



Politechnika Świętokrzyska

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Mateusz Godlewski

Numer albumu: 85163

**Oprogramowanie dydaktyczne do testowania działania
wybranych metod dla modeli szeregów czasowych na
różnych zbiorach danych.**

**Praca magisterska
na studiach II-go stopnia
na kierunku Informatyka**

Opiekun pracy dyplomowej:

dr inż. Andrzej Kułakowski

Katedra Systemów Informatycznych

Kielce, 2021

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Studia stacjonarne
Kierunek: INFORMATYKA

Rok akademicki: 2020/21

Zatwierdzam:
Prof. dr hab. inż. Barbara Łukawska
Katedra Informatyki Stosowanej
Wydział Elektrotechniki, Automatyki i Informatyki
.....
dr inż. Barbara Łukawska
Prodziekan ds. studenckich i dydaktyki

ZADANIE NA PRACĘ DYPLOMOWĄ
STUDIÓW DRUGIEGO STOPNIA

Wydano dla studenta:

inż. Mateusz Godlewski

I. Temat pracy:

Oprogramowanie dydaktyczne do testowania działania wybranych metod dla modeli szeregów czasowych na różnych zbiorach danych.

Didactic software to test the operation of selected methods for time series models on different datasets.

II. Plan pracy:

1. Charakterystyka wybranego oprogramowania dydaktycznego do testowania działania metod matematycznych.
2. Opracowanie wybranych zagadnień do konstruowania oprogramowania testującego.
3. Charakterystyka wybranego do realizacji zadania oprogramowania.
4. Projekt oprogramowania dydaktycznego do testowania działania wybranych metod dla modeli szeregów czasowych.
5. Implementacja oprogramowania dydaktycznego.
6. Testowanie działania wykonanego oprogramowania.

III. Cel pracy:

Celem pracy jest wykonanie oprogramowania dydaktycznego do testowania działania wybranych metod dla modeli szeregów czasowych na różnych zbiorach danych.

IV. Uwagi dotyczące pracy:

V. Termin oddania pracy: zgodnie z Regulaminem Studiów tj. do końca zajęć semestru dyplomowego.

VI. Konsultant:

Opiekun merytoryczny
KIEROWNIK
Katedry Informatyki Stosowanej
Wydział Elektrotechniki, Automatyki i Informatyki
.....
dr hab. inż. Paweł Sitek, prof. PŚk
(pieczęć i podpis)

Promotor pracy dyplomowej

.....
dr inż. Andrzej Kulakowski

Temat pracy dyplomowej celem jej wykonania otrzymałem(am):

Kielce, dnia 11.06.2021 r.
Godlewski Mateusz
czytelny podpis studenta



Politechnika Świętokrzyska

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Kielce, dnia 14.06.2021

Mateusz Godlewski 085163

Imię i nazwisko studenta

nr albumu

Seminarijska 28/25, 25-385 Kielce

Adres zamieszkania

Informatyka, Grafika Komputerowa, II rok, stacjonarne

Kierunek, specjalność, rok studiów, rodzaj studiów (stacjonarne, niestacjonarne)

dr inż. Andrzej Kutakowski

Opiekun pracy dyplomowej inżynierskiej/magisterskiej*

OSWIADCZENIE

Przedkładając w roku akademickim 2020/21, opiekunowi pracy dyplomowej inżynierskiej/magisterskiej*, powołanemu przez Dziekana Wydziału Elektrotechniki, Automatyki i Informatyki Politechniki Świętokrzyskiej, pracę dyplomową inżynierską/magisterską* pod tytułem:

Oprogramowanie dydaktyczne do testowania obrotowa
wybranych metod dla modeli szeregow, czasowych i innych danych

oswiadczam, że:

- 1) przedstawiona praca dyplomowa inżynierska/magisterska* została opracowana przeze mnie samodzielnie, stosownie do wskazówek merytorycznych opiekuna pracy,
- 2) przy wykonywaniu pracy dyplomowej inżynierskiej/magisterskiej* wykorzystano materiały źródłowe, w granicach dozwolonego użytku wymieniając autora, tytuł pozycji i miejsce jej publikacji,
- 3) praca dyplomowa inżynierska/magisterska* nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
- 4) przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia zawodowego/naukowego w wyższej uczelni,
- 5) niniejsza wersja pracy jest identyczna z załączoną treścią elektroniczną (na CD i w systemie Archiwum Prac Dyplomowych).

Przyjmuję do wiadomości, iż w przypadku ujawnienia naruszenia przepisów ustawy o prawie autorskim i prawach pokrewnych, praca dyplomowa inżynierska/magisterska* może być unieważniona przez Uczelnię, nawet po przeprowadzeniu obrony pracy.

Zostaję uprzedzony:

- 1) o odpowiedzialności karnej wynikającej z art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. 1994 Nr 24, poz. 83, t.j. Dz. U. 2018 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”,
- 2) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 i 2 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668, z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta. Student może być ukarany przez rektora lub komisję dyscyplinarną”.

Prawdziwość powyższego oświadczenia potwierdzam własnoręcznym podpisem.

Godlewski Mateusz

czytelny podpis studenta

*) niepotrzebne skreślić

Oprogramowanie dydaktyczne do testowania działania wybranych metod dla modeli szeregów czasowych na różnych zbiorach danych

Streszczenie

Celem mojej pracy dyplomowej było zaprojektowanie oraz stworzenie oprogramowania dydaktycznego, umożliwiającego testowanie działania wybranych metod predykcji szeregów czasowych. Oprogramowanie to umożliwia wykonanie analizy statystycznej oraz eksploracyjnej na dostarczonym przez użytkownika szeregu czasowym.

W niniejszej pracy zawarty jest opis wybranego oprogramowania służącego do testowania działania metod matematycznych, charakterystyka zagadnień których dotyczy oprogramowanie oraz charakterystyka tworzonego oprogramowania.

Oprogramowanie zostało dokładnie udokumentowane. Natomiast projekt został zilustrowany w postaci wykresów i diagramów, a jego implementacja została przedstawiona poprzez fragmenty kodów źródłowych i zrzuty ekranu.

Słowa kluczowe: Szereg czasowy, statystyka, autoregresja, średnia ruchoma, Python, Flask.

Didactic software for testing selected methods for time series models on various data sets

Summary

Purpose of my thesis was to design and create didactic software that would enable user to test selected time series prediction methods. This software enables performing statistical and exploratory analysis on the time series given by user.

Present work contains description of selected software that is using for testing mathematical operations, the characteristic of issues related to the software and characteristic of developed program itself.

The software was thoroughly documented. Project of software has been detailed illustrated in the form of charts and diagrams, and its implementation has been presented through source code fragments and screenshots.

Keywords: Time series, statistics, autoregression, moving average, Python, Flask.

SPIS TREŚCI

1. WSTĘP	11
2. CHARAKTERYSTYKA WYBRANEGO OPROGRAMOWANIA DYDAKTYCZNEGO DO TESTOWANIA DZIAŁANIA METOD MATEMATYCZNYCH	13
2.1 Oprogramowania matematyczne	13
2.2 Opis programu MATLAB	14
3. OPRACOWANIE WYBRANYCH ZAGADNIEŃ DO KONSTRUOWANIA OPROGRAMOWANIA TESTUJĄCEGO.....	17
3.1 Szereg czasowy.....	17
3.2 Analiza statystyczna szeregów czasowych.....	20
3.3 Metody predykcji szeregów czasowych	23
3.3.1 Model autoregresyjny (AR)	23
3.3.2 Średnia ruchoma (MA).....	26
3.3.3 Model autoregresyjny zintegrowany ze średnią ruchomą (ARIMA).....	27
4. CHARAKTERYSTYKA WYBRANEGO DO REALIZACJI ZADANIA OPROGRAMOWANIA.....	29
4.1 Opis funkcjonalności oprogramowania	29
4.4.1 Przesyłanie pliku	30
4.4.2 Analiza statystyczna oraz wizualizacja szeregu czasowego	31
4.4.3 Prognozowanie szeregów czasowych	33
4.2 Wykorzystane technologie.....	35
5. PROJEKT OPROGRAMOWANIA DYDAKTYCZNEGO DO TESTOWANIA DZIAŁANIA WYBRANYCH METOD DLA MODELI SZEREGÓW CZASOWYCH.....	36
5.1 Architektura oprogramowania.....	36
5.2 Interfejs użytkownika	38

6. IMPLEMENTACJA OPROGRAMOWANIA DYDAKTYCZNEGO	41
6.1 Struktura plików aplikacji	41
6.2 Model szeregu czasowego.....	43
6.3 Przesyłanie pliku	45
6.4 Analiza statystyczna	48
6.5 Modele predykcyjne	52
7. TESTOWANIE DZIAŁANIA WYKONANEGO OPROGRAMOWANIA.....	00
8. PODSUMOWANIE	00
LITERATURA.....	00
Spis rysunków	00
Spis listingów	00

Wstęp

Dynamiczny rozwój szeroko pojętej sztucznej inteligencji w ostatnim dziesięcioleciu sprawił, iż znalazła swoje zastosowanie w prawie każdej sferze życia codziennego. Uczenie maszynowe, sztuczne sieci neuronowe, czy analiza danych, wykorzystywane są w najbardziej popularnych serwisach internetowych czy aplikacjach mobilnych. Rozpoznawanie obrazów, rozpoznawanie mowy, eksploracja danych czy prognozowanie przyszłości, to jedynie kilka z wielu funkcjonalności dostarczanych przez algorytmy szeroko pojętej sztucznej inteligencji.

Celem mojej pracy dyplomowej było stworzenie oprogramowania dydaktycznego w formie aplikacji internetowej, umożliwiającego testowanie wybranych metod prognozy szeregów czasowych, na różnych zbiorach danych. Użytkownik ma możliwość wykonania analizy statystycznej, na dostarczonym przez niego zbiorze danych oraz wizualizacji zawartych w nim danych w postaci wykresów. Główną funkcjonalnością stworzonego oprogramowania, jest możliwość testowania metod predykcji przyszłych wartości szeregów czasowych, za pomocą autoregresji lub autoregresji ze zintegrowaną średnią ruchomą.

Dokument składa się z ośmiu rozdziałów przedstawiających najważniejsze aspekty pracy. Na wstępie przedstawiony został cel pracy oraz krótki opis poszczególnych rozdziałów. W rozdziale drugim pt. "Charakterystyka wybranego oprogramowania dydaktycznego do testowania działania metod matematycznych" przedstawiona została charakterystyka najpopularniejszego oprogramowania które jest wykorzystywane przy testowaniu metod matematycznych. Następny rozdział zatytułowany "Opracowanie wybranych zagadnień do konstruowania oprogramowania testującego" zawiera zagadnienia oraz definicje pojęć, które najczęściej występują w omawianym oprogramowaniu. W rozdziale numer cztery "Charakterystyka wybranego do realizacji zadania oprogramowania" Zostały zaprezentowane narzędzia oraz technologie wykorzystane do implementacji systemu, jak i opis funkcjonalności oprogramowania. W kolejnym rozdziale pt. „Projekt oprogramowania dydaktycznego do testowania działania wybranych metod dla modeli szeregów czasowych” zawarty jest projekt architektury stworzonego oprogramowania w postaci wykresów i diagramów. Następnie w rozdziale szóstym „Implementacja oprogramowania dydaktycznego.” znajduje się przedstawienie zaimplementowanych funkcjonalności systemu, wraz z kodami źródłowymi oraz zrzutami ekranu. Rozdział siódmy "Testowanie działania wykonanego oprogramowania" poświęcony został opisowi testów stworzonego oprogramowania. Ostatni, ósmy rozdział zawiera podsumowanie pracy.

2. CHARAKTERYSTYKA WYBRANEGO OPROGRAMOWANIA DYDAKTYCZNEGO DO TESTOWANIA DZIAŁANIA METOD MATEMATYCZNYCH

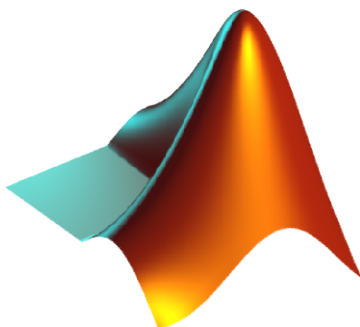
Matematyka, jako przysłowiowa „królowa nauk”, bez wątpienia w ogromnej mierze przyczyniła się do postępu cywilizacyjnego, a zwłaszcza postępu technologicznego, z którym mamy obecnie do czynienia. Znajduje ona swoje zastosowanie w medycynie, sporcie, inżynierii, ekonomii, muzyce, sztuce oraz w wielu aspektach codziennego życia. Obecnie w celach naukowych lub dydaktycznych, wykorzystuje się tzw. „oprogramowania matematyczne”.

2.1 Oprogramowania matematyczne

Oprogramowanie matematyczne, to oprogramowanie wykorzystywane do modelowania, analizy oraz obliczeń numerycznych, symbolicznych lub geometrycznych. Jako przykład najprostszego z nich, można przedstawić oprogramowanie prostego kalkulatora [1]. W tym rozdziale skupię się na bardziej zaawansowanych oprogramowaniach, które umożliwiają modelowanie oraz wykonywanie bardziej zaawansowanych operacji. Wśród najpopularniejszych rozwiązań, dostępnych na rynku wyróżniają się:

- MATLAB;
- Magnolia;
- Maple;
- Scilab.

Na potrzeby pracy, zostanie omówione najbardziej popularne oprogramowanie matematyczne, jakim jest MATLAB oraz zostaną przedstawione przykłady zastosowania tego rozwiązania na praktycznych przykładach.



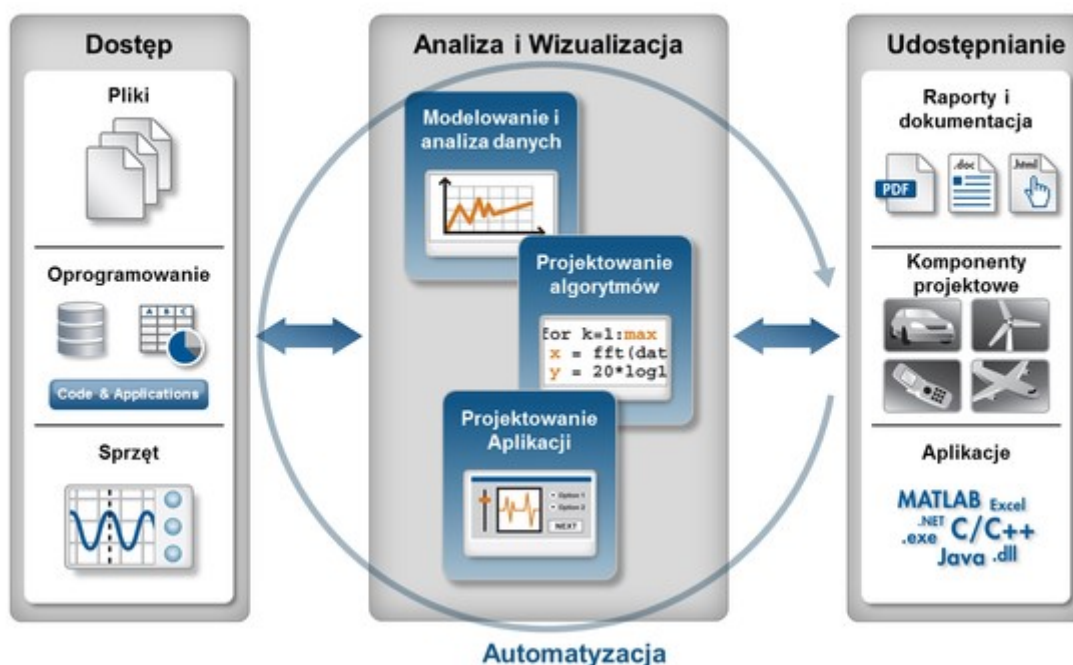
Rysunek 2.1 Logo programu MATLAB [22]

2.2 Opis programu MATLAB

MATLAB (*ang. MATrix LABoratory*) to interaktywne programistyczne środowisko przeznaczone do rozwijania algorytmów, wizualizacji i analizy danych. Rozwiązanie to pozwala również na prowadzenie obliczeń naukowych oraz inżynierskich. MATLAB oferuje możliwość szybszego rozwiązywania problemów, niż przy wykorzystaniu języków programowania [2].

Autorzy MATLAB, bardzo duży nacisk postawili na automatyzację rutynowych procesów, na każdym etapie rozwiązywania danego zadania. Poczynając od gromadzenia danych, poprzez ich analizę i wizualizację, aż do generacji końcowego wyniku w postaci raportu. Rozpoczynając pracę z programem, w bardzo prosty i intuicyjny sposób można zaimportować dane z pliku, bazy danych czy specyficznego sprzętu pomiarowego. Upřednio zaimportowane dane można zwizualizować za pomocą wielu dostępnych funkcji graficznych. W analizie danych mogą pomóc liczne gotowe procedury obliczeniowe. Na zakończenie pracy, środowisko pracy generuje raport z obliczeń [2].

