**Aldona Świrad**

Uczenie maszynowe w analizie szeregów czasowych

**Praca dyplomowa inżynierska**

Opiekun pracy:  
dr Ewa Rejwer-Kosińska

Rzeszów, 2024

Spis treści

1. Wstęp
   1. Cel pracy
   2. Zakres pracy
2. Klasyczna analiza szeregów czasowych
   1. Definicja szeregów czasowych
   2. Metody analizy klasycznej
   3. Charakterystyka i zastosowanie metod klasycznych
3. Uczenie maszynowe w analizie szeregów czasowych
   1. Wprowadzenie do uczenia maszynowego
   2. Zastosowanie uczenia maszynowego w analizie szeregów czasowych
   3. Charakterystyka i zalety podejścia opartego na uczeniu maszynowym
   4. Charakterystyka i zalety podejścia opartego na uczeniu głębokim
4. Opis zbioru danych
   1. Przygotowanie danych
5. Modele do analizy szeregów czasowych w uczeniu maszynowym
   1. Klasteryzacja - ?DTW?
   2. Drzewa decyzyjne
      1. Las losowy
      2. XGBoost – gradient boosted decisions trees
6. Modele do analizy szeregów czasowych w uczeniu głębokim
   1. RNN
   2. Long Short-Term Memory (LSTM)
   3. Prophet
   4. GAN?
   5. CNN?
7. Dyskusja
   1. Porównanie modeli i interpretacja wyników
   2. Analiza i porównanie rezultatów
   3. Wnioski
8. Podsumowanie
9. Bibliografia

Teoria – robię na bieżąco

Zrobione

Do zastanowienia – jak zostanie czasu

Wykaz symboli, oznaczeń i akronimów

*X* – symbol

*Y* – symbol

1. Wstęp

Tekst akapitu.

Tekst akapitu.

1.1. Cel i zakres pracy

Celem niniejszej pracy inżynierskiej jest zbadanie i porównanie skuteczności metod analizy szeregów czasowych z wykorzystaniem podejścia klasycznego oraz metod opartych na uczeniu maszynowym, w tym uczeniu głębokim. Pod uwagę wzięte zostaną modele RNN, LSTM i Prophet (i kolejne). Głównym celem jest zastosowanie tych modeli do prognozowania szeregów czasowych oraz ocena ich przydatności w różnych scenariuszach analizy danych czasowych.

1.2. Zakres pracy???

1. Przegląd literatury dotyczący klasycznych metod analizy szeregów czasowych, w tym ich definicji, charakterystyki i zastosowań.
2. Zbadanie roli uczenia maszynowego w analizie szeregów czasowych, w tym wprowadzenie do podstawowych pojęć i metod uczenia maszynowego oraz omówienie zalet tego podejścia.
3. Przedstawienie modeli klasteryzacji, opartych na drzewach decyzyjnych, RNN, LSTM i Prophet (i kolejnych) jako przykładowych modeli uczenia maszynowego oraz uczenia głębokiego do analizy szeregów czasowych, wraz z opisem ich działania, zastosowań i ograniczeń.
4. Porównanie wybranych modeli pod kątem ich skuteczności w prognozowaniu szeregów czasowych na podstawie eksperymentów przeprowadzonych na rzeczywistych danych.
5. Analiza i dyskusja uzyskanych wyników, wraz z wnioskami dotyczącymi przydatności poszczególnych modeli w różnych scenariuszach analizy szeregów czasowych.
6. Podsumowanie pracy, wraz z sugestiami dotyczącymi dalszych badań w tej dziedzinie oraz potencjalnych zastosowań praktycznych uzyskanych wyników.

Praca będzie opierać się głównie na analizie literatury naukowej oraz eksperymentach przeprowadzonych na rzeczywistych danych, co umożliwi kompleksowe zbadanie tematu oraz wyciągnięcie trafnych wniosków dotyczących analizy szeregów czasowych za pomocą metod klasycznych i uczenia maszynowego.

5. 2 Drzewa decyzyjne

Prezentowane badanie omawia wykorzystanie modeli opartych na drzewach decyzyjnych, takich jak lasy losowe (Random Forest) i wzmocnienie gradientowe (XGBoost), w prognozowaniu szeregów czasowych z wyraźnym trendem. Podkreśla się znaczenie różnicowania danych w procesie prognozowania, szczególnie w kontekście szeregów czasowych z trendem. Prezentowane są kroki przetwarzania danych, analizy struktury szeregu czasowego, jak również implementacja prognozatorów oraz ocena ich wydajności.

Modele oparte na drzewach decyzyjnych, takie jak drzewa decyzyjne, lasy losowe (Random Forest) i gradient boosting machines (GBMs), są znane ze swojej skuteczności i szerokiego zastosowania w różnych dziedzinach uczenia maszynowego [1]. Ich popularność wynika z kilku zalet, takich jak łatwość interpretacji, zdolność do pracy z danymi o różnej naturze (ciągłe, kategoryczne) oraz możliwość obsługi brakujących wartości. Mimo tych zalet, modele te mają istotne ograniczenia, zwłaszcza gdy chodzi o ekstrapolację, czyli prognozowanie wartości poza zakresem danych obserwowanych podczas treningu [2].

Ograniczenia te stają się szczególnie istotne przy prognozowaniu szeregów czasowych, które charakteryzują się wyraźnym trendem [3]. Tradycyjne modele oparte na drzewach decyzyjnych nie są w stanie efektywnie przewidywać wartości, które wykraczają poza zakres danych treningowych. Jest to problematyczne, ponieważ prognozy szeregów czasowych często wymagają przewidywania przyszłych wartości, które mogą znacznie odbiegać od wartości historycznych. W takich przypadkach prognozy generowane przez modele oparte na drzewach mogą znacznie odbiegać od rzeczywistego trendu, co prowadzi do niskiej dokładności prognoz.

