**Politechnika Opolska**

Wydział Elektrotechniki, Automatyki i Informatyki

Katedra Informatyki

**PRACA DYPLOMOWA**

Studia pierwszego stopnia

Stacjonarne

Kierunek studiów

*Informatyka*

**TEMAT PRACY**

Projektowanie i implementacja aplikacji mobilnej do zarządzania codziennymi zadaniami z uwzględnieniem interakcji użytkownika   
i elementów UX

|  |  |  |
| --- | --- | --- |
| Promotor:  dr. inż. Łukasz Dzierżanowski |  | Autor:  Mateusz Ferst nr albumu: 102299 |

**Opole 2025**

Projektowanie i implementacja aplikacji mobilnej do zarządzania codziennymi zadaniami z uwzględnieniem interakcji użytkownika i elementów UX

Streszczenie

W pracy przedstawiono projekt systemu z wykorzystaniem biblioteki PyQt5, składającego się z trzech kluczowych części: analizy finansowej, kalkulatora opcji oraz analizy sentymentu z Twittera. Moduł analizy finansowej opiera się ocenie kondycji finansowej wybranych spółek giełdowych. W ramach kalkulatora opcji została zaimplementowana symulacja dwóch głównych strategii handlowych na podstawie danych na żywo. Ostatnia część skupia się na analizie sentymentu na Twitterze z wykorzystaniem modeli uczenia maszynowego takich jak Random Forest, do przewidywania zamian cen akcji.

Designing and implementing a mobile application for managing daily tasks with a focus on user interaction and UX elements

Summary

The work presents a project of a system developed using PyQt5 library, containing of three key components: financial analysis, an option calculator and Twitter sentiment analysis. The financial analysis module focuses on assessing the financial condition of selected stock market companies. The option calculator implements the simulation of two main trading strategies based on live data. The final part focuses on Twitter sentiment analysis using machine learning models such as Random Forest to predict stock price changes.

Spis treści

[Zaawansowany system analizy finansowej z wykorzystaniem PyQt5 2](#_Toc185013874)

[Advanced financial analysis system using PyQt5 2](#_Toc185013875)

[1. Wstęp 5](#_Toc185013876)

[1.1. Motywacja do podjęcia tematu 5](#_Toc185013877)

[1.2. Ograniczenia pracy 5](#_Toc185013878)

[1.3. Charakter i zakres realizacji projektu 6](#_Toc185013879)

[2. Analiza tematu i kontekst teoretyczny 6](#_Toc185013880)

[2.1. Przegląd istniejących narzędzi i rozwiązań 6](#_Toc185013881)

[2.2. Wybór technologii i narzędzi zastosowanych w projekcie 7](#_Toc185013882)

[3. Wymagania i wykorzystywane narzędzia 7](#_Toc185013883)

[3.1. Wymagania niefunkcjonalne 8](#_Toc185013884)

[3.2. Wymagania funkcjonalne 8](#_Toc185013885)

[3.3. Wykorzystywane narzędzia 9](#_Toc185013886)

[4. Architektura systemu 10](#_Toc185013887)

[4.1. Kluczowe założenia systemu 11](#_Toc185013888)

[4.2. Struktura i założenia ogólne 11](#_Toc185013889)

[4.2.1. Moduły systemu i ich funkcje 11](#_Toc185013890)

[4.2.2. Integracja 12](#_Toc185013891)

[4.2.3. Mechanizmy wspierające architekture 12](#_Toc185013892)

[4.3. Wzorce projektowe i logika architektury 13](#_Toc185013893)

[4.3.1. Model-Widok-Kontroler (MVC) 13](#_Toc185013894)

[4.3.2. Model Obserwator 13](#_Toc185013895)

[4.3.3. Wzorzec Fasada 14](#_Toc185013896)

[4.3.4. Integracja wzorców w logice architektury 14](#_Toc185013897)

[4.4. Komunikacja i przepływ danych 15](#_Toc185013898)

[4.5. Skalowalność i modularność 16](#_Toc185013899)

[5. Wymagania sprzętowe i środowisko pracy 16](#_Toc185013900)

[5.1. Instalacja i konfiguracja aplikacji 16](#_Toc185013901)

[5.2. Moduł główny aplikacji 17](#_Toc185013902)

[5.3. Moduł analizy finansowej 19](#_Toc185013903)

[5.4. Moduł kalkulatora opcji 20](#_Toc185013904)

[5.4.1. Wprowadzenie do opcji 20](#_Toc185013905)

[5.4.2. Elementy składowe opcji 21](#_Toc185013906)

[5.4.3. Wycena opcji i model Blacka-Scholesa 22](#_Toc185013907)

[5.4.4. Dane wykorzystywane w kalkulatorze oraz porównanie z bazowym modelem Blacka-Scholesa 23](#_Toc185013908)

[5.4.5. Struktura modułu kalkulatora opcji 23](#_Toc185013909)

[5.5. Moduł analizy sentymentu 30](#_Toc185013910)

[5.5.1. Zbiór danych 30](#_Toc185013911)

[5.5.2. Interfejs użytkownika 31](#_Toc185013912)

[5.5.3. Model Random Forest 32](#_Toc185013913)

[5.5.4. Model Extreme Gradient Boosting 35](#_Toc185013914)

[5.5.5. Model Long Short-Term Memory 36](#_Toc185013915)

[5.5.6. Zestawienie wyników modeli 38](#_Toc185013916)

[6. Podsumowanie i wnioski 39](#_Toc185013917)

[7. Literatura 39](#_Toc185013918)

1. Wstęp

Współczesny świat charakteryzuje się dynamicznym tempem życia, które wymaga od jednostek umiejętnego zarządzania czasem i obowiązkami. Zarówno w życiu prywatnym, jak   
i zawodowym skuteczne planowanie i organizacja zadań stały się nieodzownym elementem, pomagającym sprostać rosnącym wymaganiom dnia codziennego. W tym kontekście, rozwój technologii mobilnych znacząco wpłynął na sposób zarządzania obowiązkami. Smartfony, dzięki swojej wszechstronności i dostępności, stały się narzędziem pierwszego wyboru   
w organizacji codziennych działań.

Mobilne aplikacje do zarządzania zadaniami oferują użytkownikom wsparcie nie tylko w monitorowaniu ich obowiązków, ale także w poprawie produktywności, redukcji stresu   
i budowaniu bardziej zorganizowanego stylu życia. Umożliwiają one łatwy dostęp do harmonogramów, przypomnień i planów, co czyni je nieocenionym wsparciem w codziennych wyzwaniach.

Celem niniejszej pracy dyplomowej jest opracowanie i wdrożenie aplikacji mobilnej, która umożliwi użytkownikom skuteczne zarządzanie codziennymi obowiązkami. Projekt koncentruje się na zapewnieniu użytkownikom intuicyjnej i przyjaznej w obsłudze platformy, która wykorzystuje kluczowe zasady projektowania User Experience (UX). Szczególny nacisk położono na stworzenie interfejsu, który będzie prosty, a jednocześnie funkcjonalny.

## Motywacja do podjęcia tematu

Motywacją do zajęcia się tym tematem jest chęć stworzenia narzędzia, które w prosty i intuicyjny sposób pomaga organizować codzienne obowiązki. Na rynku istnieje wiele rozbudowanych aplikacji do zarządzania zadaniami, ale ich zaawansowane funkcje często idą w parze z nadmierną złożonością, co może zniechęcać użytkowników potrzebujących czegoś prostego i praktycznego. Projekt ten ma na celu wypełnienie tej luki, oferując rozwiązanie, które łączy łatwość obsługi z podstawowymi funkcjami organizacyjnymi. To narzędzie ma pomagać użytkownikom w codziennym planowaniu w sposób przejrzysty i bez zbędnych komplikacji, wspierając bardziej zorganizowany styl życia.

## Ograniczenia pracy

Projekt posiada pewne ograniczenia wynikające z założeń. Aplikacja została stworzona jako narzędzie o podstawowej funkcjonalności, co oznacza, że brakuje w niej zaawansowanych opcji, takich jak integracja z zewnętrznymi kalendarzami, synchronizacja między urządzeniami czy współdzielenie list zadań z innymi użytkownikami. Takie ograniczenia mogą spowodować, że aplikacja nie zaspokoi potrzeb osób poszukujących bardziej zaawansowanych rozwiązań. Aplikacja może działać tylko na platformie mobilnej Android, co sprawia, że osoby korzystające z urządzeń poza tym ekosystemem nie będą miały do niej dostępu. Brak synchronizacji z chmurą oznacza także, że dane są zapisane lokalnie, co może wiązać się z ryzykiem utraty informacji podczas awarii urządzenia.

## Charakter i zakres realizacji projektu

Praca inżynierska została zrealizowana z zastosowaniem powszechnie akceptowanych reguł tworzenia aplikacji mobilnych oraz wytycznych związanych z User Experience. Proces obejmował projektowanie interfejsu użytkownika oraz implementację funkcji aplikacji, kładąc szczególny nacisk na intuicyjność oraz komfort obsługi. Zastosowane podejście opierało się na wykorzystaniu uznanych praktyk projektowania oraz narzędzi programistycznych stworzonych dla platformy Android. Modularna budowa projektu umożliwia proste wprowadzanie zmian   
i rozwijanie aplikacji w przyszłości, co pozwala na dalsze dostosowywanie jej do potrzeb użytkowników oraz ewoluujących wymagań technologicznych.

1. Analiza tematu i kontekst teoretyczny

Aplikacje mobilne zaprojektowane do zarządzania zadaniami, pomimo pozornej prostoty, opierają się czasem na zaawansowanych zasadach projektowania mających na celu zwiększenie interakcji użytkownika i zapewnienie łatwości użytkowania. Z teoretycznego punktu widzenia szczególnie ważne było zagłębienie się w podstawy User Experience, takie jak organizacja informacji, dostępność i zmniejszenie obciążenia poznawczego użytkowników. Zasady te, omawiane w literaturze, stanowią podstawę skutecznego i przyjaznego dla użytkownika projektowania aplikacji mobilnych.

## Przegląd istniejących narzędzi i rozwiązań

Przegląd dostępnych aplikacji, pozwolił na zidentyfikowanie kluczowych cech oraz rozwiązań wspierających efektywne zarządzanie zadaniami. Todoist wyróżnia się funkcjonalnością umożliwiającą organizację i priorytetyzację zadań, co sprawia, że jest szczególnie przydatna dla osób poszukujących prostych, ale skutecznych narzędzi do planowania. Microsoft To Do oferuje intuicyjny interfejs oraz integrację z ekosystemem Microsoft, co czyni ją wygodnym rozwiązaniem dla użytkowników korzystających z innych produktów tej firmy. TickTick, oprócz podstawowych funkcji listy zadań, dostarcza także opcje kalendarza i przypomnień, co umożliwia bardziej kompleksowe podejście do organizacji czasu. Any.do, z kolei, łączy przejrzystość interfejsu z szerokimi możliwościami personalizacji, pozwalając użytkownikom dostosować aplikację do swoich indywidualnych potrzeb.

