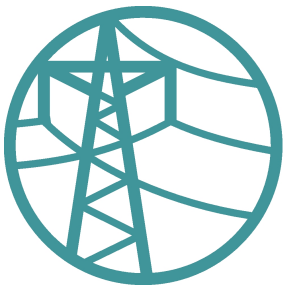


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Sieć LSTM do przewidywania cen akcji

Dawid Kowalczuk – 177107

Prowadzący:
mgr inż. Dawid Kalandyk

Rzeszów, 22 maja 2025

Spis treści

1. Wstęp

Celem niniejszego projektu jest zbadanie możliwości wykorzystania sieci LSTM do przewidywania przyszłych cen akcji na podstawie historycznych danych rynkowych. Analizie poddane zostały dane wybranych spółek notowanych na amerykańskiej giełdzie, takich jak Apple, Microsoft, Google, Amazon i Nvidia.

Zastosowanie modelu sieci LSTM w przewidywaniu zmiennych rynkowych niesie ze sobą wyzwania związane z nieliniowością oraz dużą zmiennością danych. Projekt ten ma na celu nie tylko praktyczne wykorzystanie modelu LSTM, ale także refleksję nad jego ograniczeniami.

2. Dane

2.1. Pochodzenie danych

W projekcie wykorzystano dane giełdowe zawierające notowania 501 spółek notowanych na amerykańskiej giełdzie. Dane zostały pobrane w formacie CSV i przedstawione są w tabeli [2.1]. Zawierają następujące informacje:

- Data – data notowania,
- Symbol – symbol giełdowy spółki,
- Open – cena otwarcia,
- High – cena maksymalna,
- Low – cena minimalna,
- Close – cena zamknięcia,
- Volume – wolumen obrotu.

Tabela 2.1. Losowe przykładowe dane

Data	Symbol	Open	Close	Low	High	Volume
2014-01-28	VIAB	80.63	82.26	80.63	82.31	2 493 200
2010-05-05	FITB	14.10	14.59	13.99	14.93	16 049 800
2014-10-29	WFC	51.84	52.17	51.68	52.27	16 546 900
2014-05-21	PX	130.24	130.51	130.03	130.79	684 200
2010-05-11	DNB	75.76	75.80	75.56	76.39	538 900

Na potrzeby projektu wybrano dane dotyczące pięciu spółek: AAPL (Apple), GOOGL (Alphabet), MSFT (Microsoft), AMZN (Amazon) oraz NVDA (Nvidia). Dane zostały zapisane w plikach CSV.

2.2. Analiza statystyczna danych

Analiza statystyczna została przeprowadzona na podstawie danych spółki AAPL. Wstępna analiza pozwoliła określić podstawowe cechy statystyczne [2.2]:

- zakres cen i wolumenu obrotu,
- wartości średnie oraz mediany,
- odchylenie standardowe jako miara zmienności.

Tabela 2.2. Podstawowe statystyki dla spółki AAPL

Cecha	Średnia	Mediana	Min	Max	Odch. std.
Open	313.08	318.23	90.00	702.41	185.30
Close	312.93	318.24	90.28	702.10	185.15
High	315.91	320.60	90.70	705.07	186.90
Low	309.83	316.55	89.47	699.57	183.38
Volume	94 225 775.88	80 503 850.00	11 475 900.00	470 249 500.00	60 205 187.77

2.3. Przygotowanie danych

Dane zostały posortowane według daty w porządku rosnącym. Upewniono się również, że wszystkie dane są poprawne i nie zawierają brakujących wartości. Zostały one znormalizowane do zakresu [0, 1], co zapewnia lepszą stabilność i szybkość uczenia modelu. Losowy fragment przygotowanych danych został przedstawiony w tablei [2.3]. Następnie dane podzielono na zbiory treningowe i testowe w proporcji 80/20.

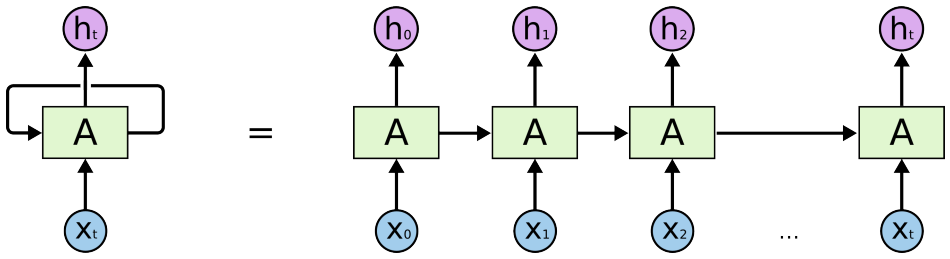
Tabela 2.3. Znormalizowane dane dla spółki AAPL

Data	Symbol	Open	Close	Low	High	Volume
2010-01-04	AAPL	0.2015	0.2022	0.2015	0.2015	0.2440
2010-01-05	AAPL	0.2035	0.2028	0.2029	0.2033	0.3030
2010-01-06	AAPL	0.2031	0.1973	0.1988	0.2027	0.2759
2010-01-07	AAPL	0.1988	0.1966	0.1960	0.1974	0.2350
2010-01-08	AAPL	0.1964	0.1989	0.1960	0.1974	0.2189

3. Wstęp teoretyczny

3.1. Rekurencyjne sieci neuronowe (RNN)

Zwyczajne sieci neuronowe nie są przystosowane do przetwarzania sekwencyjnych danych, takich jak tekst czy szeregi czasowe. Do tego typu danych wykorzystywane są Rekurencyjne sieci neuronowe (RNN), dzięki temu, że wyjście z jednego kroku czasowego jest podawane jako wejście do następnego kroku [3.1] są w stanie zapamiętywać informacje z przeszłości i wykorzystywać je przy podejmowaniu decyzji.



