdzie $\vec{x}$ stanowi rozszerzony wektor cech wejściowych $\vec{x} = [1, x1, \cdots, xn]$.

Zadaniem naszego algorytmu uczenia maszyn jest znalezienie wektora parametrów $\vec{\theta}$ dla którego błąd pomiędzy wartością oczekiwaną y, a oszacowaną $\hat{y}$ będzie możliwie najmniejszy. Proces ten nazywamy treningiem modelu.

Pierwszym krokiem było uzyskanie rozwiązania w postaci zamkniętej dla modelu regresji liniowej. Wykorzystano dostępne dane treningowe, gdzie zmienną niezależną stanowiły dni, a zmienną zależną cena dolara w stosunku do złotówki. Do obliczenia współczynników regresji, czyli a i b, użyto wzorów opartych na estymacji średnich wartości zmiennych niezależnych i zależnych.

3. Segmentowa Regresja Liniowa

Segmentowa regresja liniowa jest modyfikacją regresji liniowej, która uwzględnia zmienność trendów w danych. Zamiast stosować pojedynczy model liniowy dla całego zbioru danych, dzieli się zbiór na segmenty, gdzie każdy segment reprezentuje okres o ciągłym rosnącym lub malejącym trendzie. Dla każdego segmentu tworzony jest osobny model liniowy, który jest bardziej odpowiedni dla danego trendu.

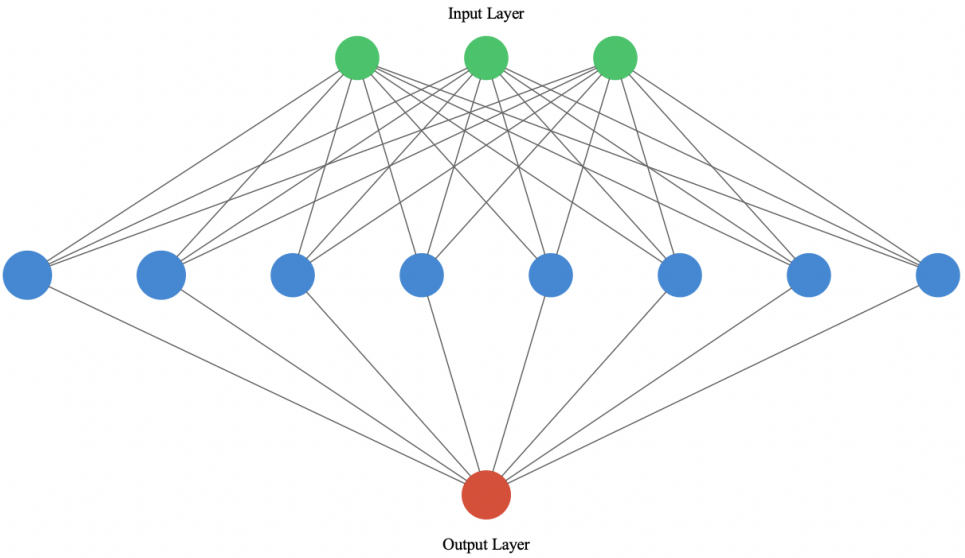
Segmentowa regresja liniowa ma zaletę adaptowania się do zmieniających się trendów w danych, co czyni ją bardziej przydatną niż regresja liniowa w przypadku danych o zmiennym trendzie. Dzięki podziałowi danych na segmenty i tworzeniu osobnych modeli dla każdego segmentu, segmentowa regresja liniowa może dostosować się do różnych trendów obecnych w danych i dostarczyć bardziej dokładne prognozy.

4. Sieć neuronowa feedforward

W sieciach neuronowych feedforward informacja przepływa przez sieć w jednym kierunku.

Proces uczenia sieci neuronowej feedforward polega na znalezieniu optymalnych wag i biasów (parametrów) dla każdego neuronu w sieci. Wagi są dostosowywane podczas treningu w oparciu o różnicę między wartościami oczekiwanymi a prognozowanymi. Algorytm wstecznej propagacji błędu jest najczęściej stosowanym sposobem uczenia sieci neuronowych feedforward.

Wizualizacja zaimplementowanej przez nas sieci neuronowej wygenerowana przy pomocy biblioteki graphviz.



3. Opis realizacji zadania

1. Regresja Liniowa

W przypadku krótszych przedziałów czasowych, gdzie analizowane były dane obejmujące np. miesiąc, model regresji liniowej wykazywał względnie dobre wyniki. Przyjęto zasadę, że 80% dni zostało wykorzystanych do treningu modelu, a 20% dni stanowiło prognozę. W takim przypadku model był w stanie w miarę dokładnie odwzorować trendy cenowe.