Aby poradzić sobie z tym wyzwaniem, można skorzystać z kilku strategii. Jedną z najczęściej stosowanych technik jest różnicowanie, które polega na obliczaniu różnic między kolejnymi obserwacjami w szeregu czasowym. Zamiast modelować bezwzględne wartości szeregu czasowego, różnicowanie koncentruje się na modelowaniu względnych zmian między kolejnymi punktami danych. Ta transformacja może pomóc modelom lepiej uchwycić wzorce w danych, zwłaszcza w obecności trendów [4].

Po zastosowaniu różnicowania, szereg czasowy jest mniej podatny na wpływ trendów, co pozwala modelom lepiej przewidywać przyszłe wartości [5]. Po wygenerowaniu prognoz, transformacja różnicowania jest odwracana, aby odzyskać wartości w pierwotnej skali. W ten sposób uzyskuje się prognozy, które są bardziej zgodne z rzeczywistym trendem w danych [6].

Biblioteka skforecast, od wersji 0.10.0 lub wyższej [12], wprowadza nowy parametr różnicowania w swoich klasach prognozujących. Parametr ten wskazuje, że proces różnicowania musi być zastosowany przed treningiem modelu. Implementacja tego procesu jest realizowana za pomocą transformatora o nazwie skforecast.preprocessing. TimeSeriesDifferentiator. Transformator ten automatycznie różnicuje dane przed treningiem modelu i odwraca różnicowanie podczas fazy prognozowania, zapewniając, że wartości prognoz są w tej samej skali co oryginalne dane.

Pozwolenie na zarządzanie wszystkimi transformacjami wewnętrznie ma kilka istotnych zalet. Po pierwsze, zapewnia, że te same transformacje są stosowane zarówno podczas treningu modelu, jak i podczas prognozowania na nowych danych [7]. Jest to szczególnie ważne, gdy nowe dane nie następują bezpośrednio po danych treningowych, na przykład, gdy model nie jest ponownie trenowany dla każdej fazy prognozowania. W takich przypadkach automatycznie dostosowuje się rozmiar ostatniego okna potrzebnego do generowania predyktorów, stosuje różnicowanie do nowych danych i odwraca je w końcowych prognozach [8].

Te transformacje są nietrywialne i bardzo podatne na błędy, dlatego skforecast stara się unikać nadmiernego komplikowania już i tak trudnego zadania prognozowania szeregów czasowych [9]. Wprowadzenie automatycznego różnicowania wewnętrznego w bibliotece skforecast ułatwia proces modelowania szeregów czasowych z trendami, zwiększając dokładność i niezawodność prognoz.

Niniejszy artykuł pokazuje, jak różnicowanie może być wykorzystane do modelowania szeregów czasowych z pozytywnym trendem, przy użyciu modeli opartych na drzewach decyzyjnych, takich jak random forest i gradient boosting (xgboost). Prezentuje on krok po kroku, jak zastosować te techniki, aby uzyskać lepsze prognozy i jak unikać typowych błędów związanych z różnicowaniem [10].

1. Wybór danych

Chcąc wybrać szereg z mocnym trendem, posłużono się portalem fred.stlouisfed.org. FRED (Federal Reserve Economic Data) to internetowa baza danych zarządzana przez Federal Reserve Bank of St. Louis. Portal ten dostarcza szeroki zakres danych ekonomicznych i finansowych z całego świata. Jego głównym celem jest udostępnianie danych, które mogą być używane przez badaczy, analityków oraz ogół społeczeństwa do analizy i interpretacji zjawisk ekonomicznych. Szereg czasowy, który posłuży do demonstracji podanych modeli i metod analizy, to "Production, Sales, Work Started and Orders: Production Volume: Economic Activity: Manufacturing for Poland". Jest to szereg czasowy, który reprezentuje wolumen produkcji w sektorze produkcyjnym w Polsce.

* 1. Analiza struktury szeregu czasowego
* Dane w tym szeregu czasowym były zbierane miesięcznie, przez to analizując jeden rok jest dostęp do 12 próbek zebranych w tym czasie. Pierwsza próbka datowana jest na styczeń 1985 r., a ostatnia na marzec 2024r.
* Szereg czasowy cechuje się bardzo wyraźnym trendem wzrostowym. Oznacza to, że wartości obserwowane w tym szeregu mają tendencję do systematycznego wzrostu w czasie, co jest typowe dla rozwijającej się gospodarki oraz sektora produkcyjnego, który rozszerza swoje moce produkcyjne.
* Możliwe, lecz nie tak wyraźne są sezonowe wzorce, które mogą występować w regularnych odstępach czasu (np. miesięcznych lub rocznych), co jest typowe dla produkcji przemysłowej, gdzie cykle produkcyjne mogą być związane z sezonowym popytem.
* Szereg czasowy wyraźnie pokazuje moment wystąpienia pandemii COVID-19, co wpłynęło na wartości obserwowane. Pandemia miała znaczący wpływ na sektor produkcyjny, powodując zakłócenia w łańcuchach dostaw, zamknięcia zakładów produkcyjnych i zmniejszenie popytu na niektóre produkty. Te zmiany są wyraźnie widoczne w danych jako gwałtowne spadki i fluktuacje w określonych okresach.
* Szereg czasowy przed dalszą analizą został edytowany w ten sposób, że wyodrębnione zostało 10 lat (120 miesięcy). Zaczynają się one od marca 2014 do marca 2024.