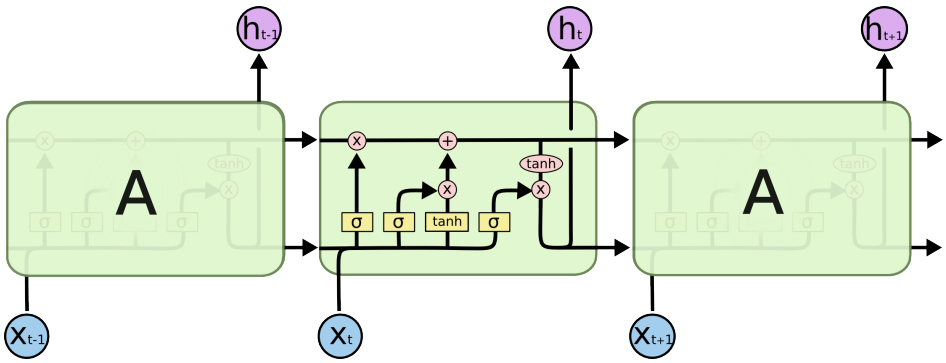
Rys. 3.1. Schemat działania rekurencyjnych sieci neuronowych (RNN)

Źródło: Olah, C. (2015)[2]

Sieci te dobrze radzą sobie z krótkimi sekwencjami. Tak jak w przypadku zdania „Ala poszła do sklepu". Gdzie słowa „Ala" i „sklepu" są ze sobą powiązane ale znajdują się relatywnie blisko siebie. Jednak w przypadku dłuższych sekwencji, takich jak „Film tego reżysera początkowo nie dotarł do większej publiczności, jednak po sukcesach kolejnych filmów i rosnącej popularności reżysera został finalnie doceniony przez krytyków", żeby zrozumieć co zostało docenione trzeba cofnąć się do początku relatywnie długiego zdania. W tym przypadku może pojawić się problem znany jako „zanikanie i eksplozja gradientów". Dlatego w przypadku długich sekwencji możemy użyć sieci LSTM.

3.2. Sieć LSTM

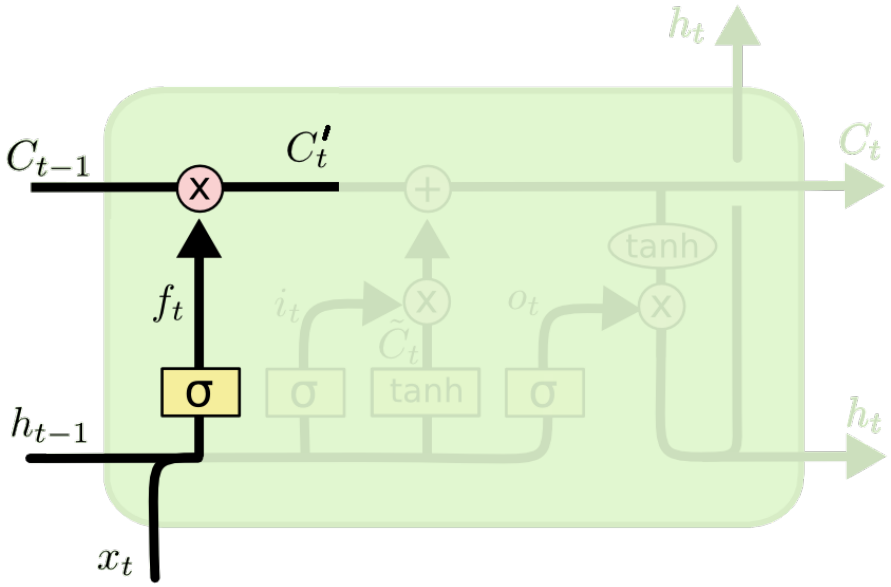
Sieć LSTM (ang. Long Short-Term Memory) jest to rozwinięcie architektury RNN, zaproponowane przez Hochreitera i Schmidhubera (1997)[1]. Sieć ta może decydować, które informacje zapamiętywać, a które zapominać, dzięki czemu jest w stanie uczyć się długoterminowych zależności. Główna różnica między LSTM a RNN polega na tym, że w sieci RNN [3.1] w bloku A znajduje się tylko jedna funkcja aktywacji, natomiast w LSTM [3.2] mamy do czynienia z bardziej złożoną strukturą.



Rys. 3.2. Schemat działania sieci LSTM Źródło: Olah, C. (2015)[2]

3.2.1. Brama zapominania

Brama zapominania [3.3](ang. forget gate) to kluczowy mechanizm w kontekście długoterminowej pamięci sieci LSTM. Jej zadaniem jest decydowanie, które informacje z poprzedniego stanu powinny zostać zapomniane, a które zachowane np. w przypadku analizy tekstu, brama może decydować, które słowa są nieistotne i można je zapomnieć. Pozwala to uniknąć problemu „zanikania i eksplozji gradientu".



Rys. 3.3. Schemat bramy zapominania

Źródło: Opracowanie własne na podstawie Olah, C. (2015)[2].

Opis symboli:

- x_t - wejście do komórki w chwili czasowej t,
- h_{t-1} - wyjście (stan ukryty) z poprzedniej komórki LSTM,
- σ - funkcja sigmoidalna. Reprezentuje wagę, jak bardzo informacja powinna być "przepuszczona".

– f_t - wyjście z bramy zapominania, czyli sygnał decydujący, jaką część informacji z poprzedniego stanu komórki C_{t-1} zachować.

– znak x - mnożenie elementowe (ang. element-wise multiplication).