2. Segmentowa Regresja Liniowa

Segmentowa regresja liniowa została zastosowana w celu uwzględnienia zmian trendu, które mogą występować w danych dotyczących prognozy ceny dolara w stosunku do złotówki. Standardowa regresja liniowa nie zawsze sprawdza się dobrze w przypadku danych, w których występują zmiany trendu, dlatego zastosowano podejście segmentowane, które pozwala uwzględnić te zmiany i dostosować model regresji do różnych segmentów danych.

Implementacja segmentowanej regresji liniowej opiera się na podziale danych na segmenty, w których zachodzi kontynuacja trendu. Dla każdego segmentu danych, model regresji liniowej jest trenowany na danych treningowych, a następnie używany do prognozowania wartości dla ostatniego punktu danych w segmencie. Proces ten jest powtarzany dla każdego segmentu, a ostateczne prognozy są łączone w celu uzyskania pełnego zestawu prognoz.

3. Sieć neuronowa feedforward

Sieć neuronowa FeedForward została zaimplementowana jako metoda prognozowania ceny dolara w stosunku do złotówki. Sieć składa się z warstw neuronów, które przekształcają dane wejściowe i generują wyniki prognoz. Implementacja sieci neuronowej FeedForward opiera się na dwóch klasach: Warstwa (Layer) i Sieć (NeuralNetwork). Oto opis ich implementacji:

1. Klasa Layer:
    - Konstruktor przyjmuje trzy parametry: input_size (rozmiar wejścia), output_size (rozmiar wyjścia) i activation (funkcję aktywacji).
    - Inicjalizowane są wagi (weights) za pomocą losowych wartości i bias (bias) ustawiane jest na zera.
    - Metoda forward wykonuje operację przekształcenia danych wejściowych (X) za pomocą wag i bias. Wynik jest przepuszczany przez funkcję aktywacji (relu lub liniową) i zwracany jako wyjście (output).
    - Metoda backward wykonuje propagację wsteczną, obliczając gradienty dla wag (gradient_weights) i bias (gradient_bias) na podstawie otrzymanego gradientu wejściowego (gradient_input).

2. Klasa NeuralNetwork:
    - Konstruktor tworzy pustą listę warstw (layers).
    - Metoda add dodaje warstwę do sieci.
    - Metoda compile przyjmuje dwa parametry: optimizer (optymalizator) i loss (funkcję straty).
    - Metoda fit trenuje sieć na podstawie danych treningowych (X) i oczekiwanych wyników (y) przez określoną liczbę epok (epochs) i rozmiar wsadu (batch_size).
    - Metoda train_on_batch wykonuje jedną iterację treningową dla wsadu danych, przeprowadzając przód (forward) i wsteczną (backward) propagację.
    - Metoda update_layer aktualizuje wagi i biasy warstwy przy użyciu optymalizatora Adam.
    - Metoda predict wykonuje predykcję na podstawie danych wejściowych (X) przez przeprowadzenie przód przez wszystkie warstwy i zwraca wynik.

Wyniki prognozowane przez sieć neuronową FeedForward mogą być uzyskane za pomocą metody predict. Sieć może być trenowana za pomocą metody fit, która aktualizuje wagi i biasy na podstawie optymalizatora i funkcji straty.

W naszym przypadku dane wejściowe (train_features) zawierają trzy kolumny odpowiadające tym trzem cechom: "Open", "High" i "Low". Cecha docelowa ("Close") jest przeniesiona do train_target.

W naszym przypadku ważną rolę odgrywa optymalizator Adam, który aktualizuje wagi i biasy. Źródło informacji
Algorytm Adam łączy zalety dwóch innych algorytmów optymalizacji: momentum i RMSProp.

Momentum uzyskuje się poprzez obliczenie średniej kroczącej gradientu (znanej również jako średnie ważone wykładniczo), a następnie użycie jej do aktualizacji parametrów "θ" (wagi, odchylenia).

$$\mathrm{m}_t = \beta \cdot \mathrm{m}_{t-1} + (1-\beta) \cdot grad(\theta_{t-1})$$

$$\theta_t := \theta_{t-1} - \alpha \cdot m_t$$

RMSProp utrzymuje kroczącą średnią kwadratów ostatnich gradientów, oznaczoną przez (v). W ten sposób nadaje większą wagę ostatnim gradientom.

