POLITECHNIKA KRAKOWSKA im. T. Kościuszki
Wydział Mechaniczny
Katedra Inżynierii i Automatyzacji Produkcji

Kierunek studiów: Automatyka i Robotyka
Specjalność: Automatyzacja Systemów Wytwarzania

STUDIA STACJONARNE

PRACA DYPLOMOWA

INŻYNIERSKA

Sebastian Gumula

Zastosowanie algorytmów uczenia maszynowego do optymalizacji procesów produkcyjnych

Application of machine learning algorithms for the optimization of production processes

Promotor:
dr inż. Piotr Lipiec

Kraków, rok akad. 2021/2022

OŚWIADCZENIE O SAMODZIELNYM WYKONANIU PRACY DYPLOMOWEJ

Oświadczam, że przedkładana przeze mnie praca dyplomowa inżynierska została napisana przeze mnie samodzielnie. Jednocześnie oświadczam, że ww. praca:

1) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2019 r. poz. 1231, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am* w sposób niedozwolony,
2) nie była wcześniej podstawą żadnej innej procedury związanej z nadawaniem tytułów zawodowych, stopni lub tytułów naukowych.

Jednocześnie wyrażam zgodę na:

1) poddanie mojej pracy kontroli za pomocą systemu antyplagiatowego oraz na umieszczenie tekstu pracy w bazie danych uczelni, w celu ochrony go przed nieuprawnionym wykorzystaniem. Oświadczam, że zostałem/am* poinformowany/a* i wyrażam zgodę, by system antyplagiatowy porównywał tekst mojej pracy z tekstem innych prac znajdujących się w bazie danych uczelni, z tekstami dostępnymi w zasobach światowego Internetu oraz z bazą porównawczą systemu antyplagiatowego,
2) to, aby moja praca pozostała w bazie danych uczelni przez okres, który uczelnia uzna za stosowny. Oświadczam, że zostałem poinformowany i wyrażam zgodę, że tekst mojej pracy stanie się elementem porównawczej bazy danych uczelni, która będzie wykorzystywana, w tym także udostępniana innym podmiotom, na zasadach określonych przez uczelnię, w celu dokonywania kontroli antyplagiatowej prac dyplomowych/doktorskich, a także innych tekstów, które powstaną w przyszłości.

Sebastian Gumula
.....
podpis

1) Wyrażam zgodę na udostępnianie mojej pracy dyplomowej w Akademickim Systemie Archiwizacji Prac do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych (Dz.U. z 2019 r. poz. 1231, z późn. zm.).

TAK/~~NIE~~*

Sebastian Gumula
.....
podpis

Jednocześnie przyjmuję do wiadomości, że w przypadku stwierdzenia popełnienia przeze mnie czynu polegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzej pracy, lub ustalenia naukowego, Rektor PK stwierdzi nieważność postępowania w sprawie nadania mi tytułu zawodowego (art. 77 ust. 5 ustawy z dnia 18 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce, (Dz.U. z 2020 r., poz. 85, z późn. zm.).

Sebastian Gumula
.....
podpis

*) – niepotrzebne skreślić

SPIS TREŚCI

WYKAZ OZNACZEŃ

LSTM - pamięć długotrwała-krótkotrwała (*ang. long-short term memory*)

ARIMA – autoregresyjny zintegrowany model średniej ruchomej (*ang. Autoregressive Integrated Moving Average)*

ADF – rozszerzony test Dickeya-Fullera *(ang. Augmented Dickey-Fuller)*

KPSS - test Kwiatkowski-Phillips-Schmidt-Shin

PP - test Phillipsa-Perrona

ACF – funkcja autokorelacji *(ang. Autocorrelation Function)*

PACF – funkcja autokorelacji cząstkowej *(ang. Partial Autocorrelation)*

ReLU – Rektyfikowana jednostka liniowa *(ang. Rectified Linear Unit)*

1. Cel i zakres pracy

Celem poniższej pracy jest zastosowanie algorytmów uczenia maszynowego, do stworzenia modeli prognostycznych dla danych produkcyjnych, które można przedstawić w postaci jednowymiarowych szeregów czasowych. Przykładami takich danych są:

- Zapotrzebowanie na dany produkt
- Ilość produkowanych komponentów
- Ilość sprzedanych towarów

Zadaniem modeli jest stworzenie dobrej jakości prognoz na podstawie danych historycznych, które umożliwią minimalizację kosztów ponoszonych przez zakład produkcyjny poprzez lepsze zrozumienie zjawisk zachodzących w badanych procesach, takich jak trend, sezonowość, autokorelacja.

Niniejsza praca dyplomowa składa się z części teoretycznej, objaśniającej proces analizy jednowymiarowych szeregów czasowych i dopasowania ich do modeli prognostycznych na przykładzie modeli ARIMA oraz LSTM.

Część praktyczna zawiera implementację modeli dla wybranych jednowymiarowych szeregów czasowych, porównanie i interpretację wyników prognoz modeli ARIMA, oraz LSTM.

Ostatnia część zawiera wnioski na temat stosowania autoregresyjnych oraz rekurencyjnych modeli prognostycznych do p procesów produkcyjnych.

W pracy poruszone zostały tematy z zakresu:

- Analizy szeregów czasowych – wizualizacja danych, dekompozycja szeregu na elementy składowe;
- Prognozowania modelem ARIMA – model AR, model MA, proces tworzenia modelu prognostycznego w oparciu o metodykę Boxa-Jenkinsa, uzyskiwanie stacjonarności danych przez różnicowanie, weryfikacja stacjonarności za pomocą testu ADF, wykorzystanie narzędzia *auto_arima* do wyznaczenia hiperparametrów ARIMA(p,d,q) poprzez minimalizację kryterium Akaikego i Bayesowskiego kryterium informacyjnego Schwarza, walidacja krzyżowa danych testowych i prognoz, akceptacja prognozy w oparciu o błąd RMSE i MAPE;
- Prognozowania modelem LSTM – budowa i działanie modelu LSTM, funkcje aktywacyjne (sigmoid, tanh), transformacja danych szeregu czasowego do problemu uczenia nadzorowanego, skalowanie danych, dobór hiperparametrów (neuron, epoka, rozmiar partii), walidacja krzyżowa danych testowych i prognoz, akceptacja prognozy w oparciu o błąd RMSE i MAPE;

2. Wstęp

Wraz z rozwojem technologicznym spowodowanym czwartą rewolucją przemysłową, coraz większa ilość procesów produkcyjnych zachodzących w przemyśle jest automatyzowana. Tworzenie wydajnych komponentów elektronicznych, takich jak nowoczesne karty graficzne i procesory umożliwia trenowanie zaawansowanych modeli sztucznej inteligencji, które znacznie przewyższają swoją sprawnością człowieka w wielu aspektach.

Większość decyzji biznesowych podejmowanych jest w przedsiębiorstwach na podstawie danych, które uzyskiwane są przez analityków z różnych źródeł, takich jak bazy danych, kostki OLAP czy hurtownie danych. Równolegle do rozwoju przedsiębiorstwa zwiększa się ilość dostępnych źródeł informacyjnych, z których dane mogą zostać pozyskiwane, przez co analizy procesów zachodzących w przedsiębiorstwie mogą być narażone na rosnące niedoszacowanie. Dodatkowo w przypadku braku automatyzacji procesu pozyskiwania danych, wiele z nich dostarczanych jest do źródeł w sposób manualny, co dodatkowo obniża ich jakość.

Wiele procesów produkcyjnych zachodzi w fabrykach w sposób cykliczny. Każdego miesiąca produkowana jest określona ilość wyrobów, zgodnie z planem produkcji. Każdy wyprodukowany komponent przechodzi również przez proces testowania, którego rezultaty uzależnione są od jakości wyprodukowanego wyrobu. Wraz z upływem czasu zmienia się również popyt na niektóre produkty, lub cena określonych usług jak i surowców niezbędnych do wytworzenia danego wyrobu. Oznacza to, że wiele procesów produkcyjnych można opisać w postaci procesu stochastycznego o stałym okresie czasowym, na przykład dziennym, miesięcznym lub rocznym.

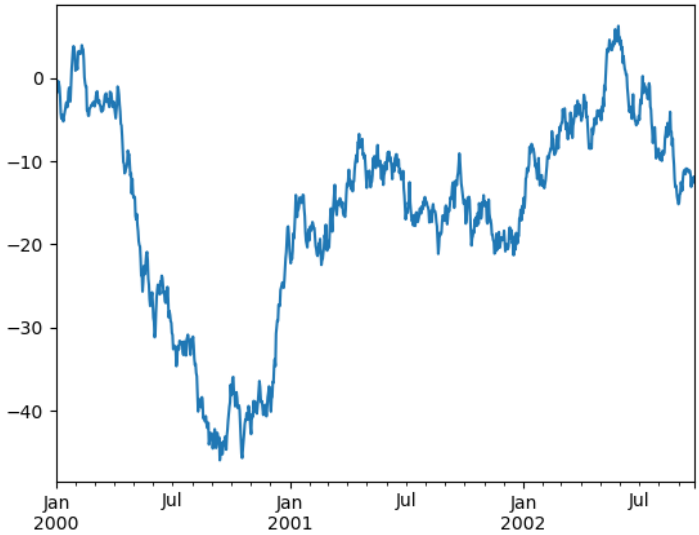
Za pomocą metod statystycznych i algorytmów uczenia maszynowego możliwe jest stworzenie modeli prognostycznych, które będą w stanie określić w oparciu o dane historyczne przyszłą wartość cechy, która wystąpi w procesie stochastycznym. Wykorzystywane są w tym celu między innymi modele autoregresyjne ze średnią kroczącą oraz rekurencyjne sieci neuronowe. Stworzenie modelu prognostycznego o wysokiej wydajności pozwoli przedsiębiorstwu na uzyskanie wiarygodnego wskaźnika, w oparciu o który będzie w stanie podejmować lepsze decyzje biznesowe. Przewidywanie popytu na produkowane wyroby w różnym okresie pozwoli na zwiększenie możliwości produkcyjnych przedsiębiorstwa a także znacznie ograniczy koszty magazynowania nadprodukcji, dzięki czemu będzie mogło zyskać ono przewagę, poprzez lepsze dopasowanie do aktualnych potrzeb klienta. Na podstawie prognoz, przedsiębiorstwo będzie mogło również oszacować roczną ilość awarii maszyn, sprzętu lub nieprawidłowo wyprodukowanych wyrobów, o ile badane zjawisko nie jest w pełni losowe, co uniemożliwiłoby jego przewidywanie.

3. Analiza jednowymiarowych szeregów czasowych

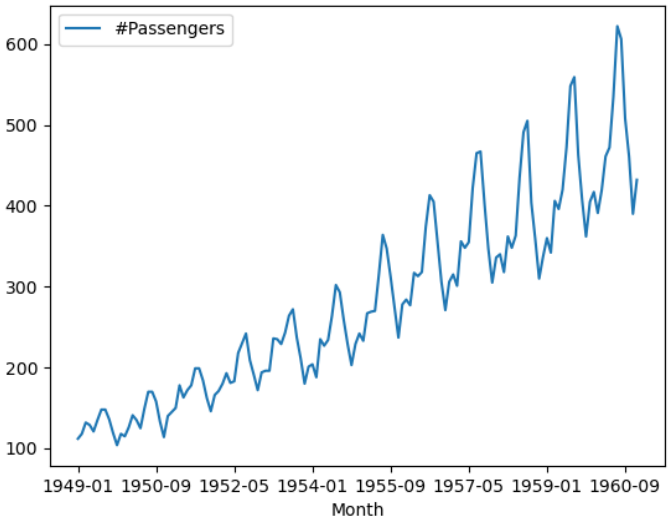
3.1. Dekompozycja szeregu czasowego

W ogólnym rozumieniu, jednowymiarowym szeregiem czasowym nazywamy uporządkowany zbiór danych, którym w określonym odstępie czasowym przypisano pewną cechę, na przykład wartość liczbową. Przykładami takich szeregów mogą być zatem: średnia miesięczna ilość pasażerów linii lotniczych, ilość urodzeń w danej populacji lub cena ropy naftowej.

Przed przystąpieniem do tworzenia modelu predykcyjnego, szereg czasowy należy poddać analizie. Celem analizy szeregów czasowych jest wyjaśnienie zjawisk w nich występujących, co jest niezbędne do zrozumienia badanego przez nas zjawiska a tym samym do dobrania odpowiednich parametrów modelu predykcyjnego.



Rys. 3.1. Jednowymiarowy szereg czasowy (dane wygenerowane losowo)



Rys. 3.2. Szereg czasowy średniej miesięcznej ilości pasażerów linii lotniczych w latach 1949-1960

Jason Brownlee [1] definiuje cztery składniki szeregów czasowych, które można wyodrębnić w trakcie analizy:

- trend – zmiana cechy, która może zachodzić w obu kierunkach (rosnący, malejący)
- sezonowość – zmiana wartości cechy, wywoływana przez występowanie krótkoterminowych cykli w szeregu czasowym
- poziom (ang. *level*) – wartość średnia cechy w całym szeregu czasowym
- szum (ang. *noise*) – zmienna losowa występująca w szeregu czasowym

Zadaniem dekompozycji szeregu czasowego jest wyodrębnienie zjawisk, takich jak trend, sezonowość i cykliczność oraz odseparowanie ich od szumów.

Jason Brownlee w swoim artykule dotyczącym metod dekompozycji szeregów czasowych wyróżnia jej dwa podstawowe modele: model addytywny oraz multiplikatywny. Zakładają one zależności matematyczne, jakie występują między poszczególnymi elementami składowymi.

Dla przykładowego jednowymiarowego szeregu czasowego $y(t)$ dla modelu addytywnego (3.1) oraz multiplikatywnego (3.2) zakładane są następujące zależności:

$$y(t) = L(t) + T(t) + S(t) + N(t) \tag{3.1}$$

$$y(t) = L(t) * T(t) * S(t) * N(t) \tag{3.2}$$

gdzie:

$L(t)$ – składowa poziomu
$T(t)$ – składowa trendu
$S(t)$ – składowa wahań sezonowych
$N(t)$ – składowa szumu

W modelu addytywnym zachodzące w szeregu czasowym zmiany są liniowe, oraz posiadają stały przyrost. Składowa trendu przyjmuje postać linii prostej, natomiast składowa wahań sezonowych ma stałą częstotliwość oraz amplitudę w całym okresie szeregu.