Schemat pracy w programie MATLAB



Rysunek 2.2 Schemat pracy w programie MATLAB [23]

Środowisko MATLAB posiada wiele zalet, do najważniejszych udogodnień tego rozwiązania zalicza się [2]:

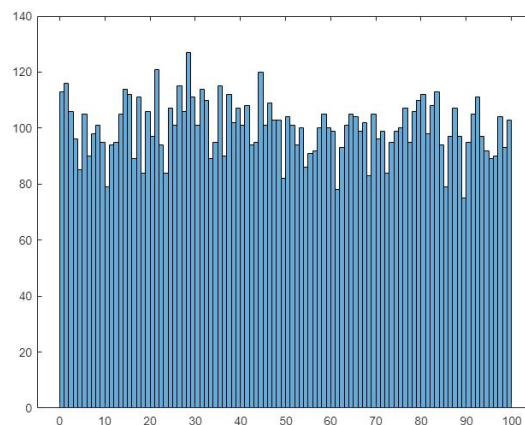
- zintegrowany język programowania wysokiego poziomu;
- narzędzia do importu danych;
- możliwość integracji własnego kodu języka C oraz Java;
- interaktywne narzędzia do wizualizacji i eksploracji danych;
- duża ilość wbudowanych funkcji obliczeniowych.

MATLAB posiada swój własny język programowania, umożliwiający pisanie w pełni funkcjonalnych programów. Korzystając z tego języka można w szybszy sposób tworzyć algorytmy, niż w przypadku innych języków programowania. Dzięki temu użytkownik nie jest obciążony wykonywaniem niskopoziomowych operacji, takich jak deklaracja zmiennych ani określania ich typów. Język ten udostępnia możliwość operacji na macierzach, co eliminuje w wielu przypadkach konieczność tworzenia zagnieżdżonych pętli „for”.

```
columns = 10000;  
rows = 1;  
bins = columns/100;  
rng(now);  
list = 100*rand(rows,columns);  
histogram(list,bins)
```

Listing 1. Przykładowy fragment kodu w języku MATLAB

Powyższy fragment kodu tworzy histogram stu losowo wygenerowanych wartości. Rezultaty uruchomienia powyższego kodu prezentują się następująco.



Rysunek 2.3 Histogram wygenerowany za pomocą programu MATLAB

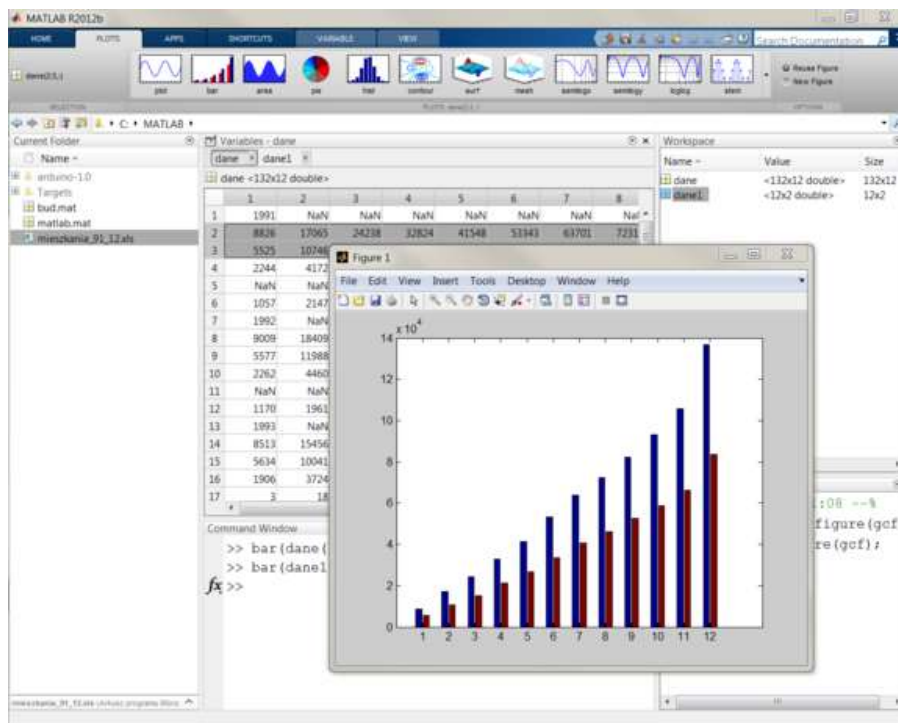
Środowisko MATLAB posiada kilka bardzo przydatnych, wbudowanych narzędzi, które usprawniają pracę z oprogramowaniem, takich jak:

- MATLAB Editor – służy do edycji i debugowania kodu.
- Code Analyzer – sprawdza kod pod kątem występujących błędów i rekomenduje zmiany w kodzie, mające na celu uzyskanie najwyższej jakości.
- MATLAB Profiler – oblicza czas, jaki jest wymagany na wykonanie poszczególnych fragmentów kodu.

Omawiane rozwiązanie umożliwia wyeksportowanie wyników pracy w postaci pojedynczych wykresów, jak i w postaci kompletnego raportu z przeprowadzonych obliczeń. Program obsługuje eksport wyników w formatach takich jak: HTML, Word, LaTeX czy PDF.

MATLAB jest oprogramowaniem posiadającym szerokie zastosowanie. Oto niektóre z przykładowych obszarów zastosowania:

- Przetwarzanie sygnałów;
- Przetwarzanie obrazów;
- Telekomunikacja;
- Projektowanie układów sterowania;
- Matematyka finansowa.



Rysunek 2.4 Interfejs programu MATLAB [24]

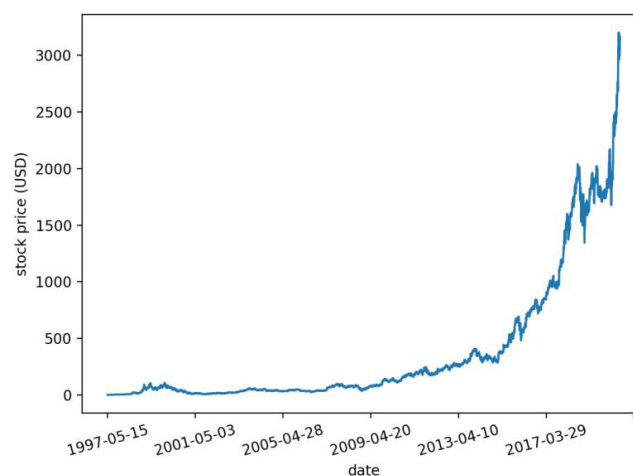
3. OPRACOWANIE WYBRANYCH ZAGADNIĘĆ DO KONSTRUOWANIA OPROGRAMOWANIA TESTUJĄCEGO

W stworzonym oprogramowaniu pojawiają się pojęcia z dziedzin matematyki oraz statystyki. Aby w pełni zrozumieć logikę aplikacji, oraz operacje, które są wykonywane przez oprogramowanie należy zapoznać się z definicjami jak:

- Szereg czasowy;
- Analiza statystyczna szeregów czasowych;
- Metody predykcji szeregów czasowych.

3.1 Szereg czasowy

Szereg czasowy to seria punktów danych zindeksowanych w porządku czasowym. Najczęściej jest on sekwencją wykonywaną w kolejnych, równych odstępach czasu. Zatem jest to sekwencja danych w czasie dyskretnym. Przykładami szeregów czasowych są wartości akcji na giełdzie, temperatura powietrza, monitorowanie ciśnienia krwi itd. Szeregi czasowe są bardzo często przedstawiane za pomocą czasowego wykresu liniowego. Szeregi czasowe są najczęściej wykorzystywane w statystyce, przetwarzaniu sygnałów, rozpoznawaniu wzorców, ekonometrii, finansach matematycznych, prognozowaniu pogody, przewidywaniu trzęsień ziemi, inżynierii sterowania [3].



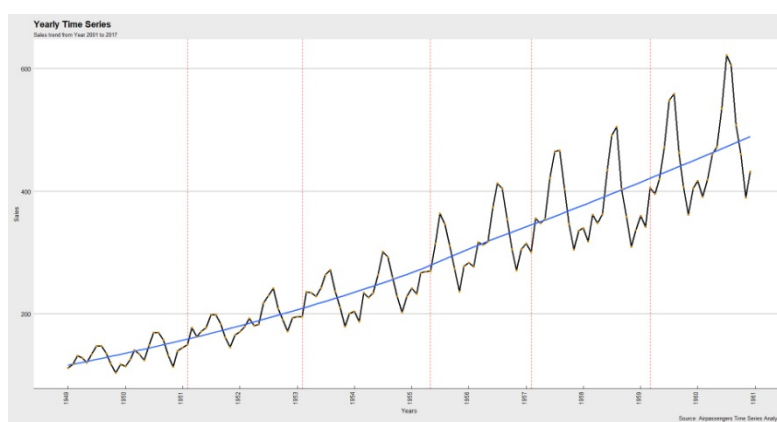
Rysunek 3.1 Wartości akcji firmy Amazon w latach 1997 – 2020

Do oznaczania ciągów czasowych stosowane są różne notacje. Najczęściej ciąg czasowy X indeksowany liczbami naturalnymi jest zapisywany jako [3]:

$$X = \{X_1, X_2, \dots\}$$

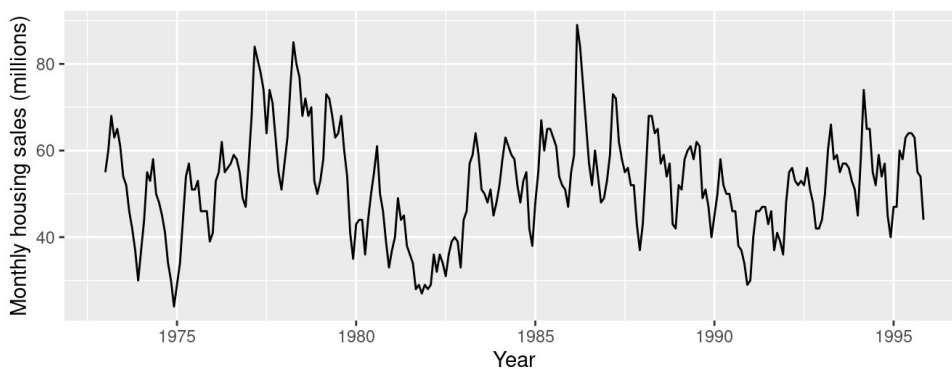
Szeregi czasowe mogą (lecz nie muszą) posiadać własności takie jak [4]:

- **Tendencja rozwojowa (trend)** - czyli długookresowe, systematyczne i jednokierunkowe zmiany wielkości zjawiska. Są wywołane przez stałe przyczyny działające w całym okresie.



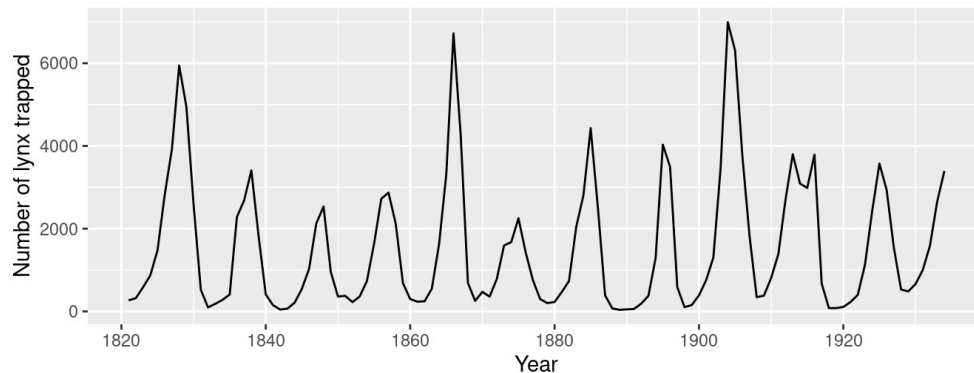
Rysunek 3.2 Trend szeregu czasowego (niebieska linia) [25]

- **Wahania okresowe (wahania sezonowe)** - są to rytmiczne, powtarzające się zmiany poziomu zjawiska, które są wywoływane przez czynniki, np. przyrodnicze. Długość cyklu wahań sezonowych, to liczba okresów składających się na pełen cykl wahań okresowych



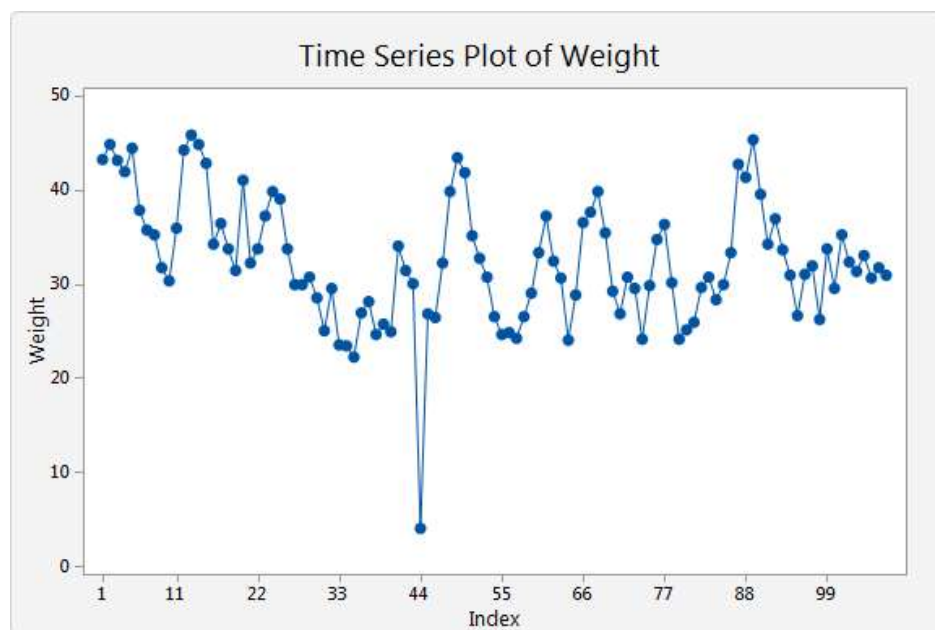
Rysunek 3.3 Szereg czasowy z widoczną sezonowością [26]

- **Wahania cyklu koniunkturalnego (wahania cykliczne)** - to oscylacje, które wymagają wieloletnich obserwacji, posiadające nieregularny charakter i różną długość cyklu. Wyróżnia się okresy recesji, depresji, ożywienia i dobrobyt. Brakuje ugruntowanej statystycznej metody ich analizy oraz ich przebieg jest trudny do przewidzenia. Są to systemowe, falowe wahania rozwoju gospodarki obserwowane w okresach dłuższych od roku i ich analiza wymaga wieloletnich obserwacji.



Rysunek 3.4 Szereg czasowy z widocznymi wahaniami cyklicznymi [27]

- **Wahania przypadkowe** - czyli nieregularne odchylenia wielkości zjawiska od poziomu, którego spodziewalibyśmy się na podstawie oddziaływania innych czynników (trendu, wahań sezonowych i cyklicznych). Ich natura jest trudna do zdefiniowania. Mogą być wywoływane przez przypadkowe różnokierunkowe działania czynników, które nie są uwzględnione w analizie.



Rysunek 3.5 Szereg czasowy z widocznym losowym odchyleniem [28]

3.2 Analiza statystyczna szeregów czasowych

Analiza statystyczna to proces zbierania i analizowania danych w celu zidentyfikowania wzorców i trendów. Jest to metoda wykorzystująca metody matematyczne w celu usunięcia wszelkich uprzedzeń podczas przeglądania informacji. Można ją również traktować jako narzędzie, które może wpływać na podejmowanie decyzji [5].

W większości przypadków (zestawów danych), podczas przeprowadzania analizy statystycznej można wyodrębnić pięć poniższych kroków [5]:

1. Opisanie charakteru analizowanych danych.
2. Zbadanie relacji danych do bazowej populacji.
3. Utworzenie modelu, aby podsumować relacje między danymi a populacją bazową.
4. Udowodnienie (lub obalenie) słuszności modelu.
5. Wykorzystanie wyników do analiz predykcyjnych, aby przewidywać przyszłe trendy.

W przypadku szeregów czasowych, niemożliwe jest zastosowanie typowej analizy statystycznej, albowiem nie możliwe jest wykonanie zbadanie relacji pomiędzy danymi. Szereg czasowy posiada tylko jeden rodzaj zmiennych, których wartość jest zmienna w dziedzinie czasu.