## Wybór technologii i narzędzi zastosowanych w projekcie

Wybór technologii i narzędzi był ważnym krokiem w procesie projektowania aplikacji mobilnej do zarządzania codziennymi zadaniami. Przy podjęciu decyzji uwzględnio wymagania funkcjonalne, intuicyjność interfejsu oraz zasady projektowania zgodne z UX. Aplikacja została zaprojektowana na platformę Android z wykorzystaniem języka Kotlin, który oferuje nowoczesne podejście do programowania i wysoką czytelność kodu. Lokalna baza danych SQLite, zarządzana za pomocą biblioteki Room, zapewnia stabilne przechowywanie danych i uproszczenie operacji na bazie.

Do budowy interfejsu użytkownika wykorzystano natywne komponenty Androida oraz wytyczne Material Design, co pozwoliło na stworzenie estetycznego i intuicyjnego systemu zarządzania zadaniami. Architektura aplikacji opiera się na wzorcu MVVM (Model-View-ViewModel), co poprawiło strukturę kodu i ułatwiło jego rozwój. W projekcie zastosowano również komponenty Android Jetpack, takie jak LiveData czy ViewModel, które wspierają zarządzanie danymi oraz ich spójność w czasie rzeczywistym. Takie podejście umożliwiło realizację założeń projektu w sposób wydajny i skalowalny.

1. Wymagania i wykorzystywane narzędzia

Ten rozdział opisuje wymagania funkcjonalne i niefunkcjonalne systemu oraz narzędzia i technologie, które będą wykorzystane do ich realizacji. Dobrze zdefiniowane wymagania pomagają w tworzeniu projektu zgodnie z założeniami, który będzie wydajny i otwarty na dalszy rozwój.

## Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają kluczowe aspekty jakościowe aplikacji, które mają wpływ na jej niezawodność, wygodę użytkowania oraz długoterminową użyteczność. Wdrożenie poniższych wymagań zapewnia poprawne działanie systemu oraz pozytywne doświadczenie użytkownika:

* Kompatybilność sprzętowa: Aplikacja powinna działać płynnie na urządzeniach mobilnych z systemem Android w wersji 12.0 (Snow Cone) lub nowszej, spełniających minimalne wymagania sprzętowe.
* Brak konieczności połączenia z internetem: Wszystkie funkcjonalności aplikacji są dostępne w trybie offline dzięki wykorzystaniu lokalnej bazy danych SQLite, co eliminuje zależność od zewnętrznych źródeł danych.
* Responsywność interfejsu: System musi zapewniać szybki czas reakcji na działania użytkownika, a interfejs powinien być dostosowany do różnych rozdzielczości ekranów, gwarantując czytelność i wygodę obsługi.
* Estetyka i użyteczność: Projekt interfejsu powinien być spójny z zasadami Material Design, z naciskiem na intuicyjność oraz minimalizację ryzyka popełnienia błędów przez użytkownika.
* Niezawodność i stabilność: Aplikacja musi działać stabilnie nawet przy dużych ilościach danych, zapewniając brak utraty informacji podczas zapisu lub modyfikacji zadań.
* Skalowalność kodu: Struktura aplikacji oparta na wzorcu MVVM pozwala na łatwe wprowadzanie nowych funkcjonalności i utrzymanie istniejącego kodu.

Spełnienie tych wymagań zapewnia wysoką jakość systemu, jego niezawodność oraz pozytywne wrażenia użytkowników podczas codziennego korzystania z aplikacji.

## Wymagania funkcjonalne

Wymagania funkcjo

zarówno prostotę o Wymagania funkcjonalne opisują funkcjonalności aplikacji, które mają na celu wspieranie użytkownika w zarządzaniu codziennymi zadaniami. Szczegółowe określenie tych wymagań pozwala na precyzyjne zdefiniowanie zakresu działania aplikacji oraz zapewnia jej intuicyjność i efektywność użytkowania.

* Dodawanie zadań: Użytkownik powinien mieć możliwość dodawania nowych zadań, w tym określenia ich nazwy oraz odnawiania zadania według tygodniowego harmonogramu.
* Edycja zadań: Aplikacja umożliwia edytowanie istniejących zadań, aby użytkownik mógł dostosowywać ich szczegóły w zależności od zmieniających się potrzeb.
* Usuwanie zadań: Użytkownik może usunąć zadania, które są już zrealizowane lub nieaktualne.
* Przegląd zadań: Aplikacja powinna zapewniać możliwość wyświetlania listy zadań w sposób uporządkowany, na przykład według daty utworzenia i statusu realizacji.
* Oznaczanie zadań jako wykonane: Użytkownik może oznaczać zadania jako wykonane, co pozwala na lepsze zarządzanie listą i skupienie się na bieżących obowiązkach.
* Tworzenie list zadań: Aplikacja umożliwia użytkownikom tworzenie własnych list zadań, co pozwala na organizację obowiązków w kategorie lub projekty.
* Przypisywanie zadań do list: Użytkownik może przypisywać zadania do konkretnych list, co ułatwia zarządzanie zadaniami i pozwala na lepsze grupowanie zadań według kontekstu lub priorytetu.
* Lokalne przechowywanie danych: Wszystkie dane związane z zadaniami są przechowywane lokalnie w bazie SQLite, co zapewnia ich dostępność w trybie offline.

Wymagania te tworzą solidne podstawy dla aplikacji, która ma na celu efektywne wsparcie użytkowników w codziennym zarządzaniu zadaniami. Przyjęte rozwiązania uwzględniają zarówno prostotę obsługi, jak i niezawodność działania, co przyczynia się do zwiększenia satysfakcji użytkowników oraz poprawy ich produktywności.

## Wykorzystywane narzędzia

Do Podczas tworzenia aplikacji mobilnej do zarządzania codziennymi zadaniami skorzystano z nowoczesnych technologii oraz narzędzi wspierających cały proces, od projektowania, przez implementację, aż po testowanie. Głównym środowiskiem wykorzystywanym do programowania było Android Studio – oficjalne IDE dla platformy Android. Zapewniało ono pełne wsparcie dla języka Kotlin, narzędzia debugujące, edytor graficzny interfejsu użytkownika oraz integrację z emulatorami Androida, co pozwoliło na sprawną implementację oraz weryfikację działania aplikacji.

Do realizacji projektu wybrano język Kotlin, który dzięki swojej zwięzłości, nowoczesnym funkcjom i zgodności z Javą umożliwił efektywne tworzenie kodu. Rozwiązania takie jak rozszerzenia funkcji czy mechanizmy zapobiegające błędom związanym z obsługą wartości null przyczyniły się do poprawy jakości i czytelności kodu. Za zarządzanie lokalną bazą danych odpowiadała biblioteka Room, która uprościła obsługę SQLite, umożliwiając wygodne wykonywanie operacji takich jak dodawanie, modyfikowanie i odczytywanie danych. Dodatkowo Room współpracuje z LiveData, co pozwala na automatyczne aktualizowanie widoków aplikacji przy zmianach w bazie.

Interfejs użytkownika zrealizowano przy użyciu Material Design Components, co umożliwiło stworzenie spójnej, estetycznej i intuicyjnej aplikacji zgodnej z wytycznymi projektowymi Google. Dzięki tym narzędziom interfejs dopasowano do różnych rozdzielczości ekranów, co zapewniło jego poprawne działanie na wielu urządzeniach. Architektura aplikacji została zaprojektowana w oparciu o wzorzec MVVM (Model-View-ViewModel), co w połączeniu z komponentami Android Jetpack, takimi jak LiveData, ViewModel i Navigation Component, ułatwiło zarządzanie danymi oraz cyklem życia aplikacji. Takie podejście zwiększyło przejrzystość kodu oraz umożliwiło łatwe wprowadzanie modyfikacji.

W trakcie pracy korzystano także z systemu kontroli wersji Git oraz platformy GitHub. Pozwoliły one na śledzenie zmian w kodzie, zarządzanie zadaniami oraz ułatwiły współpracę nad projektem. Działanie aplikacji testowano na emulatorach Androida, co umożliwiło dokładną weryfikację funkcjonowania programu na różnych urządzeniach i wersjach systemu, a także identyfikację potencjalnych problemów na wczesnym etapie.

Do tworzenia schematów i prototypów interfejsu użytkownika posłużył program Figma, który dzięki swoim funkcjom pozwolił na efektywne projektowanie zgodne z zasadami UX i Material Design. Narzędzie to okazało się niezwykle przydatne w wizualizacji i dopracowywaniu wyglądu aplikacji jeszcze przed rozpoczęciem implementacji.

Dzięki wykorzystaniu tych narzędzi udało się stworzyć aplikację spełniającą założone wymagania funkcjonalne i niefunkcjonalne, jednocześnie zachowując wysoką jakość, wydajność oraz potencjał do dalszego rozwoju.

1. Architektura systemu

W niniejszym rozdziale zostanie omówiona architektura systemu, obejmująca główne założenia, strukturę oraz sposób działania poszczególnych modułów. Celem jest przedstawienie w sposób przejrzysty logicznej organizacji komponentów systemu, ich wzajemnych powiązań oraz funkcjonalności, które realizują założenia projektu. W dalszej części rozdziału zaprezentowane zostaną szczegóły dotyczące logicznego podziału na moduły, ich roli w systemie oraz sposobu integracji między nimi. System integruje różnorodne technologie i metody analityczne, co czyni go wszechstronnym narzędziem wspierającym inwestorów w podejmowaniu decyzji.

## Kluczowe założenia systemu

System został zaprojektowany w sposób modułowy, co pozwala na łatwą rozbudowę i zarządzanie kodem. Poniżej przedstawiono kluczowe założenia systemu:

* System został stworzony do przeprowadzania zaawansowanej analizy fundamentalnej z użyciem wskaźników finansowych, takich jak wskaźnik ceny do zysku (P/E) oraz wskaźnik ceny do sprzedaży (P/S), co wspiera ocenę kondycji finansowej danej spółki.
* Kalkulator opcji został oparty na modelu Black-Scholesa, umożliwiając symulację dwóch głównych strategii inwestycyjnych oraz analizę potencjalnych wyników w czasie rzeczywistym.
* Moduł analizy sentymentu korzysta z algorytmów uczenia maszynowego do przetwarzania danych z Twittera i ich powiązania z cenami akcji Apple.
* Kod systemu został opracowany zgodnie z dobrymi praktykami programistycznymi, co ułatwia jego czytelność i utrzymanie.