W modelu multiplikatywnym zmiany są nieliniowe. Mogą się one zwiększać lub zmniejszać, wraz z upływem czasu. Składowa trendu przyjmuje postać krzywej, natomiast częstotliwość i amplituda składowej wahań sezonowych wyraźnie rośnie lub maleje w czasie.

Zdaniem doktora, dekompozycja powinna być traktowana jako narzędzie do oceny złożoności problemu przewidywania danego szeregu czasowego. Uważa on, że szeregi czasowe rzeczywistych danych są zbyt skomplikowane, by jednoznacznie określić je danym modelem matematycznym. W obrębie jednego szeregu czasowego mogą wystąpić zjawiska zarówno liniowe jak i nieliniowe, dlatego dekompozycja powinna posłużyć do wstępnego ich oszacowania.

3.2. Dekompozycja szeregu czasowego na przykładzie metody klasycznej

Podręcznik [2] autorstwa Rob'a J Hyndmana i George'a Athanasopoulosa datuje powstanie klasycznej metody dekompozycji na lata 20. XX wieku. Autorzy podkreślają, że metoda ta stanowi podstawę wielu metod dekompozycji stosowanych w dzisiejszych czasach.

Pierwszym krokiem do przeprowadzenia klasycznej dekompozycji jest użycie metody średniej ruchomej (*MA – moving average*) do oszacowania składowej trend-cykl (ang. *trend-cycle*). Średnią ruchomą rzędu m-tego opisuje następująca zależność:

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}, \tag{3.3}$$

gdzie m = 2*k + 1.

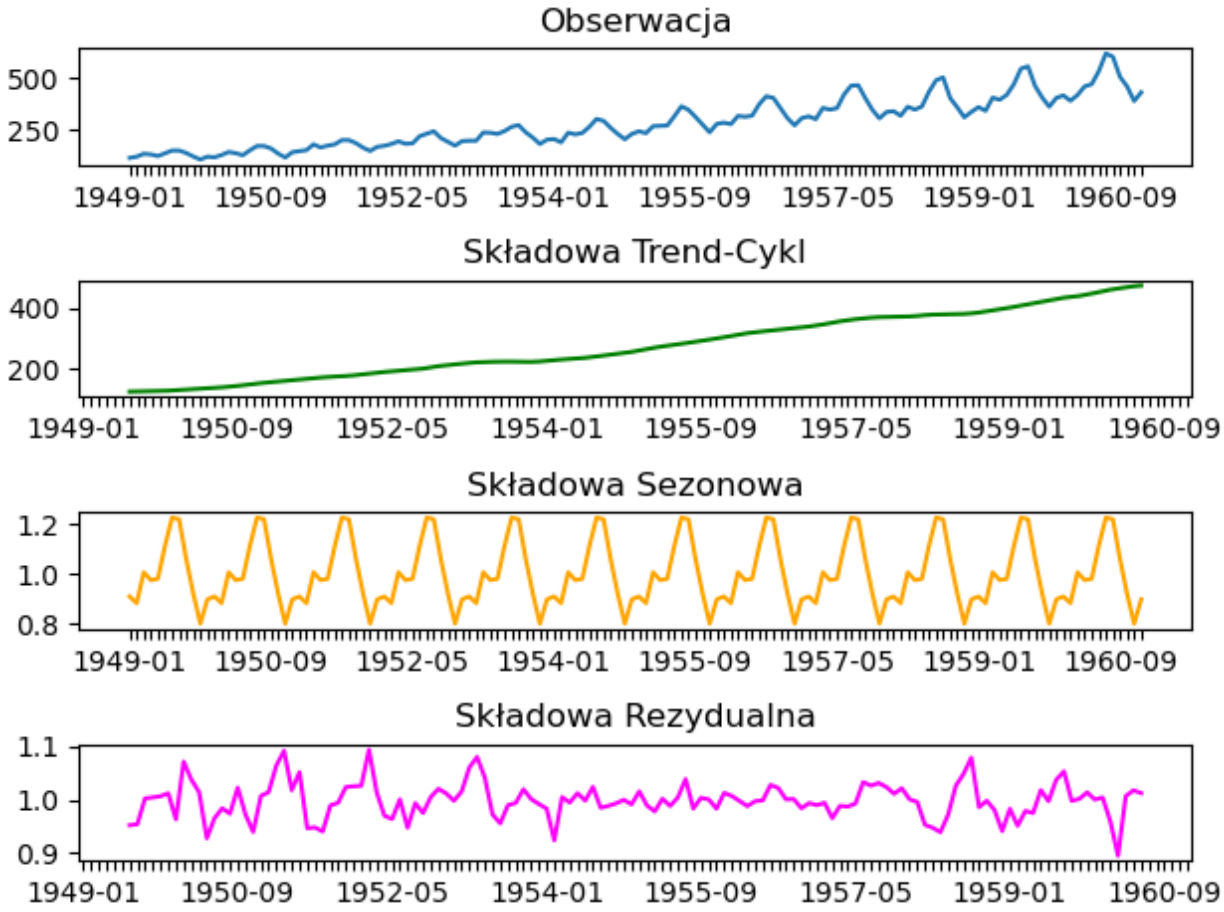
Oznacza to, że estymacja składowej trend-cykl w czasie *t* jest otrzymywana poprzez uśrednianie wartości szeregu czasowego w obrębie *k*-okresów w czasie *t*. Liczenie średniej pozwala na częściową eliminację losowości w szeregu czasowym, pozostawiając wygładzoną składową trend-cykl (*m-MA).*

Drugim krokiem jest wyznaczenie wartości okresu m dla badanego szeregu czasowego (dla kwartałów: m = 4, miesięcy: m = 12, dni: m = 7). Algorytm postępowania dla metody addytywnej jest następujący:

- Jeżeli *m* jest liczbą parzystą, oblicz składową trend-cykl T_t, przy użyciu 2**m-Ma*, jeżeli zaś nieparzystą, oblicz składową przy użyciu *m-MA.*
- Oblicz szereg zdetrendowany $y_t - T_t$ (odejmij składową trend-cykl od y_t)
- Do wyznaczenia składowej sezonowości dla każdego sezonu, uśrednij wartości szeregu zdetrendowanego dla tego sezonu (dla danych miesięcznych, składowa sezonowa dla marca jest średnią z wszystkich zdetrendowanych wartości dla marca w całym szeregu czasowym). Ostateczna wartość składowej S_t jest otrzymywana po przeprowadzeniu tego procesu dla każdego roku.
- Wartość szumu obliczana jest poprzez odjęcie składowych sezonowości i trend-cykl od szeregu *y*

Algorytm postępowania dla metody multiplikatywnej:

- Jeżeli *m* jest liczbą parzystą, oblicz składową trend-cykl T_t , przy użyciu 2**m-Ma,* jeżeli zaś nieparzystą, oblicz składową przy użyciu *m-MA.*
- Oblicz szereg zdetrendowany y_t / T_t (podziel y_t przez wartość składowej trend-cykl)
- Do wyznaczenia składowej sezonowości dla każdego sezonu, uśrednij wartości szeregu zdetrendowanego dla tego sezonu.
- Wartość szumu obliczana jest przez wydzielenie szeregu y_t przez iloczyn składowych sezonowości i trend-cykl



Rys. 3.2.1. Dekompozycja szeregu czasowego, model multiplikatywny
Źródło: opracowanie własne, na podstawie danych:
https://www.kaggle.com/rakannimer/air-passengers

Rysunek 3.2.1. przedstawia wynik dekompozycji szeregu czasowego wykonanej w oparciu o bibliotekę języka Python do modelowania statystycznego (statsmodels). Zaimplementowana w niej funkcja seasonal_decompose() pozwala na dobranie modelu matematycznego do uprzednio uporządkowanego zestawu danych (na przykład w postaci pliku .csv, gdzie każdy wiersz oznacza kolejny miesiąc oraz przypisaną do niego wartość liczbową).

Wstępna obserwacja szeregu czasowego (Rys. 3.2.) pozwala na wychwycenie widocznych różnic w średniej ilości pasażerów oraz sezonowości wraz z upływem lat. Zmiany te mają charakter nieliniowy w całym badanym okresie, dlatego przyjęty został model multiplikatywny. Jako, że zbiór danych dotyczy średnich miesięcznych wartości, dobrany został parametr m = 12.

W dokumentacji biblioteki statsmodels [3] znajduje się informacja, że klasyczna dekompozycja Census I jest metodą naiwną, dlatego sugerowane jest korzystanie z bardziej złożonych metod. W podręczniku [2] opisane zostały następujące wady metody naiwnej:

- Estymacja składowej trend-cykl jest pomijana w przypadku początkowych i końcowych obserwacji. Przykładowo dla parametru m = 12 nie można określić estymacji składowej dla pierwszych i ostatnich 6 obserwacji. Przekłada się to na brak możliwości dokładnego wyznaczenia składowej rezydualnej w tym okresie.
- Estymacja składowej trend-cykl ma tendencję do nadmiernego wygładzania gwałtownych zmian (wzrostów i spadków) w szeregu czasowym.
- Klasyczna metoda dekompozycji zakłada występowanie składowej sezonowej co rok. Dla wielu szeregów, jest to poprawne założenie (np. w przypadku sezonowości cen owoców). Nie sprawdza się ono natomiast w przypadku dłuższych szeregów, gdzie dodatkowy wpływ mają czynniki długookresowe (np. wzrost cen energii spowodowany rozwojem technologicznym oraz sytuacją polityczną danego kraju).
- Metoda Census I nie uwzględnia również odchyleń będących wynikiem zdarzeń losowych (np. wpływ lockdownu na zmianę ilości pasażerów linii lotniczych).

Powyższe wady wyeliminowane zostały w bardziej nowoczesnych metodach dekompozycji. Przykładowo, metoda Census II eliminuje błędy estymacji składowej trend-cykl dla wartości początkowych i końcowych, oraz uwzględnia dodatkowe czynniki sezonowe takie jak zmienność dnia handlowego czy efekt świąteczny.

Dobór odpowiedniej metody dekompozycji podyktowany jest zatem typem danych oraz znajomością zjawisk w nich występujących. Poprawne przeprowadzenie dekompozycji pozwoli zatem na zrozumienie analizowanego szeregu czasowego oraz dobranie odpowiedniego modelu predykcyjnego.

4. Prognozowanie szeregów czasowych modelem ARIMA.

4.1. Model autoregresyjny AR.

Proces autoregresji (AR) polega na przewidywaniu wartości zmiennej, przy wykorzystaniu liniowej zależności, wyznaczonej na podstawie jej poprzednich wartości. Termin autoregresja oznacza zatem zastosowanie statystycznej metody regresji liniowej, w której zmienną prognozowaną jest przewidywana przyszła wartość szeregu czasowego w oparciu o regresor, który stanowią poprzednie wartości zmiennej. Model matematyczny procesu autoregresyjnego rzędu *p* opisany został w literaturze przedmiotu [2] oraz ma następującą postać:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \quad (4.1)$$

gdzie:

c, ϕ - współczynniki modelu autoregresyjnego

y_i - wartości szeregu czasowego

ε_t - wartość składowej błędu (biały szum)

W modelu autoregresyjnym pierwszego rzędu AR(1), w zależności od wartości współczynników funkcji y_t, jest ona określana jako:

- biały szum, (gdy $\phi_1 = 0$)
- błądzenie losowe, (gdy $\phi_1 = 1$ oraz c = 0)
- błądzenie losowe z dryftem, (gdy $\phi_1 = 1$ oraz c =/= 0)

Do poprawnego działania modelu autoregresyjnego wymagana jest również jego stacjonarność, przez co parametry ϕ podlegają następującym ograniczeniom:

- dla modelu AR(1):
  $-1 < \phi_1 < 1$

- dla modelu AR(2):
  $-1 < \phi_2 < 1$,
  $\phi_1 + \phi_2 < 1$,
  $\phi_2 + \phi_1 < 1$.

Literatura nie podaje ograniczeń dla modeli AR wyższych rzędów, ze względu na rosnące skomplikowanie obliczeń wraz z kolejnymi rzędami. Do ich wyznaczenia wykorzystuje się metody matematyczne, np. równania Yule – Walkera [6].

4.2. Model średniej kroczącej MA.

Zasada działania modelu średniej kroczącej jest bardzo podobna do procesu AR. Główną różnicą jest przewidywanie wartości błędu w procesie przypominającym regresję. Biały szum jest procesem losowym, dlatego nie można mówić o obserwacji jego wartości. Równanie matematyczne opisujące model średniej kroczącej rzędu q wygląda następująco:

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_1\varepsilon_{t-1} + \ldots + \theta_q\varepsilon_{t-q} \tag{4.2}$$

gdzie:

c, θ - współczynniki modelu autoregresyjnego

ε_t - wartość składowej błędu (biały szum)

Autor [2] zwraca również uwagę na różnicę między modelem średniej kroczącej MA(q) a procesem wygładzania, wykorzystywanym do estymacji składowej trend-cykl w procesie dekompozycji.

Wspomina on również, że między procesami AR oraz MA występuje dualizm, dzięki czemu każdy stacjonarny model AR(p) można zapisać w postaci MA(∞). Aby przekształcenie procesu MA(q) do postaci AR(∞) było możliwe, wymagane są następujące kryteria odwracalności:

- dla modelu MA(1):
  $-1 < \theta_1 < 1$
- dla modelu MA(2):
  $-1 < \theta_2 < 1$,
  $\theta_1 - \theta_2 < 1$,
  $\theta_2 + \theta_1 > -1$,

Podobnie jak w przypadku modeli AR wyższych rzędów, do obliczenia kryteriów wykorzystywane są metody matematyczne, zaimplementowane w programach do obliczeń statystycznych lub na przykład bibliotece języka R. Przykładem takiej metody może być spełnienie warunku wynikającego z definicji odwracalności [7].

4.3. Prognozowanie w oparciu o model ARIMA

Przykładem popularnej i efektywnej metody predykcyjnej jest prognozowanie w oparciu o autoregresyjny zintegrowany model średniej ruchomej (ang. *autoregressive integrated moving average; ARIMA*). Model ten składa się z trzech procesów:

- Autoregresji (AR) – pozwalającej na uzyskanie zależności między danymi historycznymi do przewidywania wartości poprzez wykorzystanie regresji liniowej.
- Integracji (I) – wykorzystywanej w celu osiągnięcia stacjonarności szeregu czasowego.
- Średniej ruchomej (MA) – wykorzystującej zależność między wartością zaobserwowaną a błędem modelu średniej ruchomej.