Na szeregach czasowych, w ramach analizy statystycznej, można wykonać takie operacje jak:

- Preprocessing danych

Proces polegający na wstępnej obróbce danych. W tym procesie, na szeregu czasowym, wykonywane są metody:

- Walidacji danych - sprawdzenie poprawności typów danych. Najczęściej polega na sprawdzeniu czy wartości szeregu czasowego to liczby całkowite, lub zmiennoprzecinkowe.
- Sprawdzenie komplementarności danych – sprawdzenie, czy szereg nie posiada brakujących wartości. Sprawdzenie to dotyczy się zarówno wartości szeregów, jak i znaczników czasowych w dziedzinie czasu. Jeśli jedna z próbek (pomiarów) nie posiada wartości, jako wartość tej próbki przyjmowana jest uśredniona wartość sąsiednich pomiarów. Jeśli natomiast brakuje znacznika czasowego w próbce, jest ona usuwana z szeregu czasowego.

- Statystyka opisowa

Polega na wykonaniu różnego rodzaju wielkości obliczane na podstawie uzyskanych danych. Interpretacja wartości tych miar dostarcza informacji na temat charakteru rozkładu cechy. Wykonywane są przykładowo takie obliczenia jak:

- Wyznaczenie minimalnej i maksymalnej wielkości szeregu czasowego.
- Obliczenie wartości średniej arytmetycznej – obliczenie średniej arytmetycznej wszystkich wartości szeregu czasowego zgodnie ze wzorem:

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n}$$

- Obliczenie wartości odchylenia standardowego - obliczenie odchylenia standardowego wartości szeregu czasowego zgodnie ze wzorem:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

- Wyznaczenie mediany – aby obliczyć medianę, należy posortować wszystkie wartości szeregu czasowego w sposób niemalejący. Jeśli liczba obserwacji jest liczbą nieparzystą, mediana jest wartością znajdującą się „pośrodku”, czyli próbka o numerze $\frac{n+1}{2}$, natomiast w przeciwnym wypadku, gdy liczba próbek jest liczbą parzystą, mediana jest średnią arytmetyczną między dwoma środkowymi obserwacjami, czyli obserwacją $\frac{n}{2}$ oraz $\frac{n+1}{2}$.
- Wyznaczenie kwantyla – wyznaczenie kwantyla rzędu q ($0 < q < 1$), który jest liczbą Q , której 100% elementów danej zbiorowości statystycznej ma wartość nie większą niż Q .
- Wyznaczenie interkwantyla – obliczenie różnicy między trzecim a pierwszym kwantylem. Pierwszy kwantyl (dolny kwantyl), to kwantyl rzędu $\frac{1}{4}$, natomiast trzeci kwantyl, to kwantyl trzeciego rzędu (kwantyl górny) – kwantyl rzędu $\frac{3}{4}$.

- Identyfikacja punktów oddalonych

Polega na wyznaczeniu poszczególnych punktów w zbiorze danych (a w tym przypadku szeregu czasowym), których wartość jest mniejsza niż różnica pierwszego kwantyla i ilorazu liczby 1,5 oraz rozstępu międzykwantylowego lub większa, niż suma trzeciego kwantyla i ilorazu liczby 1,5 oraz rozstępu międzykwantylowego.

- Autokorelacja

Współczynnik korelacji między dwiema wartościami szeregu czasowego nazywany jest współczynnikiem autokorelacji. Współczynnik ten dla szeregu czasowego y_t , można przedstawić jako:

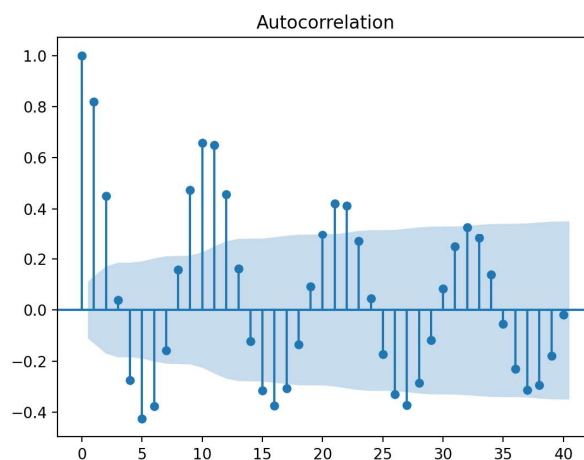
$$\text{Corr}(y_t, y_{t-k})$$

Wartość k , zwana opóźnieniem, przedstawia przerwę czasową. Autokorelacja z opóźnieniem (k) równym 1, jest korelacją pomiędzy wartościami, których różni jeden okres czasu (jedną jednostkę czasu w szeregu czasowym). Autokorelacja z opóźnieniem k , jest korelacją między wartościami, które są oddalone od siebie o k okresów czasu. Wyznaczenie współczynnika autokorelacji jest sposobem pomiaru liniowej zależności między obserwacjami w czasie t , a obserwacjami z poprzednich czasów [6]. Współczynnik autokorelacji można obliczyć posługując się wzorem [7]:

$$\rho_m = \frac{\sum_{n=1}^N [(x_n - \bar{x}) \cdot (x_{n-k} - \bar{x})]}{\sum_{n=1}^N (x_n - \bar{x})^2}$$

ρ_m - to współczynnik korelacji, k - opóźnienie, n - liczba obserwacji, x - kolejne obserwacje, x_{n-k} kolejne obserwacje opóźnione o k , \bar{x} - średnia obserwacji.

Jeśli współczynnik autokorelacji jest bliski wartości 1, między obserwowanymi wartościami zachodzi silna korelacja dodatnia, jeśli bliski wartości -1, silna korelacja ujemna, natomiast gdy współczynnik ten jest bliski zeru, brak korelacji, lub jest ona niewielka.



Rysunek 3.6 Wykres autokorelacji szeregu czasowego [29]

3.3 Metody predykcji szeregów czasowych

Predykcja (prognoza) szeregów czasowych polega na przewidywaniu przyszłych wartości szeregów czasowych, na podstawie danych historycznych. Prognozowanie szeregów czasowych to czasami tylko analiza ekspertów badających dziedzinę i oferujących swoje prognozy. Jednak w wielu nowoczesnych aplikacjach prognozowanie szeregów czasowych wykorzystuje technologie komputerowe, w tym:

- Uczenie maszynowe;
- Sztuczne sieci neuronowe;
- Maszyna wektorów nośnych;
- Ukryte modele Markova.

Jednak w ramach stworzonego oprogramowania, do predykcji przyszłych wartości zostały wykorzystane zostały modele statystyczne takie jak model autoregresyjny, oraz model autoregresyjny zintegrowany ze średnią ruchomą.

3.3.1 Model autoregresyjny (AR)

Model autoregresyjny (*ang. autoregressive model, AR model*) to parametryczny model szeregu czasowego, który wykorzystywany jest do modelowania i predykcji zjawisk naturalnych, opisanych za pomocą szeregów czasowych. Model autoregresyjny jest jedną z formuł predykcji liniowej – takie formuły dokonują predykcji wyjścia układu w oparciu o wartości na wejściu modelu (danych z przeszłości).

Model autoregresyjny można przedstawić za pomocą notacji **AR(p)**, która wskazuje na to, iż chodzi o model autoregresyjny rzędu **p**. Notacja ta prezentuje się następująco [8]:

$$X_t = C + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t$$

gdzie:

- x_t - wartość szeregu czasowego dla obserwacji t
- $\varphi_1, \dots, \varphi_p$ - parametry modelu;
- C - stała (dla uproszczenia formuły, często pomijana);
- ε_t – szum biały.

Modele autoregresyjne są niezwykle elastyczne w obsłudze szerokiej gamy szeregów czasowych o różnych właściwościach. Model autoregresyjny określa, że zmienna wyjściowa zależy liniowo od swoich wcześniejszych wartości i od składnika stochastycznego zatem model ma postać stochastycznego równania różnicowego (lub relacji rekurencyjnej, której nie należy mylić z równaniem różniczkowym) [8].

Najprostszym przykładem jest model autoregresyjny pierwszego rzędu **AR(1)** (inaczej nazywany liniowym procesem Markova), w którym wartość w danej chwili zależy wyłącznie od wartości w chwili poprzedniej i szumu. $X_t = \varphi_1 X_{t-1} + \varepsilon_t$ [9].

Podstawiając trzy kolejne wyrazy.

$$X_t = \varepsilon_t + \varphi X_{t-1}$$

$$X_{t-1} = \varepsilon_{t-1} + \varphi X_{t-2}$$

$$X_{t-2} = \varepsilon_{t-2} + \varphi X_{t-3}$$

$$\begin{aligned} X_t &= \varepsilon_t + \varphi X_{t-1} = \varepsilon_t + \varphi(\varepsilon_{t-1} + \varphi X_{t-2}) = \varepsilon_t + \varphi(\varepsilon_{t-1} + \varphi(\varepsilon_{t-2} + \varphi X_{t-3})) = \\ &= \varepsilon_t + \varphi \varepsilon_{t-1} + \varphi^2 \varepsilon_{t-2} + \varphi^3 X_{t-3} \end{aligned}$$

Bardzo ważną kwestią związaną z zastosowaniem modeli autoregresyjnych jest ustalenie odpowiedniego rzędu autoregresji. Do najbardziej popularnych metod ustalania rzędu autoregresji należą kryteria informacyjne takie jak:

- Kryterium Informacyjne Akaike'a (AIC)

Według tego kryterium, najskuteczniejszy jest model, którego najmniejsza jest wartość AIC, gdzie formuła AIC prezentuje się następująco [10]:

$$AIC = -2 \sum_j \ln \hat{\pi}_j + 2q$$

gdzie:

- $\hat{\pi}_j$ - estymowane prawdopodobieństwo uzyskania takiej właśnie wartości obserwacji **j** jaka była naprawdę uzyskana;
- q - liczba parametrów modelu.

- Bayesowskie kryterium informacyjne Schwarza (BIC)

W przypadku tego modelu wybiera się model, którego najmniejsza jest wartość [11]:

$$BIC = K \ln(N) - 2 \ln L(\theta)$$

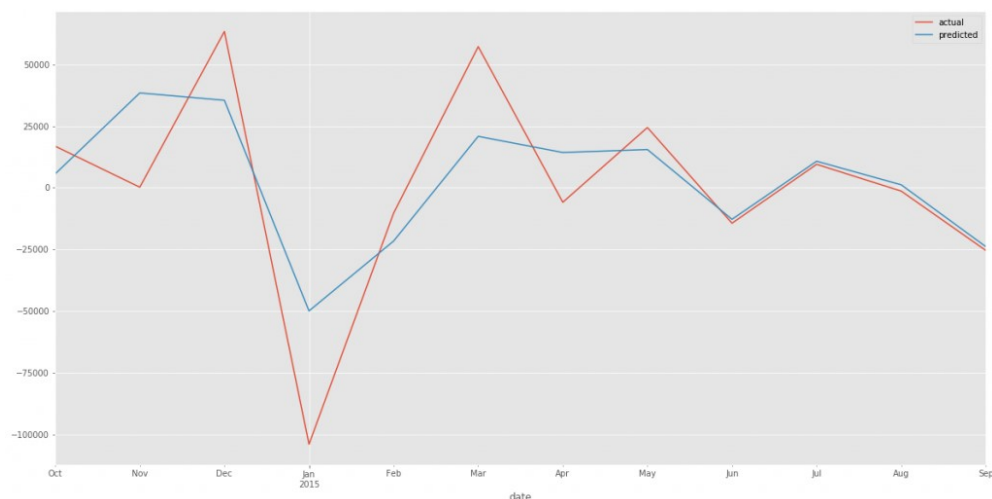
gdzie:

- N - liczba obserwacji;
 - K - liczba parametrów modelu;
 - $L(\theta)$ - funkcja wiarygodności dla oszacowanego wektora parametrów.
 - Kryterium informacyjne Hannana-Quinna (HQC)
- Według kryterium Hannana-Quinna, najbardziej precyzyjny jest model którego najniższa jest wartość [12]:

$$HQC = n \ln \left(\frac{RSS}{n} \right) + 2k \ln \ln n$$

gdzie:

- k - liczba parametrów;
- n - liczba obserwacji;
- RSS – suma kwadratów reszt wynikająca z przeprowadzonej regresji liniowej bądź też innego modelu.



Rysunek 3.7 Przykładowa prognoza szeregu czasowego z wykorzystaniem AR (pomarańczowa linia oznacza prawdziwe wartości, niebieska natomiast wartości estymowane przez model) [30]

3.3.2 Średnia ruchoma (MA)

Kolejną z metod, która jest ważnym elementem procesu predykcji przyszłych wartości szeregów czasowych, jest średnia ruchoma (*ang. moving average*). Jest to statystyczna metoda, za pomocą której można wyznaczyć wartości poszczególnych punktów poprzez obliczenie średnich z różnych podzbiorów pełnego zestawu danych. Nazywana jest również średnią kroczącą (*ang. moving mean*). Średnią ruchomą stosuje się głównie w celu wygładzania krótkoterminowych wahań. Wśród średnich ruchomych wyróżnia się takie odmiany jak:

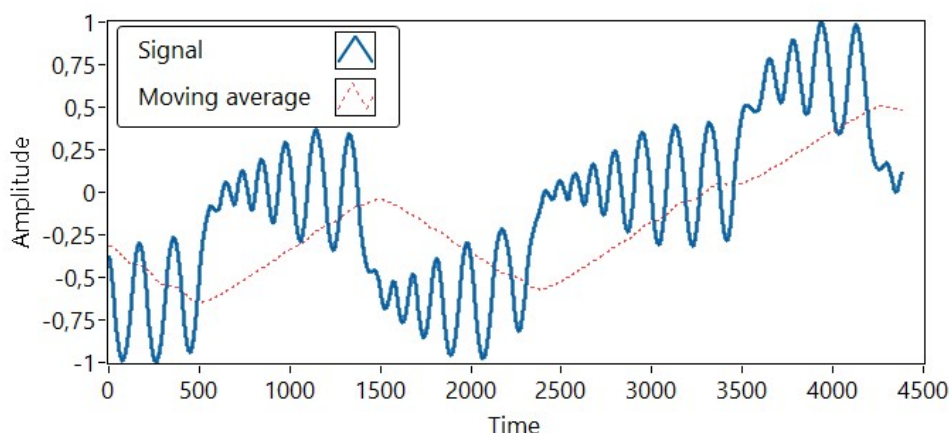
- Prosta średnia ruchoma;
- Ważona średnia ruchoma;
- Wykładnicza średnia ruchoma.;
- Skumulowana średnia ruchoma.

Biblioteki wykorzystane w tworzeniu oprogramowania wykorzystują prostą średnią ruchomą. Prosta średnia ruchoma SMA (*ang. simple moving average*) to zwykła średnia arytmetyczna wartości z ostatnich n okresów. p_0 oznacza ostatnią wartość [13].

$$SMA = \frac{p_0 + p_1 + \dots + p_{n-1}}{n}$$

gdzie:

- n – liczba obserwacji
- p_n - poszczególna obserwacja



Rysunek 3.8 Wygładzenie przebiegu z wykorzystaniem średniej ruchomej [31]

3.3.3 Model autoregresyjny zintegrowany ze średnią ruchomą (ARIMA)

Model ARIMA (*ang. autoregressive integrated moving average*) to model statystyczny wykorzystujący szereg czasowy dla lepszego zrozumienia danych, oraz prognozowania jego przyszłych wartości. Model ten jest połączeniem trzech elementów, takich jak [14]:

- Autoregresja (AR) – odnosi się do modelu, który pokazuje zmieniającą się zmienną, ulegającą regresji według własnych, opóźnionych (poprzednich) wartości.
- Zintegrowanie (I) – reprezentuje szeregi czasowe, które zostały zróżnicowane tak, aby stały się one stacjonarne.
- Średnia ruchoma (MA) – zawiera zależność między obserwacją a błędem modelu średniej ruchomej zastosowanego do opóźnionych obserwacji.

Każdy z elementów w modelu ARIMA funkcjonuje jako parametr ze standardową notacją. W przypadku tego modelu są to parametry p , q i d . Każdy z parametrów jest wielkością całkowitą, która oznacza [14]:

- p – to liczba określająca rząd predykcji w komponencie autoregresji. Wartość tego parametru definiuje ile poprzednich wartości w szeregu czasowym będzie brane pod uwagę w predykcji kolejnych wartości.
- q - określa stopień zróżnicowania, wskazuje, ile razy wartości szeregu czasowego zostają opóźnione (odjęte) w celu przekształcenia szeregu czasowy w stacjonarny.
- d - oznacza liczbę błędów prognozy w modelu i jest również określany jako rozmiar okna średniej ruchomej.