1. Przetwarzanie szeregu czasowego

Środowisko (IDE) wykorzystywane w tej analizie to Visual Studio Code Microsoft. Jest ono bardzo popularne i przyjazne dla użytkowników oraz deweloperów. Język programowania używany w tym zadaniu to Python, jeden z popularniejszych i szeroko wykorzystywanych narzędzi. Python ma szerokie zastosowanie zwłaszcza w dziedzinach Data Science, dlatego został wybrany ze względu na naturę badanego zagadnienia.

* 1. Opis potrzebnych bibliotek

W celu przeprowadzenia analizy szeregów czasowych użyto kilku istotnych bibliotek i modułów w Pythonie:

Listing 1 Wykorzystywane biblioteki

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from xgboost import XGBRegressor

from sklearn.ensemble import RandomForestRegressor

from skforecast.ForecasterAutoreg import ForecasterAutoreg

from skforecast.model\_selection import backtesting\_forecaster

from skforecast.preprocessing import TimeSeriesDifferentiator

from sklearn.metrics import mean\_absolute\_error

Poniżej znajduje się krótki opis każdej z nich:

1. NumPy - fundamentalna biblioteka dla obliczeń numerycznych w Pythonie. Umożliwia przeprowadzanie operacji na tablicach wielowymiarowych oraz zapewnia bogaty zbiór funkcji matematycznych,
2. Pandas - biblioteka służąca do manipulacji i analizy danych. Umożliwia łatwe wczytywanie, przekształcanie oraz analizowanie danych tabelarycznych za pomocą struktur danych takich jak DataFrame,
3. Matplotlib - biblioteka do tworzenia statycznych, animowanych i interaktywnych wizualizacji w Pythonie. Pyplot to jej podmoduł, który umożliwia tworzenie wykresów w sposób zbliżony do MATLABa,
4. XGBoost - wydajna i elastyczna biblioteka do uczenia gradientowego boostingu, stosowana w wielu konkursach machine learningowych. XGBRegressor to jej implementacja dla zadań regresyjnych.
5. RandomForestRegressor - moduł z biblioteki scikit-learn, który implementuje algorytm lasów losowych dla zadań regresyjnych. Lasy losowe to zbiór drzew decyzyjnych, które wspólnie podejmują decyzję,
6. ForecasterAutoreg - moduł z biblioteki skforecast, który umożliwia tworzenie modeli autoregresyjnych dla szeregów czasowych. Wspiera różne algorytmy regresji, w tym drzewa decyzyjne i XGBoost,
7. Backtesting\_forecaster - narzędzie z biblioteki skforecast, które pozwala na przeprowadzanie testów historycznych (backtesting) modeli prognozujących. Umożliwia ocenę skuteczności modeli na podstawie danych historycznych,
8. TimeSeriesDifferentiator - narzędzie z biblioteki skforecast, które umożliwia różnicowanie szeregów czasowych. Proces różnicowania pomaga w stabilizowaniu trendów i sezonowości przed trenowaniem modeli,
9. Mean\_absolute\_error - funkcja z biblioteki scikit-learn, która oblicza średni błąd bezwzględny (MAE) między przewidywanymi a rzeczywistymi wartościami.
   1. Wprowadzenie danych do środowiska

Dane pobrano w formacie CSV, a następnie plik nazwano ts1.csv.

Listing 2 Wprowadzenie danych

# Wczytanie danych i konwersja na DataFrame

ts1 = pd.read\_csv('ts1.csv')

# Wyświetlenie pięciu pierwszych wierszy

print(ts1.head())

Tabela 1 Pierwsze wiersze danych szeregu czasowego.

|  |  |
| --- | --- |
| Parametr ‘DATE’ | Parametr ‘POLPROMANMISMEI’ |
| 2014-03-01 | 93.81107 |
| 2014-04-01 | 95.06747 |
| 2014-05-01 | 93.39227 |
| 2014-06-01 | 94.64867 |
| 2014-07-01 | 95.90508 |

Do dalszych działań dodano znacznik czasu, którym będzie data. Nadano kolumnie ‘DATE’ format daty.

Listing 3 Konwersja kolumny 'DATE' na typ daty

ts1['DATE'] = pd.to\_datetime(ts1['DATE'], format='%Y-%m')

Ten fragment kodu konwertuje kolumnę 'DATE' na typ daty, używając formatu %Y-%m, gdzie %Y reprezentuje rok, a %m miesiąc.

* 1. Przetwarzanie danych

Przetwarzanie danych to zestaw operacji mających na celu przekształcenie surowych danych w użyteczną formę, która może być analizowana i interpretowana.

Listing 4 Przetwarzanie danych

# Ustawienie kolumny 'DATE' jako indeks

ts1 = ts1.set\_index('DATE')

# Ustawienie częstotliwości danych na miesięczną (start miesiąca)

ts1 = ts1.asfreq('MS')

# Wybór kolumny 'POLPROMANMISMEI'

ts1 = ts1['POLPROMANMISMEI']

# Sortowanie indeksu

ts1 = ts1.sort\_index()

ts1.head(4)

Ustawiono kolumnę 'DATE' jako indeks DataFrame, co pozwala na łatwiejsze operacje na szeregach czasowych. Nadano częstotliwość indeksu na miesięczną, gdzie 'MS' oznacza początek miesiąca.