W notacji modelu ARIMA, każdy z powyższych procesów zdefiniowany jest kolejno jako argumenty funkcji ARIMA(p,d,q), gdzie ich wartości określane są nieujemnymi liczbami całkowitymi. Parametry te są zdefiniowane następująco:

- p - rząd autoregresji
- d - stopień integracji szeregu czasowego. Określa on ilość wystąpień procesu integracji (przekształcenia danych niestacjonarnych na stacjonarne) w danym szeregu.
- q - rząd średniej ruchomej

Do poprawnego działania modelu wymagana jest zatem prawidłowa konfiguracja każdego z parametrów. Wykorzystywana jest w tym celu metodologia, opisana przez George'a Boxa i Gwylima Jenkinsa w książce: „*Time Series Analysis: Forecasting and Control"* [4]. Została opisana w niej metodyka wyznaczania parametrów dla jednowymiarowych modeli autoregresyjnych, która składa się z 3 kroków:

- Identyfikacji
- Estymacji
- Diagnostyki i prognozowania

Metoda Boxa-Jenkinsa pozwala zatem na dopasowanie modelu predykcyjnego dla dowolnego szeregu czasowego, pod warunkiem że jest on szeregiem stacjonarnym oraz nie jest on białym szumem. Jest to proces losowy o wartości średniej równej zeru, dlatego jego prognozowanie jest zbędne.

Poniżej przedstawiony został proces dopasowania modelu, zgodnie z metodą Boxa-Jenkinsa dla przykładowego szeregu czasowego miesięcznej sprzedaży szamponu na przestrzeni 3 lat, pochodzący z pakietu *tsdl* języka R.

Rys. 4.4.1. Szereg czasowy miesięcznej sprzedaży szamponu na przestrzeni 3 lat.
Źródło: opracowanie własne, na podstawie danych
https://pkg.yangzhuoranyang.com/tsdl/ [9]

4.4. Identyfikacja

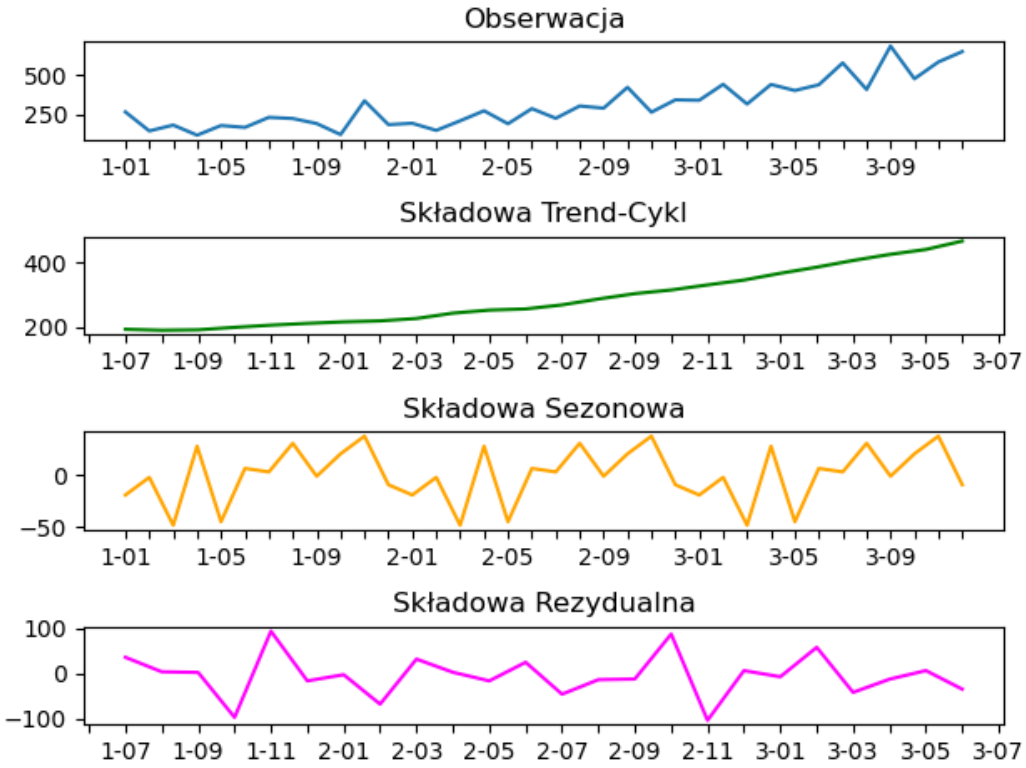
Pierwszym krokiem procesu identyfikacji jest wykluczenie, że badany szereg czasowy jest białym szumem. Cechą charakteryzującą biały szum jest zerowa wartość średniej oraz stała wariancja.

Kolejnym krokiem jest przeanalizowanie szeregu czasowego pod kątem stacjonarności. Stacjonarność jednowymiarowego szeregu czasowego oznacza sytuację, w której wartość zmiennej nie jest zależna od upływu czasu. Szeregiem niestacjonarnym jest zatem szereg posiadający cechy takie jak trend czy sezonowość, które wpływają na wartość zmiennej w różnych odstępach czasu.

W niektórych przypadkach stacjonarnych szeregów mogą występować natomiast zjawiska cykliczne, pozbawione trendu oraz sezonowości. Cykle te nie są jednolicie odległe od siebie, dlatego ich występowanie jest uniezależnione od upływu czasu. [2]

Stacjonarność szeregu czasowego można osiągnąć poprzez jego różnicowanie do momentu, aż nie będzie on wykazywał znamion niestacjonarności, tzn. jego wariancja, autokorelacja i średnia będą stałe w czasie [5]. Proces różnicowania polega na odejmowaniu wybranej wartości zmiennej szeregu czasowego y_t od wartości poprzedzającej y_{t-1} dla wszystkich wartości. Do uzyskania stacjonarności szeregu można wykorzystać również transformacje funkcjonalne takie jak logarytm naturalny lub pierwiastek kwadratowy, wykorzystywane do stabilizacji wariancji [8] .

Istnieją również formalne testy statystyczne, które ułatwiają dobór parametrów modelu predykcyjnego. Przykładami takich testów są: testy Dickeya-Fullera (DF/ADF), test Phillipsa-Perrona (PP), test Kwiatkowski-Phillips-Schmidt-Shin (KPSS).



Rys. 4.4.2. Dekompozycja szeregu czasowego
Źródło: opracowanie własne, na podstawie danych
https://pkg.yangzhuoranyang.com/tsdl [9]

Rys. 4.4.3. Różnicowanie szeregu czasowego
Źródło: opracowanie własne, na podstawie danych
https://pkg.yangzhuoranyang.com/tsdl [9]

Aby mieć pewność, że zróżnicowany szereg czasowy jest stacjonarny, został on poddany rozszerzonemu testowi Dickeya-Fullera. Polega on na badaniu hipotezy zerowej, według której każdy niestacjonarny proces zawiera pierwiastek jednostkowy, leżący w obrębie jego koła jednostkowego. W przypadku jego braku rozważana jest hipoteza alternatywna, według której wartość testu statystycznego szeregu stacjonarnego nie przekracza wartości krytycznych, znajdujących się w tablicach statystycznych testu ADF [10].

Wyniki testu Dickeya-Fullera dla jednokrotnie zróżnicowanego szeregu były następujące:

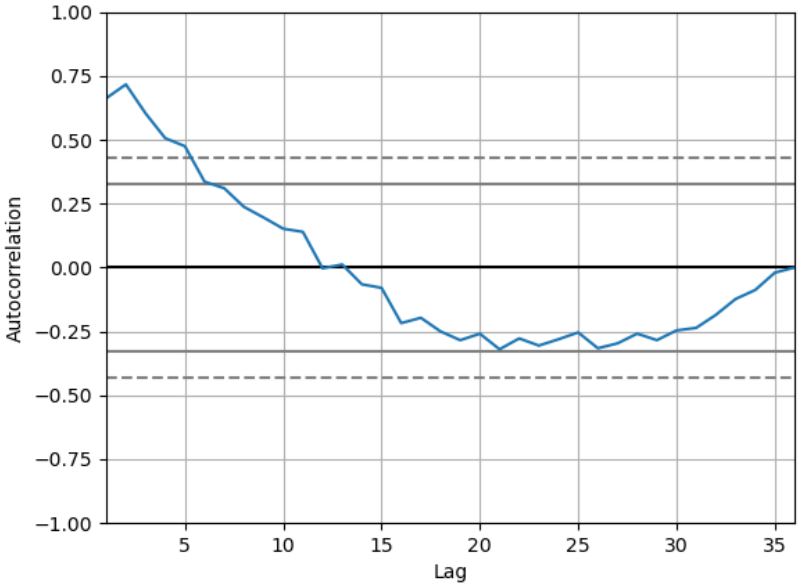
- Statystyka ADF: -7,249074055553854
- p-wartość: 1,7998574141687034e-10

Wartości krytyczne:

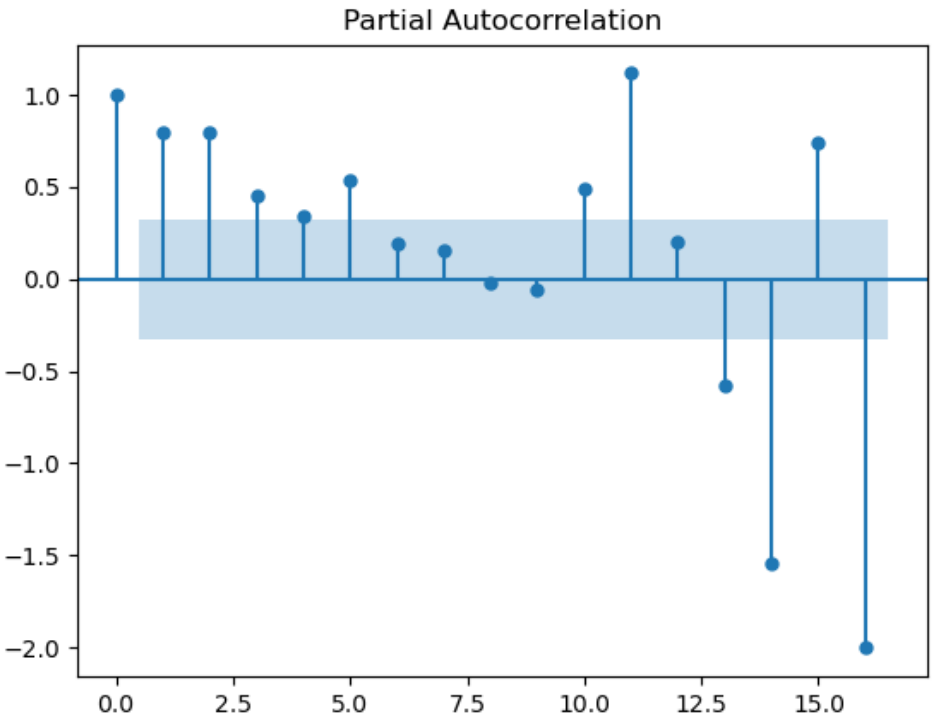
- dla rozkładu 1%: -3,6461350877925254
- dla rozkładu 5%: -2,954126991123355
- dla rozkładu 10%: -2,6159676124885216

Uzyskana wartość statystyki ADF oraz niski współczynnik p-wartości ($p << 0.05$) wskazują na stacjonarność szeregu. Wartość statystyki ADF mniejsza od wartości krytycznej 1% pozwala na odrzucenie hipotezy zerowej na rzecz hipotezy alternatywnej, co udowadnia jego stacjonarność.

Po uzyskaniu parametru $d = 1$, kolejnym krokiem procesu identyfikacji jest oszacowanie parametrów p oraz q dla badanego szeregu czasowego. Wykorzystywane są w tym celu wykresy funkcji autokorelacji oraz autokorelacji cząstkowej. Pozwalają one określić, w jakim stopniu dane historyczne skorelowane są ze sobą.



Rys. 4.4.4. Wykres ACF
Źródło: opracowanie własne, na podstawie danych https://pkg.yangzhuoranyang.com/tsdl [9]



Rys. 4.4.5. Wykres PACF
Źródło: opracowanie własne, na podstawie danych
https://pkg.yangzhuoranyang.com/tsdl [9]

Na podstawie kształtu wykresów ACF oraz PACF można oszacować z jakiego typu procesem mamy do czynienia. Istotne z punktu widzenia analizy są jedynie punkty, które przekraczają wartość graniczną (obszar zaznaczony na niebiesko). Dla poszczególnych procesów wyróżnia się następujące kształty charakterystyk:

Dla procesu AR:

- kształt charakterystyki ACF jest wykładniczy oraz maleje wraz ze wzrostem opóźnień (*ang. „lag”*)
- punkty charakterystyki PACF znajdują się powyżej wartości granicznej do momentu *p*

Dla procesu MA:

- kształt charakterystyki PACF jest wykładniczy oraz maleje wraz ze wzrostem opóźnień
- punkty charakterystyki ACF znajdują się powyżej wartości granicznej do momentu *q*

Dla procesu ARMA:

- kształty charakterystyk ACF i PACF są wykładnicze oraz maleją wraz ze wzrostem opóźnień.

Analiza powyższych charakterystyk pozwala zauważyć, że dla wartości opóźnienia (Lag = 5) punkty charakterystyk znajdują się powyżej linii granicznej. Co więcej, charakterystyka ACF ma kształt wykładniczy, dlatego można założyć, że badany model jest procesem autoregresyjnym rzędu wskazanego przez graniczną wartość opóźnienia.

Proces identyfikacji pozwolił zatem określić szereg czasowy jako proces autoregresyjny rzędu p = 5. Do uzyskania jego stacjonarności wymagane było przeprowadzenie jednego procesu różnicowania (d = 1). Zgodnie z notacją, powyższy model został zapisany jako proces ARIMA(5,1,0).

4.5. Estymacja

Po zidentyfikowaniu modelu jako procesu ARIMA(5,1,0), zgodnie z metodyką Boxa - Jenkinsa należy przystąpić do estymacji jego parametrów. Wykorzystuje się w tym celu wskaźniki dopasowania modelu, na przykład kryterium informacyjne Akaikego (AIC) lub Bayesowskie kryterium informacyjne Schwarza (BIC). Ich minimalizacja pozwala na dobór parametrów p oraz q dla badanego modelu.