– C_{t-1} - stan komórki z poprzedniego kroku czasowego.

– C'_t - zaktualizowany stan komórki po bramie zapominania.

Wyjście z bramy zapominania jest obliczane jako:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

W_f oznacza wagi bramy zapominania, b_f to bias bramy zapominania, a funkcja sigmoidalna przekształca wynik do przedziału (0, 1), wartości bliskie 0 oznaczają, że informacja powinna być zapomniana, a wartości bliskie 1, że powinna zostać zachowana.

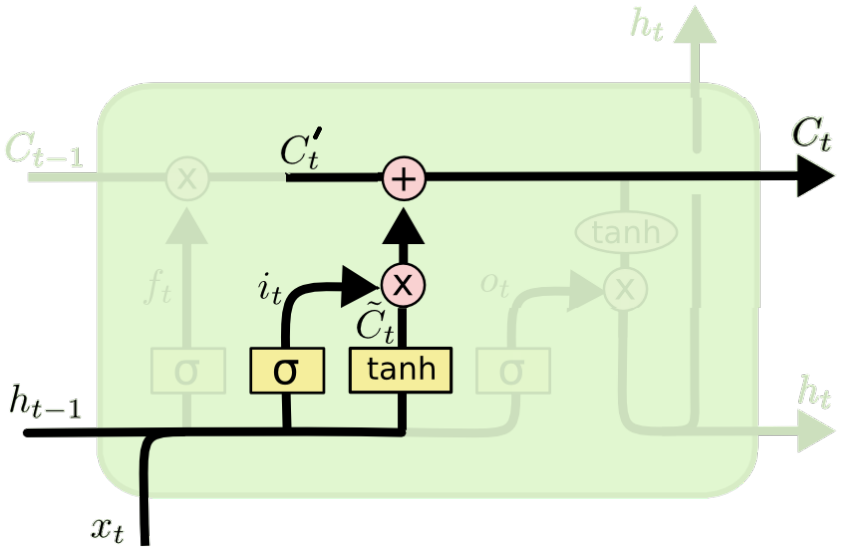
W następnym kroku przy pomocy równania:

$$C'_t = f_t \odot C_{t-1}$$

Brama decyduje, które informacje z poprzedniego stanu komórki C_{t-1} zostaną zachowane, a które zapomniane. Zapominanie pozwala uniknąć przepełnienia nieistotnymi informacjami, a zapamietywanie pozwala na długoterminowe zależności.

3.2.2. Brama wejściowa

Brama wejściowa [3.4] (ang. input gate) decyduje, które nowe informacje zostaną dodane do stanu komórki i nadaje im odpowiednią wagę. Dzięki temu zachowuje równowagę między nowymi informacjami a tymi, które już zostały zapamiętane.



Rys. 3.4. Schemat bramy wejściowej

Źródło: Opracowanie własne na podstawie Olah, C. (2015)[2].

Opis symboli:

- i_t - brama wejściowa (input gate) decyduje, które nowe informacje zostaną zachowane.
- $\tilde{C}_t$ - nowe informacje, które zostaną dodane do stanu komórki.
- C_t - nowy stan pamięci komórki po aktualizacji.
- *tanh* - funkcja aktywacji hiperboliczna, która nadaje wartościom zakres (-1, 1).

Początkowo obliczamy jak duża część nowych informacji zostanie dodana do stanu komórki:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

Następnie obliczamy propozycję aktualizacji stanu komórki:

$$\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$$

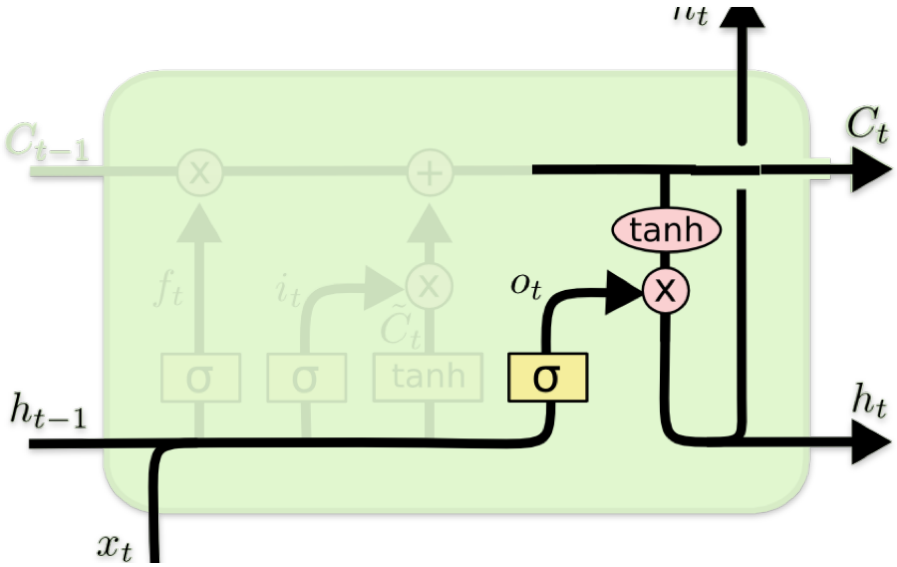
Na końcu aktualizujemy stan komórki:

$$C_t = C'_t + i_t \odot \tilde{C}_t$$

Brama wejściowa decyduje, które nowe informacje zostaną dodane do stanu komórki, a także nadaje im odpowiednią wagę. Bez tej bramy sieć LSTM straciłaby zdolność do uczenia się nowych zależności. A w połączeniu z bramą zapominania, pozwala na efektywne aktualizowanie stanu komórki.