Tabela 2 Wygląd fragmentu danych po przetworzeniu danych

|  |  |
| --- | --- |
| Parametr ‘DATE' |  |
| 2014-03-01 | 93.81107 |
| 2014-04-01 | 95.06747 |
| 2014-05-01 | 93.39227 |
| 2014-06-01 | 94.64867 |

1. Różnicowanie

W kontekście analizy szeregów czasowych, "Data differentiated" odnosi się do procesu różnicowania danych (differencing), który jest techniką stosowaną w celu uczynienia szeregu czasowego stacjonarnym. Szereg czasowy jest stacjonarny, jeśli jego statystyki, takie jak średnia i wariancja, są stałe w czasie. Różnicowanie jest szczególnie przydatne, gdy dane mają trend lub sezonowość.

Różnicowanie polega na odejmowaniu wartości poprzedniego okresu od wartości bieżącej, co formalnie zapisuje się jako:

Równanie 1 Różnicowanie

Gdzie:

– to wartości szeregu czasowego o okresie t

- to wartości szeregu czasowego o okresie t-1

- to zróżnicowana wartość szeregu czasowego o okresie t

* 1. Działanie różnicowania

Różnicowanie pomaga usunąć trend i sezonowość z szeregu czasowego, co czyni go bardziej stacjonarnym. Jest to ważny krok w analizie szeregów czasowych. Używam do tego funkcji TimeSeriesDifferentiator.

Utworzono instancję klasy TimeSeriesDifferentiator, z parametrem order=1. Oznacza on, że dokonano pierwsze różnicowanie, tj. różnicę pomiędzy każdą wartością a jej poprzednikiem. TimeSeriesDifferentiator jest częścią biblioteki specyficznej dla obróbki szeregów czasowych.

Listing 5 Instancja klasy TimeSeriesDifferenitiator

diferenciator = TimeSeriesDifferentiator(order=1)

* 1. Konwersja danych na dwa różne formaty

Przekonwertowanie danych na numpy.ndarray jest wymagane, ponieważ niektóre metody z biblioteki skforecast mogą nie obsługiwać bezpośrednio obiektów pandas.Series czy DataFrame. Do tego celu stworzone zostały dwie zmienne:

1. ts1 – pozostanie ona w formacie pandas.core.series.Series
2. ts11 – przekonwertuje ją na zmienną typu numpy.ndarray

Listing 6 Zmienne ts1 i ts11 w różnych formatach

type(ts1)

pandas.core.series.Series

# Konwersja pandas.Series na numpy.ndarray

ts11 = ts1.values

type(ts11)

numpy.ndarray

* 1. Zastosowanie różnicowania na danych

Metoda fit\_transform(data) najpierw "dopasowuje" różnicowanie do danych, a następnie przekształca dane, zwracając różnicowane wartości.

Listing 7 Zastosowanie różnicowania na danych

ts1\_diff = diferenciator.fit\_transform(ts11)

ts1\_diff.head(4)

Tabela 3 Fragment danych po różnicowaniu

|  |  |
| --- | --- |
| Parametr ‘DATE’ | Wartość danych po zróżnicowaniu |
| 2014-04-01 | 1.25640 |
| 2014-05-01 | -1.67520 |
| 2014-06-01 | 1.25640 |
| 2014-07-01 | 1.25641 |

1. Podział danych na dane testowe i treningowe

Podział danych na dane testowe i treningowe jest kluczowym krokiem w uczeniu maszynowym. Zapewnia on to, że model jest oceniany na danych, których nigdy wcześniej nie widział, co pozwala na sprawdzenie jego zdolności do generalizacji. Dokonano podziału danych w proporcji 2:1. Dane treningowe zawierają informacje o okresie pandemii.

Przeniesienie danych z okresu pandemii do danych treningowych może wpłynąć na sposób, w jaki model uczy się reprezentować ten szczególny czas. Jednak, jak można zauważyć w późniejszym etapie, może to nie mieć aż tak wielkiego wpływu, jakby się wydawało. Sugeruje to, że model potrafi efektywnie generalizować poza ten okres. Jest to ciekawy aspekt analizy danych, który może prowadzić do głębszego zrozumienia zachowań modelu w obliczu nieprzewidywalnych okoliczności, takich jak pandemia.

Ważne jest również, aby upewnić się, że dane testowe rozpoczynają się od momentu, który jest zaraz po okresie danych treningowych. To zapewnia, że model jest testowany na danych, które są rzeczywiście nowe dla niego i które powinny odzwierciedlać warunki poza okresem, na którym był trenowany.

Listing 8 Podział danych na dane treningowe i testowe

end\_train = '2021-03-01'

print(

    f"Train dates : {ts1.index.min()} --- {ts1.loc[:end\_train].index.max()}  "

    f"(n={len(ts1.loc[:end\_train])})")

print(

    f"Test dates  : {ts1.loc[end\_train:].index.min()} --- {ts1.index.max()}  "

    f"(n={len(ts1.loc[end\_train:])})")

Tabela 4 Dane po podziale na część treningową i testową

|  |  |
| --- | --- |
| Dane treningowe | 2014-03-01 00:00:00 --- 2021-03-01 00:00:00 (n=85) |
| Dane testowe | 2021-03-01 00:00:00 --- 2024-03-01 00:00:00 (n=37) |

Aby zilustrować ten podział możemy stworzyć wykresy przedstawiające dane testowe i dane treningowe.

Listing 9 Kod generujący wykresy danych testowych i danych treningowych

fig, axs = plt.subplots(1, 2, figsize=(11, 2.5))

axs = axs.ravel()

ts1.loc[:end\_train].plot(ax=axs[0], label='train')

ts1.loc[end\_train:].plot(ax=axs[0], label='test')

axs[0].legend()

axs[0].set\_title('Dane oryginalne')

ts1\_diff.loc[:end\_train].plot(ax=axs[1], label='train')

ts1\_diff.loc[end\_train:].plot(ax=axs[1], label='test')

axs[1].legend()

axs[1].set\_title('Dane zróżnicowane')

Obraz zawierający tekst, linia, Wykres, Czcionka

Opis wygenerowany automatycznie

Rysunek 1 Wykresy podziałów danych oryginalnych i danych zróżnicowanych ze względu na dane testowe i treningowe.