SARIMAX Results

Dep. Variable:	y	No. Observations:	23			
Model:	ARIMA(5, 1, 0)	Log Likelihood	-122.677			
Date:	Sat, 15 Jan 2022	AIC	257.354			
Time:	19:19:06	BIC	263.900			
Sample:	0	HQIC	258.896			
	- 23					
Covariance Type:	opg					

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8515	0.284	-2.999	0.003	-1.408	-0.295
ar.L2	-0.4084	0.334	-1.225	0.221	-1.062	0.245
ar.L3	-0.4066	0.417	-0.975	0.330	-1.224	0.411
ar.L4	-0.0210	0.506	-0.041	0.967	-1.012	0.970
ar.L5	0.0843	0.394	0.214	0.830	-0.687	0.856
sigma2	3876.3771	1641.524	2.361	0.018	659.049	7093.705

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	1.27
Prob(Q):	0.98	Prob(JB):	0.53
Heteroskedasticity (H):	1.59	Skew:	0.52
Prob(H) (two-sided):	0.55	Kurtosis:	2.47

Rys. 4.5.1: Statystyki dopasowania modelu ARIMA(5,1,0)

Do znalezienia wartości optymalnej parametrów *p,d,q* można wykorzystać narzędzie *auto_arima* [11]. Aby to zrobić, należy dokonać podziału szeregu czasowego na dane testowe oraz treningowe. Dla badanego szeregu przyjęto 66% początkowych danych jako treningowe, natomiast pozostałe posłużą do testowania modelu. Narzędzie *auto_arima* pozwala na dopasowanie wartości p,q poprzez analizę kryteriów AIC, BIC. Pozwala również na dobór parametru *d*, m.in. w oparciu o testy Dickeya-Fullera. Optymalizacja parametrów wymaga podania zakresu dla parametrów *p,d,q* w obrębie których model będzie dostrajany.

Zakres poszukiwań powinien być skończony, aby uniknąć zjawiska nadmiernego dopasowania. Przetrenowany model będzie wskazywał wówczas wartości historyczne, zamiast je przewidywać. Za wartość graniczną współczynników przyjęto wartość *p*, wykazaną w procesie identyfikacji.

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=264.590, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=254.260, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.08 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=262.590, Time=0.01 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=255.774, Time=0.07 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.11 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.11 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=252.652, Time=0.02 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=254.364, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=253.446, Time=0.04 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=253.127, Time=0.02 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=255.247, Time=0.05 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 0.615 seconds
```

Rys. 4.5.2 Poszukiwanie optymalnego modelu ARIMA

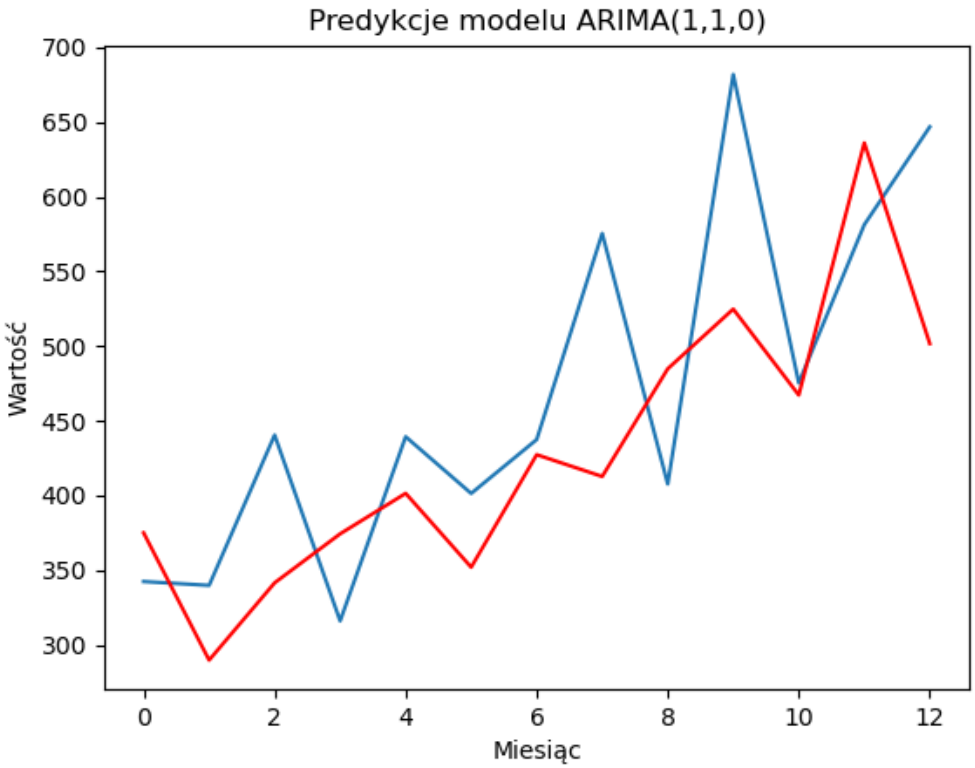
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   23
Model:               SARIMAX(1, 1, 0)   Log Likelihood                -124.326
Date:                Sat, 15 Jan 2022   AIC                           252.652
Time:                        19:21:47   BIC                           254.834
Sample:                             0   HQIC                          253.166
                                  - 23
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7029      0.199     -3.532      0.000      -1.093      -0.313
sigma2      4608.3122   1470.849      3.133      0.002    1725.501    7491.123
===================================================================================
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):                 0.81
Prob(Q):                              0.75   Prob(JB):                         0.67
Heteroskedasticity (H):               1.51   Skew:                             0.45
Prob(H) (two-sided):                  0.60   Kurtosis:                         2.71
===================================================================================
```

Rys. 4.5.3: Statystyki dopasowania modelu ARIMA(1,1,0)

Dla wartości granicznych max_p = 5, max_d = 5, max_q = 5, narzędzie wskazało model ARIMA(1,1,0) jako najlepsze dopasowanie dla danych treningowych, według kryterium minimalizacji funkcji AIC.

4.6. Diagnostyka i prognozowanie

Ostatnim krokiem jest porównanie wyników predykcji dopasowanego modelu z wyodrębnionymi danymi testowymi (walidacja krzyżowa). Poniższy wykres przedstawia porównanie wartości przewidywanych (kolor niebieski) z danymi testowymi (kolor czerwony).



Rys. 4.6.1: Prognozowanie szeregu czasowego, model ARIMA(1,1,0)

Skuteczność dopasowania modelu predykcyjnego może zostać wyznaczona poprzez obliczenie średniego procentowego błędu prognozy (MAPE) i pierwiastkowego błędu średniokwadratowego (RMSE). Dla dopasowanego modelu uzyskano następujące wartości:

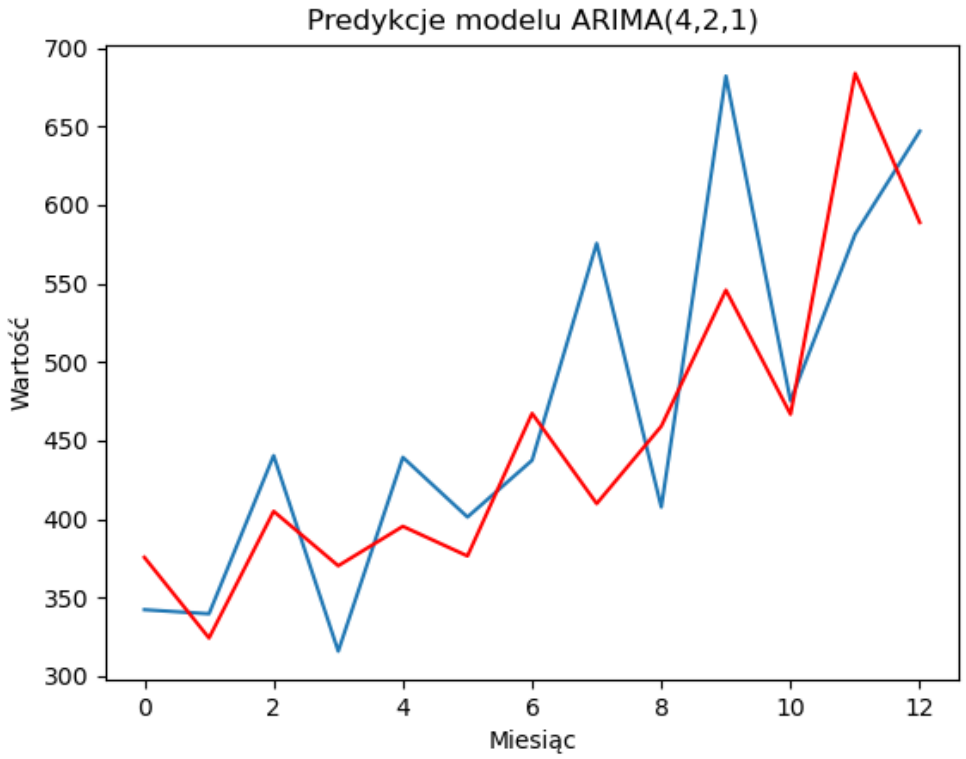
- RMSE: 88,631
- MAPE: 28,961537940877363

Analizując wyniki, model predykcyjny obarczony jest błędem (około 29%). Na poziom błędu modelu prognostycznego mogą mieć wpływ między innymi takie czynniki jak:

- Niedoszacowana wartość parametrów *p,d,q*
- Nieprawidłowy dobór danych trenujących i testowych.
- Niska jakość danych
- Błąd ludzki

Zgodnie z metodyką Boxa-Jenkinsa, w przypadu uzyskania niesatysfakcjonującego wyniku modelu predykcyjnego, należy powtórzyć procedurę strojenia modelu, w oparciu o inne parametry. W badanym przypadku, najbardziej prawdopodobną przyczyną błędu jest nieprawidłowy dobór parametrów metodą *auto_arima*. Wykorzystuje ona minimalizację tylko jednego czynnika (funkcji AIC), która w badanym przypadku nie pozwala osiągnąć dobrej prognozy.

W artykule Jasona Brownlee [13] przedstawiona została alternatywna metoda poszukiwania parametrów *p,d,q*, poprzez minimalizację średniego błędu kwadratowego (MSE). Wartość minimalna uzyskana została dla modelu ARIMA(4,2,1).



Rys. 4.6.2. Prognozowanie szeregu czasowego, model ARIMA(4,2,1)

Dla uzyskanych wartości prognoz modelu ARIMA(4,2,1) wyznaczone zostały błędy RSME oraz MAPE:

- RMSE: 74,219
- MAPE: 9,872198217540518

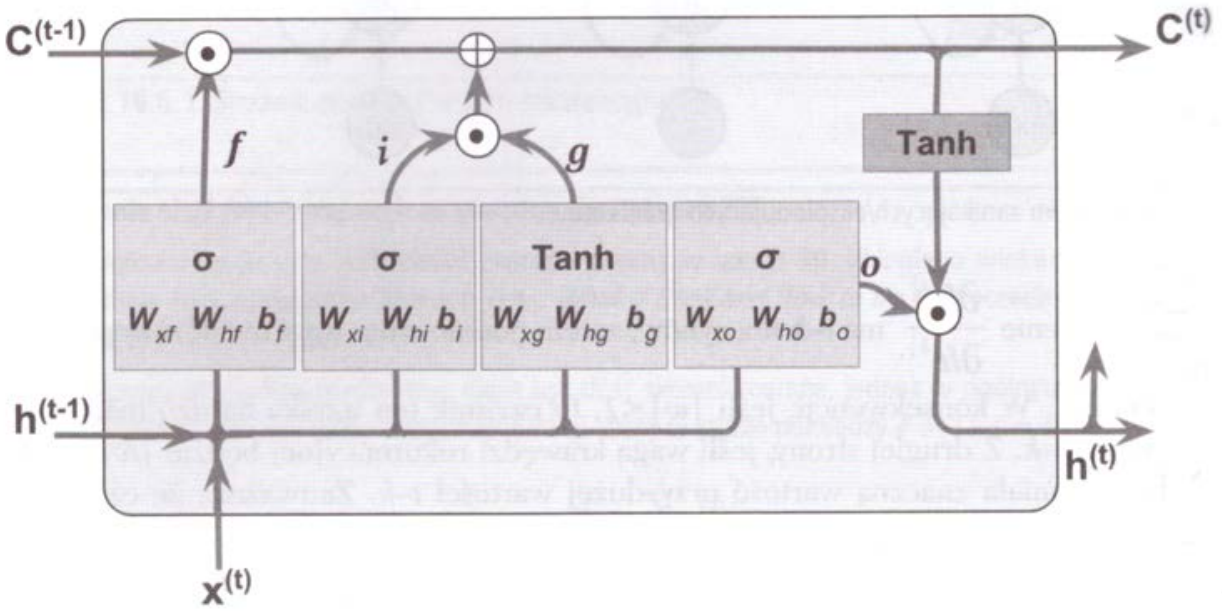
Porównując obie metody optymalizacji, dla badanego szeregu czasowego skuteczniejszy model predykcyjny osiągnięto poprzez minimalizację funkcji błędu MSE. Zastosowanie alternatywnej metody pozwoliło na ponad trzykrotne zmniejszenie błędu prognozowania. Ze względu na doświadczalny charakter strojenia modeli ARIMA za wysoce dokładny uważa się model, którego procentowy błąd prognozowania nie przekracza 10%.

5. Prognozowanie w oparciu o rekurencyjne sieci neuronowe.

5.1. Model LSTM

Przykładem alternatywnej metody prognozowania szeregów czasowych jest stworzenie modelu predykcyjnego w oparciu o rekurencyjne sieci neuronowe. Poprzez wykorzystanie sprzężenia zwrotnego oraz funkcji aktywacyjnych, rekurencyjna sieć neuronowa jest w stanie oszacować przyszłe wartości szeregu czasowego.

Przykładem takiego modelu jest rekurencyjna sieć neuronowa z blokami pamięci długotrwałej i krótkotrwałej (*ang. long-short term memory, LSTM)*. W odróżnieniu od innych rekurencyjnych sieci neuronowych, jednostka LSTM zawiera komórkę pamięci (*ang. memory cell*). Może ona przechowywać wprowadzoną do sieci informację, której wartość uwzględniana będzie w procesie prognozowania szeregu czasowego.