Tutaj termin Beta jest wprowadzany jako czynnik zapominania

$$\mathrm{v}_t = \beta \cdot \mathrm{v}_{t-1} + (1-\beta) \cdot [grad(\theta_{t-1})]^2$$

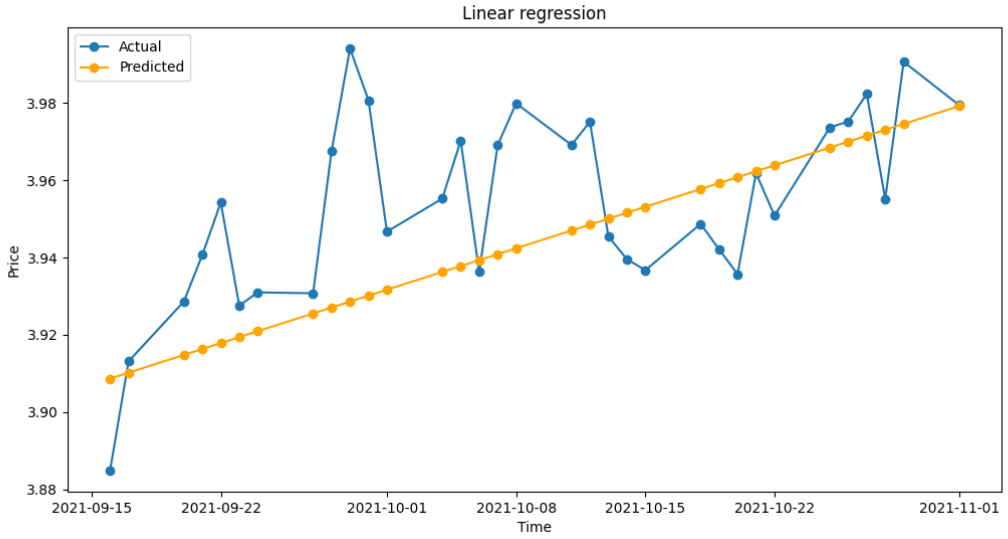
podczas aktualizacji wagi lub biasu, należy podzielić gradient poprzedniej wartości θ przez średnią kroczącą kwadratów ostatnich gradientów dla tego parametru θ, a następnie pomnożyć go przez α i oczywiście odjąć poprzednią wartość θ.

$$\theta_t := \theta_{t-1} - \alpha \cdot \frac{\mathrm{grad}(\theta_{t-1})}{\sqrt{\mathrm{v}_t + \epsilon}}$$

4. Prezentacja osiągniętych wyników

1. Regresja Liniowa

Poniżej znajduje się wykres gdzie regresja liniowa wyznacza trend spadkowy dla przedziału 2021-09-09 - 2021-11-01.
Tabela przedstawiająca wyniki metryki oceny modelu.



MSE	MAE	RMSE
0.0005	0.0187	0.0238

Jednakże, dla dłuższych przedziałów czasowych, gdzie analizowane były dane obejmujące większą ilość dni, zauważono, że model regresji liniowej stawał się znacznie mniej precyzyjny. Wynikało to z faktu, że długoterminowe zmiany trendów na rynku miały znaczny wpływ na dokładność prognoz. Model nie był w stanie uwzględnić nieliniowych zmian ani zakrzywień w danych, co prowadziło do większych błędów w prognozach. Poniżej znajduje się wykres gdzie regresja liniowa wyznacza trend spadkowy dla przedziału 2023-02-15 - 2023-05-01.