1. Prognozowanie używając drzew losowych i wzmocnienia gradientowego

Utworzono dwa autoregresyjne prognozatory. Jeden za pomocą RandomForestRegressor z biblioteki scikit-learn, a drugi z użyciem XGBoost.

RandomForestRegressor i XGBRegressor są dwoma popularnymi modelami używanymi do problemów regresji w uczeniu maszynowym. Oto krótkie opisy obu modeli:

Bardziej opisac algorytmy. I wypłaszczenie+ notebook colab.

1. RandomForestRegressor - jest modelem zespołowym opartym na drzewach decyzyjnych. Tworzy on wiele drzew decyzyjnych podczas treningu i łączy ich przewidywania w celu uzyskania bardziej stabilnych i dokładnych prognoz. Każde drzewo decyzyjne w lesie losowym jest trenowane na losowo wybranym podzbiorze danych treningowych i losowo wybranych funkcjach, co pomaga uniknąć przetrenowania i zwiększyć różnorodność drzew w lesie. Podczas prognozowania, wyniki z poszczególnych drzew są uśredniane, co pomaga zredukować wariancję modelu,
2. XGBRegressor - implementacja algorytmu Gradient Boosting Machine, która wykorzystuje bibliotekę XGBoost. Jest to silny model zespołowy, który buduje sekwencyjnie drzewa decyzyjne, a następnie poprawia błędy poprzednich drzew poprzez dopasowywanie nowych drzew do reszt (gradientu) poprzednich przewidywań. XGBoost wykorzystuje wiele technik optymalizacyjnych, takich jak regularyzacja, próbkowanie wierszy i kolumn, aby zapobiec przetrenowaniu i zwiększyć wydajność modelu. Ponadto, XGBoost umożliwia elastyczne dostosowywanie hiperparametrów, co pozwala na optymalizację modelu pod kątem konkretnych danych i problemu.

Oba modele, RandomForestRegressor i XGBRegressor, są potężnymi narzędziami w dziedzinie uczenia maszynowego i znajdują zastosowanie w szerokim zakresie problemów regresji, od przewidywania cen nieruchomości po prognozowanie przychodów biznesowych. Oba zostały wytrenowane na danych od 2014-03-01 do 2024-03-01 i generują prognozy na kolejne 36 miesięcy (3 lata).

* 1. Prognozowanie bez różnicowania

Prognozowanie bez uwzględniania różnicowania w danych może prowadzić do kilku istotnych konsekwencji. Po pierwsze, model, zwłaszcza z wyraźnym trendem, może nie być w stanie uchwycić subtelnych różnic między różnymi grupami lub warunkami, co może prowadzić do przeszacowania lub niedoszacowania pewnych zjawisk. Na przykład, w przypadku analizy danych związanych z pandemią, brak różnicowania może sprawić, że model nie będzie w stanie uwzględnić różnych strategii zarządzania pandemią, które mogą mieć wpływ na dane. Ponadto, prognozowanie bez różnicowania może prowadzić do uproszczeń i generalizacji, które mogą nie odzwierciedlać rzeczywistości w pełni. Aby to udowodnić wykonam prognozy utworzone bez wcześniejszego różnicowania.

Listing 10 Przygotowanie do tworzenia prognoz bez różnicowania

steps = len(ts1.loc[end\_train:])

forecaster\_rf = ForecasterAutoreg(

                    regressor = RandomForestRegressor(random\_state=963),

                    lags      = 12  
 # brak różnicowania

                )

forecaster\_gb = ForecasterAutoreg(

                    regressor = XGBRegressor(random\_state=963),

                    lags      = 12

# brak różnicowania

Listing 11 Trenowanie danych bez różnicowania

# Train

forecaster\_rf.fit(ts1.loc[:end\_train])

forecaster\_gb.fit(ts1.loc[:end\_train])

Listing 12 Predykcja danych bez różnicowania

# Predict

predictions\_rf = forecaster\_rf.predict(steps=steps)

predictions\_gb = forecaster\_gb.predict(steps=steps)

Błąd MEA (Mean Absolute Error) jest jednym z powszechnie stosowanych wskaźników oceny wydajności modelu w uczeniu maszynowym i analizie danych. Jest to miara średniej wartości bezwzględnej różnicy między wartościami przewidywanymi przez model a rzeczywistymi wartościami.

Aby obliczyć błąd MEA, dla każdej obserwacji obliczana jest różnica między wartością przewidywaną przez model a rzeczywistą wartością, a następnie wszystkie te różnice są uśredniane, ignorując ich kierunek (czyli wartości bezwzględne).

Błąd MEA jest wyrażony w tych samych jednostkach co oryginalne dane, co ułatwia interpretację. Im niższa wartość błędu MEA, tym lepiej dopasowany jest model do danych. Jest to użyteczne narzędzie zarówno podczas tworzenia modelu, jak i jego oceny, ponieważ umożliwia porównanie wydajności różnych modeli lub ocenę, czy model osiąga satysfakcjonujące rezultaty.