3.2.3. Brama wyjściowa

Brama wyjściowa [3.5] (ang. output gate) ostatni element komórki LSTM, który decyduje, które informacje z aktualnego stanu komórki zostaną przekazane do wyjścia sieci jako stan ukryty. Stanowi ona równocześnie role ostaniego filtra.



Rys. 3.5. Schemat bramy wyjściowej

Źródło: Opracowanie własne na podstawie Olah, C. (2015)[2].

Opis symboli:

- o_t - brama wyjściowa (output gate) decyduje, które informacje z aktualnego stanu komórki zostaną przekazane do wyjścia sieci.
- h_t - stan ukryty (hidden state) sieci LSTM, który jest przekazywany do następnej komórki.

Wartość bramy wyjściowej jest obliczana jako:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

Finalne wyjście z komórki LSTM jest obliczane jako:

$$h_t = o_t \odot tanh(C_t)$$

Brama wyjściowa kontroluje które informacje z aktualnego stanu komórki są istotne dla bieżących obliczeń i powinny zostać przekazane do następnej komórki.

3.2.4. Pełny przepływ informacji w komórce LSTM

Pełny proces przetwarzania informacji w pojedynczym kroku czasowym zaczyna się od podania na wejście x_t - bieżące wejście, c_{t-1} - poprzedni stan komórki oraz h_{t-1} - stan ukryty z poprzedniego kroku czasowego. Następnie wszystkie trzy bramy f_t, i_t i o_t są obliczane równocześnie, razem z kandydatem $\tilde{C}_t$ do aktualizacji stanu komórki. Następnie stan komórki C_t jest aktualizowany. Potem następuje obliczenie stanu ukrytego h_t który na końcu jest przekazywany do następnej komórki LSTM.

3.3. Backpropagation Through Time (BPTT)

(Fragment opracowany przy użyciu modelu językowego ChatGPT (OpenAI)[4]

Uczenie sieci LSTM polega na minimalizacji funkcji kosztu, jest to osiągane poprzez dostosowywanie wag w sieci. Proces ten jest realizowany przy pomocy algorytmu wstecznej propagacji błedów w czasie (ang. Backpropagation Through Time, BPTT). Algorytm ten jest rozszerzeniem klasycznego algorytmu wstecznej propagacji błędów (ang. Backpropagation)[3].

Dla sieci LSTM dla danego kroku czasowego t mamy x_t - wejście, h_{t-1} - stan ukryty z poprzedniego kroku czasowego oraz C_{t-1} - stan komórki z poprzedniego kroku czasowego jako dane wejściowe. I zgodnie z poprzednimi punktami mamy:

$$z_t = [h_{t-1}, x_t]$$

$$a_f = W_f z_t + b_f$$

$$a_i = W_i z_t + b_i$$

$$a_o = W_o z_t + b_o$$

$$a_C = W_C z_t + b_C$$

$$f_t = \sigma(a_f)$$

$$i_t = \sigma(a_i)$$

$$o_t = \sigma(a_o)$$

$$\tilde{C}_t = tanh(a_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot tanh(C_t)$$

Celem jest obliczenie gradientów względem parametrów $(W_f, W_i, W_o, W_C, b_f, b_i, b_o, b_C)$ na podstawie błędu $\mathcal{L}$ (np. MSE lub cross-entropy).

Zakładamy, że strata $\mathcal{L}$ zależy od wyjścia h_t:

$$\mathcal{L} = \mathcal{L}(h_t, y_t)$$

Aby znaleźć gradienty, musimy obliczyć pochodne funkcji straty względem każdej bramy.

Gradienty od straty względem wyjścia:

$$\delta_t^h := \frac{\partial \mathcal{L}}{\partial h_t}$$

Ale:

$$h_t = o_t \odot tanh(C_t)$$

z tego wynika:

$$\frac{\partial h_t}{\partial o_t} = tanh(C_t)$$

$$\frac{\partial h_t}{\partial o_t} = o_t \odot (1 - \tanh^2(c_t))$$

więc:

$$\delta_t^o = \delta_t^h \odot tanh(C_t)$$

$$\delta_t^c = \delta_t^h \odot o_t \odot (1 - \tanh^2(C_t)) + \delta_{t+1}^c \odot f_{t+1}$$

Dodajemy $\delta_{t+1}^c \odot f_{t+1}$ bo c_{t+1} zależy od c_t.

Gradienty względem bramek: Z równania stanu komórki:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Zatem:

$$\delta_t^f = \delta_t^c \odot c_{t-1}$$

$$\delta_t^i = \delta_t^c \odot \tilde{C}_t$$

$$\delta_t^{\tilde{C}} = \delta_t^c \odot i_t$$

Gradienty względem aktywacji przed nieliniowościami: Używamy pochodnych funkcji sigmoidalnej i tangensa hiperbolicznego:

- $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
- $\tanh'(x) = 1 - \tanh^2(x)$

Otrzymujemy:

$$\delta_t^{a_f} = \delta_t^f \odot f_t \odot (1 - f_t)$$

$$\delta_t^{a_i} = \delta_t^i \odot i_t \odot (1 - i_t)$$

$$\delta_t^{a_o} = \delta_t^o \odot o_t \odot (1 - o_t)$$

$$\delta_t^{a_C} = \delta_t^{\tilde{C}} \odot (1 - \tilde{C}_t^2)$$

Gradienty względem wag:

$$\frac{\partial \mathcal{L}}{\partial W_f} = \delta_t^{a_f} \odot z_t^T$$

$$\frac{\partial \mathcal{L}}{\partial W_i} = \delta_t^{a_i} \odot z_t^T$$

$$\frac{\partial \mathcal{L}}{\partial W_o} = \delta_t^{a_o} \odot z_t^T$$

$$\frac{\partial \mathcal{L}}{\partial W_C} = \delta_t^{a_C} \odot z_t^T$$

Podobnie dla biasów:

$$\frac{\partial \mathcal{L}}{\partial b_f} = \delta_t^{a_f}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \delta_t^{a_i}$$

$$\frac{\partial \mathcal{L}}{\partial b_o} = \delta_t^{a_o}$$

$$\frac{\partial \mathcal{L}}{\partial b_C} = \delta_t^{a_C}$$

Gradienty względem h_{t-1} i x_t: Pochodna po $z_t = [h_{t-1}, x_t]$:

$$\delta_t^z = W_f^T \delta_t^{a_f} + W_i^T \delta_t^{a_i} + W_o^T \delta_t^{a_o} + W_C^T \delta_t^{a_C}$$