## Struktura i założenia ogólne

Architektura systemu została zaprojektowana z naciskiem na elastyczność, łatwość skalowania oraz prostotę utrzymania, co czyni ją solidnym fundamentem dla zaawansowanej aplikacji analitycznej. Kluczowym założeniem projektu było zastosowanie modularności, gdzie każdy komponent odpowiada za konkretne zadania, a jednocześnie może być rozwijany i modyfikowany niezależnie od pozostałych. Całość opiera się na wzorcu Model-Widok-Kontroler (MVC), który zapewnia przejrzystość w podziale ról między poszczególnymi warstwami aplikacji. Wzorzec MVC pełni centralną rolę w architekturze systemu. Podział aplikacji na warstwy modelu, widoku i kontrolera umożliwia oddzielenie logiki biznesowej od interfejsu użytkownika. Model odpowiada za obsługę danych – ich przetwarzanie i przechowywanie, widok zajmuje się wizualizacją wyników, a kontroler zarządza przepływem informacji między modelem a widokiem, reagując na działania użytkownika. Dzięki zastosowaniu MVC każda warstwa może być rozwijana w izolacji, co pozwala na wprowadzanie zmian w jednej części aplikacji bez konieczności modyfikowania pozostałych. Taka struktura sprzyja również testowaniu oraz poprawie niezawodności kodu.

### Moduły systemu i ich funkcje

System składa się z czterech głównych modułów, które odpowiadają za różne aspekty działania aplikacji:

**Moduł główny aplikacji** – pełni rolę centrum zarządzania, gdzie użytkownik wprowadza dane, takie jak symbol giełdowy, oraz nawigacja do innych funkcji aplikacji. Umożliwia także wyświetlanie wykresów świecowych i podstawowych danych giełdowych, wykorzystując bibliotekę yFinance.

**Moduł analizy finansowej** – realizuje analizę fundamentalną, przetwarzając dane giełdowe pobrane z yFinance. Oblicza wskaźniki takie jak P/S i P/E oraz generuje wizualizacje, które pozwalają użytkownikowi ocenić kondycję finansową wybranej spółki.

**Kalkulator opcji** – służy do symulacji strategii inwestycyjnych przy wykorzystaniu modelu Black-Scholesa . Pobiera dane z API Finnhub, takie jak zmienność, wolumen, aby umożliwić użytkownikowi analizę potencjalnych wyników finansowych.

**Moduł analizy sentymentu** – korzysta z algorytmów uczenia maszynowego, takich jak Random Forest, LSTM czy XGBoost , do analizy danych z mediów społecznościowych oraz przewidywania wpływu nastrojów rynkowych na ceny akcji.

### Integracja

Przepływ danych w systemie został zaprojektowany w sposób logiczny i uporządkowany. Symbol spółki wprowadzony w module głównym zasila moduł analizy finansowej, zapewniając spójność w analizach. Kluczowe założenia architektoniczne, takie jak luźne powiązania między modułami i standaryzacja komunikacji, umożliwiają sprawną wymianę danych między komponentami bez ryzyka zakłóceń i ich działaniu. Poniżej prezentuje się diagram architektury systemu:

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rys 4.1 Diagram architektury systemu, który ilustruje interakcje między modułami

### Mechanizmy wspierające architekturę

System wykorzystuje mechanizmy wielowątkowości i asynchroniczności, co pozwala na jednoczesne pobieranie z różnych źródeł oraz ich przetwarzanie bez obciążania interfejsu użytkownika. Dzięki temu aplikacja pozostaje responsywna nawet podczas wykonywania operacji wymagających dużych zasobów obliczeniowych. Przyjęta architektura systemu, bazująca na wzorcu MVC i podejściu modułowym, zapewnia dużą elastyczność wprowadzania nowych funkcji oraz wysoki poziom organizacji kodu.

## Wzorce projektowe i logika architektury

W niniejszym podrozdziale zostaną przedstawione wzorce projektowe, które stanowią podstawę zaprojektowanego systemu oraz ich wpływ na strukturę i działanie aplikacji. Zastosowane rozwiązania nie tylko usprawniają komunikację między poszczególnymi elementami systemu, ale także umożliwiają efektywne zarządzanie przepływem danych i obsługę złożonych operacji. Podrozdział skupi się na wzorcach projektowych, takich jak Model-Widok-Kontroler (MVC), wzorzec Obserwator oraz Fasada, które odegrały kluczową rolę w budowie systemu. Zostanie także omówione zastosowanie asynchroniczności i wielowątkowości jako narzędzi zwiększających wydajność i płynność działania aplikacji. Analiza pokaże, jak te wzorce i mechanizmy zostały wykorzystane do stworzenia stabilnego i funkcjonalnego rozwiązania.

### Model-Widok-Kontroler (MVC)

Aplikacja została zaprojektowana w oparciu o wzorzec Model-Widok-Kontroler, co umożliwia klarowny podział odpowiedzialności pomiędzy interfejsem użytkownika (widok), logiką biznesową (model) oraz logiką sterowania (kontroler). Takie podejście ułatwia zarządzanie kodem, testowanie i rozwój aplikacji, jednocześnie zapewniając spójność strukturalną w całym systemie. Każdy moduł aplikacji działa zgodnie z zasadami MVC, co wspiera jednolitą organizację.

**Widok** odpowiada za prezentację danych i interakcję z użytkownikiem, wykorzystując bibliotekę PyQt5. Dzięki jej komponentom, takim jak przyciski, pola tekstowe czy dynamiczne wykresy, użytkownik może łatwo nawigować po funkcjach aplikacji, takich jak analiza finansowa czy kalkulator opcji. Przykładem jest wykres świecowy z danymi z yFinance, który umożliwia wizualizację zmian cen akcji, lub tabela z wynikami symulacji w kalkulatorze opcji.

**Kontroler** pośredniczy między widokiem a modelem, obsługując zdarzenia generowane przez użytkownika, takie jak wprowadzanie symbolu giełdowego czy uruchomienie analizy. Przetwarza dane wejściowe, wywołuje odpowiednie metody w modelu i aktualizuje widok na podstawie uzyskanych wyników.

**Model** zarządza danymi i logiką biznesową, obsługując komunikację z zewnętrznymi źródłami, takimi jak yFinance czy Finnhub. W module analizy finansowej model przetwarza dane o wskaźnikach, takich jak P/E i P/S.

### Model Obserwator

W zaprojektowanym systemie zastosowano również wzorzec Obserwator, który umożliwia dynamiczną aktualizację widoku w odpowiedzi na zamiany danych. Dzięki temu interfejs użytkownika jest zawsze zsynchronizowany z modelem, co znacząco poprawia responsywność aplikacji. W systemie Obserwator działa na zasadzie powiązania między modelem, który zarządza danymi a widokiem, który je prezentuje. Mechanizm ten odgrywa kluczową rolę w takich obszarach jak aktualizacja wykresów świecowych czy prezentacji wyników kalkulatora opcji. Mechanizm działania wzorca opiera się na rejestracji widoków i obserwatorów modelu. Model przechowuje listę widoków i wysyła im powiadomienia o każdej zmianie w danych. Widoki reagują na te powiadomienia, odświeżając interfejs użytkownika. Zastosowanie wzorca Obserwator przynosi wiele korzyści. Dzięki automatycznej aktualizacji danych użytkownik zawsze widzi aktualne informacje bez konieczności ręcznego odświeżania interfejsu, co zwiększa płynność działania aplikacji. W praktyce wzorzec ten wspiera kluczowe moduły aplikacji. Na przykład w module kalkulatora opcji model dostarcza wyniki strategii inwestycyjnych bezpośrednio do widoku, co pozwala użytkownikowi na natychmiastowe analizowanie danych.

### Wzorzec Fasada

W zaprojektowanej aplikacji wzorzec Fasada został wykorzystany w celu uproszczenia komunikacji między modułami systemu a zewnętrznymi bibliotekami, takimi jak yFinance czy Finnhub API. Dzięki temu wzorcowi inne komponenty aplikacji, takie jak kalkulator opcji czy moduł analizy finansowej, mogą korzystać z danych giełdowych w uproszczonej ujednoliconej formie, bez potrzeby zajmowania się szczegółami implementacyjnymi czy złożonością obsługi zewnętrznych źródeł. Wzorzec Fasada odgrywa kluczową rolę jako warstwa pośrednicząca, która zapewnia:

* **Ukrycie złożoności**: Fasada maskuje szczegóły związane z pobieraniem i przetwarzaniem danych, oferując jednolity interfejs dla pozostałych modułów. Na przykład moduł wizualizacji nie musi zarządzać skomplikowaną logiką przetwarzania danych z yFinance – wszystkie niezbędne informacje dostarczone są w przystępnej formie.
* **Centralizację logiki**: Dzięki fasadzie wszystkie operacje związane z zarządzaniem danymi odbywają się w jednym miejscu, co pozwala na efektywną organizację kodu i wprowadzanie zmian.

W module kalkulatora opcji kiedy użytkownik wprowadza symbol spółki i datę wygaśnięcia opcji, fasada pobiera odpowiednie dane z Finnhub API, przetwarza je (np. oblicza wartości średnie bid/ask) i zwraca gotowy zestaw informacji dla kalkulatora opcji. W module analizy finansowej fasada integruje dane z yFinance, przetwarzając informacje o cenach otwarcia, zamknięcia, maksimum i minimum dla wybranej spółki. Dane te są następnie przygotowywane w formacie umożliwiającym łatwe generowanie wykresów. Wzorzec Fasada upraszcza kod, separując logikę przetwarzania danych od interfejsu użytkownika, co sprawiam że system jest bardziej czytelny i łatwy w zarządzaniu. Dzięki centralizacji logiki zmiany w bibliotekach czy źródłach danych wymagają modyfikacji wyłącznie w fasadzie, co zwiększa elastyczność aplikacji.

### Integracja wzorców w logice architektury

W architekturze zaprojektowanego systemu integracja wzorców projektowych odgrywa kluczową rolę w zapewnieniu spójnego i efektywnego działania aplikacji. Implementacja modelu MVC w module głównym wygląda następująco:

* Model: Klasa AppState pełni kluczową rolę w zarządzaniu stanem aplikacji, przechowując dane takie jak aktualnie wybrany symbol giełdowy.
* Widok: Klasa MainWindow odpowiada za tworzenie i obsługę interfejsu użytkownika z wykorzystaniem PyQt5, natomiast MplCanvas służy do graficznej prezentacji danych giełdowych w postaci wykresów. Widok umożliwia użytkownikowi wprowadzanie danych i wizualizacje wyników analiz.
* Kontroler: W klasie MainWindow znajdują się metody zdarzenia, takie jak search\_stock() i update\_data(). Ich zadaniem jest reagowanie na działania użytkownika, inicjowanie odpowiednich operacji w modelu oraz aktualizacja widoku.

Wzorzec Obserwator został zaimplementowany w systemie przy użyciu mechanizmów sygnałów i slotów dostępnych w PyQt5, co umożliwia automatyczną synchronizację widoku z modelem. Przykładowo w klasie AppState następuje zmiana symbolu spółki, widok otrzymuje powiadomienie o tej zmianie i automatycznie aktualizuje wykres. W module kalkulatora opcji zastosowano wzorzec Fasada poprzez klasę OptionDataLoader. Klasa ta upraszcza interakcję z zewnętrznym API Finnhub, ukrywając złożoność komunikacji i oferując prosty interfejs. Dzięki temu pozostałe komponenty mogą korzystać z danych opcji bez konieczności znajomości szczegółów technicznych API.