Rys. 4.4.5. Model LSTM
Źródło: Sebastian Raschka, Vahid Mirjalili „Python. Uczenie maszynowe. Wydanie 2.” [14]

Powyższy rysunek przedstawia schemat modelu LSTM. Można w nim wyróżnić następujące bloki:

- Pamięć aktualnej jednostki $C^{(t)}$
- Pamięć poprzedniej jednostki $C^{(t-1)}$
- Sygnał wyjściowy aktualnej jednostki ukrytej $h^{(t)}$
- Sygnał wyjściowy poprzedniej jednostki ukrytej $h^{(t-1)}$
- Sygnał wejściowy $x^{(t)}$
- Bramka wejściowa *i*
- Węzeł wejściowy *g*
- Bramka wyjściowa *o*
- Bramka zapominająca *f*
- Sigmoidalna funkcja aktywacyjna (σ)
- Funkcja aktywacyjna tangensa hiperbolicznego (tanh)

Bloki pamięci $C^{(t)}$ i $C^{(t-1)}$ wykorzystywane są do przenoszenia wartości pamięci między poszczególnymi jednostkami. Do opisu przepływu informacji wykorzystywany jest zapis Θ oraz $\oplus$, jako oznaczenie iloczynu oraz sumy odpowiadających sobie elementów [15]. Stan komórki w czasie *t* przyjmuje postać:

$$C^{(t)} = (C^{(t-1)} \Theta f_t) \oplus (i_t \Theta g_t) \tag{5.1}$$

Ze wzoru 5.1. wynika, że na wartość pamięci jednostki LSTM w czasie *t* mają wpływ iloczyn stanu komórki w czasie *t-1* oraz funkcji sigmoidalnej, pełniącej rolę bramki zapominającej. Służy ona do ustalenia, w jakim stopniu informacja zawarta w sygnałach $x^{(t)}$ oraz $h^{(t-1)}$ jest warta zapamiętania (funkcja przyjmuje wartość > 0) lub zapomnienia (funkcja przyjmuje wartość 0). Jej wartość opisuje następująca zależność:

$$f_t = \sigma(W_{xf}\, x^{(t)} + W_{hf}\, h^{(t-1)} + b_f) \tag{5.2}$$

gdzie:

W_{xf} – waga funkcji wejściowej $x^{(t)}$
W_{hf} – waga funkcji ukrytej $h^{(t-1)}$
b_f – bias

Zadaniem bramki wejściowej *i* oraz węzła wejściowego *g* jest aktualizacja stanu $C^{(t)}$. Podobnie jak w przypadku bramki zapominającej, funkcja sigmoid decyduje o tym, które wartości oraz w jakim stopniu są aktualizowane. Stany *i* oraz *g* można obliczyć ze wzorów 5.3 i 5.4:

$$i_t = \sigma(W_{xi}\, x^{(t)} + W_{hi}\, h^{(t-1)} + b_i) \quad (5.3)$$

$$g_t = \tanh(W_{xg}\, x^{(t)} + W_{hg}\, h^{(t-1)} + b_g) \quad (5.4)$$

Funkcja aktywacyjna tangensa hiperbolicznego jest wykorzystywana do przypisywania wag do tych wartości, z zakresu od -1 do 1. Wykorzystanie tej funkcji pozwala również na przezwyciężenie problemu niestabilnego gradientu (zanikającego lub eksplodującego), który znacznie utrudnia skuteczne uczenie sieci neuronowej.

Bramka wyjściowa *o* decyduje o tym, które informacje zostaną przekazane do jednostki ukrytej h_t. Po przejściu informacji przez funkcję sigmoidalną, wyznaczony ze wzoru 5.1 stan komórki $C^{(t)}$ przepuszczany jest funkcję tangensa hiperbolicznego, a następnie jest on przemnażany przez wartość wyjściową *o*. W literaturze [14] iloczyn ten nazywany jest pobudzeniem jednostki ukrytej w takcie *t*. Wartości *o* oraz pobudzenia $h^{(t)}$ można wyznaczyć za pomocą równań:

$$o_t = \sigma(W_{xo}\, x^{(t)} + W_{ho}\, h^{(t-1)} + b_o) \quad (5.5)$$

$$g_t = \tanh(W_{xg}\, x^{(t)} + W_{hg}\, h^{(t-1)} + b_g) \quad (5.6)$$

$$h^{(t)} = o_t \,\Theta\, \tanh(C^{(t)}) \quad (5.7)$$

5.2. Transformacja danych szeregu czasowego.

Podobnie jak w przypadku modeli ARIMA, aby dopasować model predykcyjny do badanego szeregu czasowego, analizę należy rozpocząć od ogólnego zrozumienia danych, z którymi mamy do czynienia. Następnie muszą one zostać przetworzone, zanim zostaną dopasowane do modelu prognostycznego.

Jason Brownlee w swojej książce oraz artykułach [16,17] wyróżnia następujące 3 kroki procesu transformacji danych:

- Uzyskanie stacjonarności danych
- Przekształcenie szeregu czasowego w problem uczenia nadzorowanego
- Skalowanie danych

Poniżej przedstawiono procedurę transformacji danych dla zbioru *shampoo.csv*, użytego do modelowania procesu ARIMA w rozdziale 4. Do stworzenia modeli prognostycznych wykorzystana została biblioteka *Keras*, zawierająca implementację modelu LSTM. Procedura uzyskiwania stacjonarności danych dla zbioru *shampoo.csv* została opisana w rozdziale 4.4. dlatego została pominięta.

Uczenie nadzorowane to sposób uczenia maszynowego, w którym proces nauki jest nadzorowany przez człowieka. Wymaga on podziału danych na komponenty wejściowe *X* oraz wyjściowe *Y*. Komponenty *X* to dane, które dostarczane są maszynie do przetworzenia, celem uzyskania wyników znajdujących się w zbiorze *Y*, na przykład:

$$X = [10, 20, 30, 40]$$
$$Y = [20, 30, 40, 50]$$

Zadaniem maszyny jest znalezienie takiego sposobu przetwarzania danych *X*, aby były one w stanie uzyskać rezultat zawarty w zbiorze *Y*.

W przypadku szeregów czasowych komponentem *X* są dane historyczne, na przykład pochodzące z okresu *t-1*, które mają zostać przekształcone w dane aktualne w okresie *t*. Przekształcenie szeregu czasowego polega zatem na stworzeniu listy przesuniętej w stosunku do danych o jeden okres, na przykład jeden miesiąc dla danych miesięcznych.

W bibliotece *Pandas* zaimplementowana jest funkcja *shift()*, która pozwala na wykonanie operacji przesunięcia danych o zdefiniowany przez użytkownika okres. Utworzono zatem nową listę, która dodana została następnie do zbioru danych. Za pomocą funkcji *concat()* dane zostały powiązane ze sobą wzdłuż kolumn.

```
In [33]: runfile('C:/Users/Admin/Desktop/data/LSTM_shampoo.py', wdir='C:/
Users/Admin/Desktop/data')
       Sales  Sales
Month
1-01     NaN  266.0
1-02   266.0  145.9
1-03   145.9  183.1
1-04   183.1  119.3
1-05   119.3  180.3

In [34]:
```

Rys. 5.2.1 Przetwarzanie danych na problem uczenia nadzorowanego

Na rysunku 5.2.1 można zaobserwować przesunięcie danych względem siebie o jedną pozycję. W wyniku operacji *shift()* w kolumnie reprezentującej komponent *X* pojawiła się wartość numeryczna NaN. Aby zapewnić prawidłową interpretację tej wartości przez sieć neuronową, należy zastąpić wszystkie powstałe wartości NaN zerami. Funkcja *fillna()* biblioteki Pandas pozwala na wykrycie tych wartości, oraz zamianę ich na wartość 0 poprzez interpolację [18].

```
In [37]: runfile('C:/Users/Admin/Desktop/data/LSTM_shampoo.py', wdir='C:/
Users/Admin/Desktop/data')
       Sales  Sales
Month
1-01     0.0  266.0
1-02   266.0  145.9
1-03   145.9  183.1
1-04   183.1  119.3
1-05   119.3  180.3
```

Rys. 5.2.2 Eliminacja wartości NaN

Wykonanie powyższych czynności pozwoliło na przekształcenie szeregu czasowego w problem uczenia nadzorowanego o danych wejściowych *X*, reprezentujących wartości *t-1* oraz danych wyjściowych *Y* przedstawiających wartości aktualne *t*.

W rozdziale 5.1 opisane zostały rodzaje funkcji aktywacyjnych, występujących w klasycznym modelu LSTM. Wartości przyjmowane przez funkcję tangensa hiperbolicznego znajdują się w przedziale od -1 do 1, dlatego dane badanego szeregu czasowego muszą zostać przeskalowane w taki sposób, by mogły zostać przetworzone przez funkcje aktywacyjne sieci neuronowej.

Wykorzystywana jest w tym celu funkcja *MinMaxScaler()* biblioteki *scikit-learn*, która pozwala na automatyczne przeskalowanie wartości szeregu czasowego do wartości znajdujących się w zadanym przedziale, w tym przypadku między -1 a 1.

```
In [53]: runfile('C:/Users/Admin/Desktop/data/LSTM_shampoo.py', wdir='C:/
Users/Admin/Desktop/data')
       Sales
Month
1-01   266.0
1-02   145.9
1-03   183.1
1-04   119.3
1-05   180.3
Wartosci przeskalowane
0   -0.478585
1   -0.905456
2   -0.773236
3   -1.000000
4   -0.783188
dtype: float64
```

Rys. 5.2.3 Skalowanie danych

5.3. Prognozowanie sekwencyjnym modelem LSTM

Tworzenie modelu sieci neuronowej odbywa się poprzez łączenie ze sobą warstw (*ang. layers*). W zależności od wykorzystywanej warstwy, wymagane jest dopasowanie kształtu danych tak, aby dana warstwa była w stanie poprawnie zinterpretować przyjmowane dane.

Aby móc wykorzystać warstwę LSTM zaimplementowaną w bibliotece *keras*, należy przekształcić dane wejściowe do struktury trójwymiarowego tensora, w którym zawierać będą się następujące informacje wykorzystywane przez model:

- *Sample* (próbka) – wartość wskazująca na ilość rzędów danych w zbiorze
- *TimeStep* (znacznik czasowy) – wartość wskazująca, ile razy dane zostały przekazane do modelu
- *Feature* (cecha) - wartość wskazująca, ile kolumn zawiera każda próbka

Kolejnym krokiem jest zdefiniowanie następujących hiperparametrów modelu LSTM:

- Liczba neuronów – wartość wskazująca na ilość neuronów występujących w jednej warstwie LSTM
- Ilość epok – ilość iteracji procesu uczenia
- Rozmiar partii (*ang. batch size*) – ilość jednostek treningowych, użytych w jednej iteracji procesu uczenia

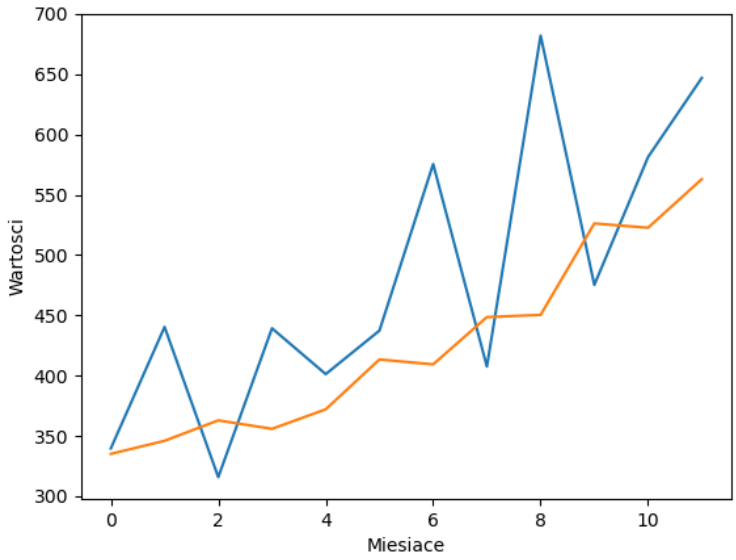
Po zdefiniowaniu warstw oraz parametrów modelu, należy zdefiniować algorytm optymalizujący oraz funkcję straty, na podstawie których przeprowadzany będzie proces uczenia. W badanym przypadku dobrano algorytm *Adam* oraz funkcję straty średniego błędu kwadratowego (MSE).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (1, 1)                    12