Następnie dzielimy ten wektor spowrotem:

$$\delta_t^{h_{t-1}} = \delta_t^z[:m]$$

$$\delta_t^{x_t} = \delta_t^z[m:]$$

Wszystkie powyższe gradienty muszą być sumowane przez wszystkie kroki czasowe od $t = T$ do $t = 1$, to właśnie nazywamy Backpropagation Through Time (BPTT).

Otrzymujemy zatem np.:

$$\frac{\partial \mathcal{L}}{\partial W_f} = \sum_{t=1}^{T} \delta_t^{a_f} \odot z_t^T$$

4. Eksperymenty

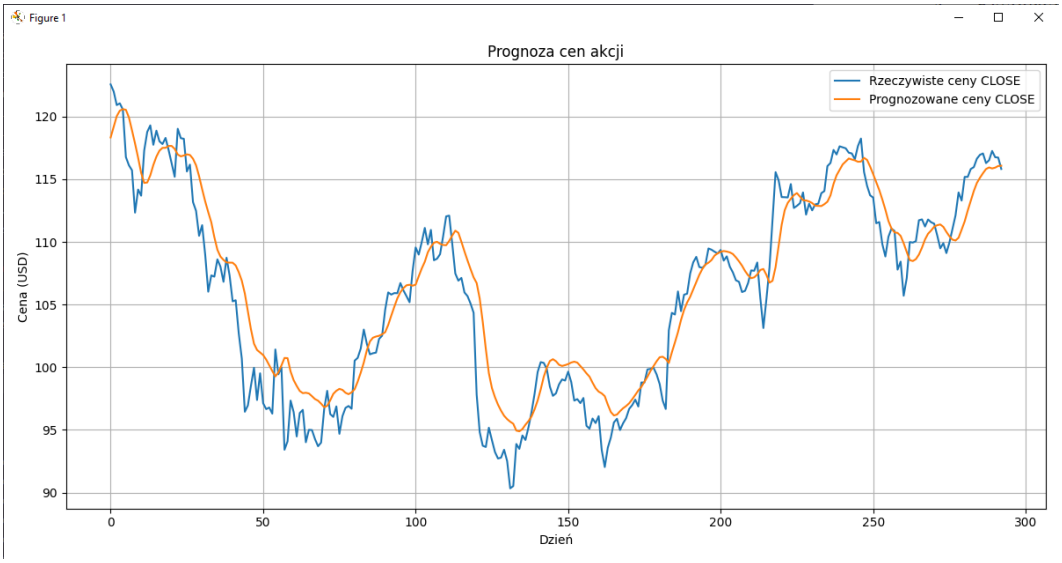
4.1. Standardowy model LSTM - punkt odniesienia

Przed rozpoczęciem eksperymentów przygotowano model referencyjny [4.1], stanowiący punkt wyjścia do dalszej analizy. Model ten wykorzystuje podstawową konfigurację warstwy LSTM i jest trenowany na wcześniej zaprezentowanych danych. Celem jest predykcja ceny zamknięcia (Close) na podstawie danych z poprzednich dni. Model ten będzie dalej zmieniany na potrzeby eksperymentów.

Listing 4.1. Implementacja modelu LSTM do predykcji cen akcji

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size=64, num_layers=2):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out, _ = self.lstm(x)
        return self.fc(out[:, -1])

input_size = X_train.shape[2]
model = LSTMModel(input_size=input_size)

X_train_t = torch.tensor(X_train, dtype=torch.float32)
y_train_t = torch.tensor(y_train, dtype=torch.float32)
X_test_t = torch.tensor(X_test, dtype=torch.float32)
y_test_t = torch.tensor(y_test, dtype=torch.float32)

train_dataset = TensorDataset(X_train_t, y_train_t)
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)