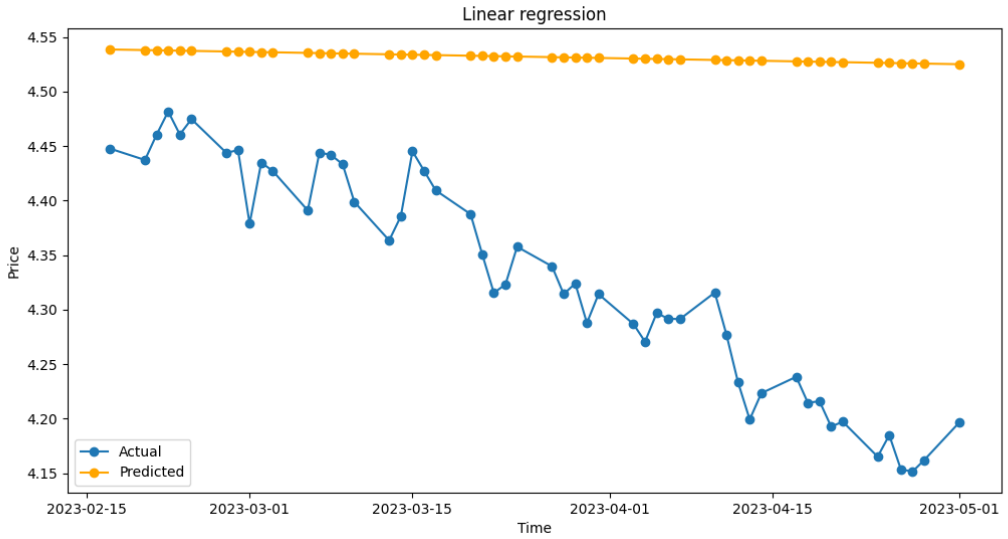


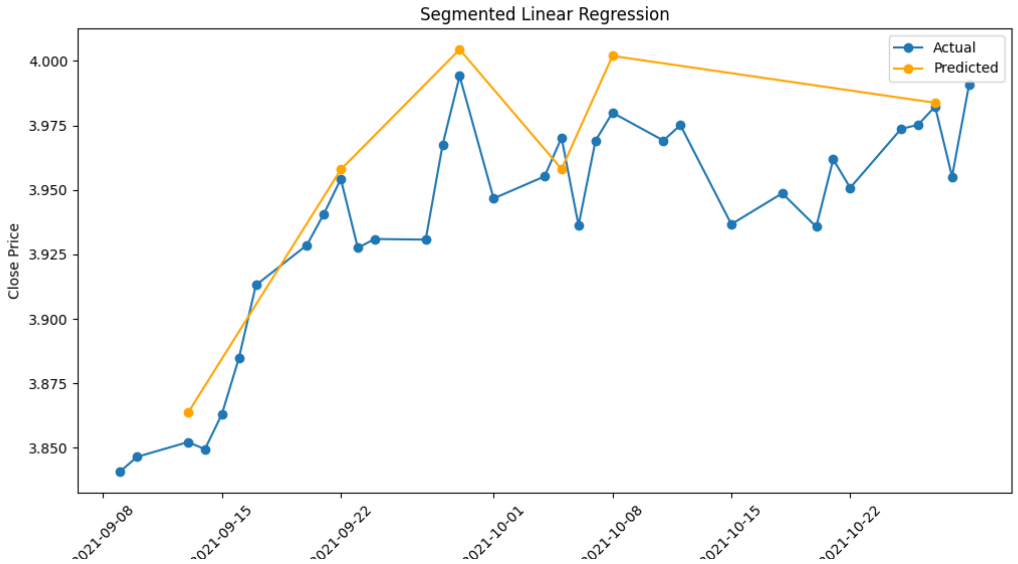
Tabela przedstawiająca wyniki metryki oceny modelu.

MSE	MAE	RMSE
0.0503	0.2029	0.2244

Podsumowując, implementacja regresji liniowej była skuteczna dla krótszych przedziałów czasowych, gdzie trendy cenowe były bardziej stabilne. Jednakże, dla dłuższych przedziałów czasowych, gdzie zmiany trendów były bardziej dynamiczne, regresja liniowa stawała się mniej dokładna.

2. Segmentowa Regresja Liniowa

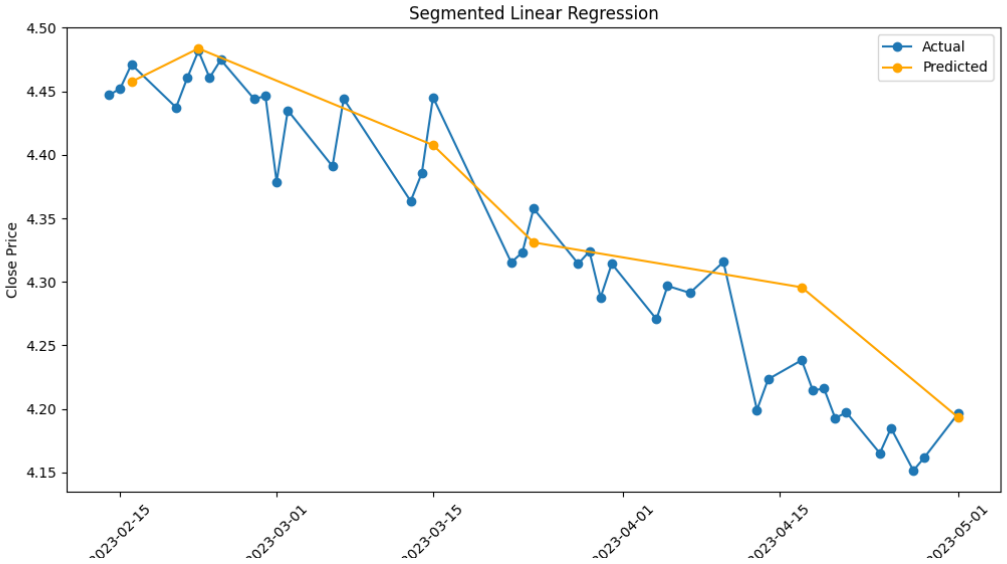
Poniżej znajduje się wykres dla tego samego przedziału co w akapicie 3, czyli dla przedziału 2021-09-09 - 2021-11-01.



MSE	MAE	RMSE
0.0001	0.0101	0.0121

Z wykresu wynika że segmentowa regresja liniowa lepiej radzi sobie w przypadku zmian trendu. Tak samo widać to po wartości MSE, która jest około 5 razy mniejsza.