 dense (Dense)               (1, 1)                    2

=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0
```

Rys. 5.3.1 Uczenie modelu LSTM, 1 neuron, 100 epok

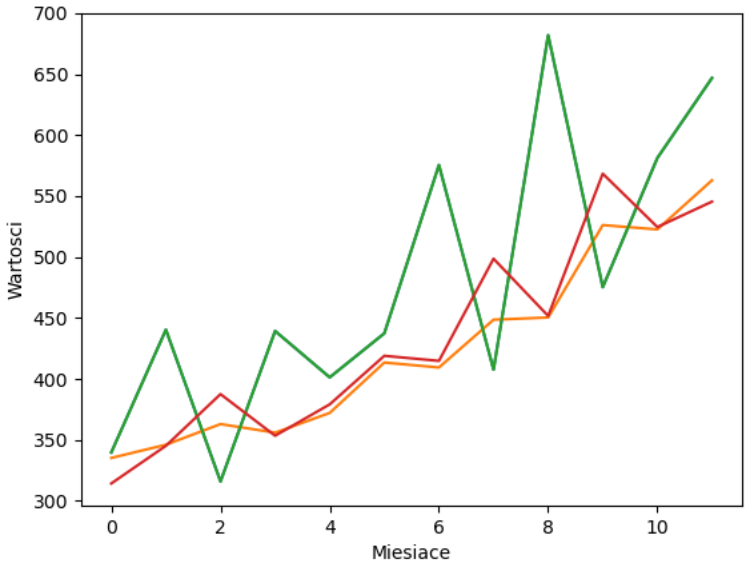


Rys. 5.3.2 Walidacja krzyżowa modelu LSTM, 1 neuron, 100 epok

Dla modelu o zdefiniowanych parametrach (1 neuron, 100 epok) wytrenowany został model, który poddano walidacji krzyżowej ze zbiorem danych testowych. Otrzymano wartości:

- RMSE: 105,032
- MAPE: 17,51%

Podobnie jak w przypadku modeli ARIMA, dobór hiperparametrów sieci neuronowej ma charakter doświadczalny. Co więcej, modele prognostyczne wytrenowane przy tych samych parametrach wejściowych będą różnić się od siebie, co przedstawione zostało na rysunku 5.3.3.

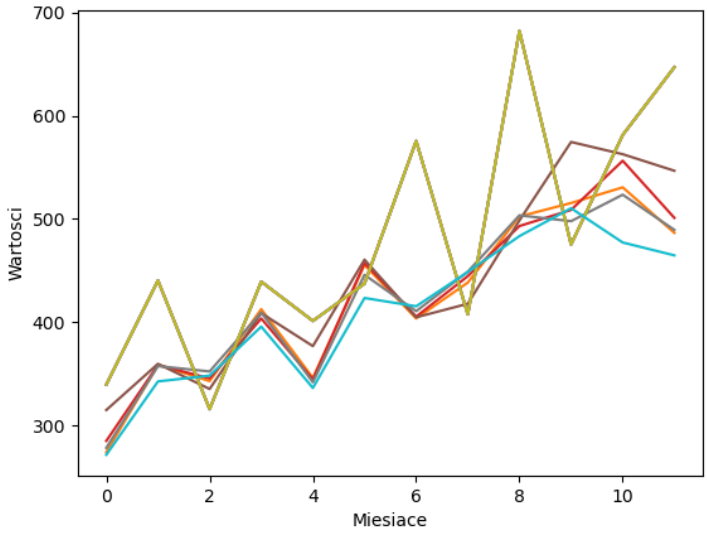


Rys. 5.3.3 Walidacja krzyżowa dwóch modeli LSTM (1 neuron, 100 epok)

Prognozy modeli wytrenowanych dla wstępnie dobranych hiperparametrów obarczone były zbyt dużym błędem (MAPE ~ 19%), dlatego proces doboru został powtórzony.

W artykule [19] Jasona Brownlee opisana została metoda poszukiwania hiperparametrów modeli poprzez porównywanie ich wyników testów RMSE. Pozwala to na stwierdzenie czy zmiana konkretnego hiperparametru ma wpływ na uczenie się modelu.

W drugim cyklu wytrenowano pięć modeli LSTM, zwiększając 10-krotnie liczbę epok przy zachowaniu poprzednich wartości pozostałych hiperparametrów.

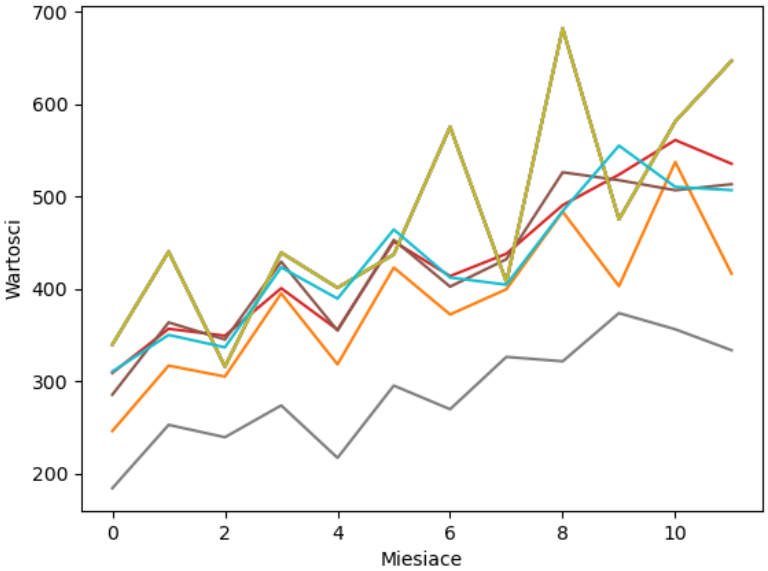


Rys. 5.3.4 Walidacja krzyżowa pięciu modeli LSTM (1 neuron, 1000 epok)

Modele w procesie walidacji krzyżowej uzyskały następujące wyniki:

- Model 1: RMSE = 95,044, MAPE = 14,67%
- Model 2: RMSE = 93,268, MAPE = 14,25%
- Model 3: RMSE = 87,888, MAPE = 12,39%
- Model 4: RMSE = 93,904, MAPE = 14,66%
- Model 5: RMSE = 105,478, MAPE = 16,85%

Na podstawie wyników, można stwierdzić zwiększenie skuteczności prognoz, wraz z ilością epok. Modele uzyskały średni wynik RMSE równy 95,1148 oraz MAPE równy 14.64857456. W kolejnym kroku zbadany został wpływ zwiększenia ilości neuronów na wyniki prognoz.



Rys. 5.3.5 Walidacja krzyżowa pięciu modeli LSTM (2 neurony, 1000 epok)

Uzyskana średnia wartość RMSE wynosiła 120,417 natomiast wartość MAPE była równa 19,252%. Wprowadzenie dodatkowego neuronu spowodowało generowanie modeli o bardzo złych wynikach prognoz. Aby wyeliminować przypadkowość tego zjawiska, zwiększono ilość wytrenowanych modeli do 30.

Na 30 wytrenowanych modeli, aż 10 uzyskało wartość MAPE > 20%, średnia wartość RMSE wynosiła 118,728, średnia wartość MAPE była równa 18,33%. Wskazuje to na znaczne pogorszenie prognoz po dodaniu kolejnego neuronu. Zwiększenie rozmiaru partii również nie miało wpływu na poprawienie jakości prognoz, dlatego w kolejnych cyklach zwiększona została jedynie liczba epok, gdyż tylko ten parametr miał pozytywny wpływ na usprawnienie modelu.

Na 30 wytrenowanych modeli, najlepszy wynik prognoz został osiągnięty dla modelu o parametrach: 1 neuron, 1200 epok, rozmiar partii 1. Osiągnął on wynik RMSE równy 87,423 oraz MAPE na poziomie 11,07%.

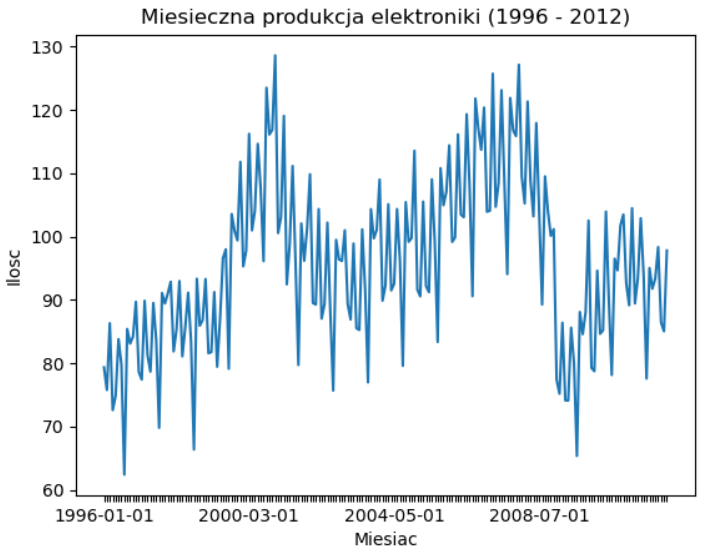
6. Prognozowanie szeregów czasowych modelami LSTM i ARIMA.

Poniższy rozdział zawiera porównanie wyników prognoz dla wybranych szeregów czasowych. Analizie poddane zostały następujące szeregi czasowe:

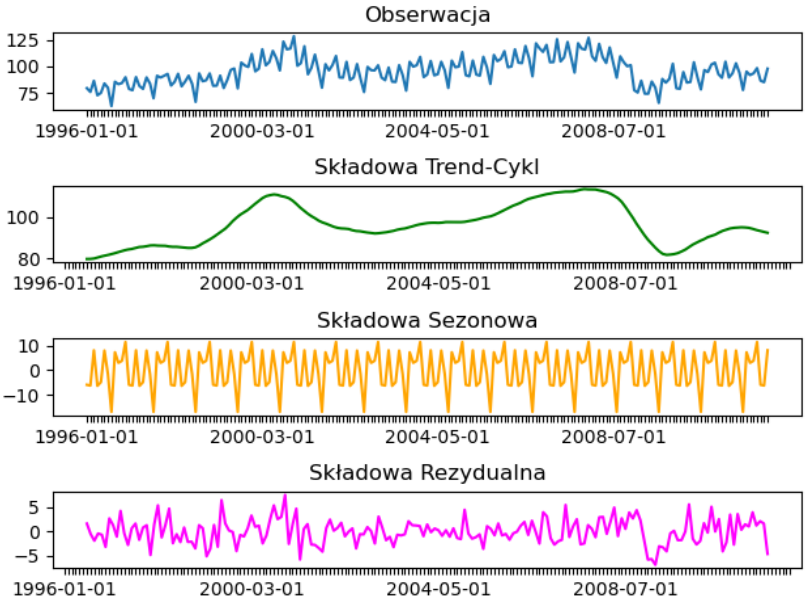
- zbiór danych, przedstawiający średnią miesięczną produkcję sprzętu elektronicznego w Europie (lata 1996 – 2012) [20]
- zbiór danych, przedstawiający średnią miesięczną sprzedaż samochodów w Quebec (lata 1960 – 1968) [21]

6.1. Prognozowanie średniej miesięcznej produkcji sprzętu elektronicznego

Przykładem jednowymiarowego szeregu czasowego jest pochodzący z Eurostatu zbiór danych, przedstawiający miesięczną produkcję sprzętu elektronicznego w Europie. Prognoza tego zjawiska może umożliwić lepsze dopasowanie produkcji do obecnego zapotrzebowania na sprzęt elektroniczny. W pierwszym kroku zbiór został zwizualizowany oraz poddany dekompozycji:

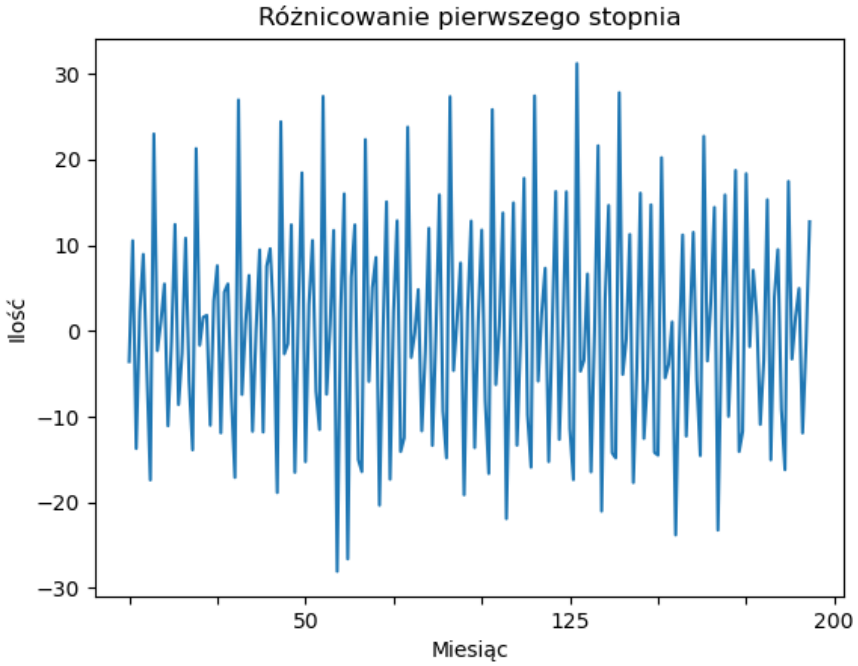


Rys. 6.1.1. Miesięczna produkcja elektroniki w Europie

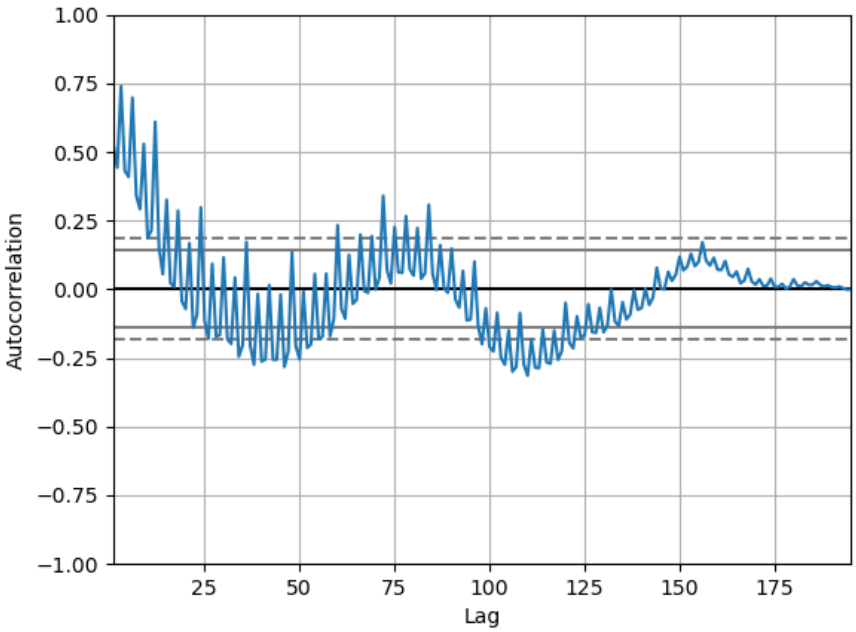


Rys. 6.1.2. Dekompozycja szeregu czasowego

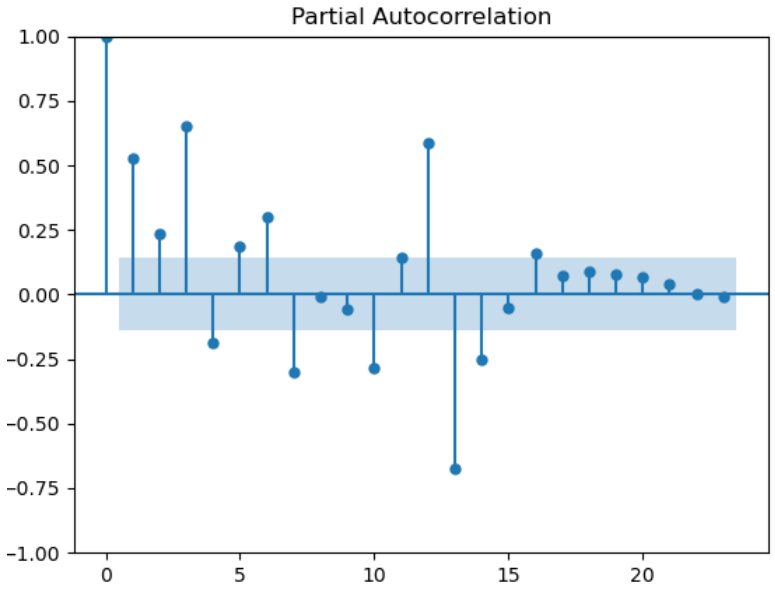
Drugim krokiem było uzyskanie stacjonarności danych:



Rys. 6.1.3. Różnicowanie szeregu czasowego



Rys. 6.1.4. Wykres funkcji autokorelacji szeregu czasowego



Rys. 6.1.5. Wykres funkcji autokorelacji cząstkowej szeregu czasowego

Wyniki testu Dickeya-Fullera i wartości krytycznych dla zróżnicowanych danych:

- Statystyka ADF: -3,2112221677217723
- p-wartość: 0,019344216566089284
- 1%: -3,467631519151906
- 5%: -2,8779183721695567
- 10%: -2,575501353364474

Badany szereg uzyskał stacjonarność po jednokrotnym różnicowaniu. Statystyka ADF zawiera się w przedziale wartości krytycznych, natomiast p-wartość < 0,05.

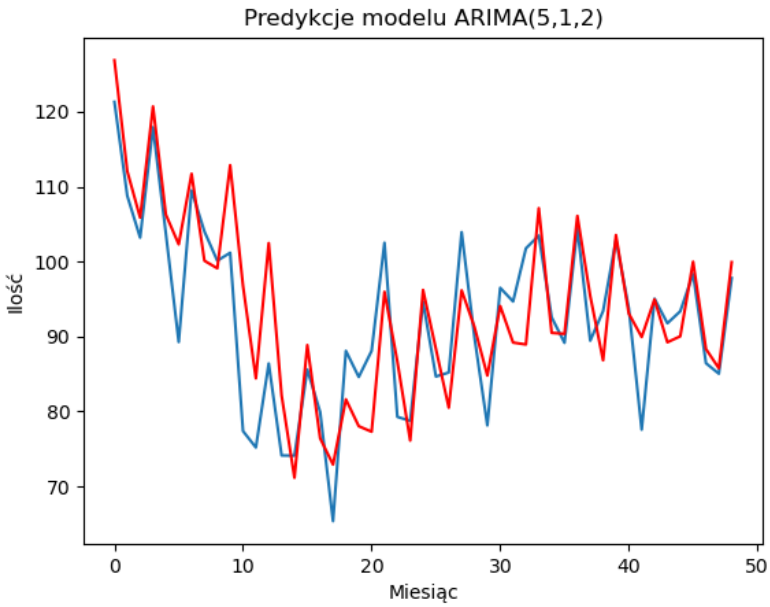
SARIMAX Results

Dep. Variable:	y	No. Observations:	146
Model:	SARIMAX(5, 1, 2)	Log Likelihood	-475.771
Date:	Sun, 30 Jan 2022	AIC	967.541
Time:	23:16:53	BIC	991.355
Sample:	0	HQIC	977.218
	- 146		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.3364	0.085	-15.760	0.000	-1.503	-1.170
ar.L2	-1.7148	0.144	-11.874	0.000	-1.998	-1.432
ar.L3	-1.1630	0.186	-6.236	0.000	-1.529	-0.797
ar.L4	-0.8220	0.142	-5.779	0.000	-1.101	-0.543
ar.L5	-0.4391	0.083	-5.309	0.000	-0.601	-0.277
ma.L1	0.8419	0.041	20.700	0.000	0.762	0.922
ma.L2	0.9140	0.044	20.698	0.000	0.827	1.001
sigma2	37.7787	4.841	7.804	0.000	28.291	47.267

Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):	7.83
Prob(Q):	0.85	Prob(JB):	0.02
Heteroskedasticity (H):	0.89	Skew:	-0.56
Prob(H) (two-sided):	0.69	Kurtosis:	3.19

Rys. 6.1.6. Statystyki dopasowania modelu ARIMA(5,1,2)

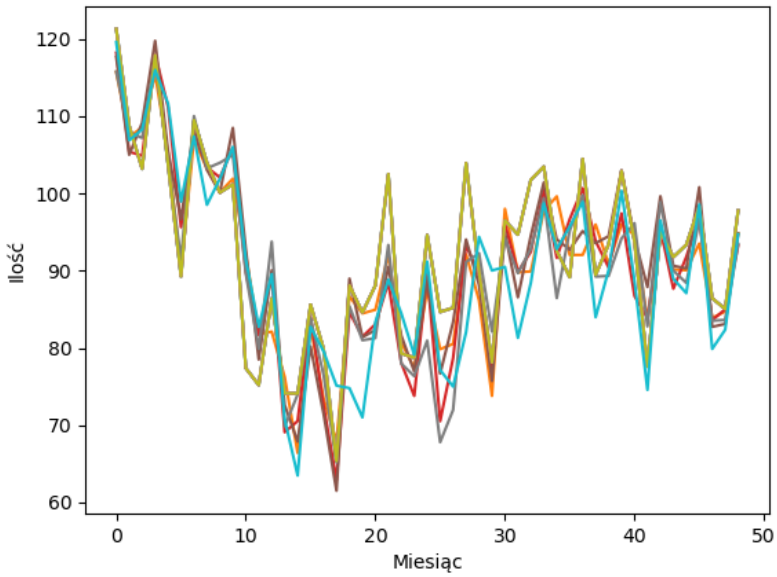


Rys. 6.1.7. Statystyki dopasowania modelu ARIMA(5,1,2)

Dla analizowanych danych z okresu 12 lat, wyznaczono zbiór treningowy (75%) oraz testowy (25%). Na ich podstawie za pomocą narzędzia *auto_arima* wyznaczone zostały parametry *p,d,q* poprzez minimalizację funkcji AIC oraz BIC. W procesie walidacji krzyżowej danych testowych z wartościami prognoz wyznaczone zostały błędy RMSE oraz MAPE, które wynosiły:

- RMSE: 6,696
- MAPE: 5,157

Za pomocą modelu ARIMA uzyskano model prognostyczny, o bardzo wysokiej klasie dokładności (około 95%).



Rys. 6.1.8. Walidacja krzyżowa pięciu modeli LSTM (32 neuroy, 2000 epok)

Za pomocą modelu LSTM, dla hiperparametrów:

- 32 neurony
- Rozmiar partii = 1
- 2000 epok

Wytrenowanych zostało 5 modeli, o bardzo zbliżonych wartościach RMSE oraz MAPE:

- Wartość średnia RMSE: 5,992
- Wartość średnia MAPE: 5,51%