Listing 13 Obliczenie błędu MAE danych bez różnicowania

# Error

error\_rf = mean\_absolute\_error(ts1.loc[end\_train:], predictions\_rf)

error\_gb = mean\_absolute\_error(ts1.loc[end\_train:], predictions\_gb)

print(f"Error (MAE) Random Forest: {error\_rf:.2f}")

print(f"Error (MAE) Gradient Boosting: {error\_gb:.2f}")

Tabela 5 Błędy MEA dla danych bez różnicowania

|  |  |
| --- | --- |
| Nazwa algorytmu | Błąd MAE |
| Random Forest | 15.56 |
| Gradient Boosting | 12.78 |

Otrzymane błędy nie są duże, lecz nadal są większe niż gdyby przeprowadzić wcześniej różnicowanie. Chcąc sprawdzić jak dane modele poradziły sobie w prognozowaniu można narysować odpowiedni wykres.

Listing 14 Kod generujący wykres prognozowania bez różnicowania

fig, ax = plt.subplots(figsize=(7, 3), sharex=True, sharey=True)

ts1.loc[:end\_train].plot(ax=ax, label='train')

ts1.loc[end\_train:].plot(ax=ax, label='test')

predictions\_rf.plot(ax=ax, label='Random Forest')

predictions\_gb.plot(ax=ax, label='Gradient Boosting')

ax.set\_title(f'Prognozowanie bez różnicowania')

ax.set\_xlabel('')

ax.legend()

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

Rysunek 2 Wykres prognozowania bez różnicowania

Wykres pokazuje, że żaden z modeli nie jest w stanie dokładnie przewidzieć trendu. Po kilku krokach prognozy stają się niemal stałe, zbliżone do maksymalnych wartości obserwowanych w danych treningowych.

Następnie dwa nowe prognozatory są trenowane przy użyciu tej samej konfiguracji, ale z argumentem differentiation = 1. Aktywuje to wewnętrzny proces różnicowania szeregu czasowego przed trenowaniem modelu i odwraca różnicowanie, znane również jako całkowanie, dla wartości prognozowanych.

* 1. Prognozowanie z różnicowaniem

Prognozowanie z różnicowaniem dla danych z wyraźnym trendem ma na celu uwzględnienie zmian w trendzie w różnych grupach lub warunkach w danych w celu uzyskania bardziej precyzyjnych prognoz. Gdy dane wykazują wyraźny trend, różnice między grupami mogą wpływać na dynamikę tego trendu, dlatego ważne jest uwzględnienie tych różnic w procesie prognozowania.

Listing 15 Przygotowanie do tworzenia prognoz z różnicowaniem

steps = len(ts1.loc[end\_train:])

forecaster\_rf = ForecasterAutoreg(

                    regressor       = RandomForestRegressor(random\_state=910),

                    lags            = 12,

                    differentiation = 1 #dodajemy różnicowanie

                )

forecaster\_gb = ForecasterAutoreg(

                    regressor       = XGBRegressor(random\_state=910),

                    lags            = 12,

                    differentiation = 1 #dodajemy różnicowanie

                )

Listing 16 Trenowanie danych z różnicowaniem

# Train

forecaster\_rf.fit(ts1.loc[:end\_train])

forecaster\_gb.fit(ts1.loc[:end\_train])

Listing 17 Predykcja danych z różnicowaniem

# Predict

predictions\_rf = forecaster\_rf.predict(steps=steps)

predictions\_gb = forecaster\_gb.predict(steps=steps)

Listing 18 Obliczenie błędu MEA danych z różnicowaniem

# Error

error\_rf = mean\_absolute\_error(ts1.loc[end\_train:], predictions\_rf)

error\_gb = mean\_absolute\_error(ts1.loc[end\_train:], predictions\_gb)

print(f"Error (MAE) Random Forest: {error\_rf:.2f}")

print(f"Error (MAE) Gradient Boosting: {error\_gb:.2f}")

Tabela 6 Błędy MAE dla danych z różnicowaniem

|  |  |
| --- | --- |
| Nazwa algorytmu | Błąd MAE |
| Random Forest | 8.01 |
| Gradient Boosting | 4.56 |

Można zauważyć, że błędy MAE są zdecydowanie mniejsze niż gdy nie wykonałam różnicowania. Zmiany można również zbadać na wykresie. Dane nie są idealnie dopasowane, co jest poprawne, lecz mogłyby być lepiej dopasowane, lecz występująca pandemia może mieć wpływ na działanie algorytmów. Niemniej jednak, dane dalej są prognozowane w zadawalający sposób. Zwłaszcza dużo lepiej spisuje się Gradient Boosting.

Listing 19 Kod generujący wykres prognozowania z różnicowaniem

fig, ax = plt.subplots(figsize=(7, 3), sharex=True, sharey=True)

ts1.loc[:end\_train].plot(ax=ax, label='train')

ts1.loc[end\_train:].plot(ax=ax, label='test')

predictions\_rf.plot(ax=ax, label='Random Forest')

predictions\_gb.plot(ax=ax, label='Gradient Boosting')

ax.set\_title(f'Prognozowanie z różnicowaniem')

ax.set\_xlabel('')

ax.legend()

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

Rysunek 3 Wykres prognozowania z różnicowaniem

Poprzedni przykład pokazał, jak łatwo jest wprowadzić różnicowanie do procesu prognozowania dzięki funkcjonalnościom dostępnym w skforecast. Jednakże, aby osiągnąć płynną interakcję, konieczne jest zastosowanie kilku nietrywialnych transformacji.

1. Transformator TimeSeriesDifferentiator

TimeSeriesDifferentiator to niestandardowy transformator, który został zaprojektowany do przetwarzania danych szeregów czasowych w kontekście uczenia maszynowego, a szczególnie jako część przetwarzania wstępnego w bibliotece Scikit-learn (sklearn). Jako transformator Scikit-learna, TimeSeriesDifferentiator implementuje metody fit, transform, fit\_transform oraz inverse\_transform, które są standardowymi metodami interfejsu API Scikit-learn.