Poniżej znajduje się wykres dla tego samego przedziału co w akapicie 3, gdzie wartości prognozowane przez regresja liniowa bardzo odbiegały od rzeczywistych ze względu na dłuższy przedział czasowy, czyli dla przedziału 2023-02-15 - 2023-05-01.
W tym przypadku widać szczególną poprawę w porównaniu ze zwykłą regresją liniową.



MSE	MAE	RMSE
0.0009	0.0233	0.0305

Podsumowując, w porównaniu do zwykłej regresji, regresja segmentowa może uwzględniać lokalne zmiany trendu, co może prowadzić do bardziej precyzyjnych prognoz w przypadku danych, w których trend się zmienia.

3. Sieć neuronowa feedforward

Poniżej znajduje się rezultat predykcji ceny za pomocą zaimplementowanej przez nas sieci Feedforward dla tego samego przedziału czasu jak w poprzednich przykładach 2021-09-09 - 2021-11-01.

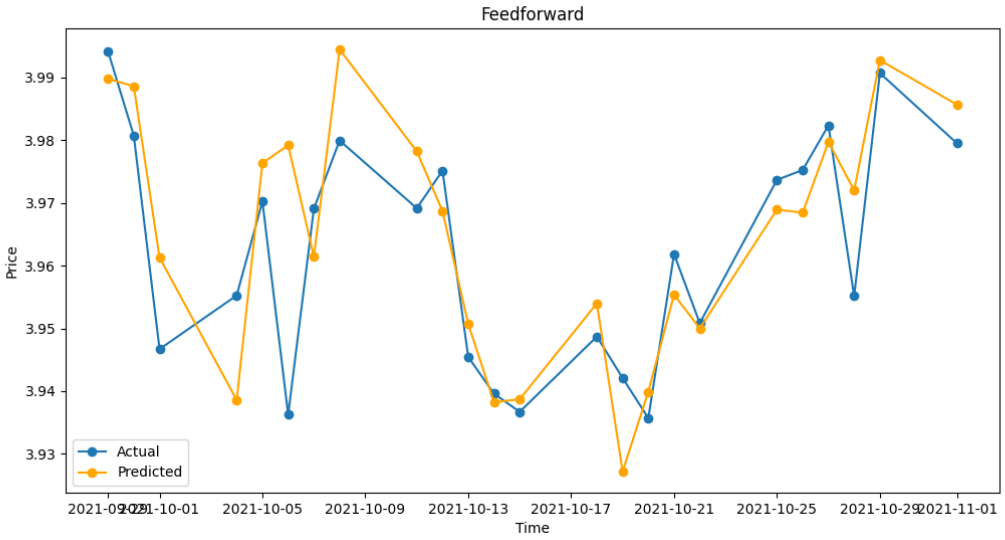
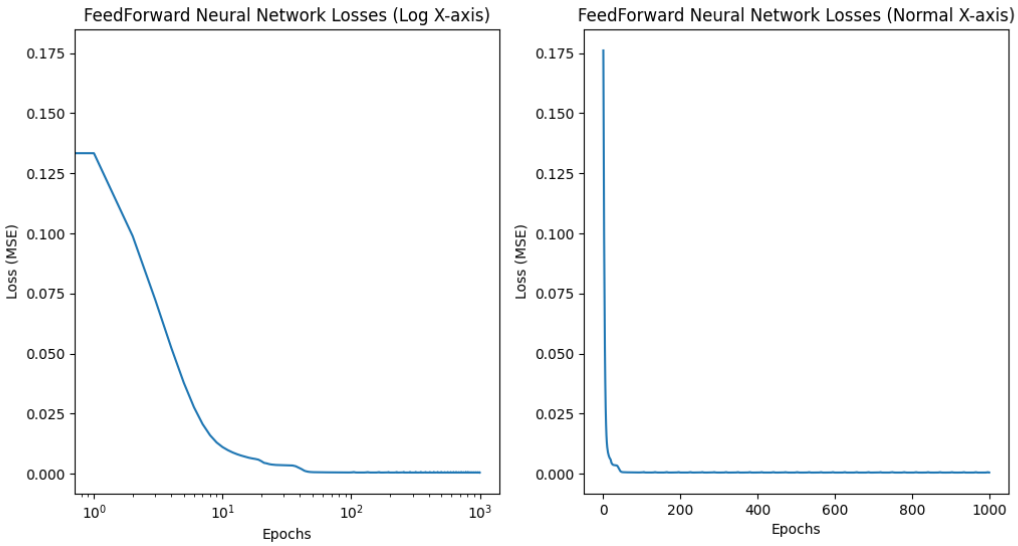