## Komunikacja i przepływ danych

W niniejszym podrozdziale omówiony zostanie mechanizm przepływu danych w systemie oraz sposób, w jaki poszczególne moduły współpracują ze sobą. Moduł główny aplikacji przyjmuje symbol spółki w polu wyszukiwania. Na podstawie tego symbolu moduł pobiera dane giełdowe z zewnętrznego źródła (biblioteki yFinance). Dane są przechowywane w klasie AppState, która odpowiada za reprezentację bieżącego stanu aplikacji. Widok MainWindow aktualizuje następnie wykresy i tabele, prezentując użytkownikowi przetworzone wyniki. W module analizy finansowej przepływ danych przebiega w podobny sposób. Moduł główny przekazuje symbol spółki do modułu analizy finansowej, który na jego podstawie inicjuje pobieranie danych finansowych z biblioteki yFinance. Następnie, pomniejsze komponenty wykonują odpowiednie obliczenia, takie jak wyliczanie wskaźników finansowych. Wyniki są przetwarzane przez generator wykresów, który przygotowuje wizualizacje. Gotowe wykresy są prezentowane w widoku głównym modułu. W kalkulatorze opcji użytkownik wprowadza symbol spółki, co skutkuje dynamicznym pobieraniem aktualnej ceny akcji z API. Użytkownik wybiera następnie datę wygaśnięcia oraz cenę wykonania kontraktu. Na podstawie tych danych moduł pobiera szczegółowe informacje o wybranym kontrakcie z API. Wyniki symulacji strategii są przedstawiane w formie tabeli z nałożonym gradientem kolorystycznym. Moduł analizy sentymentu wykorzystuje dane historyczne zgromadzone z platformy Kaggle, które zostały wcześniej rozszerzone o dodatkowe dane, związane z realnymi wartościami spółki Apple przy użyciu biblioteki yFinance. Na tym finalnym zbiorze zastosowano algorytmy uczenia maszynowego takie jak Random Forest, czy XGBoost, aby zbadać sentyment.

## Skalowalność i modularność

Skalowalność i modularność są kluczowymi cechami architektury zaprojektowanego systemu, które pozwalają na jego rozwój oraz dostosowanie do zmieniających się wymagań użytkowników. System został skonstruowany jako zbiór niezależnych modułów, z których każdy odpowiada za określoną funkcjonalność. Taki podział pozwala na rozwijanie, testowanie i wdrażanie poszczególnych elementów niezależnie, co znacząco przyspiesza proces tworzenia oprogramowania i ułatwia zarządzanie projektem. Modularność umożliwia łatwe diagnozowanie i eliminowanie błędów w wybranych częściach systemu, bez wpływu na pozostałe komponenty. Architektura została zaprojektowana z myślą o łatwym skalowaniu zarówno pod względem funkcjonalności, jak i wydajności. Struktura systemu wspiera integrację dodatkowych modułów, które mogą wprowadzać nowe funkcje takie jak analiza techniczna, prognozowanie czy alerty. Jednocześnie istniejące moduły mogą być modyfikowane i rozbudowywane, bez zakłócania działania innych części systemu. Wykorzystanie mechanizmów wielowątkowości pozwala na równoległe przetwarzanie danych i obsługę wielu operacji jednocześnie, co zwiększa wydajność aplikacji. Moduły komunikują się poprzez zdefiniowane interfejsy, co minimalizuje obciążenie systemu i optymalizuje wykorzystanie zasobów. Pobieranie danych z zewnętrznych źródeł odbywa się asynchronicznie, co pozwala na nieprzerwane działanie interfejsu i szybkie reagowanie na działanie użytkownika. Zastosowanie technologii takich jak Numpy czy Pandas wspomaga efektywne przetwarzanie danych i zarządzanie zasobami. Przyjęta architektura systemu oparta na modularności zapewnia solidną podstawę dla dalszego rozwoju aplikacji. Dzięki niej system może być łatwo rozszerzany o nowe funkcjonalności.

1. Wymagania sprzętowe i środowisko pracy

Aby uruchomić aplikację, użytkownik musi posiadać komputer z systemem operacyjnym, aplikację oraz dostęp do Internetu. Minimalne wymagania sprzętowe obejmują standardowy procesor wielordzeniowy, 8 GB pamięci RAM oraz około 3 GB wolnego miejsca na dysku, co wystarcza do płynnego działania aplikacji. Jednak bez stabilnego połączenia internetowego aplikacja nie będzie w stanie efektywnie funkcjonować, ponieważ większość modułów opiera się na danych z zewnętrznych źródeł. Te źródła dostarczają zarówno bieżące, jak i historyczne dane niezbędne do analiz oraz kalkulacji w kalkulatorze opcji. Bez aktywnego połączenia z Internetem użytkownik będzie mógł jedynie korzystać z wizualizacji wyników analiz uczenia maszynowego przy załadowanych wcześniej danych. Aplikacja została zaprojektowana z myślą o maksymalnej elastyczności, działając w trybie okienkowym i dostosowując swój interfejs do rozdzielczości ekranu użytkownika, co zapewnia wygodę i czytelność podczas pracy.

## Instalacja i konfiguracja aplikacji

Aby uruchomić aplikację, użytkownik musi najpierw pobrać jej kod źródłowy i zainstalować wymagane zależności. Proces instalacji został zaprojektowany z myślą o łatwości obsługi, zarówno dla osób z doświadczeniem w programowaniu, jak i dla użytkowników o podstawowej wiedzy technicznej. Kluczowym krokiem jest wcześniejsze zainstalowanie Pythona oraz edytora kodu, takiego jak Visual Studio Code. Przygotowanie środowiska pracy obejmuje instalację Pythona i Visual Studio Code, które można pobrać z ich oficjalnych stron internetowych. Python powinien być w wersji 3.9 lub nowszej. Podczas instalacji należy zaznaczyć opcję dodania Pythona do zmiennej systemowej PATH, co umożliwi korzystanie z niego w terminalu. Po skonfigurowaniu Pythona należy zainstalować Visual Studio Code, a następnie dodać do niego rozszerzenie Python, które ułatwia pracę z językiem, m.in. poprzez obsługę środowisk wirtualnych, linterów i podpowiedzi kodu. Instalacja aplikacji polega na pobraniu plików źródłowych z repozytorium GitHub. Proces rozpoczyna się od pobrania projektu, co można zrobić poprzez przejście na stronę repozytorium {LINK ? EngineeringThesis }, kliknięcie przycisku Code i wybranie opcji Download ZIP. Po pobraniu pliku należy go rozpakować w wybranym miejscu na komputerze. Alternatywnie można skorzystać z narzędzia Git, które umożliwia szybsze i łatwiejsze pobranie projektu za pomocą komendy w terminalu:

1. git clone https://github.com/maggyy666/EngineeringThesis.git

Po pobraniu plików projekt należy otworzyć w edytorze Visual Studio Code, wybierając opcję File > Open Folder, i upewnić się, że Python w wersji 3.9 lub nowszej jest zainstalowany w systemie. Kolejnym krokiem jest instalacja wymaganych bibliotek, których lista znajduje się w pliku requirements.txt w folderze projektu. W tym celu należy otworzyć terminal w Visual Studio Code i wykonać polecenie:

1. pip install -r requirements.txt

Gdy wszystkie zależności zostaną zainstalowane, aplikację można uruchomić za pomocą polecenia:

1. python app\_main.py

Po poprawnym wykonaniu tej komendy główne okno aplikacji zostanie wyświetlone, umożliwiając użytkownikowi korzystanie z pełnej funkcjonalności systemu. Skorzystanie z narzędzia Git do pobrania projekt może dodatkowo ułatwić przyszłe aktualizacje aplikacji.

## Moduł główny aplikacji

Moduł główny aplikacji umożliwia użytkownikowi interakcję z systemem oraz dostęp do innych kluczowych modułów, takich jak analiza finansowa, kalkulator opcji czy moduł analizy sentymentu. Kluczowe cele tego modułu to zapewnienie intuicyjnego interfejsu użytkownika oraz integracja danych pobieranych z zewnętrznych źródeł, w tym wypadku yFinance. Interfejs użytkownika został zaprojektowany przy użyciu biblioteki PyQt5, która oferuje podstawowe komponenty GUI. Jednakże estetyczny i spójny wygląd uzyskano dzięki zastosowaniu pliku stylów styles.qss. Plik ten definiuje stylizację poszczególnych elementów interfejsu, takich jak tło okna, kolory tekstu, obramowania czy efekty hover. Dzięki oddzieleniu stylów w osobnym pliku uniknięto nadmiernego polegania na domyślnych ustawieniach PyQt5. Na górze okna aplikacji znajduje się pasek nawigacyjny, który umożliwia szybkie przełączanie pomiędzy modułami. Dostępne są opcje, takie jak „Financial Analysis”, „Options Calculator” oraz „Sentiment ML”. Kliknięcie odpowiedniego przycisku powoduje otwarcie nowej instancji aplikacji. Poniżej zaprezentowano wygląd głównego modułu aplikacji, który zawiera dynamiczny wykres świecowy oraz tabelę spółek:

Obraz zawierający zrzut ekranu, Oprogramowanie graficzne, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Rys 5.1 Wygląd głównego modułu aplikacji

Pod paskiem nawigacyjnym znajduje się główny obszar roboczy, w którym wyświetlany jest wykres świecowy obrazujący zmiany cen akcji wybranej spółki. Do stworzenia tego elementu wykorzystano bibliotekę mplfinance, która pozwala na wydajne generowanie interaktywnych wykresów z dynamicznie zmieniającym się zakresem danych. Scrollowanie myszą umożliwia zarówno przybliżanie, jak i oddalanie widoku wykresu, co automatycznie dostosowuje zakres danych. Przy większym oddaleniu wyświetlany jest dłuższy okres historyczny, natomiast przybliżenie zawęża widok do krótszego przedziału czasowego. Pole wyszukiwania umieszczone bezpośrednio pod wykresem umożliwia użytkownikowi wprowadzanie symbolu giełdowego spółki, dla której mają zostać pobrane dane. Po wprowadzeniu symbolu i zatwierdzeniu wyboru aplikacja automatycznie pobiera dane dotyczące wskazanej spółki. Wyświetlane informacje są następnie aktualizowane dynamicznie na wykresie, oraz w sekcji znajdującej się w lewym górnym rogu wykresu, obejmującej bieżącą cenę akcji. Tabela spółek, umieszczona na dole aplikacji, przedstawia szczegółowe dane finansowe dotyczące wybranych spółek giełdowych, takich jak Apple, Microsoft czy Tesla. Głównym celem tabeli jest dostarczenie informacji o kapitalizacji rynkowej, wskaźnikach finansowych czy liczbie pracowników. Po kliknięciu na wybraną spółkę w tabeli, aplikacja automatycznie aktualizuje wykres świecowy, prezentując dane związane z wybraną spółką. Mechanizm nawigacji między modułami został zrealizowany poprzez wywołanie osobnych procesów dla każdego modułu. Dzięki temu aplikacja główna działa niezależnie od innych funkcjonalności. Przykładowo, kliknięcie opcji „Financial Analysis” powoduje uruchomienie odrębnego modułu analizy finansowej. W trakcie tego procesu aktualnie wybrany symbol giełdowy jest przekazywany za pomocą klasy AppState, która centralizuje zarządzanie stanem aplikacji.