# === Trening modelu ===
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

for epoch in range(10):
    model.train()
    for xb, yb in train_loader:
        optimizer.zero_grad()
        output = model(xb)
        loss = criterion(output, yb)
        loss.backward()
        optimizer.step()
    print(f"Epoch {epoch+1}, Loss: {loss.item():.6f}")
```

Model ten przewiduje [4.1] ceny akcji ze średnią wartością błędu wynoszącą 0.000237, obliczoną na podstawie 10 prób. Co oznacza, że średnio przewidywana cena akcji różni się od rzeczywistej o 2.392$ podczas gdy średnia dzienna zmienność akcji wynosi 4.151$.



Rys. 4.1. Przewidywana i rzeczywista cena akcji AAPL

4.2. Eksperymenty z różnymi hiperparametrami

Celem tego eksperymentu jest zbadanie wpływu różnych hiperparametrów na jakość przewidywań modelu LSTM. W szczególności skupimy się na:

- liczbie warstw LSTM,
- liczbie neuronów w warstwie LSTM,
- rozmiarze partii (batch size),
- liczbie epok,
- współczynniku uczenia (learning rate).

Liczba warstw LSTM [4.1] ma istotny wpływ na wynik uczenia. Zwiększenie liczby warstw LSTM z 1 do 2 prowadzi do znacznego zmniejszenia błędu, co sugeruje, że model lepiej uczy się złożonych wzorców w danych. Jednak dalsze zwiększanie liczby warstw do 3 i 4 prowadzi do ponownego wzrostu błędu.

Tabela 4.1. Liczba warstw LSTM

Liczba warstw	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
1	0.000066	0.001133	0.000289	0.000194	0.000416	0.000420
2	0.000043	0.000103	0.000109	0.000228	0.000348	0.000166
3	0.002081	0.000345	0.000127	0.000543	0.000377	0.000695
4	0.000155	0.000483	0.000304	0.000282	0.000276	0.000300

Liczba neuronów [4.2] ma wpływ nie tylko na jakość przewidywań, ale w szczególności na czas uczenia modelu. W przypadku 32 neuronów model uczył się najszybciej, jednak jakość przewidywań była najgorsza. Zwiększenie liczby neuronów do 64 prowadziło do znacznego zmniejszenia błędu, co stanowi rozsądny kompromis w stosunku do czasu uczenia. Zwiększenie liczby neuronów do 128 prowadzi do dalszego zmniejszenia błędu, jednak czas uczenia nieproporcjonalnie wzrasta.

Tabela 4.2. Liczba neuronów w warstwie LSTM

Liczba neuronów	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
32	0.001175	0.000381	0.000388	0.000567	0.000660	0.000434
64	0.000215	0.000513	0.000107	0.000445	0.000147	0.000285
128	0.000151	0.000175	0.000255	0.000122	0.000244	0.000189

Rozmiar partii [4.3] ma niewielki wpływ na jakość przewidywań, jednak znacząco wpływa na czas uczenia modelu. W przypadku rozmiaru partii 32 model uczył się najszybciej, zachowując przy tym dobrą jakość przewidywań. Zwiększenie rozmiaru partii do 64 prowadzi do dalszego zmniejszenia błędu, jednak czas uczenia wzrasta. Rozmiar partii równy 128 prowadzi do znacznego wzrostu czasu uczenia.

Tabela 4.3. Rozmiar partii (batch size)

Rozmiar partii	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
32	0.000585	0.000464	0.000380	0.000487	0.000322	0.000448
64	0.000441	0.000366	0.000589	0.000496	0.000204	0.000419
128	0.001203	0.000405	0.000489	0.000315	0.000211	0.000543

Liczba epok [4.4] ma istotny wpływ na jakość przewidywań. Zwiększanie liczby epok ma pozytywny wpływ na jakość przewidywań, jednak ponownie jest to związane z czasem uczenia modelu.

Tabela 4.4. Liczba epok

Liczba epok	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
5	0.001107	0.000390	0.001237	0.000424	0.000389	0.000709
10	0.000362	0.000374	0.000329	0.000316	0.000727	0.000422
20	0.001278	0.000260	0.000533	0.000045	0.000234	0.000470
30	0.000239	0.000041	0.000088	0.000328	0.000189	0.000177
50	0.000065	0.000271	0.000048	0.000305	0.000360	0.000210

Współczynnik uczenia [4.5] również ma bardzo istotny wpływ na jakość przewidywań. Dobranie za małego współczynnika podobnie jak dobranie zbyt dużego współczynnika prowadzi do drastycznego wzrostu błędu.

Tabela 4.5. Współczynnik uczenia (learning rate)

Współczynnik uczenia	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
0.0001	0.000906	0.000778	0.002880	0.000872	0.000291	0.001145
0.001	0.000115	0.000480	0.000073	0.000202	0.000177	0.000209