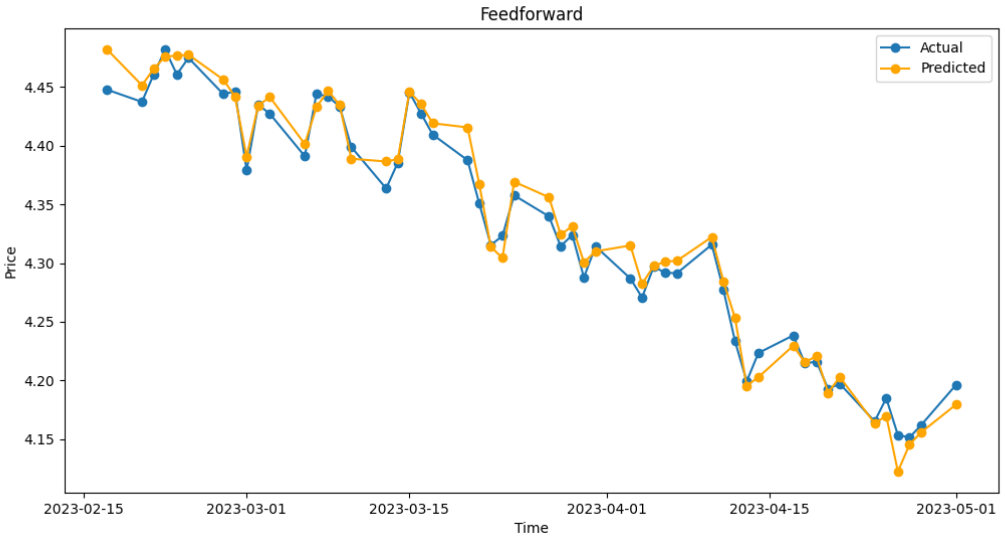
Tabela przedstawiająca wyniki metryki oceny modelu.

MSE	MAE	RMSE
0.0001	0.0087	0.0122

Wykresy przedstawiające wartości straty dla każdej epoki podczas procesu uczenia.



Tak jak w poprzednich przypadkach naszą sieć przetestowaliśmy dla dłuższego okresu czasu 2022-02-15 - 2023-05-01.



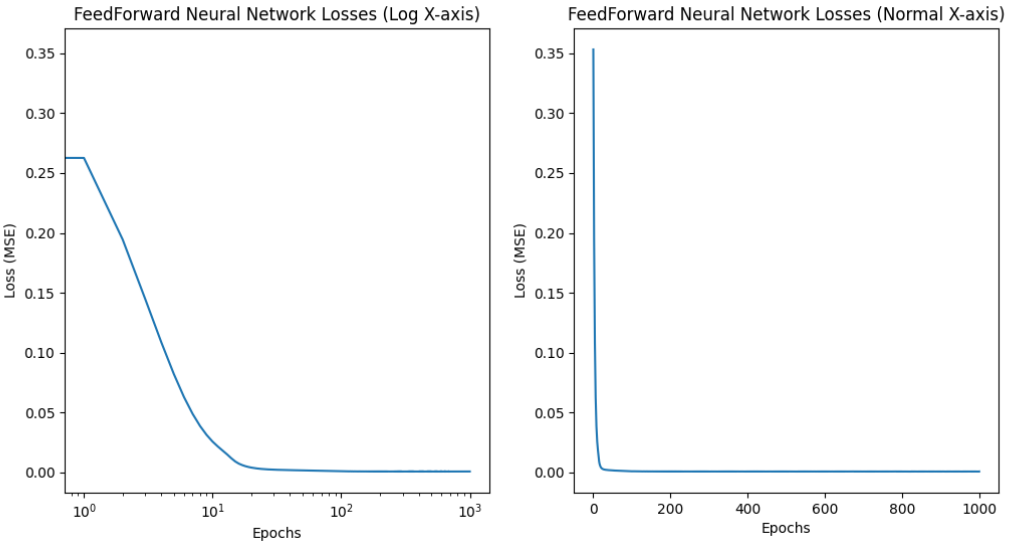
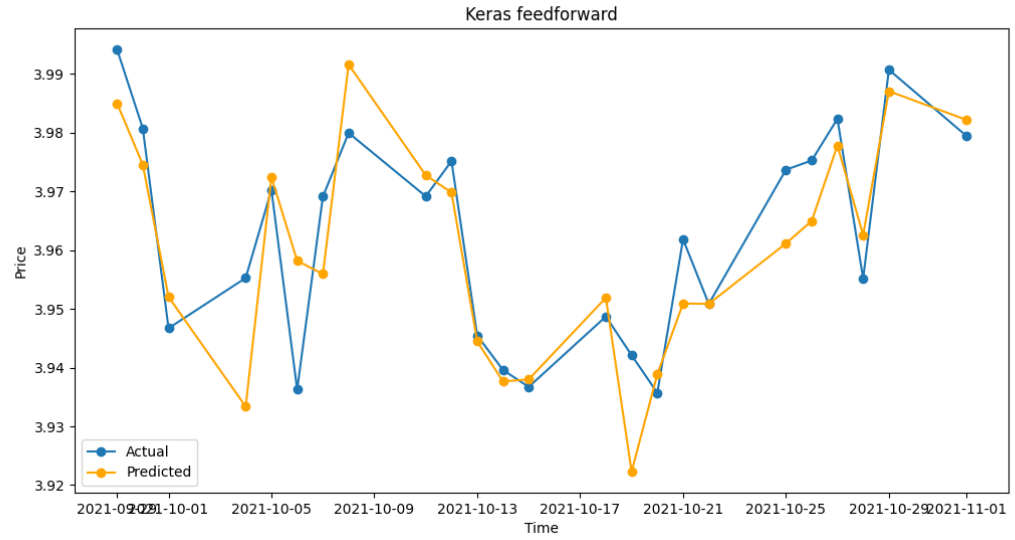