## Moduł analizy finansowej

Moduł analizy finansowej w aplikacji został zaprojektowany z myślą o zapewnieniu użytkownikowi kompleksowych informacji na temat stanu finansowego wybranej spółki giełdowej. Funkcjonalność tego modułu obejmuje prezentację kluczowych danych finansowych, takich jak struktura kapitału, ocena wartości rynkowej, wskaźnik wzrostu i rentowności, analiza dywidend oraz ogólna kondycja finansowa spółki. Informacje te są przedstawiane za pomocą wykresów, które użytkownik może wygodnie przeglądać w ramach jednego okna aplikacji. Implementacja modułu opiera się na integracji kilku kluczowych komponentów. Pierwszy etapem jest pobranie danych finansowych danej spółki za pomocą biblioteki yFinance. Następnie dane te są przetwarzane przez dedykowane moduły analityczne, takie jak dividend\_analysis czy growth\_and\_profitability, z których każdy odpowiada za obliczanie specyficznych wskaźników finansowych. Wyniki tych obliczeń są przekazywane do generatora wykresów (plot\_generator), który przekształca dane liczbowe w różnorodne formy wizualne, takie jak wykresy kołowe, słupkowe czy liniowe. Kluczową cechą modułu analizy finansowej jest jego podział na sekcje tematyczne, z których każda odpowiada za wizualizację danych dotyczących określonego aspektu działalności spółki [10]. Poniżej przedstawione wybrane sekcje tego modułu:

Sekcja struktury kapitału przedstawia dane dotyczące udziałów akcji wolnorynkowych oraz kontrolowanych, w tym ich procentowy udział w całkowitej kapitalizacji rynkowej.

Sekcja wartości rynkowej umożliwia porównanie kapitalizacji rynkowej z innymi kluczowymi wskaźnikami, takimi jak przychody netto, P/E (Price-to-Earnings) oraz P/S (Price-to-Sales).

Sekcja dotycząca dywidend zawiera szczegóły historyczne dotyczące wypłat dywidednd, takie ja wskaźniki wypłaty, regularność wypłat oraz terminy ostatnich wypłat. Te dane są szczególnie przydatne dla inwestorów zainteresowanych spółkami dywidendowymi [11].

Poniższej przedstawiono widok modułu analizy finansowej dla symbolu „AAPL” skupiający się na sekcji stabilności finansowej (ang. Financial Health):

Obraz zawierający zrzut ekranu, Oprogramowanie graficzne, Oprogramowanie multimedialne, modelowanie 3D

Opis wygenerowany automatycznie

Rys 5.2 Widok modułu analizy finansowej dla symbolu "AAPL"

Dla zobrazowania działania modułu zaprezentowany zostanie przykład analizy dywidend na podstawie spółki Apple (AAPL). Przedstawiony proces obejmuje pobieranie danych, obliczanie wskaźników oraz ich wizualizację. Moduł pozyskuje dane z biblioteki yfinance na temat dywidend. W przypadku spółki AAPL zebrano następujące informacje:

**Dividend Yield TTM**: 0.35% – stopa dywidendy na przestrzeni ostatnich 12 miesięcy.

**Last Payment**: $0.50 – ostatnia wypłata dywidendy na jedną akcję.

**Payout Ratio**: 7.08% – procent zysków przeznaczonych na dywidendy.

**Earnings Retained**: 92.92% – procent zysków zatrzymanych w firmie na reinwestycje.

Na podstawie zebranych danych obliczono kluczowe wskaźniki:

**Stopa dywidendy**: Wyliczana jako stosunek dywidendy na akcję do średniej ceny akcji. W przypadku Apple wynosi jedynie 0.35%, co sugeruje, że firma nie jest typowym wyborem dla inwestorów poszukujących wysokiego dochodu pasywnego.

**Payout Ratio i Earnings Retained**: Wysokie zatrzymanie zysków (92.92%) wskazuje, że Apple przeznacza większość swoich zysków na rozwój i ekspansję, a nie na wypłatę dywidend.

**Historia wypłat na akcję**: Analiza pokazuje stabilność wypłat na przestrzeni lat, co może być atrakcyjne dla inwestorów preferujących przewidywalność.

Obliczone wskaźniki zostałyp rzedstawione za pomocą dedykowanych wykresów takich jak Dividend Pie Chart, oraz Dividend History Chart:

Obraz zawierający zrzut ekranu, Oprogramowanie multimedialne, Oprogramowanie graficzne, oprogramowanie

Opis wygenerowany automatycznie

**Rys 5.3** Wizualizacja wskaźników dotyczących dywidend w module analizy finansowej dla symbolu „AAPL”

## Moduł kalkulatora opcji

Kalkulator opcji to kluczowy moduł aplikacji przeznaczony do analizy instrumentów finansowych, jakimi są opcje. Jego głównym celem jest umożliwienie użytkownikowi obliczanie wartości opcji europejskich typu CALL i PUT, symulacji potencjalnych zysków i strat w różnych scenariuszach cenowych oraz zrozumienia wpływu istotnych parametrów rynkowych na wycenę opcji. Zanim jednak nastąpi szczegółowe omówienie technicznych aspektów kalkulatora oraz jego implementacji, konieczne jest przedstawienie podstawowych informacji o opcjach finansowych. W tym kontekście zostaną omówione ich klasyfikacja, podstawowe rodzaj (CALL i PUT), a także fundamenty ich wyceny, co pozwoli lepiej zrozumieć znaczenie i zakres funkcjonowania oferowanych przez moduł kalkulatora.

### Wprowadzenie do opcji

Opcje to jedne z kluczowych instrumentów pochodnych, których wartość zależy od innego aktywa, zwanego instrumentem bazowym (ang. underlying asset). Są to kontrakty zawierane między dwiema stronami, które dają nabywcy prawo, ale nie nakładają obowiązku, do kupna lub sprzedaży określonego instrumentu bazowego w przyszłości po ustalonej wcześniej cenie, nazywanej ceną wykonania (ang. strike price). W zamian za to prawo nabywca płaci sprzedawcy określoną kwotę, zwaną premią opcyjną (ang. premium). Dzięki swojej specyfice opcje oferują inwestorom wyjątkową elastyczność, umożliwiając budowanie złożonych strategii inwestycyjnych. Opcje można podzielić na dwa podstawowe rodzaje, które różnią się pod względem kierunku oczekiwanego ruchu cen np. akcji spółki:

* Opcje typu CALL (kupna): Opcja CALL uprawnia jej posiadacza do zakupu akcji spółki w przyszłości po ustalonej wcześniej cenie wykonania. Tego typu opcja jest korzystna w sytuacji, gdy cena akcji wzrasta powyżej ceny wykonania, ponieważ pozwala nabyć aktywa po niższej cenie niż rynkowa. Przykładowo, jeżeli cena wykonania wynosi 100 USD, a cena rynkowa akcji 120 USD, to zysk wynosi 20 USD.
* Opcje typu PUT (sprzedaży): Opcja PUT daje posiadaczowi takie samo prawo sprzedaży akcji w przyszłości po wcześniej ustalonej cenie. Tego rodzaju opcja korzystna jest, gdy cena akcji spada poniżej ceny wykonania, ponieważ pozwala sprzedać aktywa po wyższej cenie.

### Elementy składowe opcji

Elementy składowe opcji to kluczowe aspekty, które definiują charakterystykę oraz wartość każdego kontraktu opcyjnego. Zrozumienie tych elementów jest niezbędne do skutecznej analizy i wyceny opcji.

* Instrument bazowy (ang. underlying asset): Aktywo, którego cena decyduje o wartości opcji. Mogą to być np. akcje(AAPL – Apple), indeksy giełdowe (S&P 500), waluty (USD/EUR), surowce (złoto, ropa) lub inne aktywa finansowe.
* Cena wykonania (ang. strike price): Wartość, po której posiadacz opcji ma prawo kupić (CALL) lub sprzedać (PUT) np. akcje danej spółki. Jest ustalana w momencie zawarcia kontraktu i pozostaje niezmienna przez cały okres jego trwania.
* Data wygaśnięcia (ang. expiration date): Ostateczny termin, w którym opcja może być wykonana. Po tej dacie opcja wygasa i traci swoją wartość, jeśli nie została wykorzystana. Okresy wygaśnięcia mogą wynosić od jednego dnia (opcje jednodniowe) do kilku lat (opcje długoterminowe).
* Cena opcji (premium): Kwota, którą nabywca opcji płaci jej wystawcy za prawo wynikające z kontraktu. Premia zależy od wielu czynników, w tym ceny akcji spółki, czasu wygaśnięcia, zmienności rynkowej oraz stóp procentowych.
* Współczynniki opcyjnie (ang. greeks): Parametry, które pokazują wrażliwość ceny opcji na różne zmienne rynkowe:

Delta(Δ): Określa, jak zmienia się cena opcji w reakcji na zmianę ceny akcji.

Gamma(Γ): Opisuje zmianę delty w odpowiedzi na zmiany ceny bazowej.

Theta(Θ): Wskazuje, jak upływ czasu wpływa na wartość opcji (spadek wartości w czasie).

Vega(ν): Pokazuje, jak zmienia się cena opcji w reakcji na zmienność rynku.

Rho(ρ): Określa wrażliwość ceny opcji na zmiany stóp procentowych.

Zmienność rynku (ang. implied volatility):

Wskazuje na oczekiwany zakres wahań ceny akcji w przyszłości. Wyższa zmienność zwiększa wartość opcji, ponieważ rośnie prawdopodobieństwo, że cena osiągnie poziom korzystny dla posiadacza kontraktu.

Każdy z tych elementów odgrywa kluczową rolę w określeniu wartości oraz zachowania opcji w różnych warunkach.