0.01	0.000168	0.000110	0.000306	0.000121	0.000288	0.000199
0.1	0.005461	0.001912	0.018205	0.001483	0.007643	0.006941

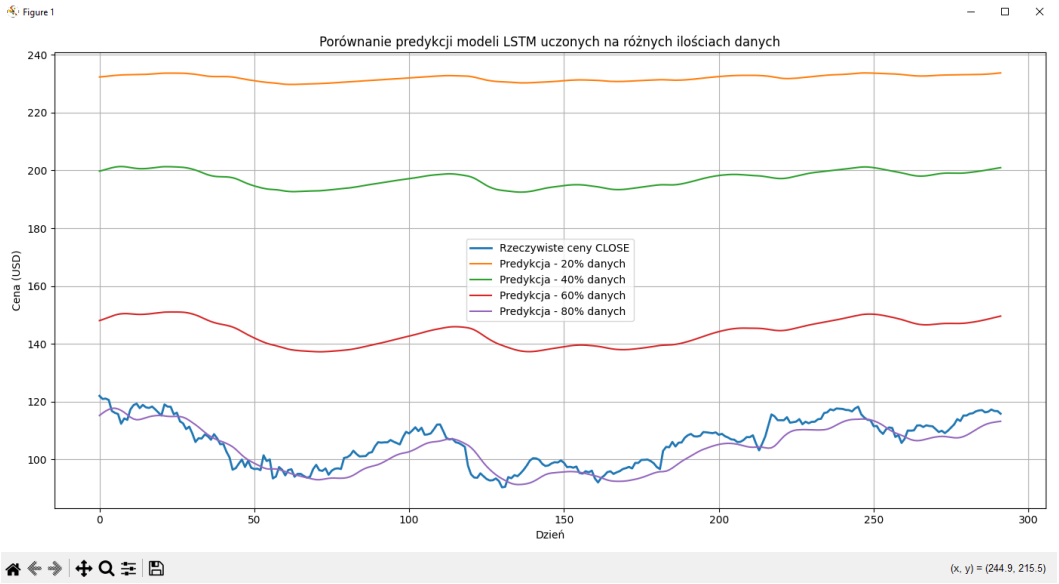
4.3. Wpływ rozmiaru danych treningowych na jakość modelu

Dane treningowe [4.6] są jedną z najważniejszych części procesu uczenia maszynowego. Bez odpowiedniej ilości danych model nie będzie w stanie nauczyć się wzorców i zależności w danych. W przypadku modelu LSTM rozmiar danych treningowych ma kluczowe znaczenie dla jakości przewidywań. Zwiększenie rozmiaru danych treningowych prowadzi do znacznego zmniejszenia błędu, co sugeruje, że model lepiej uczy się złożonych wzorców w danych. Dodatkowo, za mała ilość danych treningowych może prowadzić do znacznego pogorszenia jakości prognoz.

Tabela 4.6. Rozmiar danych treningowych

Rozmiar danych treningowych	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
20%	0.046427	0.046427	0.040606	0.036801	0.041407	0.043950
40%	0.024827	0.024827	0.027965	0.019870	0.023085	0.024115
60%	0.005702	0.005702	0.011146	0.017610	0.012690	0.010572
80%	0.000038	0.000038	0.000031	0.000040	0.000036	0.000037

Porównanie przewidywanych i rzeczywistych cen akcji przewidzianych przez modele uczone na różnych rozmiarach danych treningowych przedstawiono na rysunku [4.2]. Jak widać, jedyny użyteczny model to model uczony na 80% wszystkich danych.



Rys. 4.2. Porównanie przewidywanych cen akcji modeli uczonych na różnych rozmiarach danych
Źródło: Kod użyty do generowania wykresu napisany przy pomocy modelu językowego ChatGPT[5].

4.4. Wpływy długości sekwencji na jakość modelu

W sieci LSTM bardzo ważną rolę odgrywa pamięć długoterminowa, dlatego też wpływ długości sekwencji [4.7] na jakość modelu jest wyjątkowo ciekawy. Zwiększenie długości sekwencji prowadzi do zmniejszania się błedu, jest to tak jak wcześniej okupione wydłużonym czasem uczenia modelu, ale niestety w tym przypadku dochodzi do tego również fakt, że przy dłuższych sekwencjach ograniczamy ilość przewidywanych danych.

Tabela 4.7. Długość sekwencji

Długość sekwencji	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
10	0.000475	0.000370	0.000367	0.000711	0.000469	0.000478
20	0.000289	0.004154	0.000461	0.000260	0.000227	0.001078
30	0.000399	0.000285	0.000854	0.000402	0.000215	0.000431
60	0.000236	0.000421	0.000197	0.000164	0.000240	0.000252
100	0.000260	0.000153	0.000136	0.000302	0.000085	0.000187

4.5. Wpływ danych na jakość modelu

Jakość modelu LSTM jest ściśle związana z jakością danych [4.8], na których jest trenowany. W przypadku danych giełdowych, dla danych możemy wyróżnić kilka istotnych cech [4.9], które mogą mieć wpływ na jakość modelu:

- średnia cena zamknięcia (close),
- średni wolumen (volume),

– odchylenie standardowe ceny zamknięcia (close).

Tabela 4.8. Dane

Dane	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
AAPL	0.000130	0.000171	0.000439	0.000277	0.000244	0.000252
AMZN	0.000155	0.000155	0.000107	0.000340	0.000217	0.000195
GOOGL	0.000369	0.000384	0.000740	0.000772	0.000517	0.000556
MSFT	0.000288	0.000364	0.000721	0.000239	0.000512	0.000423
NVDA	0.000043	0.000033	0.000061	0.000032	0.000030	0.000040

Tabela 4.9. Dane statystyczne danych wejściowych

Dane	Średnia cena zamknięcia (close)	Średni wolumen (volume)	Odchylenie standardowe (close)
AAPL	312.93	94225775.88	185.15
AMZN	337.90	4607595.97	189.11
GOOGL	675.58	4096042.91	161.87
MSFT	37.13	45797836.66	10.81
NVDA	22.03	12610197.33	15.91

Możemy zauważyć, że dla GOOGL które ma najwyższą średnią cenę zamknięcia, ma również najwyższy błąd. Natomiast dla MSFT i NVDA, które mają najniższe średnie ceny zamknięcia, mają również niższy błąd. Widzimy także, że NVDA która ma niskie odchylenie standardowe, ma również najniższy błąd. Niestety nie możemy tego samego powiedzieć o MSFT, które pomimo niskiego odchylenia standardowego ma wysoki błąd. Z wyników możemy zauważyć, że nie ma jednoznacznej zależności między średnim wolumenem i odchyleniem standardowym a jakością modelu. Można jednak przypuszczać, że średnia cena zamknięcia ma wpływ na jakość modelu.