Proces odwrotnej transformacji, inverse\_transform, można zastosować tylko do tego samego szeregu czasowego, który wcześniej był różnicowany za pomocą tego samego obiektu TimeSeriesDifferentiator. To ograniczenie wynika z konieczności użycia początkowych n wartości szeregu czasowego ,gdzie n równa się rzędowi różnicowania, do pomyślnego odwrócenia różnicowania. Te wartości są przechowywane podczas wykonywania metody fit.

W transformatorze TimeSeriesDifferentiator dostępna jest dodatkowa metoda inverse\_transform\_next\_window. Metoda ta została zaprojektowana do użycia wewnątrz prognozatorów (Forecasters) w celu odwrócenia różnicowania wartości prognozowanych. Jeśli regressor prognozatora jest trenowany na różnicowanym szeregu czasowym, to prognozowane wartości również będą różnicowane. Metoda inverse\_transform\_next\_window pozwala na przywrócenie prognoz do oryginalnej skali, zakładając, że zaczynają się one bezpośrednio po ostatnich obserwowanych wartościach (last\_window).

* 1. Różnicowanie wewnętrzne, a przetwarzanie wstępne

Prognozatory zarządzają procesem różnicowania wewnętrznie, więc nie ma potrzeby dodatkowego przetwarzania wstępnego szeregu czasowego ani postprocesowania prognoz. Porównane zostaną wyniki obu podejść.

Listing 20 Różnicowanie szeregu czasowego za pomocą TimeSeriesDifferentiator

diferenciator = TimeSeriesDifferentiator(order=1)

data\_diff = diferenciator.fit\_transform(ts11)

data\_diff = pd.Series(data\_diff, index=ts1.index).dropna()

forecaster = ForecasterAutoreg(

                 regressor = RandomForestRegressor(random\_state=963),

                 lags      = 15

             )

forecaster.fit(y=data\_diff.loc[:end\_train])

predictions\_diff = forecaster.predict(steps=steps)

# Revert differentiation to obtain final predictions

last\_value\_train = ts1.loc[:end\_train].iloc[[-1]]

predictions\_1 = pd.concat([last\_value\_train, predictions\_diff]).cumsum()[1:]

predictions\_1 = predictions\_1.asfreq('MS')

predictions\_1.name = 'pred'

predictions\_1.head(5)

Tabela 7 Fragment danych po zróżnicowaniu szeregu czasowego za pomocą TimeSeriesDifferentiator

|  |  |
| --- | --- |
| Data | Prognoza |
| 2021-04-01 | 139.062341 |
| 2021-05-01 | 139.041399 |
| 2021-06-01 | 138.315471 |
| 2021-07-01 | 138.961815 |
| 2021-08-01 | 138.351763 |

Listing 21 Błędy MAE danych po uzyciu TimeSeriesDifferentiator

# Error

error\_1 = mean\_absolute\_error(ts1.loc[end\_train:], predictions\_1)

error\_1

8.622837859459462

Można zauważyć, że błąd MAE jest średniej wielkości, raczej interpretowany jako nieduży. Dobrze zwiastuje to na naszą prognozę.

Listing 22 Kod generujący wykres prognozowania TimeSeriesDifferentiator

fig, ax = plt.subplots(figsize=(7, 3), sharex=True, sharey=True)

ts1.loc[:end\_train].plot(ax=ax, label='train')

ts1.loc[end\_train:].plot(ax=ax, label='test')

predictions\_1['pred'].plot(ax=ax, label='TimeSeriesDifferentiator')

ax.set\_title(f'Prognozowanie TimeSeriesDifferentiator')

ax.set\_xlabel('')

ax.legend()

**Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie**

Rysunek 4 Wykres prognozowania po użyciu TimeSeriesDifferentiator

Jak można zauważyć, że ten sposób prognozowania wydaje się najbardziej trafny i można uznać go za najlepszy spośród badanych.

Podsumowanie modeli opartych na drzewach decyzyjnych

Wprowadzenie różnicowania do modelowania szeregów czasowych z wyraźnym trendem okazuje się kluczowym krokiem, który pozwala na lepsze przewidywanie przyszłych wartości. Ten artykuł naukowy eksploruje skuteczność modeli opartych na drzewach decyzyjnych, takich jak random forest i gradient boosting, w połączeniu z techniką różnicowania, aby zaradzić problemom ekstrapolacji w prognozowaniu szeregów czasowych. Przy użyciu biblioteki skforecast, wprowadzono nowy parametr różnicowania, który automatyzuje proces różnicowania wewnętrznego. Analiza danych pokazuje wyraźny trend wzrostowy, który jest typowy dla sektora produkcyjnego. Dzięki różnicowaniu, modele są w stanie lepiej przewidywać przyszłe wartości, zwłaszcza w obecności trendów. Eksperymenty z modelami Random Forest i XGBoost zarówno z, jak i bez różnicowania, wykazały, że modele z różnicowaniem osiągają niższe błędy MAE, co sugeruje ich większą dokładność w przewidywaniu. Różnicowanie pozwala uniknąć przeszacowania i niedoszacowania, co prowadzi do lepszej generalizacji modeli w przypadku szeregów czasowych z trendami. Dodatkowo, narzędzie TimeSeriesDifferentiator biblioteki skforecast umożliwia łatwe stosowanie różnicowania w analizie szeregów czasowych, co zwiększa wygodę i efektywność procesu modelowania.