```
<keras.engine.sequential.Sequential object at 0x00000207F256EF10>
Model: "sequential_62"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_62 (LSTM)              (1, 32)                   4352

 dense_62 (Dense)            (1, 1)                    33

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
None
Test RMSE 2: 5.375
4.6882640954005055
```

Rys. 6.1.9. Sekwencyjny model LSTM

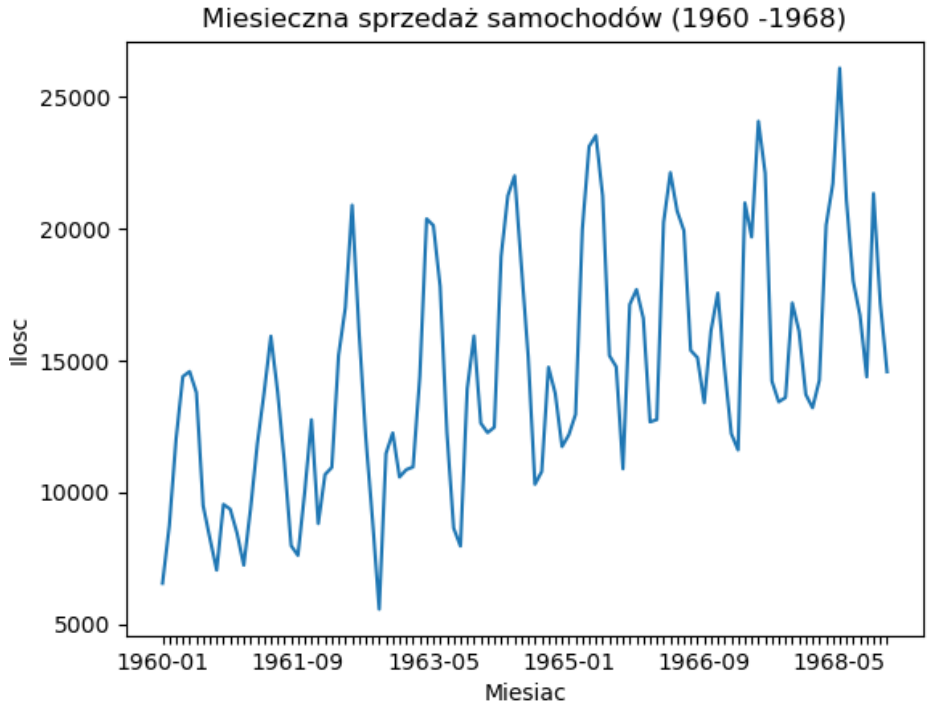
Najlepsze wyniki RMSE oraz MAPE osiągnięte przez sieć wytrenowaną dla zadanych hiperparametrów wynosiły:

- RMSE: 5,375
- MAPE: 4,688%

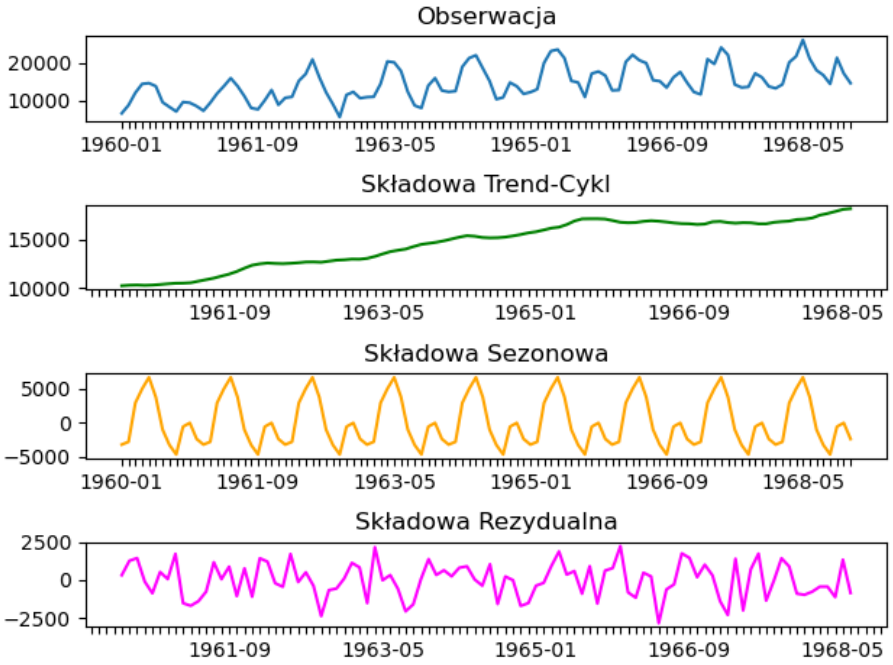
Model LSTM pozwolił zatem na wygenerowanie modelu prognostycznego o bardzo wysokiej klasie dokładności (około 95.5%). Model pozytywnie reagował na zwiększanie ilości neuronów, dla epok w przedziale 1000 – 2000, dlatego istnieje możliwość znalezienia modeli o jeszcze większej dokładności, jednakże ze względu na stosunkowo długi czas trenowania sieci, uzyskany model został zaakceptowany.

6.2. Prognozowanie średniej miesięcznej sprzedaży samochodów

Kolejnym przykładem jednowymiarowego szeregu czasowego jest zbiór danych, opisujący ilość sprzedawanych samochodów w mieście Quebec. Prognozowanie tego zjawiska pozwoli na oszacowanie, w których okresach popyt na zakup samochodu jest największy, dzięki czemu fabryka produkująca komponenty do samochodów będzie mogła zoptymalizować procesy ich wytwarzania, magazynowania oraz sprzedaży.



Rys. 6.2.1. Ilość sprzedanych samochodów w Quebec(1960 – 1968)



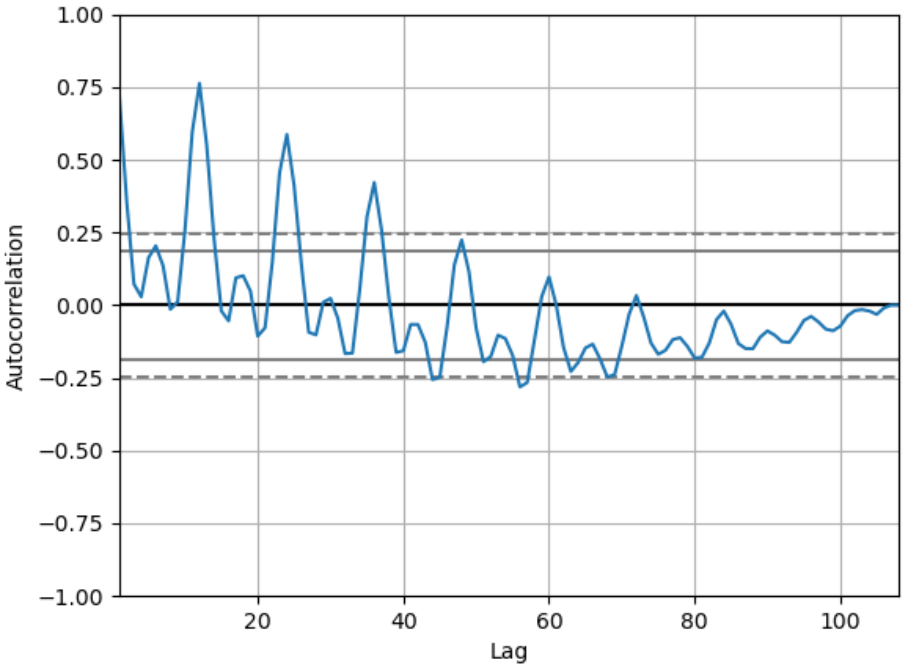
Rys. 6.2.2. Dekompozycja szeregu czasowego

Rys. 6.2.3. Różnicowanie szeregu czasowego

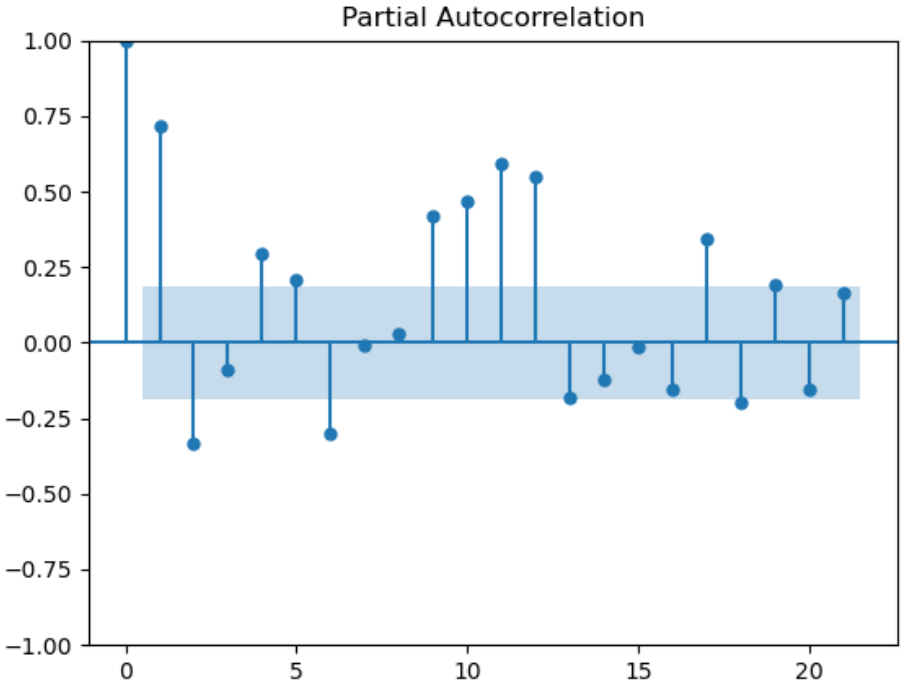
Różnicowanie pierwszego stopnia pozwoliło na uzyskanie stacjonarnego szeregu czasowego. Wartości testu Dickeya-Fullera wynosiły:

- Statystyka ADF: -6,1997992308007
- p-wartość: 5,844752599276674e-08

Statystyka ADF nie przekraczała żadnej wartości krytycznej (1%: -3,50) oraz p-wartość << 0,05. Jednokrotnie różnicowany szereg czasowy jest zatem stacjonarny.



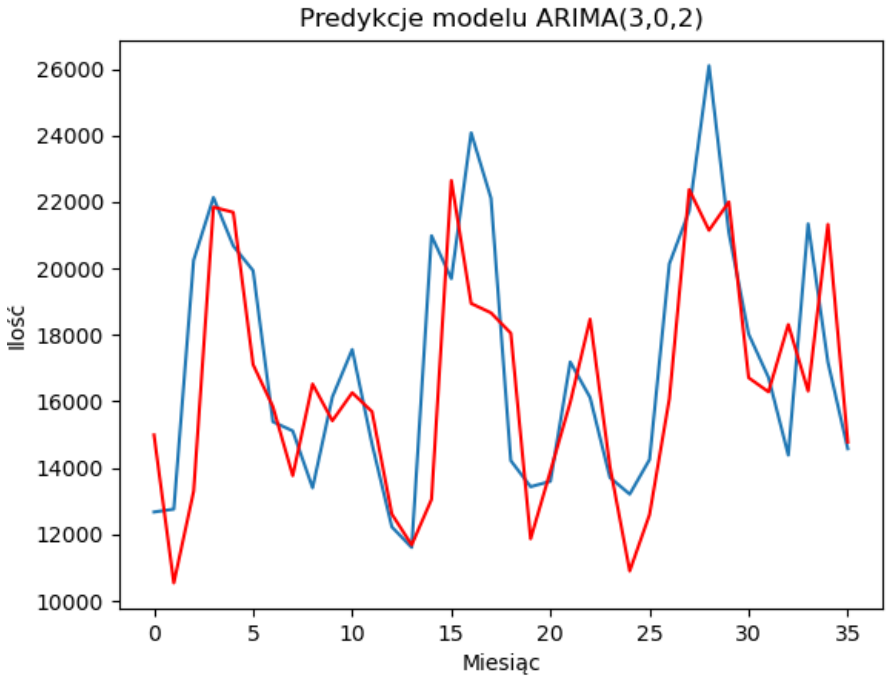
Rys. 6.2.4. Wykres funkcji autokorelacji szeregu czasowego



Rys. 6.2.5. Wykres funkcji autokorelacji cząstkowej szeregu czasowego