4.6. Wpływ ograniczenia liczby wejść na jakość modelu

Początkowo do modelu referencyjnego dodano wszystkie dostępne dane, mianowicie:

– Open - cena otwarcia,

– High - najwyższa cena,

– Low - najniższa cena,

– Close - cena zamknięcia,

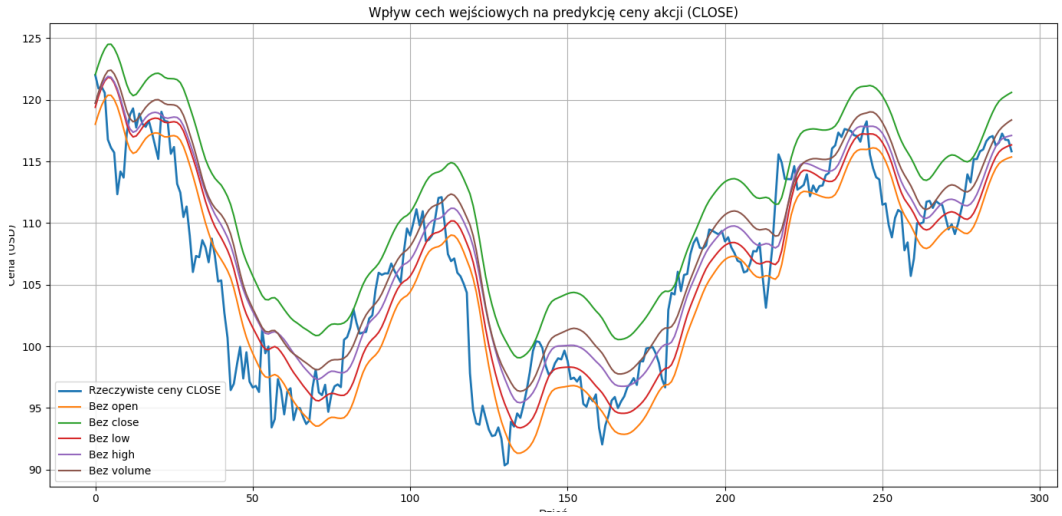
– Volume - wolumen.

Jednakże, nie wszystkie są tak samo istotne. Dlatego też przeprowadzono eksperyment mający na celu zbadanie wpływu ograniczenia liczby wejść na jakość modelu [4.10].

Tabela 4.10. Jakość modelu z pominięciem jednego z wejść

Pominięte wejście	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
Open	0.000036	0.000149	0.000029	0.000034	0.000031	0.000056
High	0.000039	0.000065	0.000066	0.000063	0.000032	0.000053
Low	0.000034	0.000044	0.000039	0.000089	0.000102	0.000062
Close	0.000099	0.000136	0.000141	0.000087	0.000091	0.000111
Volume	0.000047	0.000030	0.000056	0.000032	0.000031	0.000039

Jak widać, pominięcie jednego z wejść nie ma większego wpływu na jakość modelu. I dalej jest on w stanie przewidzieć ceny akcji [4.3]. Jednakże pominięcie ceny zamknięcia (Close) prowadzi do zauważalnego wzrostu błędu.



Rys. 4.3. Przewidywana i rzeczywista cena akcji bez pojedynczych cech

Źródło: Kod użyty do generowania wykresu napisany przy pomocy modelu językowego ChatGPT[5].

Ponadto model jest w stanie zachować dobrą jakość przewidywań dla tylko jednego wejścia [4.11]. Ale jego uczenie zajmuje więcej czasu. Jedynie dla cechy volume, która nie zawiera żadnych informacji o cenach akcji, model nie jest w stanie działać poprawnie.

Tabela 4.11. Jakość modelu dla jednego wejścia

Użyte wejście	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Średnia
Open	0.000538	0.000255	0.000600	0.001781	0.000738	0.000782
High	0.000422	0.000575	0.000321	0.000411	0.000440	0.000434
Low	0.000393	0.000313	0.000779	0.001294	0.000316	0.000619
Close	0.000264	0.000385	0.000459	0.000160	0.000402	0.000334
Volume	0.048075	0.024002	0.032752	0.043629	0.047220	0.039136

5. Podsumowanie i wnioski

W niniejszym artykule zbadano skuteczność modelu LSTM w kontekście przewidywania cen akcji na podstawie danych historycznych. Przeprowadzone eksperymenty wykazały, że model LSTM jest w stanie skutecznie przewidywać ceny akcji, ale musimy odpowiednio dobrać parametry.

Najważniejsze wnioski z przeprowadzonych badań to:

- Model LSTM jest w stanie skutecznie przewidywać ceny akcji na podstawie danych historycznych.
- Liczba warstw i liczba neuronów w modelu mają istotny wpływ na jego jakość. Zwiększenie tych parametrów poprawia wyniki, ale wydłuża czas uczenia.
- Rozmiar zbioru danych treningowych jest kluczowy - model uczony na 80% danych osiąga znacznie lepsze wyniki niż pozostałe.
- dłuższe sekwencje poprawiają wyniki, choć zmniejszają liczbę możliwych prognoz.
- Dane wejściowe mają istotny wpływ na jakość modelu, a cecha cena zamknięcia (Close) jest najważniejsza. Podczas gdy cecha (Volume) ma najmniejszy wpływ na jakość modelu i nie jest w stanie działać jako jedyne wejście.
- Wpływ jakości danych (np. średniej ceny, odchylenia standardowego) może być istotny, choć nie wykazano jednoznacznych zależności dla wszystkich spółek.

Podsumowując, model LSTM wykazuje dużą skuteczność w przewidywaniu cen akcji, o ile zostanie odpowiednio skonfigurowany. Uzyskane wyniki potwierdzają potencjał tego podejścia, jednocześnie wskazując na jego ograniczenia, takie jak wrażliwość na dane wejściowe oraz czasochłonność treningu w przypadku bardziej złożonych modeli.

Literatura

[1] Hochreiter, Sepp, and Jürgen Schmidhuber (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

[2] Olah, C. (2015). *Understanding LSTM Networks*. Dostępne online: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[3] Rumelhart D.E., Hinton G.E., & Williams R.J. (1986). *Learning representations by back-propagating errors*.

[4] Model językowy ChatGPT (OpenAI). `https://chatgpt.com/share/6813a15f-9698-8001-b900-2921b5488658`

[5] Model językowy ChatGPT (OpenAI). `https://chatgpt.com/share/682e7cd0-992c-8001-b34f-571a97016815`