### Wycena opcji i model Blacka-Scholesa

Model Blacka-Scholesa, opracowany przez Fishera Blacka i Myrona Scholesa w 1973 roku, to fundamentalne narzędzie służące do wyceny opcji europejskich typu CALL i PUT. Opiera się on na założeniu, że ceny aktywów podążają za geometrycznym ruchem Browna, co oznacza, że zmiany cen są losowe, a ich logarytmy mają rozkład normalny. Model ten zrewolucjonizował sposób wyceny opcji, oferując matematyczne podejście do analizy wartości. Dzięki swojej prostocie i precyzji stał się jednym z najczęściej stosowanych narzędzi w finansach, umożliwiając inwestorom lepsze zrozumienie dynamiki cen opcji. Model Blacka-Scholesa dostarcza wzory na wycenę opcji CALL:

|  |  |
| --- | --- |
|  | (3.1) |

Opcja PUT:

|  |  |
| --- | --- |
|  | (3.2) |

Gdzie:

* – cena opcji CALL,
* – cena opcji PUT,
* – bieżąca cena akcji,
* – cena wykonania,
* – czas do wygaśnięcia (w latach),
* – stopa wolna od ryzyka,
* – zmienność implikowana,
* – standardowa dystrybucja rozkładu normalnego, opisana wzorem:
* – zmienne pośrednie obliczane według wzorów:

|  |  |
| --- | --- |
|  | (3.3) |

|  |  |
| --- | --- |
|  | (3.4) |
|  |  |

W modelu Blacka-Scholesa oraz reprezentują prawdopodobieństwo związane z ruchem ceny akcji w kierunku korzystnym dla posiadacza opcji. Funkcje te uwzględniają zmienność () oraz czas do wygaśnięcia opcji, co pozwala na realistyczną ocenę ryzyka i potencjalnych zysków. Model uwzględnia również wpływ wartości pieniądza w czasie poprzez czynniki dyskontowania (), gdzie oznacza stopę wolną od ryzyka, a czas do wygaśnięcia opcji. W praktyce rynkowej podstawowy model Blacka-Scholesa jest często modyfikowany, aby uwzględnić dodatkowe czynniki. Na przykład model Mertona wprowadza korekty związane z wypłatami dywidend. Model Hestona z kolei umożliwia uwzględnianie zmiennej zmienności (), co lepiej odzwierciedla rzeczywiste warunki rynkowe. W zaimplementowanym kalkulatorze opcji wykorzystano klasyczną wersję modelu Blacka-Scholesa, która nie uwzględnia wypłat dywidend..

### Dane wykorzystywane w kalkulatorze oraz porównanie z bazowym modelem Blacka-Scholesa

W poniższym fragmencie omówione zostaną różnice pomiędzy założeniami klasycznego modelu Blacka-Scholesa a implementacją kalkulatora opcji w aplikacji. Analiza koncentruje się na kluczowych elementach takich jak zmienność implikowana czy stała stopa procentowa.

Stała zmienność implikowana ()

W kalkulatorze przyjęto założenie, że zmienność implikowana pozostaje stała przez cały okres obowiązywania opcji. Zgodność z założeniem modelu Blacka-Schoesa polega na traktowaniu zmienności jako wartości stałej i znanej, co upraszcza wycenę. Ograniczenie to jest charakterystyczne dla tego modelu, który nie uwzględnia dynamicznych zmian zmienności w rzeczywistych warunkach.

Stała stopa procentowa ():

Kalkulator zakłada stałą wartość stopy procentowej wynoszącą 1% rocznie. To założenie jest zgodne z bazowym modelem, w którym stopa wolna od ryzyka pozostaje stała w całym okresie do wygaśnięcia opcji. Implementacja kalkulatora pomija potencjalne wahania stóp procentowych, co odpowiada teoretycznym założeniom modelu.

Brak uwzględnienia dywidend

Zarówno kalkulator, jak i podstawowy model przyjmują, że instrument bazowy (np. akcje) nie generuje wypłat w formie dywidend. Ograniczenie to oznacza, że model najlepiej sprawdza się dla aktywów takich jak obligacje czy fundusze ETF, które nie wypłacają dywidend.

Typ opcji – wyłącznie europejskie

Kalkulator wspiera wyłącznie wycenę opcji europejskich, które można zrealizować jedynie w dniu wygaśnięcia.

### Struktura modułu kalkulatora opcji

Moduł kalkulatora opcji został zaprojektowany jako narzędzie umożliwiające przeprowadzenie szczegółowej analizy dwóch kluczowych strategii inwestycyjnych – Long CALL oraz Long PUT. Obie strategie pozwalają na zrozumienie mechanizmu działania opcji oraz przewidywanie potencjalnych zysków i strat w zależności od zmian cen akcji oraz upływu czasu. W ramach analizy opcji CALL kalkulator umożliwia ocenę i symulację scenariuszy zakładających wzrost ceny akcji powyżej ustalonej ceny wykonania. Analogicznie, moduł dla opcji PUT pozwala na zbadanie sytuacji, w których spadek ceny akcji przynosi potencjalny zysk. Wyniki analiz obejmują kluczowe wskaźniki, takie jak punkt przełamania (ang. break even), maksymalne ryzyko oraz potencjalny zysk w różnych scenariuszach. Działanie modułu rozpoczyna się od wprowadzenia symbolu akcji, który identyfikuje instrument bazowy. Po wprowadzeniu symbolu system automatycznie łączy się z zewnętrznym API w celu pobrania aktualnej cenyt akcji, co zapewnia dostęp do najnowszych danych rynkowych. Na tym etapie użytkownikowi wyświetlana jest bieżąca wartość instrumentu bazowego oraz lista dostępnych dat wygaśnięcia opcji, umożliwiająca wybór odpowiedniego terminu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Rys 5.4 Główne okno aplikacji po wpisaniu symbolu spółki, z zaktualizowaną ceną akcji.

Kolejnym krokiem jest wybór ceny wykonania opcji (ang. strike price), która stanowi kluczowy parametr determinujący wartość tego instrumentu. Dla uproszczenia procesu kalkulator prezentuje tabelę z podziałem na opcje typu CALL oraz PUT dla wybranej daty wygaśnięcia.

Tabela zawiera szczegółowe informacje, takie jak ceny bid, mid i ask, oraz odpowiadające im ceny wykonania.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Rys 5.5 Tabela opcji CALL i PUT z cenami bid, mid, ask oraz strike price dla wybranej daty wygaśnięcia.

Po wyborze opcji użytkownik może skonfigurować dodatkowe parametry, takie jak liczba kontraktów oraz typ działania – zakup opcji (ang. Buy) lub jej wystawienie (ang. Write). Użytkownik ma również możliwość określenia zakresu cenowego akcji, co pozwala na przeprowadzenie symulacji potencjalnych zysków i strat w zależności od zmiany cen na rynku. W przypadku braku ręcznej konfiguracji zakresu, program automatycznie ustala wartości graficzne na podstawie wybranej opcji. Po wprowadzeniu wszystkich parametrów kalkulator dokonuje obliczeń, których wyniki prezentowane są w formie tabeli. Tabela zawiera prognozowane zyski i straty w zależności od ceny aktywa bazowego w dniu wygaśnięcia oraqz różnych momentach czasu. Wyniki są wzbogacone o wizualne oznaczenia – scenariusze korzystne są oznaczone na zielono, a niekorzystne na czerwono co ułatwia interpretację danych.

Obraz zawierający zrzut ekranu, tekst, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Rys 5.6 Tabela wyników kalkulatora opcji CALL dla akcji AAPL z prognozowanymi zyskami i stratami.

Przedstawiony przykład ilustruje strategię Long CALL, w której inwestor oczekuje wzrosu cen akcji Apple (AAPL) powyżej ustalonej ceny wykonania (strike price) przed dniem wygaśnięcia opcji. W analizie wykorzystano poniższe dane wejściowe oraz obliczono wyniki:

* **Symbol akcji**: AAPL
* **Bieżąca cena akcji**: 242.65 USD
* **Typ strategii**: Long CALL (zakup opcji)
* **Cena jednostkowa opcji (premium)**: 8.77 USD
* **Cena wykonania (strike price):** 255 USD
* **Liczba kontraktów**: 1 (standardowo 1 kontrakt = 100 akcji)
* **Zakres cenowy akcji (automatycznie):** Od 204 USD do 267.75 USD
* **Data wygaśnięcia**: 2025-04-17

Wyniki analizy Long CALL wskazują, że całkowity koszt wejścia w inwestycję wynosi 877.00 USD. Kwota ta została obliczona jako iloczyn ceny jednostkowej, wynoszącej 8.77 USD, oraz wielkości kontraktu, która standardowo obejmuje 100 akcji. Koszt zakupu opcji stanowi jednocześnie maksymalne ryzyko inwestycji. W przypadku, gdy cena akcji Apple nie przekroczy poziomu 225 USD, czyli ceny wykonania opcji, w dniu jej wygaśnięcia, inwestor poniesie stratę równą tej kwocie, ponieważ opcja stanie się bezwartościowa. Z drugiej strony, potencjalny zysk inwestycji nie jest ograniczony („Maximum return: infinite”). W teorii, jeśli cena akcji znacznie wzrośnie powyżej ceny wykonania, inwestor może czerpać korzyści proporcjonalne do tego wzrostu. Analiza wykazała również, że próg rentowności (ang. breakeven), przy którym inwestor wychodzi na zero, wynosi 263.77 USD. Oznacza to, że w dniu wygaśnięcia cena akcji musi wzrosnąć o kwotę równą sumie ceny wykonania opcji (255 USD) oraz jednostkowego kosztu opcji (8.77 USD), aby inwestycja stała się opłacalna.

Przekroczenie tego poziomu gwarantuje zyski, które będą rosły z dalszym wzrostem ceny akcji. Ostatnia kolumna w tabeli wyników prezentuje procentowy zwrot z inwestycji dla każdej ceny akcji w dniu wygaśnięcia, obliczana jako:

**PRZYKŁAD LICZENIA // jakos to nazwac? // scenariusz taki o**

Dla przedstawionego przykładu zostanie policzona wartość dla jednej komórki tej tabelki aby zobrazować jakie procesy kalkulator przeprowadza, sprawdzamy wynik, jaką wartość osiągnie opcja dnia 11.12.2024, jeżeli dzisiaj mamy 4.12.2024, przy założeniu, że akcja osiągnie poziom 261 USD. Poniżej wprowadzamy dane, które kalkulator otrzymuje z API:

* Cena akcji (S): 261.00 USD
* Cena wykonania opcji (K): 255.0 USD
* Czas do wygaśnięcia (T):
* Zmienność (IV / sigma): 0.2103 (21.03%)
* Koszt opcji (premium): 8.77 USD za jedną akcję
* Liczba akcji na kontrakt: 100
* Koszt wejścia na kontrakt:

Krok 1: Obliczanie oraz :

mierzy względną pozycję bieżącej ceny akcji (S) względem ceny wykonania (K), uwzględniając wpływ stopy procentowej (r), zmienności () oraz czasu do wygaśnięcia (T). jest wartością , która uwzględnia dodatkowy wpływ zmienności rynkowej w odniesieniu do czasu (T). Jest przesunięciem o wielkość .

Najpierw należy policzyć logarytm stosunku bieżącej ceny akcji (S) do ceny wykonania (K):

Sumujemy stopę procentową (r) i połowę kwadratu zmienności ():

Sumujemy te wyniki, mnożąc przez czas do wygaśnięcia opcji (T):

Mnożymy zmienność () przez pierwiastek kwadratowy z czasu do wygaśnięcia ():

Dzielimy wyniki, uzyskując wartość

Odejmujemy od , dzięki czemu uzyskujemy wynik

Krok 2: Obliczanie i

i reprezentują wartości standardowego rozkładu normalnego dla oraz . Oznaczają prawdopodobieństwo, że cena akcji znajdzie się powyżej lub poniżej ceny wykonania.