```
Best model:  ARIMA(3,0,2)(0,0,0)[0] intercept
Total fit time: 1.520 seconds
                               SARIMAX Results
==========================================================================
Dep. Variable:                      y   No. Observations:                   72
Model:               SARIMAX(3, 0, 2)   Log Likelihood                -657.096
Date:                Mon, 31 Jan 2022   AIC                           1328.193
Time:                        17:27:10   BIC                           1344.129
Sample:                             0   HQIC                          1334.537
                                  - 72
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
intercept   5051.2581   1841.773      2.743      0.006    1441.450    8661.067
ar.L1          1.5066      0.141     10.693      0.000       1.230       1.783
ar.L2         -1.4096      0.153     -9.219      0.000      -1.709      -1.110
ar.L3          0.5219      0.135      3.873      0.000       0.258       0.786
ma.L1         -0.5561      0.093     -5.991      0.000      -0.738      -0.374
ma.L2          0.9410      0.125      7.502      0.000       0.695       1.187
sigma2       5.78e+06      0.410   1.41e+07      0.000    5.78e+06    5.78e+06
==========================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 0.29
Prob(Q):                              0.90   Prob(JB):                         0.86
Heteroskedasticity (H):               1.83   Skew:                             0.13
Prob(H) (two-sided):                  0.14   Kurtosis:                         2.83
==========================================================================
```

Rys. 6.2.6. Statystyki dopasowania modelu ARIMA(3,0,2)

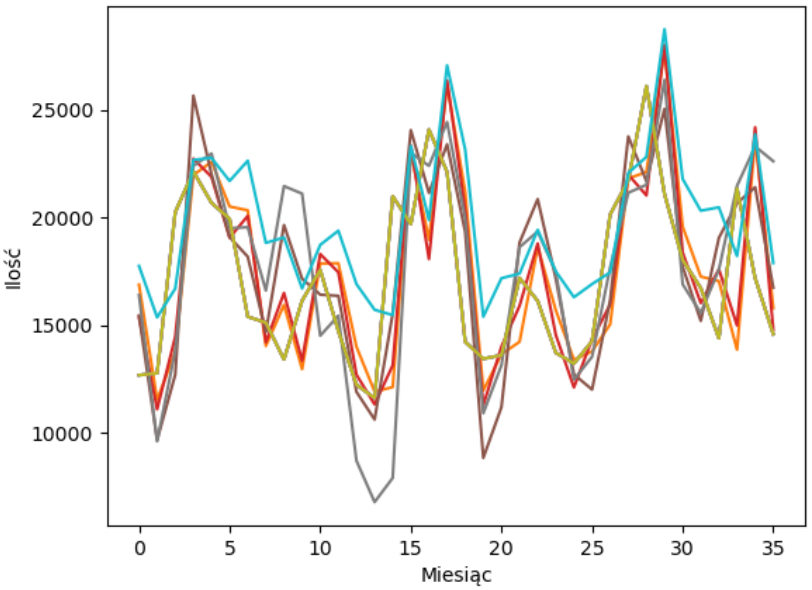


Rys. 6.2.7. Walidacja krzyżowa szeregu czasowego

Dla jednokrotnie zróżnicowanego zbioru danych trenujących, narzędzie *auto_arima* dobrało hiperparametry modelu ARIMA(3,0,2). Pozwoliło to na stworzenie modelu prognostycznego o następujących wynikach:

- RMSE: 3015,018
- MAPE: 12,896%

Wytrenowany model pozwala na uzyskanie dobrych wyników prognoz (około 87% dokładności).



Rys. 6.2.8. Walidacja krzyżowa pięciu modeli LSTM (2 neuroy, 500 epok)

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_2 (LSTM)                (1, 2)                    32

 dense_2 (Dense)              (1, 1)                    3

=================================================================
Total params: 35
Trainable params: 35
Non-trainable params: 0
_________________________________________________________________
None
Test RMSE 1: 3608.197
15.490152507069022
```

Rys. 6.2.9. Sekwencyjny model LSTM

Za pomocą modelu LSTM, dla hiperparametrów:

- 2 neurony
- Rozmiar partii = 1
- 500 epok

Wytrenowanych zostało 5 modeli, spośród których najlepsze wartości błędów prognostycznych wynosiły:

- RMSE: 3608.197
- MAPE: 15.49%

Wytrenowany model pozwolił na uzyskanie dobrych wyników prognoz (około 84,5%)

7. Wnioski końcowe.

Postępowanie zgodnie z metodyką Boxa-Jenkinsa pozwoliło na opracowanie wysokiej jakości modeli prognostycznych, zarówno dla modeli ARIMA jak i LSTM.

W rozdziałach czwartym i piątym przedstawiony został proces tworzenia modeli prognostycznych dla jednowymiarowego szeregu czasowego miesięcznej ilości sprzedanych butelek szamponu. Na podstawie danych uzyskane zostały dwa modele prognostyczne, o różnych wartościach błędu prognostycznego (model ARIMA – 9,87%, model LSTM – 11,08%).

Porównując ze sobą obie metody, lepszy rezultat osiągnął model autoregresyjny. Warto jednak zauważyć, że badany zbiór danych był niewielkich rozmiarów (36 punktów). Modele prognostyczne w oparciu o rekurencyjne sieci neuronowe z reguły lepiej sprawdzają się dla większych zbiorów danych, ze względu na zapamiętywanie wartości historycznych.

W rozdziale szóstym analizowano większy zbiór danych (195 punktów) ilości wyprodukowanego sprzętu elektronicznego. Uzyskane na ich podstawie modele prognostyczne miały zbliżone wartości błędów MAPE (model ARIMA – 5,157%, model LSTM – 4,688%). Co więcej, model LSTM wykazywał tendencję do polepszania wyniku prognoz, wraz z dodawaniem kolejnych neuronów. Potwierdza to skuteczność modeli LSTM rosnącą wraz z ilością danych w zbiorze.

Drugi z analizowanych szeregów czasowych pozwolił na wytrenowanie modeli o niższej dokładności (model ARIMA – 87%, model LSTM – 84,5%). Spadek jakości prognoz może być spowodowany dość silną sezonowością występującą w badanym szeregu czasowym.

Dla tego typu szeregów czasowych lepsze wartości prognoz można uzyskać przez wykorzystanie sezonowych modeli ARIMA. Wykorzystują one dodatkowy zestaw parametrów sezonowości P,D,Q oraz m, dzięki którym można określić częstotliwość występowania zjawiska sezonowego.

Aby usprawnić prognozy modelu LSTM, można zwiększyć ilość warstw ukrytych w nim występujących. Dodawanie kolejnych warstw pozwala na stworzenie modelu uczącego się głębiej, co może wpłynąć korzystnie na poprawę wyników prognoz. Można również rozważyć zmianę funkcji aktywacyjnej, występującej w modelu, na przykład na funkcję rektyfikowanej jednostki liniowej ReLU.

Ze względu na doświadczalny charakter badania szeregów czasowych, każda wykonana prognoza powinna zostać zweryfikowana, poprzez wykonanie wielu modeli oraz stopniowe wykluczanie tych, które wykazują mniejszą skuteczność. Porównanie modeli bazujących na różnych założeniach matematycznych pozwoli również na uzyskanie informacji o jakości prognozowanych danych. W przypadku każdego z przedstawionych modeli prognostycznych różnice w prognozach między modelami LSTM a ARIMA nie przekraczały 3%, dlatego obniżona jakość prognoz w przypadku analizowanego szeregu czasowego może być wynikiem niskiej jakości danych w badanym zbiorze.

Warto również nadmienić, że dość częstym problemem pojawiającym się w analizie danych jest ich niska jakość lub braki w danych. Warto wówczas skorzystać z metod uzupełniających, takich jak algorytm k-najbliższych sąsiadów lub wielokrotna imputacja, które pozwolą na uzyskanie brakujących wartości. Uzupełnianie danych poprzez liczenie średniej prowadzi do eliminacji autokorelacji występującej w szeregu czasowym, dlatego zalecane jest korzystanie z metod, które ograniczą ewentualne straty do minimum.

Literatura

[1] https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/
[2] https://otextx.com/fpp2/
[3] https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
[4] https://otexts.com/fpp2/
[5] https://www.statsoft.pl/textbook/stathome_stat.html?https%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fsttimser.html
[6] https://hal.archives-ouvertes.fr/hal-01965577/document
[7] https://www.stat.berkeley.edu/~bartlett/courses/153-fall2010/lectures/5.pdf
[8] https://www.ibm.com/docs/pl/spss-modeler/SaaS?topic=data-series-transformations
[9] https://pkg.yangzhuoranyang.com/tsdl/
[10] http://coin.wne.uw.edu.pl/pstrawinski/dane_fin/adf03.pdf
[11] https://alkaline-ml.com/pmdarima/tips_and_tricks.html
[12] https://alkaline-ml.com/pmdarima/0.9.0/modules/generated/pyramid.arima.auto_arima.html
[13] https://machinelearningmastery.com/grid-search-arima-hyperparameters-with-python/
[14] Sebastian Raschka, Vahid Mirjalili: *Python. Uczenie maszynowe. Wydanie 2.*
[15] Sebastian Raschka, Vahid Mirjalili: *Python. Uczenie maszynowe. Wydanie 2.*
[16] Jason Brownlee: *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*
[17] https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/
[18] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html
[19] https://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting/
[20] https://rdrr.io/cran/fpp2/man/elecequip.html
[21] https://data.world/perceptron/monthly-car-sales-quebec-1960

Summary

The purpose of the thesis *Application of machine learning algorithms for the optimization of production processes* is to create predictive models for production processes described by univariate time series. The objective of these models is to create a good quality forecasts, based on the historical data, which will provide a better understanding of the phenomena occuring in the studied processes, such as trend, seasonality, autocorrelation.

This thesis consists of a theoretical part, explaining the process of time series analysis and fitting them to forecasting models, based on the long-short term memory neural networks and autoregressive integrated moving average models. The following sections describe:

- data visualization, decomposition of a time series into its components;
- ARIMA model forecasting - autoregressive model, moving average model, development of a forecasting model based on Box-Jenkins methodology, obtaining data stationarity, verification of stationarity with augmented Dickey-Fuller test, use of auto_arima tool for determining ARIMA(p,d,q) hyperparameters by minimizing Akaike and Bayesian information criterions, cross validation of model, acceptance of predictive model based on root mean squared error (RMSE) and mean absolute percentage error (MAPE);
- LSTM model forecasting - LSTM model operation, activation functions (sigmoid, tanh), transformation of time series data to a supervised learning problem, data scaling, tuning of hyperparameters (neuron, epoch, batch size), cross-validation of model, acceptance of predictive model based on root mean squared error (RMSE) and mean absolute percentage error (MAPE);

The second part includes a comparison of forecasting results for two exemplary univariate time series, which represent certain production processes. The use of Box-Jenkins methodology contributed to achieving high quality predictive models.

The last section contains conclusions about the application of machine learning algorithms for univariate time series forecasting. It presents the forecast error results obtained by each model and describes how their performance can be improved. It also mentions common problems related to data analysis, such as missing data and overall poor quality of real data. In such cases it is worth to use supplementary methods, such as k-nearest neighbors algorithm or multiple imputation, which will allow to obtain the missing values without significant modification of level, trend, seasonality and autocorrelation.