Tabela przedstawiająca wyniki metryki oceny modelu.

MSE	MAE	RMSE
0.0001	0.01055	0.0132

4. Sieć neuronowa feedforward z biblioteki Keras

Aby zweryfikować poprawność naszej implementacji porównaliśmy naszą sieć z gotową implementacja z biblioteki Keras przy wykorzystaniu klas keras.Sequential oraz keras.layers.Dense. Tak prezentują się wyniki uzyskane dla tych samych danych wejściowych co w poprzednich przypadkach.



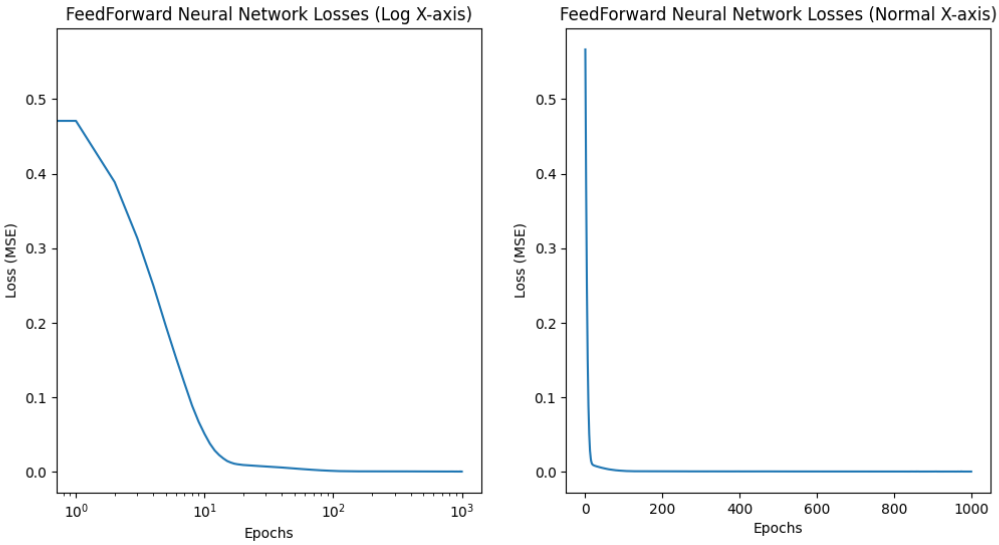
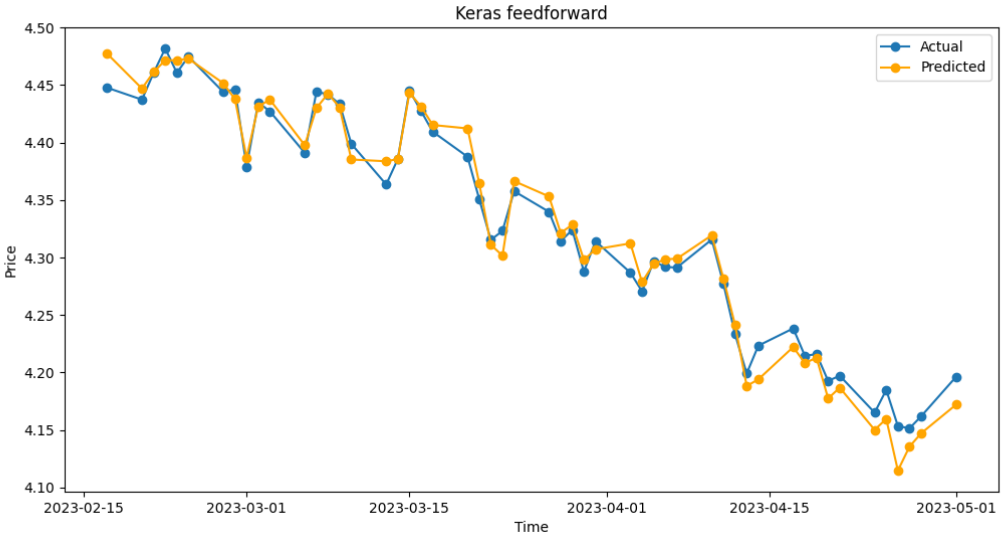
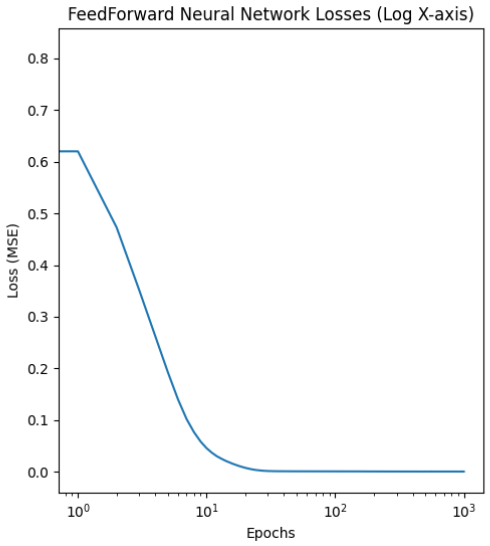