Jeżeli wartość któregoś z parametrów zostanie ustawiona na 0, będzie to oznaczało, iż komponent ten nie zostanie wykorzystany. W ten sposób, model ARIMA może zostać przekształcony w dowolną kombinację komponentów modelu.

Komponenty modelu ARIMA, takie jak autoregresja oraz średnia ruchoma zostały omówione w poprzednich podrozdziałach. Nie został natomiast wspomniany komponent integracji szeregów czasowych. Do prognozy szeregów czasowych wykorzystywane są również modele takie jak ARMA. Lecz ich zastosowanie ma jedynie miejsce w przypadku, gdy szereg czasowy jest stacjonarny. Model ARIMA, który jest rozszerzeniem modelu ARMA o dodatkowy komponent integracji, może zostać zastosowany w przypadku szeregów

niestacjonarnych, pod warunkiem, iż szereg czasowy na którym operuje model, może zostać sprowadzony do postaci stacjonarnej.

Szereg czasowy jest wtedy stacjonarny, kiedy ma stały poziom w czasie (średnią) oraz odchylenie standardowe od średniego poziomu (wariancję). Stacjonarność szeregu można sprawdzić za pomocą wykresu autokorelacji: szeregi niestacjonarne mają około 6 opóźnień istotnie różnych od zera. Jedną z metod sprowadzenia szeregu czasowego do stacjonarności jest jego zróżnicowanie (*ang. differencing*). Zróżnicowanie polega na obliczeniu różnic pomiędzy kolejnymi wartościami. Pierwsze z nich można oznaczyć jako:

$$\Delta y_t = y_t - y_{t-1}$$

Różnice między tymi wartościami będą reprezentować spadki i wzrosty wartości w szeregu czasowym. Tak więc przykładowy szereg może mieć przebieg wykazujący trend, a po zróżnicowaniu otrzymane zostaną wartości, które w większości przypadków będą stacjonarne. Istnieją przypadki, gdzie zróżnicowanie pierwszego stopnia nie jest wystarczające do uzyskania przebiegu stacjonarnego. To którego stopnia różnice obliczane są w procesie zróżnicowania, definiuje wcześniej wspomniany parametr **d** w modelu ARIMA [15].

Dotychczas zostały omówione wszystkie komponenty modelu ARIMA. Biorąc pod uwagę wszystkie elementy, model ten można opisać za pomocą następującej formuły [16].

$$Y_c = c + \phi_1 Y_{t-1} + \dots + \phi_p Y_{t-p} + \theta_1 e_{t-1} + \dots + \theta_q e_{t-q} + e_t$$

- Y_t = zmienna objaśniana w dziedzinie czasu;
- c = stała (dla uproszczenia formuły, często pomijana);
- ϕ = współczynnik każdego parametru p ;
- θ = współczynnik każdego parametru q ;
- e_t = szum biały.

4. CHARAKTERYSTYKA WYBRANEGO DO REALIZACJI ZADANIA OPROGRAMOWANIA

Głównym założeniem tworzonego oprogramowania było zapoznanie użytkownika z pojęciem szeregów czasowych oraz przedstawienie możliwości metod analitycznych oraz eksploracyjnych, jakie można na owych szeregach czasowych wykonywać. Stworzona aplikacja umożliwia przesłanie przez użytkownika własnego pliku w formacie CSV, zawierającego szereg czasowy. Jeśli plik przesłany przez użytkownika jest poprawnego formatu, dane w nim zawarte podlegają natychmiastowo analizie statystycznej oraz wizualizacji. Osoba korzystająca z aplikacji posiada możliwość metody prognozy przyszłych wartości szeregów czasowych oraz parametryzacji metody prognozującej. Jako rezultat wykonania metody predykcyjnej, wygenerowany zostaje prognozowany przebieg w postaci wykresu liniowego, wraz z współczynnikami błędu predykcji.

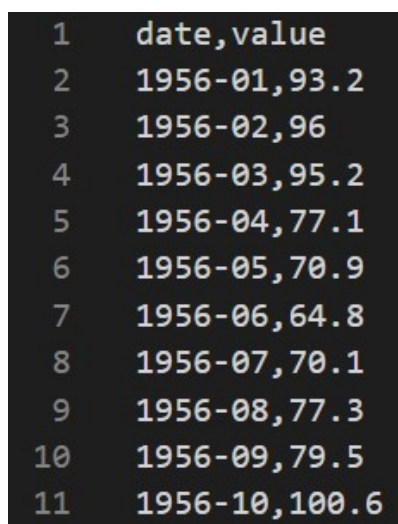
4.1 Opis funkcjonalności oprogramowania

Ten podrozdział jest poświęcony szczegółowym opisom poszczególnych funkcjonalności oprogramowania. Na podstawie owych charakterystyk zostanie zaprojektowana architektura systemu, oraz zaimplementowane zostaną poszczególne jego funkcjonalności. Z racji tego, iż oprogramowanie to poświęcone jest głównie szeregom czasowym, omówione zostaną takie funkcje programu jak:

- Przesyłanie pliku;
- Analiza statystyczna oraz wizualizacja szeregu czasowego;
- Prognozowanie szeregów czasowych.

4.1.1 Przesyłanie pliku

Jednym z udogodnień tworzonego oprogramowania jest możliwość przesłania przez użytkownika do aplikacji dowolnego pliku w formacie CSV, w którym znajduje się szereg czasowy. W przesłanym przez użytkownika pliku, powinny zawarte być dwie kolumny. W jednej z nich powinny znajdować się znaczniki czasowe w postaci ciągu znaków. Mogą one zawierać datę w dowolnym formacie, lub inny znacznik czasu poszczególnych obserwacji. Druga z kolumn natomiast, powinna przechowywać wartości zmiennej której dotyczy obserwacja. Notowania te, powinny być wartościami liczbowymi całkowitymi lub zmiennoprzecinkowymi. Kolejność kolumn nie ma znaczenia, ważne natomiast jest, żeby w pierwszym wierszu, w kolumnie przechowującej znaczniki czasu znajdował się ciąg znaków z wartością „date”, natomiast w kolumnie zawierającej wartości, ciąg znaków o wartości „value”. Kolumny te powinny być rozdzielone przecinkiem. Poniżej przedstawiony jest początek zawartości przykładowego pliku CSV o poprawnym formacie.



```
1 date,value
2 1956-01,93.2
3 1956-02,96
4 1956-03,95.2
5 1956-04,77.1
6 1956-05,70.9
7 1956-06,64.8
8 1956-07,70.1
9 1956-08,77.3
10 1956-09,79.5
11 1956-10,100.6
```

Rysunek 4.1 Przykładowa zawartość pliku CSV zawierającego szereg czasowy

Przesyłanie pliku powinno odbywać się na pomocą prostego formularza. Po naciśnięciu odpowiedniego przycisku przez użytkownika, powinno ukazać się okno eksploratora, z możliwością wskazania przez niego pliku z dowolnej lokalizacji na dysku, do której ma dostęp. Oprogramowanie powinno być zabezpieczone przed możliwością przesłania przez użytkownika pliku o formacie innym niż CSV. Jednocześnie użytkownik zostanie poinformowany przed przesłaniem pliku, o wymaganiach co do jego formatu oraz zaleceniach w kwestii poprawności zawartych w nim danych.

4.1.2 Analiza statystyczna oraz wizualizacja szeregu czasowego

Przesłany przez użytkownika plik z szeregiem czasowym, który spełnia wspomniane w poprzednim podrozdziale wymagania zostaje natychmiast poddany metodom analizy statystycznej oraz wizualizacji. Jest to kolejna, z ważnych funkcjonalności oferowanych przez tworzone oprogramowanie. Metody statystyczne, którym podlegają dane dostarczone w pliku, są bardziej szczegółowo opisane w rozdziale 3.2 *Analiza statystyczna szeregów czasowych*.

Przed właściwą analizą danych zawartych w pliku, szereg czasowy zostaje zweryfikowany pod kątem:

- komplementarności danych – sprawdzenie czy w żadnej z kolumn, nie występuje brakująca wartość.
- walidacji danych – czy w kolumnach znajdują się dane poprawnego typu.
- unifikacji danych – dotyczy wyłącznie wartości szeregu czasowego, polega na zaokrągleniu wszystkich wartości szeregu do tej samej liczby miejsc po przecinku.

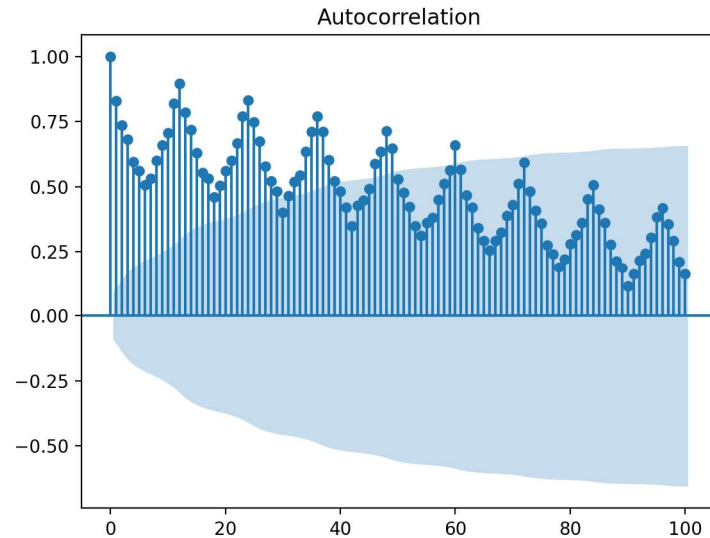
Po zweryfikowaniu danych oraz wykonaniu podstawowych metod statystycznych, jako rezultat opisywanej funkcjonalności, użytkownikowi wyświetli się tabela z wartościami takimi jak:

- Wartość minimalna;
- Wartość maksymalna;
- Średnia arytmetyczna wartości;
- Mediana wartości;
- Odchylenie standardowe;
- Interkwartyl.

Również w tym samym widoku, obok tabeli z danymi statystycznymi, użytkownik powinien ujrzeć wizualizację szeregu czasowego w postaci wykresów:

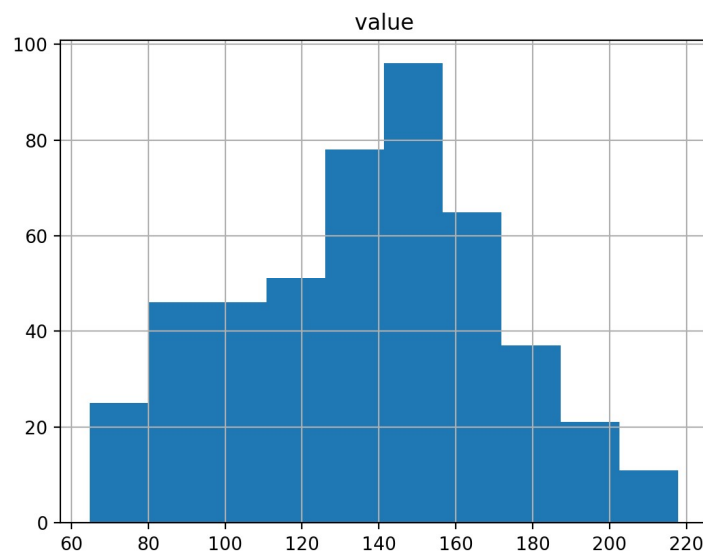
- Wykresu przebiegu szeregu czasowego – wykres przedstawiający cały cykl przebiegu wykresu czasowego. Na osi X opisane są tu znaczniki czasowe reprezentujące przedziały pomiaru, natomiast na osi Y opisane są wartości obserwacji. Przykładowy wykres znajduje się na rysunku 3.1.

- Wykresu autokorelacji – wykres przedstawiający wartości współczynników autokorelacji pomiędzy 100 poprzednimi wartościami. Wartości te zawierają się w przedziale od -1.0 to 1.0.



Rysunek 4.2 Przykładowy wykres autokorelacji szeregu czasowego dla 100 poprzednich wartości

- Histogramu rozkładu zmiennej – wykres przedstawiający liczebność obserwacji w zadanych przedziałach.



Rysunek 4.3 Przykładowy histogram rozkładu zmiennej szeregu czasowego

4.1.3 Prognozowanie szeregów czasowych

Główną funkcjonalnością tworzonego systemu, jest możliwość testowania wybranych metod predykcji przez użytkownika, na przesłanym przez niego szeregu czasowym. Użytkownik ma do wyboru dwa modele umożliwiające prognozowanie przyszłych wartości szeregów czasowych, model autoregresyjny, oraz zintegrowany model autoregresyjny ze średnią ruchomą. Modele te zostały bardziej szczegółowo opisane w rozdziale 3.3 *Metody predykcji szeregów czasowych*. Testowanie działania metod prognozujących polega na wytrenowaniu modelu za pomocą zadanej ilości obserwacji, a następnie zweryfikowaniu jego skuteczności poprzez porównanie wyników prognozy z rzeczywistym przebiegiem. Użytkownik ma możliwość parametryzacji obu tych metod. Poprzez odpowiedni dobór wartości parametrów, może optymalizować model w celu zwiększenia jego wydajności oraz zminimalizowania błędu predykcji. Parametryzacja odbywa się poprzez formularz w którym możliwe jest wprowadzenie wartości liczbowej parametru, lub wybór jednej z wielu możliwych opcji.

W przypadku modelu autoregresyjnego, możliwe jest ustawienie wartości takich parametrów jak:

- Stosunek podziału zbioru wartości – parametr ten jest wartością zmiennoprzecinkową definiującą w jakich proporcjach zbiór wartości szeregu czasowego ma zostać podzielony na podzbiór treningowy oraz testowy. Przykładowo jeśli użytkownik dostarczy szereg czasowy o liczbie obserwacji równej 300, oraz jako wartość tego parametru poda 0.6, to model zostanie wytrenowany na 180 obserwacji, natomiast pozostałe 120 posłużą przy testowaniu jego skuteczności. Jako domyślna wartość tego parametru zostanie przyjęta 0.8.
- Kryterium doboru parametru opóźnienia – parametr definiujący, według którego kryterium ma zostać dobrana wartość rzędu predykcji (ilość poprzednich obserwacji, na podstawie których zostanie wykonana prognoza). Dla autoregresyjnych modeli dostępne są kryteria takie jak:
 - Kryterium Informacyjne Akaike’a (AIC);
 - Bayesowskie kryterium informacyjne Schwarza (BIC);
 - Kryterium informacyjne Hannana-Quinna (HQC).

Ich matematyczne formuły znajdują się w rozdziale 3.3.1 *Model autoregresyjny*.

W przypadku zintegrowanego modelu autoregresyjnego ze średnią ruchomą, użytkownik może dostosowywać parametry takie jak:

- Stosunek podziału zbioru wartości – definicja tego parametru jest taka sama, jak w przypadku parametru modelu autoregresyjnego, opisanego na poprzedniej stronie.
- Parametr q – wartość tego parametru oznacza rząd predykcji. Domyślna wartość tego parametru to 10.
- Parametr d – stopień zróżnicowania szeregu czasowego. Domyślnie
- Parametr p – określa rozmiar okna średniej ruchomej.

Przeznaczenie oraz bardziej szczegółowe znaczenie powyższych parametrów znajduje się w rozdziale 3.3.3 *Model autoregresyjny zintegrowany ze średnią ruchomą (ARIMA)*.

Jako wynik metody prognozującej, użytkownikowi zostanie wygenerowany widok, na którym znajdować się będzie zestawienie prognozowanych wartości wraz z realnym przebiegiem w postaci wykresu liniowego, ilość obserwacji którymi model był trenowany oraz rezultaty predykcji, zależne od zastosowanej metody.

Jeśli jako metoda prognozująca, wybrana została metoda autoregresji, wynikiem będzie:

- Rząd predykcji, który został wybrany na podstawie kryterium wskazanego przez użytkownika.
- Wartość błędu średniokwadratowego (RMSE) – określa skuteczność modelu prognozującego. Im mniejsza wartość błędu RMSE, tym prognozy modelu były bardziej precyzyjne. Błąd RMSE określa się wzorem:

$$RMSE = \sqrt{(f - o)^2}$$

gdzie:

- f - wielkość prognozowana;
- o - wielkość rzeczywista.

Jeśli natomiast prognoza przeprowadzona była za pomocą modelu ARIMA, wartościami które określają błąd predykcji, będą wyniki formuł matematycznych kryteriów wspomnianych w przypadku omawiania parametrów modelu autoregresyjnego, czyli: Kryterium Informacyjnego Akaike'a (AIC), Bayesowskiego kryterium informacyjnego Schwarza (BIC) oraz Kryterium informacyjnego Hannana-Quinna (HQC).