Krok 3: Obliczanie ceny opcji (CALL Price):

Cena opcji obliczana jest za pomocą modelu Blacka-Scholesa:

Co daje nam cenę opcji:

Krok 4: Obliczanie zysku:

Manualne obliczenia przeprowadzone zgodnie z modelem Blacka-Scholesa wykazały, że wartość zysku dla opcji CALL na akcje AAPL wynosi 775.79 USD. Dotyczy to opcji z datą wygaśnięcia 14 kwietnia 2025 roku, ceną wykonania 255 USD oraz premią 8.77 USD na kontrakt. W obliczeniach założono, że cena akcji w dniu 11 grudnia 2024 roku wynosi 261 USD. Porównanie wyników manualnych z wynikami generowanymi przez kalkulator opcji wykazuje ich pełną zgodność, co potwierdza poprawność implementacji algorytmów i ich zgodność z założeniami teoretycznymi.

Kalkulator automatycznie wykonuje analogiczne obliczenia dla każdego dnia w określonym przedziale czasowym oraz dla wszystkich wartości cenowych w zdefiniowanym zakresie. Dzięki temu narzędzie umożliwia kompleksową analizę potencjalnych scenariuszy rynkowych oraz ich wpływu na zysk lub stratę, co znacząco wspiera proces podejmowania decyzji inwestycyjnych.

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Rys 5.7 Fragment tabeli wyników kalkulatora opcji CALL dla akcji AAPL z zaznaczonym sprawdzanym wynikiem

Przeprowadzone obliczenia, zweryfikowane ręcznymi wyliczeniami, wykazały pełną zgodność wyników kalkulatora z teoretycznymi założeniami modelu. To potwierdza poprawność implementacji algorytmów. Kalkulator upraszcza złożone procesy matematyczne, umożliwiając użytkownikom skupienie się na analizie strategii inwestycyjnych i podejmowaniu świadomych decyzji. Podsumowując, moduł kalkulatora opcji stanowi istotne rozszerzenie funkcjonalności aplikacji, oferując użytkownikom dokładnie i intuicyjne narzędzie do analizy opcji finansowych. Jego wsparcie w ocenie scenariuszy rynkowych i optymalizacji strategii inwestycyjnych przyczynia się do podjęcia bardziej precyzyjnych decyzji.

## Moduł analizy sentymentu

Analiza sentymentu stanowi jeden z kluczowych modułów aplikacji, mający na celu zbadanie wpływu nastrojów rynkowych wyrażanych w mediach społecznościowych, na rzeczywiste ceny akcji. Moduł ten opiera się na danych dotyczących sentymentu dla akcji Apple, pobranych z platformy Kaggle, które zostały wzbogacone o rzeczywiste dane dotyczące cen akcji z giełdy. Do przeprowadzenia analizy zaimplementowano trzy modele uczenia maszynowego: Random Forest, XGBoost oraz LSTM. Wybór tych modeli wynika z ich zdolności do przetwarzania różnorodnych danych – od strukturalnych (Random Forest, XGBoost) po sekwencyjne (LSTM), co jest kluczowe w analizie czasowej takich danych jak sentyment i ceny akcji. Dane zebrane w modelu zostały przetworzone, a wyniki predykcji uzyskane za pomocą każdego modelu zostały zaprezentowane w postaci wykresów. Wizualizacje te pokazują zarówno prognozowane ceny akcji, jak i rzeczywiste dane rynkowe, co pozwala na ocenę skuteczności zastosowanych algorytmów w przewidywaniu przyszłych wartości. Przed szczegółowym omówieniem technicznych aspektów implementacji oraz wyników analizy, konieczne jest przedstawienie charakterystyki wykorzystanego zbioru danych oraz metod przetwarzania, które przyczyniły się do uzyskania końcowych wyników.

### Zbiór danych

Do analizy sentymewntu akcji Apple wykorzystano zestaw danych pobrany z platformy Kaggle, który zwiera informacje o sentymencie znormalizowanym w zakresie od -1 (negatywny) do 1 (pozytywny), a także liczbę wpisów na Twitterze dotyczących AAPL w poszczególnych dniach. Kluczowe kolumny tego zbioru to:

* date – data wpisu na Twitterze,
* ts\_polarity – wartość sentymentu,
* twitter\_volume – liczba wpisów na Twitterze dotyczących AAPL

Przykładowy wiersz w zestawie danych: 2016-01-01, 0.11969256317245512, 417 Aby powiązać dane o sentymencie z rzeczywistymi zmianami cen akcji, zbiór został rozszerzony o informacje giełdowe pobrane za pomocą biblioteki yFinance. Proces ten obejmował pobranie danych takich jak cena otwarcia (Open), najwyższa cena w danym dniu (High), najniższa cena w danym dniu (Low), cena zamknięcia (Close), skorygowana cena zamknięcia (Adj Close) oraz wolumen transakcji (Volume). Dzięki temu możliwe było przeprowadzenie analizy zależności między nastrojami wyrażanymi na Twitterze a rzeczywistymi zmianami cen akcji.

List. 5.1. Kod rozszerzający bazowy zbiór danych

|  |
| --- |
| import yfinance  import pandas as pd  df = pd.read\_csv('AAPL.csv')  stock\_symbol = 'AAPL'  stock\_data = yfinance.download(stock\_symbol, start=df['date'].min(), end=df['date'].max())  stock\_data.reset\_index(inplace=True)  stock\_data = stock\_data[['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]  df['date'] = pd.to\_datetime(df['date'])  stock\_data['Date'] = pd.to\_datetime(stock\_data['Date'])  merged\_df = pd.merge(df, stock\_data, left\_on='date', right\_on='Date')  merged\_df.drop(columns=['Date'], inplace=True) merged\_df.to\_csv('merged\_stock\_data.csv', index=False) |

Połączenie tych informacji pozwoliło na przeprowadzenie analizy korelacji między nastrojami na cenami akcji oraz na przygotowanie zbioru danych do wykorzystania w modelach uczenia maszynowego. Zestaw danych został podzielony na różne okresy czasowe, obejmujące zarówno pełny zakres zbioru (lata 2016-2019), jak i krótsze zakresy (np. 2016-2017), co umożliwiło ocenę skuteczności modeli na różnych wielkościach danych. Dzięki temu analiza mogła uwzględnić potencjalne różnice w zachowaniu sentymentu i cen akcji w różnych okresach. Proces przygotowania zbioru danych ograniczył się do jego połączenia z danymi giełdowymi i przekształcenia w formę gotową do trenowania modeli. Modele te pozwoliły na przewidywanie przyszłych cen akcji na podstawie historycznych danych sentymentu i cenowych.

### Interfejs użytkownika

Interfejs użytkownika modułu analizy sentymentu został zaprojektowany z myślą o czytelnej prezentacji wyników predykcji. Kluczowym elementem interfejsu jest interaktywny wykres, który ilustruje rzeczywiste ceny akcji (Real) oraz prognozowane wartości (Predicted) wygenerowane przez wybrane modele uczenia maszynowego. Wykres ten umożliwia użytkownikowi ocenę dokładności prognoz, a także skuteczność w różnych przedziałach czasowych. Widok interfejsu prezentuje wyniki modelu Random Forest na pełnym zbiorze danych, co pozwala użytkownikowi na szybkie zapoznanie się z charakterystyką predykcji. Wykres oferuje interaktywne funkcje, takie jak przybliżanie i przesuwanie danych, umożliwiające szczegółową analizę wybranych fragmentów. Funkcjonalności te zostały zaimplementowane za pomocą narzędzi nawigacyjnych biblioteki Matplotlib, takich jak NavigationToolbar2QT, co zwiększa elastyczność pracy z wykresem. Poniżej przedstawiono widok interfejsu użytkownika z aktywną wizualizacją wyników modelu Random Forestna pełnym zestawie danych:

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, Oprogramowanie graficzne

Opis wygenerowany automatycznie

Rys 5.8 Widok główny segmentu analizy sentymentu

Panel boczny interfejsu zawiera opcje wyboru modelu predykcyjnego oraz zakresu czasowego analizy. Użytkownik może zdecydować się na jeden z trzech modeli: Random Forest, XGBoost oraz LSTM. Każdy z tych modeli został zaprojektowany do specyficznego rodzaju analizy

- Random Forest i XGBoost doskonale radzą sobie z analizą danych strukturalnych, podczas gdy LSTM jest bardziej odpowiedni do analiz sekwencyjnych danych czasowych. Zakresy czasowe obejmują cały zestaw danych oraz wybrane przedziały, takie jak 2016-2017, 2017-2018 czy 2018-2019. Górny pasek nawigacyjny pozwala użytkownikowi na szybkie przełączanie się pomiędzy modułami aplikacji, takimi jak wizualizacja danych giełdowych czy kalkulator opcji. Takie rozwiązanie zapewnia płynne przechodzenie między funkcjami programu.

### Model Random Forest

Do analizy cen akcji Apple zastosowano model Random Forest jako jedno z narzędzi przewidywania. Model ten opiera się na zespole drzew decyzyjnych. Dane wejściowe zostały przetworzone i podzielone na zbiór treningowy, oraz testowy, aby umożliwić ocenę jakości predykcji. Wyniki analizy obejmowały różne okresy czasowe, takie jak pełny zestaw danych i podzbiory roczne (np. 2016-2017). Oceniono je na podstawie statystyk takich średni błąd kwadratowy (MSE) czy współczynnik determinacji (). Model Random Forest sprawdza się szczególnie dobrze w zadaniach opartych na danych tabelarycznych, które zawierają zarówno zmienne ciągłe, jak i kategoryczne. W analizie predykcji wartości akcji Apple wybrano ten model ze względu na jego zdolność do redukowania błędów wynikających z pojedynczych drzew decyzyjnych. Model został skonfigurowany w sposób zapewniający [???]. W tej konfiguracji zastosowano 500 drzew decyzyjnych, co zwiększa stabilność i precyzję modelu poprzez zredukowanie wpływu pojedynczych, mniej dokładnych drzew. Maksymalna głębokość drzewa została ograniczona do dwóch poziomów, aby zapobiec zjawisku przeuczenia (overfittingu), które często występuje przy pracy na mniejszych zbiorach danych. Dzięki temu model lepiej generalizuje wyniki na nowych danych. Ponadto, w tej implementacji wyłączono mechanizm bootstrapingu, co oznacza, że każde drzewo było trenowane na całym zbiorze danych, a nie na jego losowych próbkach. Dzięki temu model efektywnie wykorzystuje pełnię dostępnych informacji. Dodatkowo, minimalna liczba próbek wymagana w liściu drzewa została ustawiona na 2, co zapobiega tworzeniu liści na zbyt małych podzbiorach danych i minimalizuje ryzyko nadmiernego dopasowania do zbioru treningowego. Ograniczona głębokość drzew pomaga uchwycić kluczowe zależności w danych bez nadmiernego dostosowania do szumu lub przypadkowych wzorców w zbiorze treningowym. Na pełnym zbiorze danych obejmującym lata 2016-2019 model Random Forest osiągnął następujące wyniki:

* **Mean Squared Error (MSE)** na poziomie 3.70 oraz **Root Mean Squared Error (RMSE)** wynoszące 1.9235 wskazują, że średni błąd prognozy wynosi około 2 USD. W kontekście zmienności cen akcji jest to wynik bardzo zadowalający.
* **R-squared (R²)** na poziomie 0.82 oznacza, że model wyjaśnia ponad 82% zmienności w danych, co wskazuje na jego bardzo dobrą zdolność do odwzorowania zależności pomiędzy zmiennymi.