Literatura

[1] A. Nielsen. (2019). Practical Time Series Analysis: Prediction with Statistics and Machine Learning (pp. 243-250). Helion

[2] M. Muller (2007). Dynamic Time Wrapping (pp. 69-84). Springer Berlin Heidelberg

[3] Joan Serr`a, Josep Ll. Arcos (2018). An Empirical Evaluation of Similarity Measures for Time Series Classification. o Knowledge-based Systems

[4] Ch. A. Ratanamahatana , J. Lin , D. Gunopulos , E. Keogh , M. Vlachos , G. Das. (2012). Mining Time Series Data (pp. 1049-1073). Springer

[5] S.Makridakis, E. Spiliotis ,V. Assimakopoulos(2018). The M4 Competition: Results, findings, conclusion and way forward. Elsevier

[6] A. Géron (2020). Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Wydanie II, (rozdział 15.), Helion

[7] L.Zhu. N. Laptev (2017). Deep and Confident Prediction for Time Series at Uber, ICDWM, New Orleans

[8] Z. Che, S. Purushotham, K. Cho, D. Sontag , Y. Liu (2018) Recurrent Neural Networks for Multivariate Time Series with Missing Values., Nature

[9] P. Esling, C. Agon (2012). Time-series data mining. ACM Computing Surveys

[10] Davis, H. T., & Nelson, W. F. C. (1935). The analysis of time series. In H. T. Davis & W. F. C. Nelson, Elements of statistics with applications to economic data (pp. 119–145). Principia Press.

Źródła internetowe

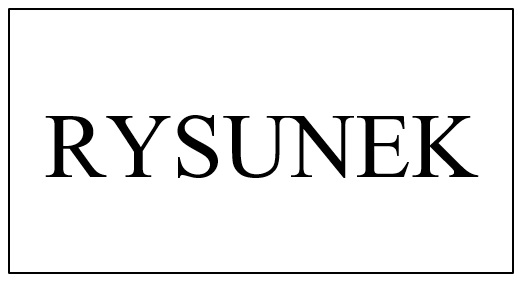
[11] https://fred.stlouisfed.org/series/POLPROMANMISMEI (dostęp: 03.05.2024)

[12] https:// skforecast.org (dostęp: 03.05.2024).

1.1.1. Tytuł zakresu

Tekst akapitu.

Tekst akapitu.



Rys. 1.1. Tytuł rysunku

Tekst akapitu.

Tekst akapitu.

Tabela 1.1. Tytuł tabeli

|  |
| --- |
| Nagłówek tabeli |
| Dana tekstowa |
| Dana liczbowa |

Tekst akapitu.

Tekst akapitu.

(1.1)

Tekst akapitu.

Tekst akapitu.

1. Pierwszy poziom wypunktowania.

- drugi poziom wypunktowania,

- koniec drugiego poziomu wypunktowania.

Tekst akapitu.

1.1.2. Tytuł zakresu

Tekst akapitu.

Tekst akapitu.

1.2. Tytuł podrozdziału

Tekst akapitu.

Tekst akapitu.

1.2.1. Tytuł zakresu

Tekst akapitu.

Tekst akapitu.

1.2.2. Tytuł zakresu

Tekst akapitu.

Tekst akapitu.

2. Tytuł rozdziału

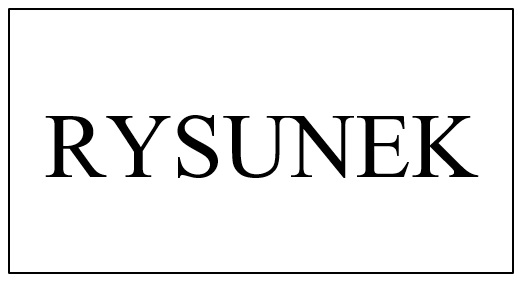
Tekst akapitu.

2.1. Tytuł podrozdziału

Tekst akapitu.

2.1.1. Tytuł zakresu

Tekst akapitu.



Rys. 2.1. Tytuł rysunku

Tekst akapitu.

Tabela 2.1. Tytuł tabeli

|  |
| --- |
| Nagłówek tabeli |
| Dana tekstowa |
| Dana liczbowa |

Tekst akapitu.

(2.1)

Tekst akapitu.

2.1.2. Tytuł zakresu

Tekst akapitu.

2.2. Tytuł podrozdziału

Tekst akapitu.

2.2.1. Tytuł zakresu

Tekst akapitu.

2.2.2. Tytuł zakresu

Tekst akapitu.

Załączniki

Literatura

[1] Pozycja literatury

[2] Pozycja literatury

|  |  |
| --- | --- |
|  | Sygnatura: |
| POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza | Rzeszów, Rok |
| Wydział Elektrotechniki i Informatyki |  |
|  |  |

**STRESZCZENIE (wybierz rodzaj pracy)**

**TYTUŁ PRACY**

Autor: Imię Nazwisko, nr albumu: (wybierz symbol studiów)-123456 Opiekun: (tytuł naukowy przed) Imię Nazwisko(tytuł naukowy po)

Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

(tekst streszczenia - max. 10 wierszy)

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, Rok

Faculty of Electrical and Computer Engineering

**DIPLOMA THESIS (wybierz rodzaj pracy) ABSTRACT**

**TYTUŁ PRACY W WERSJI ANGIELSKIEJ**

Author: Imię Nazwisko, code: (wybierz symbol studiów) -123456 Supervisor: (tytuł naukowy przed) Imię Nazwisko (tytuł naukowy po)

Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

(tekst streszczenia w jęz. angielskim - max. 10 wierszy)