4.2 Wykorzystane technologie

Oprogramowanie dydaktyczne, będące tematem pracy, zostało stworzone w formie aplikacji internetowej. Dostęp do aplikacji możliwy jest z każdego urządzenia posiadającego przeglądarkę internetową, oraz dostęp do internetu. Najbardziej istotne języki programowania oraz narzędzia które zostały wykorzystane przy tworzeniu rozwiązania to:

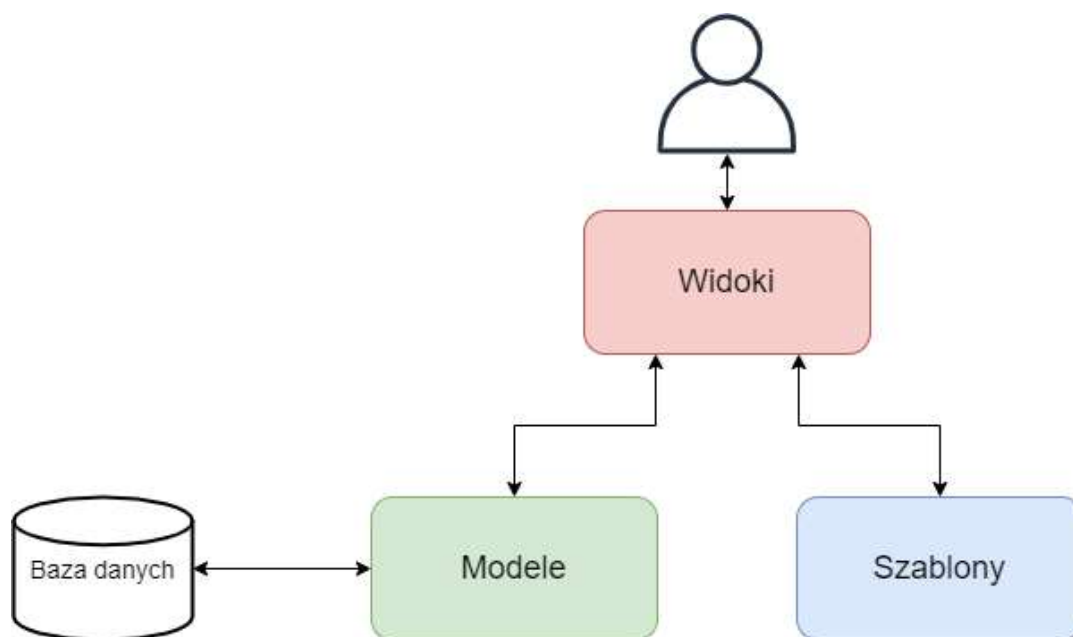
- Język Python w wersji 3.6.8 - interpretowany, zorientowany obiektowo język programowania. Dzięki wysokopoziomym, wbudowanym strukturą danych oraz dynamicznemu typowaniu jest on idealnym językiem programowania do szybkiego i prostego tworzenia aplikacji. Język Python obsługuje wiele modułów, pakietów oraz bibliotek, co zachęca do tworzenia modułowego programu i ponownego wykorzystania tworzonego kodu. Python jak i jego standardowa biblioteka są dostępne bezpłatnie w postaci kodu źródłowego lub binarnego na wszystkich platformach [17]. Cała logika biznesowa tworzonego systemu została zaprogramowana właśnie w tym języku.
- Micro-framework Flask w wersji 1.1.2 – microframework języka Python, zapewniający narzędzia oraz biblioteki, które umożliwiają szybkie i proste budowanie aplikacji internetowych. Flask jest uważany za microframework, ponieważ nie zawiera on w sobie zależności do zewnętrznych bibliotek, co ogranicza jego bazową funkcjonalność, jednocześnie tworząc go lekkim i łatwym w instalacji. Rozwiązanie to posiada dużą ilość wtyczek dostarczających dowolną funkcjonalność do tworzonych aplikacji [18]. Biblioteka ta została wykorzystana w projekcie jako rdzeń aplikacji, w którym odbywały się wszystkie obliczenia, a ich wynikiem były renderowane strony internetowe.
- Biblioteka Matplotlib w wersji 3.3.2 – najbardziej popularna biblioteka w języku Python wykorzystywana do wizualizacji danych. Jest to międzyplatformowa biblioteka do tworzenia 2 lub 3 wymiarowych wykresów bazując na danych zawartych w tablicach. Główną zaletą tego rozwiązania jest możliwość wykorzystania jej możliwości na dowolnym systemie operacyjnym [19]. Biblioteka Matplotlib posłużyła do wizualizacji przebiegu szeregów czasowych oraz wykresów z nimi związanymi.
- Biblioteka Statsmodels w wersji 0.10.2 – to moduł języka Python, udostępniający klasy i funkcje do estymacji różnych modeli statystycznych, a także do przeprowadzania testów statystycznych i statystycznej eksploracji danych. Biblioteka ta została użyta w stworzonym systemie do tworzenia modeli predykcyjnych [20].

5. PROJEKT OPROGRAMOWANIA DYDAKTYCZNEGO DO TESTOWANIA DZIAŁANIA WYBRANYCH METOD DLA MODELI SZEREGÓW CZASOWYCH

5.1 Architektura oprogramowania

Ważną kwestią podczas tworzenia oprogramowania był dobór odpowiedniej architektury aplikacji. Wybór odpowiedniej architektury ma istotny wpływ na łatwość w rozwoju i utrzymaniu oprogramowania.

Z racji tego, iż do stworzenia serwera aplikacji, został wykorzystany micro-framework Flask, nie została odgórnie narzucona architektura aplikacji. Spośród wielu dostępnych architektur, wybrana została architektura MVT (ang. *Model View Template*). Struktura ta, najlepiej wpasowuje się w potrzeby tworzonego oprogramowania. Architektura ta dzieli poszczególne warstwy aplikacji, co umożliwia modyfikację dowolnej z nich, bez obawy o działanie pozostałych.



Rysunek 5.1 Schemat architektury aplikacji MVT

Można zauważyć, że na powyższym rysunku widoczny jest komponent bazy danych. Z racji tego, iż w tworzonej aplikacji nie ma potrzeby wykorzystania żadnego z systemów baz danych, komponent ten zostanie zastąpiony danymi znajdującymi się w pliku przesłanym przez użytkownika.

Jak zaprezentowane zostało na rysunku 5.1 *Schemat architektury aplikacji MVT*. Architektura MVT składa się z trzech następujących komponentów [21]:

Modeli (ang. *models*)

Warstwa ta odpowiada za operacje wykonywane na modelach danych, które wykorzystywane są w aplikacji. W większości przypadków, dane mają swoje odzwierciedlenie w bazach danych.

Jednak jak zostało wspomniane na poprzedniej stronie, komponent bazy danych został zastąpiony w tym przypadku plikami przesłanymi przez użytkownika, które są przechowywane po stronie serwera. W przypadku tworzonego oprogramowania, w warstwie modeli będą znajdować się klasy reprezentujące szereg czasowy oraz klasy pochodzące z bibliotek zewnętrznych, będące reprezentacją statystycznych modeli predykcyjnych.

Widoków (ang. *views*)

W warstwie tej zawarta jest cała logika biznesowa aplikacji. Widoki pośredniczą pomiędzy dwoma pozostałymi warstwami. Metody znajdujące się w tej warstwie operują na danych pobranych z warstwy modeli, wykonują zaimplementowaną logikę a następnie przekazują wyniki operacji do warstwy szablonów.

W tworzonym rozwiązaniu zaimplementowane zostaną cztery widoki, każdy z nich będzie korespondować z jednym z szablonów:

- Widok strony głównej
- Widok analizy statystycznej
- Widok wyników predykcji modelu autoregresyjnego (AR)
- Widok wyników modelu autoregresyjnego zintegrowanego ze średnią ruchomą (ARIMA)

Szablonów (ang. *templates*)

Warstwa reprezentatywna, odpowiedzialna za prezentację danych otrzymanych od metod znajdujących się w warstwie widoków. Dane te trafiają do statycznych szablonów, które następnie są renderowane użytkownikowi w przeglądarce internetowej.

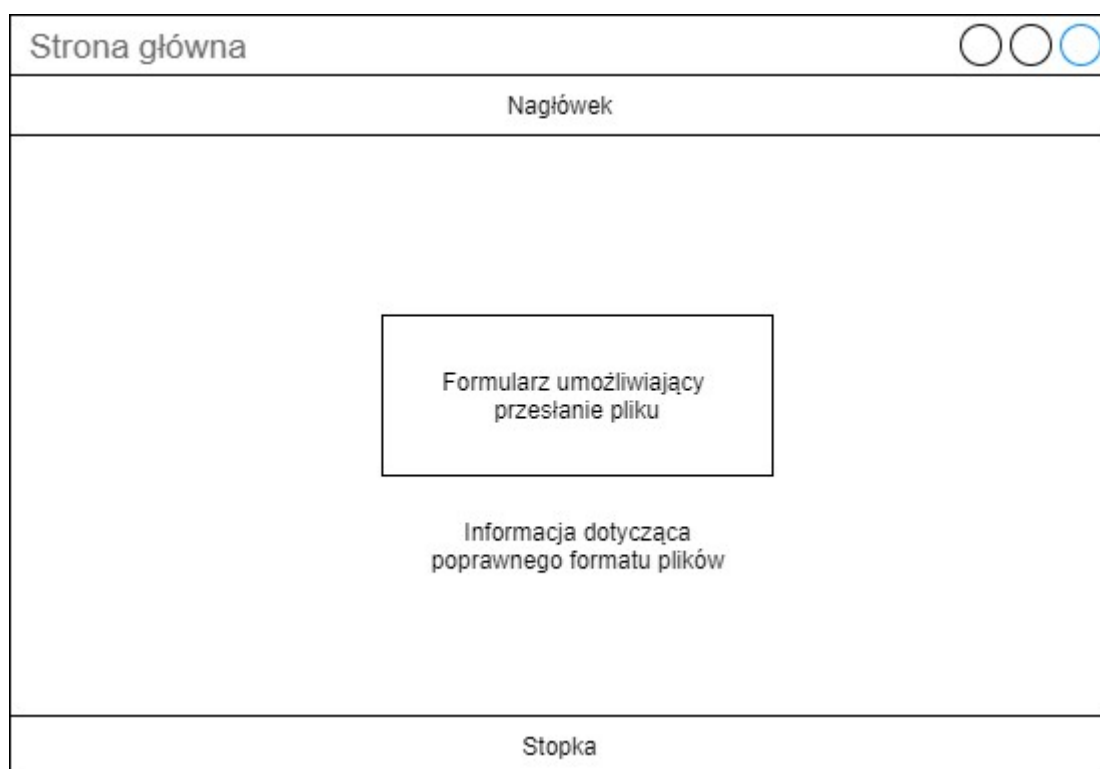
W stworzonym systemie zostaną wykorzystane cztery szablony. Każdy z nich będzie odpowiedzialny za prezentację wyników otrzymanych z wymienionych wyżej metod, opisywany przy okazji opisu warstwy szablonów.

5.2 Interfejs użytkownika

Przyjazny użytkownikowi interfejs użytkownika jest jednym z najważniejszych elementów projektowania oprogramowania użytkowego. Dobre zaprojektowanie tego elementu wpływa na komfort korzystania z aplikacji przez użytkownika. Im lepiej zostanie zaprojektowany interfejs, tym użytkownik chętnie będzie korzystać z oprogramowania.

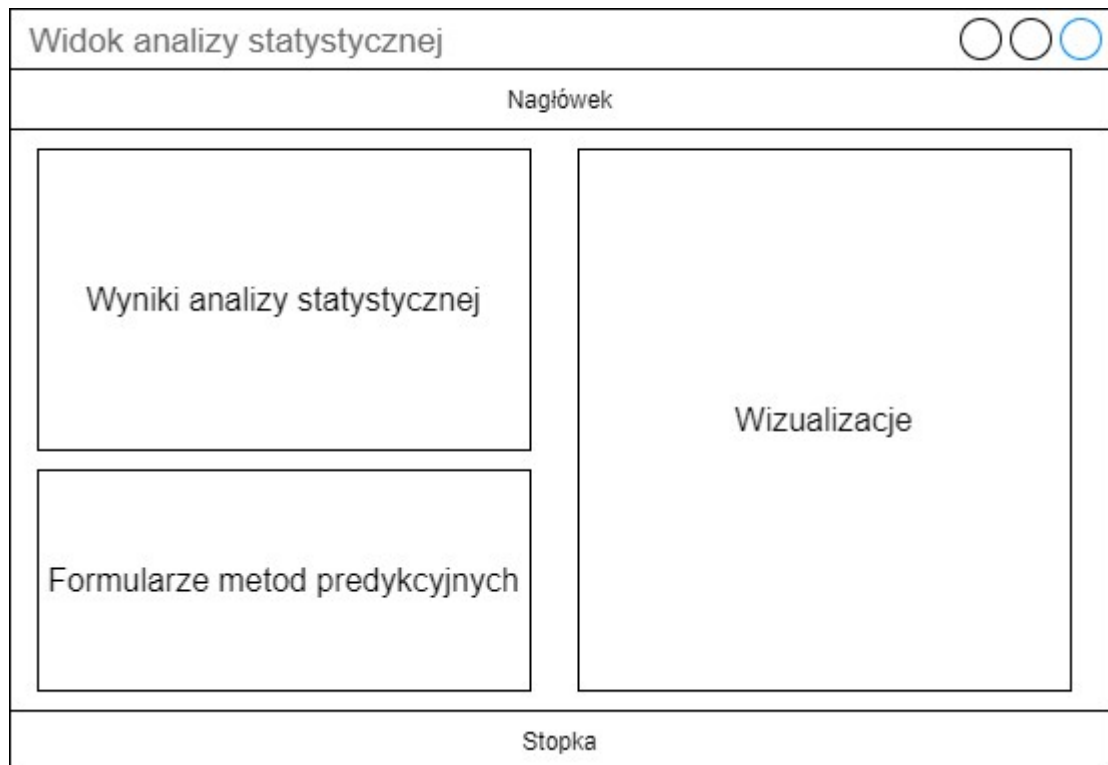
Interfejsem użytkownika w tworzonym rozwiązaniu będą statyczne szablony napisane z wykorzystaniem języka HTML oraz CSS, renderowane po stronie użytkownika, w przeglądarce internetowej. Przy tworzeniu oprogramowania zostały zaprojektowane trzy różne szablony, które zostaną wyświetlone w zależności od obecnie wykonywanej operacji. Poniżej znajduje się lista widoków oraz projekty interfejsu z nimi związane:

- Widok strony głównej – widok który ukazuje się użytkownikowi podczas uruchomienia aplikacji. Na środku interfejsu widoczny jest formularz, umożliwiający przesłanie przez użytkownika na serwer pliku, zawierającego szereg czasowy. Pod formularzem umieszczony jest tekst informujący o poprawnym formacie pliku oraz poprawnej strukturze danych w nim zawartych.



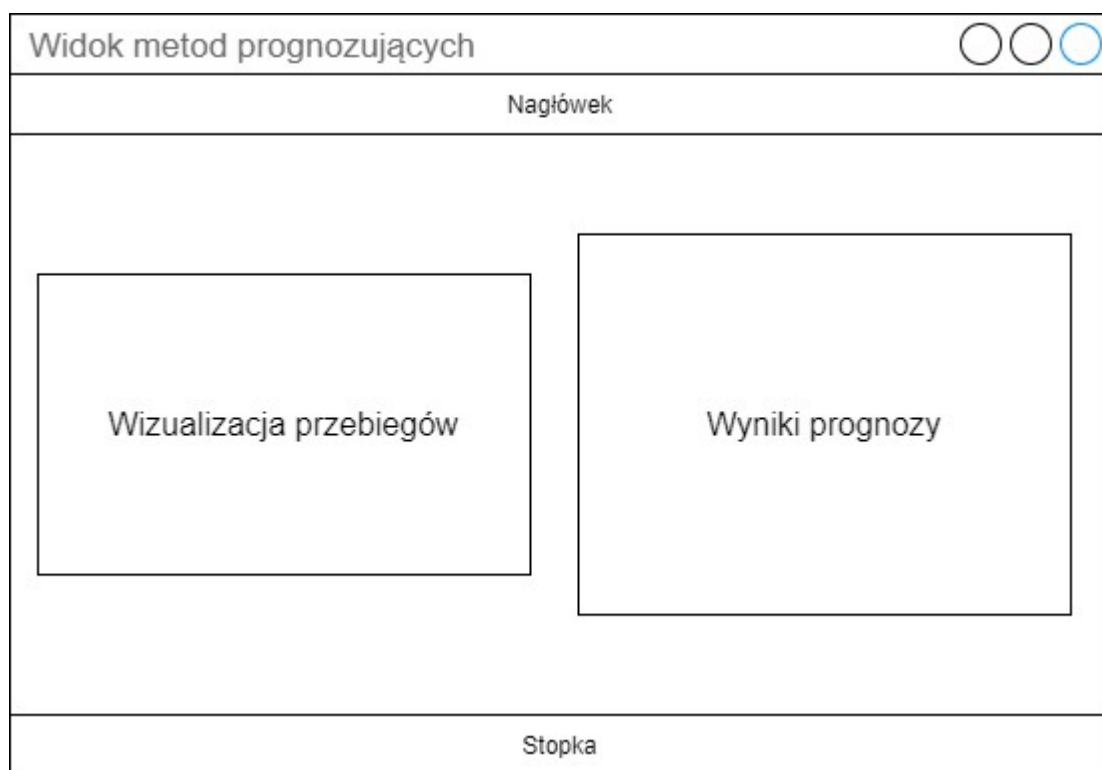
Rysunek 5.2 Projekt interfejsu – widok strony głównej

- Widok wyniku analizy statystycznej – po pomyślnym przesłaniu pliku, użytkownik zostanie przeniesiony do kolejnego widoku. W interfejsie, po lewej stronie ekranu wyświetlone zostaną wyniki analizy statystycznej szeregu czasowego. Pod nimi, użytkownik znajdzie formularze umożliwiające przetestowanie wybranej metody prognozującej przebieg szeregu czasowego. Prawa część interfejsu jest całkowicie poświęcona wizualizacji szeregów czasowych w postaci wykresów.