Tabela przedstawiająca wyniki metryki oceny modelu.

MSE	MAE	RMSE
9.85 * 10^-5	0.0076	0.0099

Tak jak w poprzednich przypadkach sieć z biblioteki Keras przetestowaliśmy dla dłuższego okresu czasu 2022-02-15 - 2023-05-01 by na koniec zestawić wyniki.





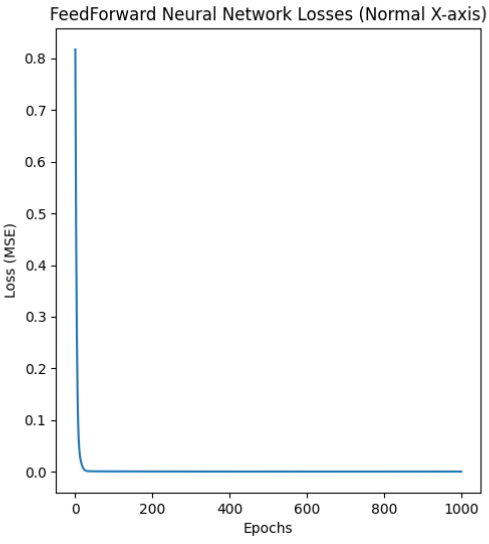


Tabela przedstawiająca wyniki metryki oceny modelu.

MSE	MAE	RMSE
0.0001	0.0111	0.0139

5. Dyskusja

W ramach projektu przeprowadziliśmy analizę prognozowania cen dolara amerykańskiego do polskiego złotego przy użyciu trzech modeli: regresji liniowej, regresji segmentowej liniowej oraz własnego modelu sieci neuronowej feedforward.

Stwierdziliśmy, że regresja liniowa może być trafną metodą prognozowania krótkoterminowych trendów, na przykład dla kilku dni. Jednak w przypadku dłuższych okresów czasowych, ze względu na liczne zmiany trendu, staje się ona niepraktyczna i nieefektywna.

Zauważyliśmy, że aktywa finansowe często podlegają zmianom trendu, co sprawia, że regresja liniowa nie jest skutecznym narzędziem do prognozowania ich cen. W takim przypadku regresja segmentowa liniowa okazała się znacznie lepszym rozwiązaniem. Potrafiła dobrze prognozować zmiany trendu i dostarczała wartościowe informacje na temat ogólnego kierunku, jednak nadal nie była wystarczająco dokładna dla precyzyjnej prognozy cen.

Wreszcie, porównaliśmy nasz własny model sieci neuronowej feedforward z gotowym modelem z biblioteki Keras. Okazało się, że nasz własny model osiągał dobre wyniki prognozowania, a dla dłuższych okresów czasowych stał się jeszcze bardziej dokładny. Porównując go z modelem z biblioteki Keras, oba modele wykazywały podobną skuteczność.

Podsumowując, dla prognozowania cen aktywów finansowych zmiennych trendowo, regresja liniowa nie jest odpowiednim narzędziem. Regresja segmentowa liniowa może radzić sobie lepiej ze zmianami trendu, ale wciąż nie zapewnia precyzyjnej prognozy cen. Natomiast model sieci neuronowej feedforward, zarówno nasz własny jak i gotowy z biblioteki Keras, może dostarczać dokładniejszych prognoz, zwłaszcza dla dłuższych okresów czasowych.