Rysunek 5.3 Projekt interfejsu – widok analizy statystycznej

- Widok wyniku prognozy – do widoku tego użytkownik zostanie przeniesiony jedynie w wyniku wykorzystania jednego z formularzy służących do testowania wybranej metody predykcji. Niezależnie od wykorzystanej metody, użytkownikowi ukaże się ten sam interfejs, różniący się jedynie zawartością sekcji „Wyniki prognozy”, w prawej części ekranu. Po lewej natomiast zostaną ukazane rzeczywiste oraz prognozowane przebiegi szeregu czasowego.

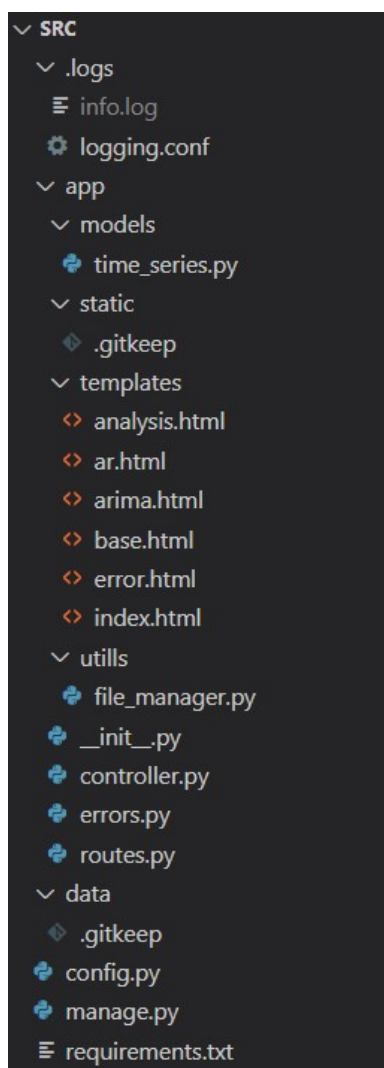


Rysunek 5.4 Projekt interfejsu – widok wyniku prognozy

6. IMPLEMENTACJA OPROGRAMOWANIA DYDAKTYCZNEGO

6.1 Struktura plików aplikacji

Z racji tego, iż do stworzenia serwera aplikacji został wykorzystany framework Flask, nie została odgórnie narzucona struktura plików aplikacji. Struktura plików projektu została zaprojektowana tak, żeby prezentowała logiczną spójność pomiędzy poszczególnymi komponentami aplikacji. Zaimplementowane funkcjonalności, o podobnym przeznaczeniu zostały zapisane w tych samych skryptach, które następnie zostały przydzielone do odpowiednich modułów (folderów). Poniżej znajduje się zrzut ekranu, prezentujący strukturę plików oraz folderów już zaimplementowanego rozwiązania.

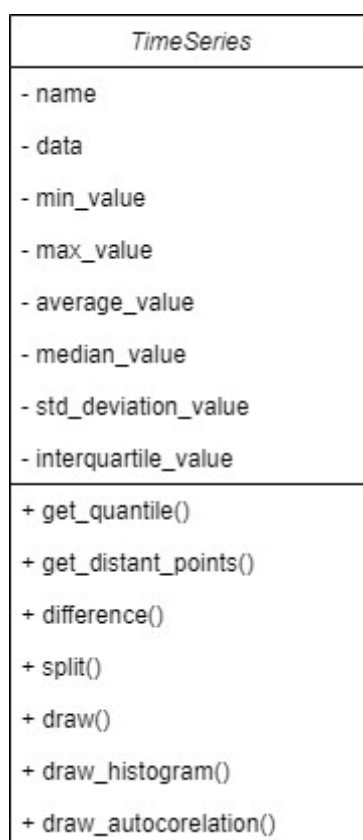


Rysunek 6.1 Struktura plików aplikacji

Każdy z folderów jest reprezentacją pojedynczego modułu. Moduł ten może zawierać pliki lub kolejne moduły. Pierwszy z modułów widocznych na rysunku 6.1 *Struktura plików aplikacji* to moduł **logs**. Przechowywany jest tam plik z logami aplikacji, oraz plik odpowiedzialny za konfigurację owych logów. Zawarte tam pliki mogą być przydatne w przy testowaniu nowych funkcjonalności oraz naprawianiu błędów w aplikacji. Kolejnym, głównym modułem całego rozwiązania jest folder **app**. Zawiera on w sobie mniejsze moduły takie jak: **models** – który zawiera plik w którym zapisana jest struktura, będącą reprezentacją szeregu czasowego w formie klasy w języku Python; **static** – w którym przechowywane są pliki graficzne wygenerowane podczas działania programu; **templates** – zawierający wszystkie szablony języka HTML, opisywane szczegółowo w rozdziale 5. *PROJEKT OPROGRAMOWANIA DYDAKTYCZNEGO DO TESTOWANIA DZIAŁANIA WYBRANYCH METOD DLA MODELI SZEREGÓW CZASOWYCH*; **utils** – będący modułem przechowującym skrypty które posiadają implementację funkcjonalności wykorzystywanych w kontekście całej aplikacji. Poza modułami, w folderze app znajdują się również najważniejsze dla całego projektu pliki, takie jak: **__init__.py** – będący rdzeniem całej architektury aplikacji; **routes.py** – odpowiedzialny za przekierowywanie żądań do poszczególnych metod; **errors.py** – obsługującym wszelkie rodzaje błędów aplikacji oraz **controller.py**- skrypt w którym zapisana jest cała logika biznesowa aplikacji. Obok głównego modułu app, znajduje się jeszcze folder **data**, który przechowuje pliki CSV przesłane do aplikacji przez użytkownika. Poza wszelkimi modułami znajdują się pliki: **config.py** – plik konfiguracyjny; **manage.py** – służący do uruchamiania serwera aplikacji oraz **requirements.txt** – będący plikiem tekstowym przechowującym nazwy pakietów oraz ich wersje, które są potrzebne do prawidłowego działania oprogramowania.

6.2 Model szeregu czasowego

Tworzone oprogramowanie w głównej mierze dotyczy szeregu czasowego, oraz operacji na nim wykonywanych. Dlatego też, w omawianym oprogramowaniu, szereg czasowy został zaimplementowany w formie klasy języka Python. Klasa ta znajduje się w pliku **time_series.py** w folderze **app/models**. Obiekty klasy **TimeSeries** tworzone są z wykorzystaniem zawartości pliku CSV przesłanego przez użytkownika. Poniżej przedstawiona jest diagram klasy, będącej reprezentacją szeregu czasowego.



Rysunek 6.2 Diagram klasy *TimeSeries*

Wszystkie pola w powyższej klasie, z wyjątkiem pola **name** oraz **data**, przechowują wartości zmiennoprzecinkowe. Klasa ta posiada następujące pola:

- **name** – ciąg znaków nazwą szeregu czasowego.
- **data** – pole przechowujące obiekt typu **DataFrame** pochodzący z biblioteki **pandas**. Obiekt ten zawiera dane, które znajdowały się w pliku, na podstawie którego tworzony jest obiekt klasy **TimeSeries**.
- **min_value** – najmniejsza wartość w szeregu czasowym.
- **max_value** – największa wartość w szeregu czasowym.

- **average_value** – wartość średnia szeregu czasowego.
- **median_value** – mediana wartości szeregu czasowego.
- **std_deviation_value** – odchylenie standardowe wartości szeregu czasowego.
- **interquartile** – interkwartyl, czyli różnica między pierwszym a trzecim kwartylem.

Poza licznymi polami, w klasie znajdują się również metody publiczne, będące implementacją metod statystycznych oraz wizualizacji. Implementacja owych jest omawiana w ramach dalszych podrozdziałów.

- **get_quantile** – metoda zwracająca kwantyl rzędu **p** przekazanego przez argument tej metody.
- **get_distant_points** – zwraca zbiór punktów oddalonych.
- **split** – podziela wartości szeregu czasowego na dwa podzbiory w proporcjach zdefiniowanych przez argument tej metody.
- **difference** – różnicuje szereg czasowy o **n** razy, gdzie **n** jest przekazywane przez argument tej metody.
- **draw** – tworzy diagram przebiegu szeregu czasowego i zapisuje go do pliku.
- **draw_histogram** - tworzy histogram wartości szeregu czasowego i zapisuje go do pliku.
- **draw_autocorrelation** - tworzy diagram autokorelacji wartości szeregu czasowego i zapisuje go do pliku.

6.3 Przesyłanie pliku

Aby można było testować szeregi czasowe, użytkownik musi przesłać do serwera aplikacji plik CSV, na którego podstawie utworzona zostanie instancja, opisywanej w poprzednim podrozdziale klasy `TimeSeries`. Pierwszą z funkcjonalności, która została zaimplementowana w omawianym oprogramowaniu było przesyłanie pliku przez użytkownika.

Użytkownik, w głównym widoku posiada prosty formularz, który umożliwia wskazanie dowolnego pliku CSV znajdującego się na jego urządzeniu. Formularz ten został zaimplementowany w pliku **index.html**.

```
<!-- uploading file form -->
<form action = "http://localhost:5000/upload" method = "POST" enctype = "multipart
/form-data">
  <input class="form-control-
file" id = "fileInput" type = "file" name = "file" accept = ".csv"/ style='width:3
0em'>
  <input class="form-control mt-3 btn-
info" id = "submitFile" type = "submit"/ disabled>
</form>

<!-- JS script, that prevents pressing upload, where there is no file choosen -->
<script>
  document.getElementById("fileInput").onchange = function() {
    if(this.value) {
      document.getElementById("submitFile").disabled = false;
    }
  }
</script>
```

Listing 2. Implementacja formularza do przesyłania pliku w pliku index.html

Formularz został stworzony z wykorzystaniem znaczników `form`. W parametrze `action` zdefiniowany został adres, pod który mają zostać przesłane dane (w tym przypadku plik) po naciśnięciu przycisku „Prześlij”, przez użytkownika. Formularz składa się z dwóch znaczników. Pierwszy z nich, z klasą `form-control-file` odpowiada za przycisk „Wybierz plik”. Dzięki ustawieniu w tym znaczniku parametru `accept` jako „.csv” formularz może przyjmować jedynie pliki z rozszerzeniem CSV. Drugi z nich, typu `submit` reprezentuje w interfejsie przycisk „Prześlij”. Poniżej formularza znajduje się krótki skrypt w języku JavaScript, który uniemożliwia użytkownikowi naciśnięcie przycisku „Prześlij” w przypadku, gdy nie został przez niego wybrany żaden plik.

Zaimplementowany na stronie głównej formularz, prezentuje się następująco.

Analiza szeregów czasowych

Prześlij plik z szeregiem czasowym

Wybierz plik Nie wybrano pliku

Prześlij

Przesłany plik powinien posiadać rozszerzenie CSV. Plik powinien posiadać dwie kolumny:

- data - znaczniki czasowe.
- value - wartości.

Kolejność tych kolumn nie ma znaczenia. Kolumny te powinny być rozdzielone przecinkiem.

© 2021 Mateusz Godlewski

Rysunek 6.3 Widok strony głównej aplikacji

Po naciśnięciu przycisku „Prześlij” aplikacja wysyła wybrany przez użytkownika plik pod adres **localhost:5000/upload** metodą POST. W skrypcie **routes.py** znajduje się metoda **upload**, która obsługuje żądania wysyłane pod wspomniany adres.

```
@APP.route("/upload", methods=["POST"])
def upload():
    """This route is called by file uploading form at home page.
    Calls method, that upload file send by request to application server storage
    directory.
    After successful file uploading, this method redirects to 'analysis' endpoint.
    """

    # calling upload file method, that uploads file to application server
    # and returns absolute path to uploaded file
    file_path = FileManager.upload_file(request=request)

    # adding 'file_path' to current session variables
    session["file_path"] = file_path

    # redirection to analysis endpoint
    return redirect(url_for("analysis"), code=307)
```

Listing 3. Metoda upLoad z pliku routes.py

Powyższa funkcja wywołuje metodę `upload` klasy `FileManager`, przekazując do niej jako argument, obiekt `request`, zawierający w sobie przesłany szereg czasowy. Wywołana metoda zapisuje plik w folderze **data** serwera aplikacji oraz zwraca bezpośrednią ścieżkę do tego pliku, która jest przypisywana do obecnej sesji użytkownika.

```
@staticmethod
def upload_file(request: Request) -> str:
    """Uploads file received by request to application data directory,
    saves it and returns absolute path to this file."""
    # retrieves file from request
    _file = request.files["file"]

    # absolute path to file
    file_path = os.path.join(config.DATA_DIR, secure_filename(_file.filename))

    # saves received file
    _file.save(file_path)
```

Listing 4. Metoda `upload_file` z pliku `file_manager.py`

Po zakończeniu metody, następuje przekierowanie do znajdującej się w tym samym skrypcie, metody `analysis`. Funkcja `analysis` pobiera ścieżkę do pliku ze zmiennych zapisanych w sesji obecnego użytkownika oraz wywołuje metodę `analysis` z pliku `controller.py`.

```
@APP.route("/analysis", methods=["POST"])
def analysis():
    """This route calls method that analyse and visualise time series."""

    # retrieving path to file from session
    file_path = session["file_path"]

    # calling analysis method
    return controller.analysis(file_path=file_path)
```

Listing 5. Metoda `analysis` ze skryptu `routes.py`

6.4 Analiza statystyczna

Jedną z najważniejszych funkcjonalności, które oferuje tworzone rozwiązanie jest analiza statystyczna szeregu czasowego. Proces analizy, rozpoczyna się natychmiast, po poprawnym przesłaniu pliku do serwera aplikacji. Użytkownik po naciśnięciu przycisku „Prześlij”, zostaje przekierowany do widoku analizy statystycznej. Za całą logikę w tym procesie odpowiada metoda `analysis` w skrypcie `controller.py`.

```
def analysis(file_path: str):
    """Renders template with statistical information and plots of time series
    included in file, which path is given by 'file_path' argument.
    """
    # dictionary that will contain all data for rendering
    data = {}

    # loads content of file
    data_file = FileManager.read_file(file_name=file_path)

    # creation of TimeSeries object
    name = FileManager.get_file_name_from_(path=file_path)
    time_series = TimeSeries(dataset=data_file, name=name)
    data["analyse"] = time_series.info

    # visualisation of created time series
    plots = _visualisation(file_path=file_path)
    if plots:
        data["plots"] = plots
    return render_template("analysis.html", data=data)
```

Listing 6. Metoda `analysis` ze skryptu `controller.py`

W tej funkcji do zmiennej `data_file`, przypisywana jest zawartość przesłanego przez użytkownika pliku, która została uprzednio wczytana za pomocą metody `read_file`, klasy `FileManager`. Następnie, na podstawie danych wczytanych z pliku, oraz nazwie owego pliku, tworzona jest nowa instancja klasy `TimeSeries`.

W momencie tworzenia nowego obiektu klasy, wywoływana jest metoda, zwana konstruktorem. W przypadku klasy `TimeSeries`, konstruktor ten wywołuje szereg metod, mających na celu weryfikację danych oraz ewentualną ich korekcję.


```

def __init__(self, dataset: pd.DataFrame, name: str) -> object:
    """Constructor, that creates TimeSeries object.
    Before object creation, given dataset is validated for data type correction.
    After successful validation, dataset missing values
    are completed and numerical values are standardized.
    """
    # data type validation
    if self._data_type_validation(dataset=dataset):
        self.name = name
        self.data = self._data_complement(dataset=dataset)
        self.data = self._data_unification(dataset=dataset)
    else:
        app.logging.error("Given dataset data types are incorrect!")
        raise TypeError

```

Listing 7. Konstruktor klasy TimeSeries

Powyższy konstruktor na początku sprawdza za pomocą metody `_data_type_validation`, czy dane zawarte w szeregu czasowym, są poprawnego typu. W przypadku niepowodzenia, użytkownikowi zwracany jest odpowiedni komunikat błędu, w przeciwnym wypadku, sprawdzana jest komplementarność danych z wykorzystaniem metody `_data_complement`, oraz unifikacja wartości za pomocą funkcji `_data_unification`.

Gdy instancja tej klasy zostanie prawidłowo stworzona, do zmiennej `data` zostaje przypisana wartość pola `info`. Pole to w klasie `TimeSeries` jest słownikiem, zawierającą wszystkie statystyczne dane dotyczące szeregu czasowego. Każda z wartości które znajdują się w tym słowniku obliczana jest w momencie pobierania tego pola.

```

@property
def info(self) -> dict:
    """Returns all properties of TimeSeries object."""
    return {
        "minimum_value": self.min_value,
        "maximum_value": self.max_value,
        "average_value": self.average_value,
        "median_value": self.median_value,
        "standard_deviation_value": self.std_deviation_value,
        "interquartile_value": self.interquartile_value,
    }

```

Listing 8. Pole info klasy TimeSeries

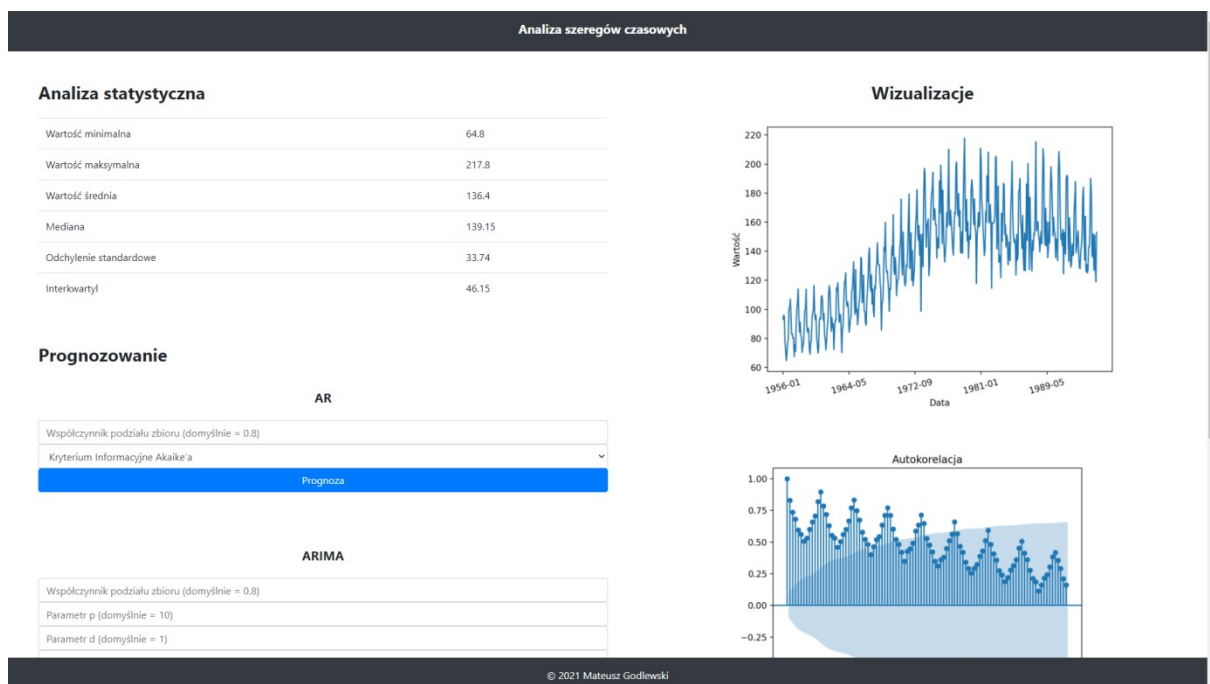
Kolejnym krokiem analizy statystycznej jest stworzenie wizualizacji szeregu czasowego w postaci wykresów. Służy do tego metoda `_visualisation`, do której przekazana jest ścieżka do pliku zawierającego szereg czasowy. Na podstawie przekazanej ścieżki, metoda ta odczytuje zawartość pliku i na jego podstawie tworzy nową instancję klasy `TimeSeries`. Następnie wywołuje kolejno metody `draw`, `draw_autocorrelation` oraz

`draw_histogram` stworzonej instancji, które generują poszczególne wykresy oraz zapisują je w postaci plików graficznych z rozszerzeniem png, w folderze **static**. Każda ze wspomnianych metod, zwraca jako rezultat swojego działania bezpośrednią ścieżkę do wygenerowanego wykresu, która zapisywana jest do zmiennej `plots`, która ostatecznie jest zwracana przez omawianą metodę `_visualisation` jako wynik tej metody.

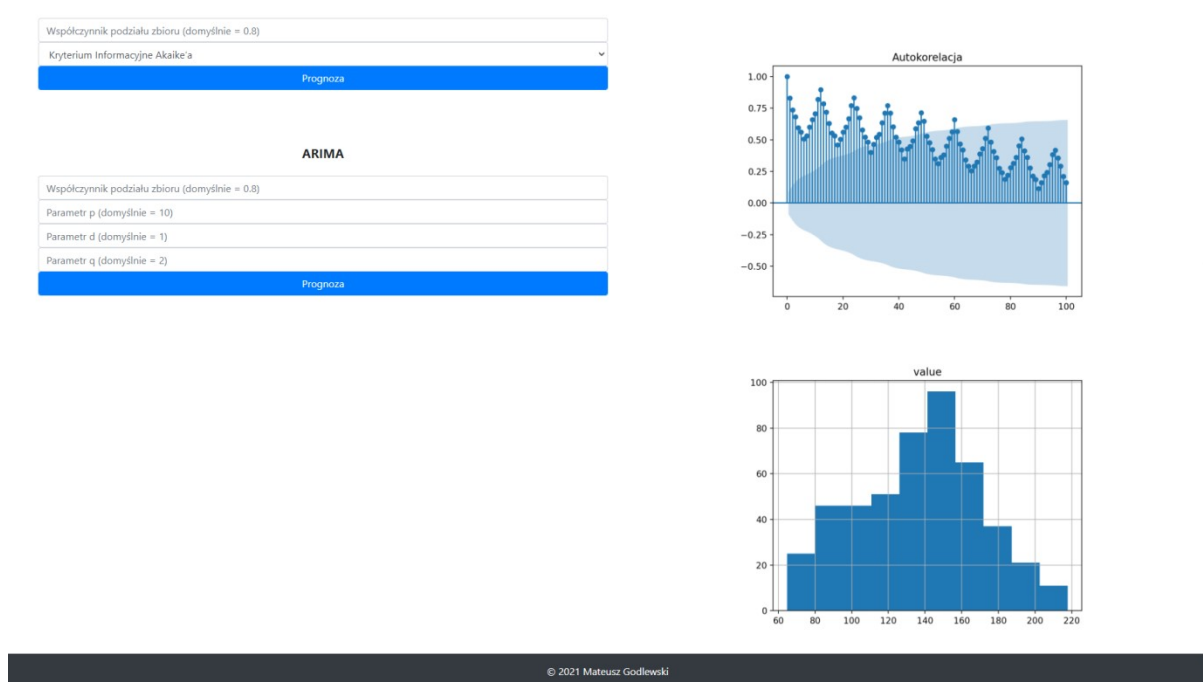
```
def _visualisation(file_path: str) -> list:
    """Creates plots of time series, that path is given as 'file_path' argument.
    This method returns list of all created and saved plots names.
    This method creates three types of plots:
    All created plots are saved to data directory.
    """
    # loads content of file
    data_file = FileManager.read_file(file_name=file_path)
    # creation of TimeSeries object
    file_name = os.path.split(file_path)[-1].split(".")[0]
    time_series = TimeSeries(dataset=data_file, name=file_name)
    # drawing plots and saving them to files
    plots = []
    plots.append(time_series.draw())
    plots.append(time_series.draw_autocorelation(lags=100))
    plots.append(time_series.draw_histogram())
    return plots
```

Listing 9. Metoda `_visualisation`

Metoda `analysis`, w której odbywają się wszystkie omawiane w tym podrozdziale operacje, przekazuje zmienną `data` do szablonu **analysis.html**, który bazując na otrzymanych danych, renderuje użytkownikowi pełen widok analizy statystycznej.



Rysunek 6.4 Widok analizy statystycznej



Rysunek 6.5 Widok analizy statystycznej cd.

Na powyższym widoku, po lewej stronie ekranu, widoczne są dwa formularze. Z ich wykorzystaniem, użytkownik, poprzez odpowiedni dobór parametrów i naciśnięcie przycisku „Prognoza” może wytestować działanie jednej z dwóch dostępnych metod predykcji szeregu czasowego. Implementacja tej funkcjonalności zostanie omówiona w kolejnym podrozdziale.

6.5 Modele predykcyjne

Główną z funkcjonalności, którą powinno posiadać tworzone oprogramowanie, jest możliwość testowania modeli statystycznych na wybranych szeregach czasowych, które umożliwiają predykcję ich przyszłych wartości. W zaimplementowanym oprogramowaniu, użytkownik może testować wybrane modele z wykorzystaniem formularzy, widocznych na rysunkach 6.4 oraz 6.5. Formularze te, zaimplementowane zostały w szablonie **analysis.html**.

```
<form action="/ar" method="POST">
  <input name="split_ratio" type="text" placeholder="Współczynnik podziału zbioru
(domyślnie = 0.8)" class="form-control" pattern="^\d*(\.\d{0,2})?$">
  <select name="ar_ic" class="form-control">
    <option value="aic">Kryterium Informacyjne Akaike'a</option>
    <option value="bic">Bayesowskie kryterium informacyjne Schwarza</option>
    <option value="hqic">Kryterium informacyjne Hannana-Quinna</option>
  </select>
  <button type="submit" formmethod="post" class="btn btn-primary text-white form-
control">Prognoza</button>
</form>
```

Listing 10. Formularz metody AR

Powyższy formularz służy do testowania metody autoregresji. Posiada on dwa pola, oraz przycisk zatwierdzający wysłanie. W pierwszym polu, użytkownik może podać wartość zmiennoprzecinkową z zakresu 0.0 do 1.0. Pole to jest chronione przed wpisaniem przez użytkownika wartości spoza tego zakresu, bądź ciągu znaków, dzięki wyrażeniu regularnemu przypisanemu do znacznika **pattern**. Wartość tego pola odpowiada za stosunek, w jakim zostanie podzielony zbiór wartości szeregu czasowego na podzbiory treningowe oraz testowe. Drugim polem jest rozwijana lista wyboru, w której może zostać wybrana jedna z wartości odpowiadającej kryterium doboru rzędu predykcji. Wszystkie z opcji zostały zdefiniowane w znacznikach **option**. Przycisk „Prognoza” wysyła zawartość formularza pod adres „/ar”.

```
<form action="/arima" method="POST">
  <input name="split_ratio" type="text" placeholder="Współczynnik podziału zbioru
(domyślnie = 0.8)" class="form-control" pattern="^\d*(\.\d{0,2})?$">
  <input name="arima_ar" type="text" placeholder="Parametr p (domyślnie = 10)" cla
ss="form-control">
  <input name="arima_i" type="text" placeholder="Parametr d (domyślnie = 1)" class
="form-control">
  <input name="arima_ma" type="text" placeholder="Parametr q (domyślnie = 2)" clas
s="form-control">
  <button type="submit" formmethod="post" class="btn btn-primary text-white form-
control">Prognoza</button>
</form>
```

Listing 11. Formularz metody ARIMA

Formularz służący do testowania metody zintegrowanej autoregresji ze średnią ruchomą, posiada 4 pola, oraz przycisk zatwierdzający przesłanie. Pierwsze pole ma identyczne zastosowanie, jak pierwsze z pól w poprzednio omawianym formularzu. W kolejne trzy pola formularza można wprowadzić wartości całkowite, które odpowiadają odpowiednio wartościom parametrów **p**, **q** oraz **d** metody **ARIMA**. Po naciśnięciu przez użytkownika przycisku „Prześlij”, wartości przez niego podane w formularzu są przesyłane pod adres „/arima”

W zależności od tego, z którego formularza skorzysta użytkownik, odpowiednie żądanie zostanie wysłane, a następnie obsłużone przez serwer aplikacji. Obsługa obu z żądań przebiega w bardzo podobny sposób. Ze zmiennych sesji użytkownika pobierana jest bezpośrednia ścieżka do pliku, która przypisywana jest do zmiennej `file_path`, następnie wszystkie wartości przekazane przez formularz, wraz z ich kluczami zostają przypisane do zmiennej słownikowej `parameters`. Obie te zmienne są przekazywane do odpowiedniej metody znajdującej się w pliku **controller.py**.

Za prognozowanie przyszłych wartości szeregu czasowego, z wykorzystaniem metody autoregresji, odpowiada metoda `forecast_ar`. Jako argumenty, metoda ta przyjmuje ścieżkę do pliku zawierającego szereg czasowy, oraz parametry przekazane przez użytkownika za pomocą formularza. Na początku działania tej funkcji, wczytywana jest zawartość pliku znajdującego się pod podaną ścieżką, oraz na podstawie jego wartości tworzony jest obiekt klasy `TimeSeries`. Następnie, wartości szeregu czasowego dzielone są na podzbiór treningowy oraz testowy z wykorzystaniem metody `split`. Podział ten wykonany jest na podstawie wartości argumentu `split_ratio`. Podzbiór treningowy powstały w wyniku tego podziału, wykorzystany jest do tworzenia instancji klasy `AR` pochodzącej z biblioteki **statsmodels**. Klasa ta jest reprezentacją modelu autoregresyjnego. Następnie z wykorzystaniem metody `fit`, do której jako argument zostało przekazane kryterium doboru rzędu predykcji wybrane przez użytkownika, model zostaje wytrenowany na dane dostarczone w momencie jego tworzenia. Następnym krokiem jest predykcja **n** przyszłych wartości szeregu czasowego, z wykorzystaniem metody `predict`, gdzie **n** oznacza liczebność zbioru testowego. Wartości zwrócone w rezultacie predykcji są porównywane z wartościami w zbiorze testowym i na ich podstawie obliczana jest wartość błędu średniokwadratowego. Następnie, tworzony jest wykres zawierający dwa przebiegi. Jeden z nich to wartości zbioru testowego, drugi natomiast, prezentuje wyniki prognozy.

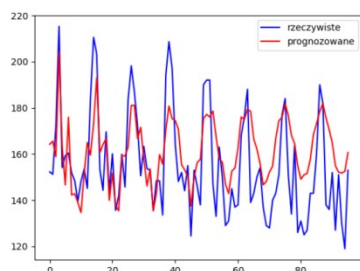
Wszystkie wyniki predykcji, takie jak: rząd predykcji, kryterium doboru rzędu predykcji, wartość błędu RMSE oraz liczba trenowanych obserwacji są zapisywane w zmiennej `data`, a następnie przekazane w celu renderowania widoku do szablonu **ar.html**.

```
def forecast_ar(file_path: str, parameters: dict):
    # dictionary that will contain all data for rendering
    data = {}
    # preparing parameters
    data["split_ratio"] = float(parameters.get("split_ratio", 0.8))
    data["ic"] = parameters.get("ar_ic", "")
    # loads content of file
    data_file = FileManager.read_file(file_name=file_path)
    # creation of TimeSeries object
    name = FileManager.get_file_name_from_(path=file_path)
    time_series = TimeSeries(dataset=data_file, name=name)
    # split time series to train and test datasets
    train_data, test_data = time_series.split(ratio=data["split_ratio"])
    # creation of auto regression model
    model = AR(train_data)
    # auto regression model training
    trained_model = model.fit(ic=data["ic"])
    # forecast
    start = len(train_data)
    end = start + len(test_data) - 1
    forecast_results = trained_model.predict(start=start, end=end)
    # RMSE - root-mean-square error
    rmse = sqrt(mean_squared_error(test_data, forecast_results))
    # plotting forecasting results and saving to file
    plt.plot(test_data, color="blue", label="rzeczywiste")
    plt.plot(forecast_results, color="red", label="prognozowane")
    plt.legend(loc="upper right")
    plot_name = f"{time_series.name}_forecast_ar_{data['ic']}_{data['split_ratio']}.png"
    plot_path = os.path.join(config.STATIC_DIR, plot_name)
    plt.savefig(plot_path)
    # clearing matplotlib plot
    plt.clf()
    # preparing data for template rendering
    data["forecast_plot"] = plot_name
    data["lag"] = trained_model.k_ar
    data["tobs"] = trained_model.n_totobs
    data["rmse"] = rmse
    data["ic"] = config.IC_METHODS[data["ic"]]
    return render_template("ar.html", data=data)
```

Listing 12. Metoda `forecast_ar`

Wynikiem powyższej metody, jest wygenerowany szablon **ar.html**, który odpowiedzialny jest za renderowanie użytkownikowi widoku wyniku predykcji. Na owym widoku, po lewej stronie ekranu, widnieje wykres prezentujący realny przebieg szeregu czasowego oraz przebieg prognozowany przez metodę autoregresji. Po prawej stronie ekranu znajduje się tabela zawierająca numeryczne wyniki prognozy.

Wyniki prognozy modelu AR



Kryterium doboru rzędu predykcji	Kryterium Informacyjne Akaike'a
Rząd predykcji	16
Ilość trenowanych obserwacji	380
Wartość RMSE - błędu średniokwadratowego	17.568628622994403

Rysunek 6.6 Widok wyniku predykcji metodą AR

Metodą odpowiedzialną za prognozowanie szeregów czasowych z wykorzystaniem metody ARIMA, jest natomiast funkcja `forecast_arima`. Metoda ta przyjmuje takie same argumenty, jak w przypadku wcześniej omawianej funkcji `forecast_ar`. Na początku jej działania, ustalane są wartości dla parametrów **p**, **q** oraz **d**. W przypadku gdy użytkownik nie podał wartości w jednym z pól odpowiadającym danemu parametrowi, przypisywana jest do niego wartość domyślna. Początek działania tej metody jest podobny jak w przypadku funkcji odpowiedzialnej za prognozowanie metodą autoregresji. Tak samo tworzony jest nowy obiekt klasy `TimeSeries`, oraz wartości szeregu czasowego dzielone są w ten sam sposób na dwa podzbiory. Metody te jednak różnią się dalszymi etapami działania. Na podstawie podzbioru treningowego, oraz wcześniej przygotowanych parametrów, tworzony jest obiekt klasy `ARIMA` pochodzącej z biblioteki **statsmodels**. Klasa ta jest odzwierciedleniem zintegrowanego modelu autoregresyjnego ze średniom ruchomom. W kolejnych krokach model ten jest trenowany z wykorzystaniem metody `fit`, oraz wykonywana jest predykcja **n** przyszłych wartości, gdzie **n** oznacza liczebność zbioru testowego. Następnie zbiór testowy oraz wyniki predykcji zostają wykorzystane w celu utworzenia grafu przedstawiającego oba te przebiegi. Na samym końcu działania tej metody, do zmiennej `data` przypisywane są numeryczne wyniki prognozy takie jak liczba trenowanych obserwacji czy błędy skuteczności modelu predykcyjnego dla każdego z trzech kryteriów wykorzystywanych w ramach tworzonego oprogramowania.

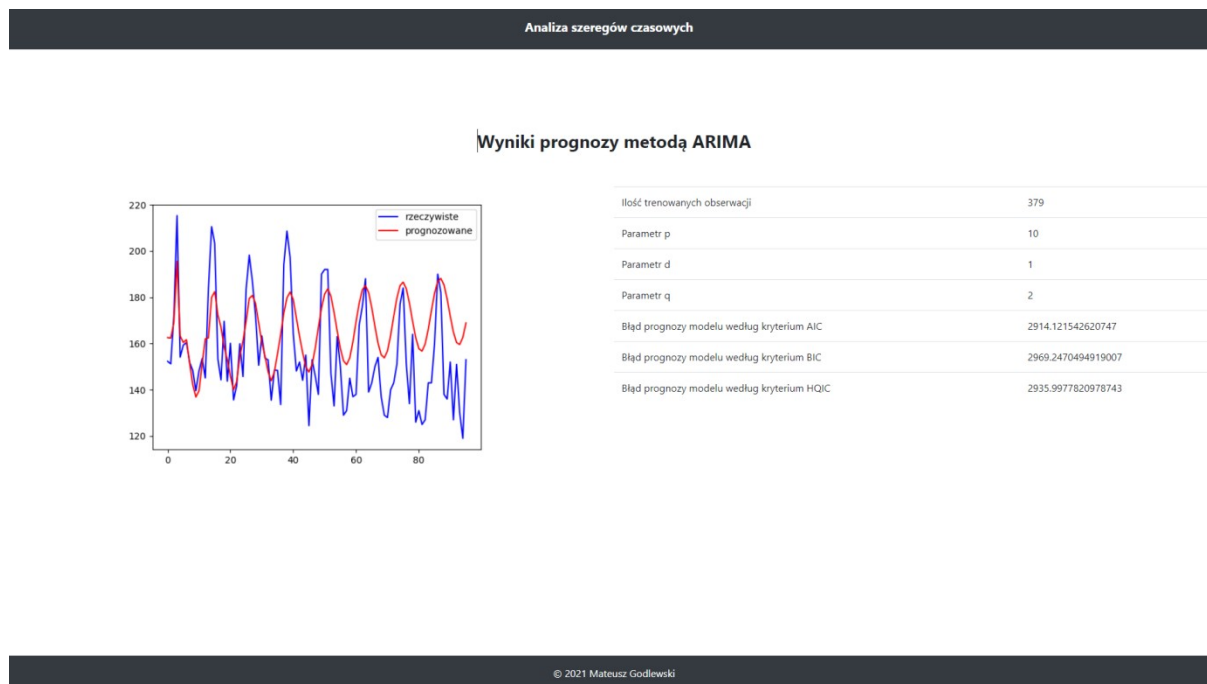
```

def forecast_arima(file_path: str, parameters: dict):
    # dictionary that will contain all data for rendering
    data = {}
    # preparing parameters
    data["split_ratio"] = float(parameters.get("split_ratio", 0.8))
    data["p"] = int(parameters.get("arima_ar", 10))
    data["q"] = int(parameters.get("arima_i", 1))
    data["d"] = int(parameters.get("arima_ma", 2))
    # loads content of file
    data_file = FileManager.read_file(file_name=file_path)
    # creation of TimeSeries object
    name = FileManager.get_file_name_from_(path=file_path)
    time_series = TimeSeries(dataset=data_file, name=name)
    # split time series to train and test datasets
    train_data, test_data = time_series.split(ratio=data["split_ratio"])
    # creation of ARIMA model
    model = ARIMA(
        train_data,
        order = (
            data["p"],
            data["q"],
            data["d"]
        )
    )
    # ARIMA model training
    trained_model = model.fit()
    # forecast range
    steps = len(test_data)
    # forecasting
    forecast_results = trained_model.forecast(steps=steps)[0]
    # plotting forecasting results and saving to file
    plt.plot(test_data, color="blue", label="rzeczywiste") # plotting real values
    plt.plot(forecast_results, color="red", label="prognozowane") # plotting predicted values
    plt.legend(loc="upper right")
    plot_name = f"{time_series.name}_forecast_arima_{data['p']}_{data['q']}_{data['d']}.png" # name of plotted file
    plot_path = os.path.join(config.STATIC_DIR, plot_name) # plotted file path
    plt.savefig(plot_path) # saving plot to file
    # clearing matplotlib plot
    plt.clf()
    # preparing data for template rendering
    data["forecast_plot"] = plot_name
    data["tobs"] = trained_model.n_totobs
    data["aic"] = trained_model.aic
    data["bic"] = trained_model.bic
    data["hqic"] = trained_model.hqic
    return render_template("arima.html", data=data)

```

Listing 13. Metoda forecast_arima

Wynikiem działania powyższej metody, tak samo jak w przypadku poprzednio omawianej funkcji `forecast_ar`, jest widok wyników predykcji. Widok prezentuje się niemal identyczny sposób jak widok wyników predykcji metodą **AR**. Wyjątkiem są dane wyświetlane po prawej stronie ekranu, które są różne w przypadku zastosowanej metody **ARIMA**. Do renderowania poniższego widoku wykorzystany został szablon **arima.html**.



Rysunek 6.7 Widok wyniku predykcji metodą ARIMA

7. TESTOWANIE DZIAŁANIA WYKONANEGO OPROGRAMOWANIA

LITERATURA

Bibliografia literatury

- [1] „Mathematical software”, https://en.wikipedia.org/wiki/Mathematical_software, [przełgądane dnia 01.06.2021r]
- [2] „MATLAB”, <http://www.ont.com.pl/produkty/lista-produktow/matlab>, [przełgądane dnia 01.06.2021r]
- [3] „Time series”, https://en.wikipedia.org/wiki/Time_series, [przełgądane dnia 01.06.2021r]
- [4] „Szereg czasowy”, https://mfiles.pl/pl/index.php/Szereg_czasowy, [przełgądane dnia 01.06.2021r]
- [5] „What Is Statistical Analysis?”, <https://www.businessnewsdaily.com/6000-statistical-analysis.html>, [przełgądane dnia 01.06.2021r]
- [6] „Autoregressive Models”, <https://online.stat.psu.edu/stat501/lesson/14/14.1>, [przełgądane dnia 07.06.2021r]
- [7] „Wzór na autokorelację”, https://www.nauowiec.org/wzory/statystyka/autokorelacja_411.html, [przełgądane dnia 07.06.2021r]
- [8] „Model AR”, https://en.wikipedia.org/wiki/Autoregressive_model, [przełgądane dnia 07.06.2021r]
- [9] „Model autoregresyjny”, [https://brain.fuw.edu.pl/edu/index.php/Model_autoregresyjny_\(AR\)](https://brain.fuw.edu.pl/edu/index.php/Model_autoregresyjny_(AR)), [przełgądane dnia 07.06.2021r]
- [10] „Kryterium informacyjne Akaikego”, https://pl.wikipedia.org/wiki/Kryterium_informacyjne_Akaikego, [przełgądane dnia 07.06.2021r]
- [11] „Bayesian information criterion”, https://en.wikipedia.org/wiki/Bayesian_information_criterion, [przełgądane dnia 07.06.2021r]

- [12] „Kryterium informacyjne Hannana-Quinna”,
https://pl.wikipedia.org/wiki/Kryterium_informacyjne_Hannana-Quinna, [przełądane dnia 07.06.2021r]
- [13] „Średnia ruchoma”, https://pl.wikipedia.org/wiki/%C5%9Arednia_ruchoma,
[przełądane dnia 08.06.2021r]
- [14] „Autoregressive integrated moving average”,
https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average, [przełądane dnia 08.06.2021r]
- [15] „Co trzeba wiedzieć korzystając z modelu ARIMA i które parametry są kluczowe”,
https://support.predictivesolutions.pl/ekspress/download/EB54/mat/model_ARIMA_istotnosc_parametrow.pdf, [przełądane dnia 08.06.2021r]
- [16] „Dow Jones Sustainability Index: Use of forecasting models to assist decision making”,
<https://www.revistaespacios.com/a15v36n11/15361121.html>, [przełądane dnia 08.06.2021r]
- [17] „What is Python? Executive Summary”, <https://www.python.org/doc/essays/blurb/>,
[przełądane dnia 17.06.2021r]
- [18] „What is flask?”, <https://pymbook.readthedocs.io/en/latest/flask.html>, [przełądane dnia 17.06.2021r]
- [19] „Visualization with Matplotlib”,
<https://jakevdp.github.io/PythonDataScienceHandbook/04.00-introduction-to-matplotlib.html>,
[przełądane dnia 17.06.2021r]
- [20] „Statsmodels”, <https://en.wikipedia.org/wiki/Statsmodels>, [przełądane dnia 17.06.2021r]
- [21] „Difference between MVC and MVT architecture”,
<https://www.geekinsta.com/difference-between-mvc-and-mvt/>, [przełądane dnia 17.06.2021r]

Bibliografia rysunków

- [22] http://www.ont.com.pl/wp-content/uploads/2014/10/content_schemat_pracy.png, [przełądane dnia 01.06.2021r]
- [23] https://upload.wikimedia.org/wikipedia/commons/thumb/2/21/Matlab_Logo.png/667px-Matlab_Logo.png, [przełądane dnia 01.06.2021r]
- [24] http://www.ont.com.pl/wp-content/uploads/2014/10/content_matlab.png, [przełądane dnia 01.06.2021r]
- [25] <https://i.stack.imgur.com/eFUp8.jpg>, [przełądane dnia 01.06.2021r]
- [26] https://robjhyndman.com/hyndsight/2011-12-14-cyclicts_files/figure-html/unnamed-chunk-2-1.png, [przełądane dnia 01.06.2021r]
- [27] https://robjhyndman.com/hyndsight/2011-12-14-cyclicts_files/figure-html/unnamed-chunk-1-1.png, [przełądane dnia 01.06.2021r]
- [28] https://support.minitab.com/en-us/minitab-express/1/tsplot_weight_with_outlier_single_y_simple_4col.xml_Graph_cmd1o1.png, [przełądane dnia 01.06.2021r]
- [29] https://www.statsmodels.org/stable/plots/graphics_tsa_plot_acf.hires.png, [przełądane dnia 07.06.2021r]
- [30] https://pythondata.com/wp-content/uploads/2019/01/comparison_plt-1024x521.png, [przełądane dnia 08.06.2021r]
- [31] <https://www.researchgate.net/profile/Denis-Butusov/publication/323860289/figure/fig3/AS:606644442517510@1521646714418/An-example-of-a-moving-average-with-the-smoothing-interval-of-1000-points-Time-scale-is.png>, [przełądane dnia 08.06.2021r]

Spis rysunków

Rysunek 2.1 Logo programu MATLAB	s 13
Rysunek 2.2 Schemat pracy w programie MATLAB	s 14
Rysunek 2.3 Histogram wygenerowany za pomocą programu MATLAB	s 15
Rysunek 2.4 Interfejs programu MATLAB	s 16
Rysunek 3.1 Wartości akcji firmy Amazon w latach 1997 – 2020	s 17
Rysunek 3.2 Trend szeregu czasowego.....	s 18
Rysunek 3.3 Szereg czasowy z widoczną sezonowością	s 18
Rysunek 3.4 Szereg czasowy z widocznymi wahaniami cyklicznymi	s 19
Rysunek 3.5 Szereg czasowy z widocznym losowym odchyleniem.....	s 19
Rysunek 3.6 Wykres autokorelacji szeregu czasowego	s 22
Rysunek 3.7 Przykładowa prognoza szeregu czasowego z wykorzystaniem AR.....	s 25
Rysunek 3.8 Wygładzenie przebiegu z wykorzystaniem średniej ruchomej	s 26
Rysunek 4.1 Przykładowa zawartość pliku CSV zawierającego szereg czasowy	s 30
Rysunek 4.2 Przykładowy wykres autokorelacji szeregu czasowego dla 100 poprzednich wartości	s 32
Rysunek 4.3 Przykładowy histogram rozkładu zmiennej szeregu czasowego	s 32
Rysunek 5.1 Schemat architektury aplikacji MVT	s 36
Rysunek 5.2 Projekt interfejsu – widok strony głównej	s 38
Rysunek 5.3 Projekt interfejsu – widok analizy statystycznej	s 39
Rysunek 5.4 Projekt interfejsu – widok wyniku prognozy	s 40
Rysunek 6.1 Struktura plików aplikacji	s 41
Rysunek 6.2 Diagram klasy TimeSeries	s 43
Rysunek 6.3 Widok strony głównej aplikacji	s 46
Rysunek 6.4 Widok analizy statystycznej	s 50
Rysunek 6.5 Widok analizy statystycznej cd.	s 51

Rysunek 6.6 Widok wyniku predykcji metodą AR.....	s 55
Rysunek 6.7 Widok wyniku predykcji metodą ARIMA.....	s 57

Spis listingów

Listing 1. Przykładowy fragment kodu w języku MATLAB.....	s 15
Listing 2. Implementacja formularza do przesyłania pliku w pliku index.html	s 45
Listing 3. Metoda <code>upload_file</code> z pliku <code>routes.py</code>	s 46
Listing 4. Metoda <code>upload_file</code> z pliku <code>file_manager.py</code>	s 47
Listing 5. Metoda <code>analysis</code> ze skryptu <code>routes.py</code>	s 47
Listing 6. Metoda <code>analysis</code> ze skryptu <code>controller.py</code>	s 48
Listing 7. Konstruktor klasy <code>TimeSeries</code>	s 49
Listing 8. Pole <code>info</code> klasy <code>TimeSeries</code>	s 49
Listing 9. Metoda <code>_visualisation</code>	s 50
Listing 10. Formularz metody AR	s 52
Listing 11. Formularz metody ARIMA.....	s 52
Listing 12. Metoda <code>forecast_ar</code>	s 54
Listing 13. Metoda <code>forecast_arima</code>	s 56