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rys 5.9 Wizualizacja wyników modelu Random Forest na pełnym zbiorze danych

Na pomniejszych zbiorach danych Random Forest również nie sprawdził się najlepiej. Charakterystyczna blokowość predykcji wskazuje, że model podąża za ogólnym trendem, ale przy nagłych skokach lub spadkach cen akcji jego prognozy są niedokładne. Sugeruje to, że model ma ograniczoną zdolność do przewidywania dynamicznych zmian rynkowych.

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

Opis wygenerowany automatycznie

Rys 5.10 Wizualizacja wyników modelu Random Forest na pomniejszych zbiorach

Model Random Forest, mimo swoich zalet, takich jak stabilność i zdolność do wychwytywania ogólnych trendów, wykazał pewne ograniczenia podczas analizy dynamicznych zmian cen akcji. Szczególnym wyzwaniem okazało się przewidywanie nagłych wahań, takich jak gwałtowne wzrosty lub spadki cen. Blokowy charakter predykcji modelu wynika z samej struktury Random Forest, która działa na zasadzie agregacji wyników wielu pojedynczych drzew decyzyjnych. Każde z tych drzew podejmuje decyzje na podstawie określonych przedziałów wartości danych wejściowych, co skutkuje ograniczoną zdolnością modelu do uchwycenia nagłych zmian w czasie rzeczywistym.

### Model Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) to zaawansowany algorytm uczenia maszynowego oparty na metodzie gradientowego wspomagania drzew decyzyjnych (Gradient Boosted Decision Trees, GBDT). Jest szczególnie ceniony w zadaniach takich jak prognozowanie wartości liczbowych i klasyfikacja, dzięki swojej wysokiej precyzji predykcji. Działanie XGBoost opiera się na iteracyjnym tworzeniu drzew decyzyjnych, w którym każde kolejne drzewo stara się skorygować błędy popełnione przez wcześniejsze. Proces ten umożliwia modelowi efektywne dopasowanie się do skomplikowanych zależności. Wbudowane mechanizmy regularyzacji takich jak L1(Regresja Lasso) i L2(Regresja grzbietowa) przeciwdziałają przeuczeniu. Algorytm został zoptymalizowany pod kątem przetwarzania dużych zbiorów danych. XGBoost wykorzystuje wielowątkowość, co znacząco przyspiesza proces uczenia. Dodatkowym atutem jest obsługa brakujących wartości w sposób automatyczny.

Obraz zawierający tekst, zrzut ekranu, linia, diagram

Opis wygenerowany automatycznie

Rys 5.12 Wizualizacja wyników modelu XGBoost na pełnym zbiorze danych

Pomimo licznych zalet, XGBoost nie jest idealnym narzędziem do analizy danych szeregów czasowych, takich jak historyczne ceny akcji. Głównym ograniczeniem algorytmu jest brak uwzględnienia kolejności danych w sposób naturalny, co oznacza, że nie posiada on „pamięci” pozwalającej uwzględniąć zależności czasowe i historyczne. W przypadku dynamicznych danych, gdzie krótkoterminowe wahania cen mają znaczenie, XGBoost może mieć trudności z dokładnym odwzorowaniem złożonych wzorców. W przeprowadzonym eksperymencie XGBoost został wykorztstany do prognozowania cen zamknięcia akcji Apple na podstawie danych z lat 2016-2019. Dane wejściowe obejmowały historyczne ceny w 3-dniowych oknach czasowych. Analiza statystyk dla poszczególnych lat ujawniła, że współczynnik determinacji dla mniejszych przedziałów czasowych nie przekraczał 0.55. Oznacza to, że model napotyka trudności w uchwyceniu krótkoterminowych skoków cen, co jest typowe dla danych szeregów czasowych.

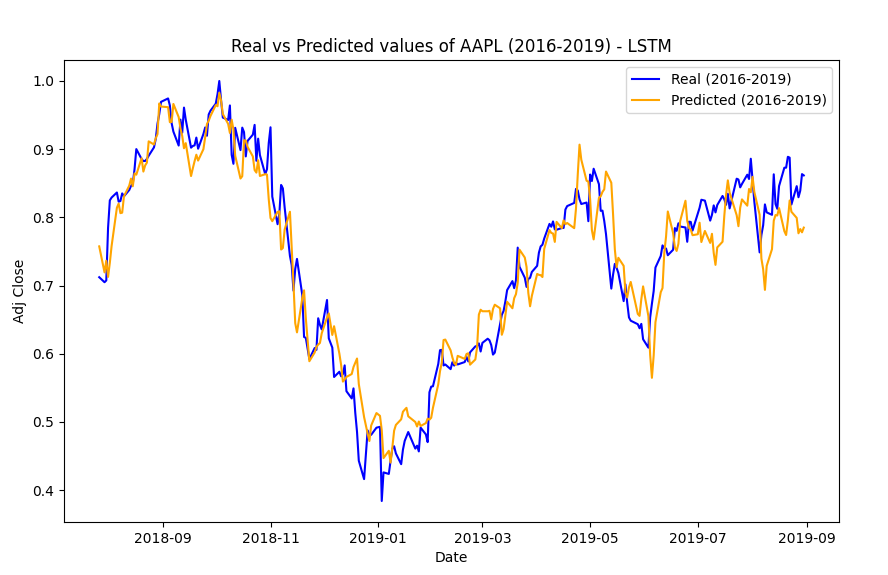
Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

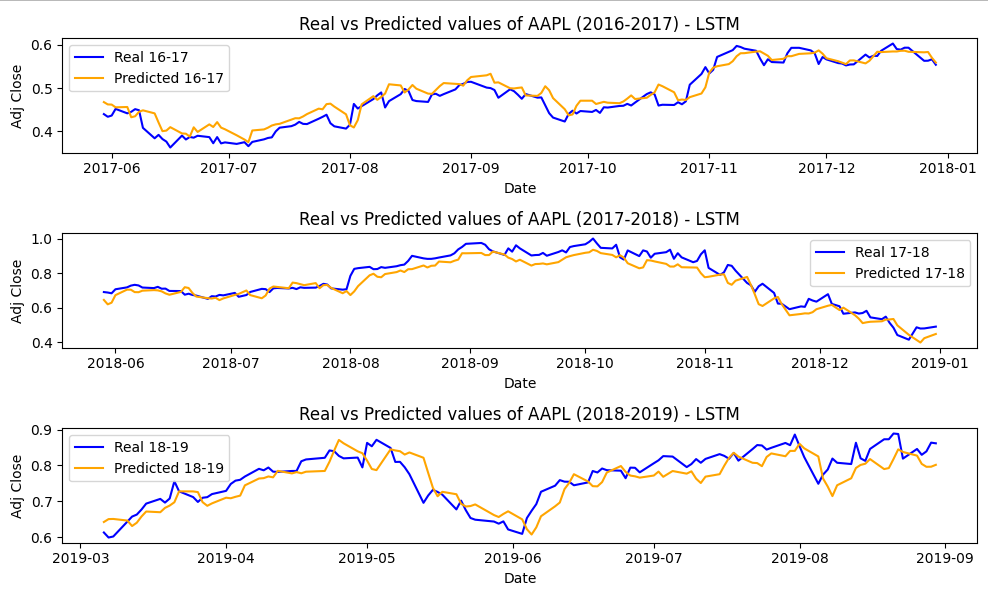
Rys 5.13 Wizualizacja wyników modelu XGBoost na pełnym zbiorze danych

### Model Long Short-Term Memory

LSTM (Long Short-Term Merory) to zaawansowana architektura rekurencyjnej sieci neuronowej (RNN), efektywna w przetwarzaniu i modelowaniu danych sekwencyjnych. Tradycyjne RNN często napotykają na trudności w uchwyceniu długoterminowych zależności z powodu problemu zaniku gradientów. LSTM rozwiązuje ten problem, wprowadzając kroki pamięci oraz mechanizmy bramek: wejściowej (input gate), zapominania (forget gate) i wyjściowej (output gate). Te elementy umożliwiają sieci decydowanie, które informacje zachować, które zapomnieć, a które przekazać do kolejnych kroków. Dzięki tej konstrukcji LSTM skutecznie identyfikuje istotne wzorce w dłuższych ciągach danych. W eksperymencie zastosowano model LSTM zbudowany przy użyciu biblioteki TensorFlow Keras. Kluczowym elementem jego konfiguracji było podzielenie na danych wejściowych na 3-dniowe sekwencje. Takie podejście pozwalało modelowi na analizę krótkoterminowych zależności czasowych i wyciąganie wzorców z poprzednich dni. Architektura modelu składała się z trzech warstw LSTM, z których każda zawierała 50 neuronów. Dzięki temu model miał wystarczającą złożoność, aby uchwycić istotne zachowania, jednocześnie unikając nadmiernego skomplikowania, które mogły prowadzić do przeuczenia. Aby dodatkowo ograniczyć ryzyko przeuczenia, pomiędzy warstwami LSTM zastosowano warstwy Dropout z ustalonym współczynnikiem 0.2, co oznacza losowe wyłączanie 20% neuronów w trakcie treningu. Ostatnia warstwa modelu, tzw. Warstwa wyjścia, została zbudowana z pojedynczego neuronu Dense, którego zadaniem było przewidywanie końcowej wartości ceny ackji. Proces trenowania modelu odbywal się przsez 50 epok, z wykorzystaniem wielkości partii (batch\_size) wynoszącej 32. Próbki danych nie były mieszane, co jest szczególnie ważne w przypadku danych sekwencyjnych. Dane podzielono na zbiór treningowy, obejmujący 70% danych oraz testowy, stanowiący pozostałe 30%.



Rys 5.11 Wizualizacja wyników modelu LSTM na pełnym zbiorze danych



Rys 5.12 Wizualizacja wyników modelu LSTM na pomniejszych zbiorach danych

Wyniki ilościowe uzyskane dla modelu LSTM na pełnym zbiorze danych (R² ~0,90, RMSE ~7,3%, MSE ~0,0053) potwierdzają jego bardzo dobre dopasowanie do historycznych cen akcji. Dzięki wykorzystywaniu mechanizmów komórek pamięci długoterminowej i bramek kontrolujących przepływ informacji, model LSTM efektywnie zarówno długoterminowe oraz bardziej lokalne trendy. W porównaniu z modelami takimi jak XGBoost czy Random Forest, LSTM okazał się bardziej skuteczny w przewidywaniu dynamiki rynku.

### Zestawienie wyników modeli

Poniżej przedstawiono zestawienie wyników trzech modeli – LSTM, XGBoost i Random Forest – w prognozowaniu cen akcji Apple na pełnym zbiorze danych oraz na różnych okresach czasowych. Porównania dokonano na podstawie współczynnika determinacji R2 oraz miał błędu (RMSE,MSE).

**Tab 5.1** Zestawienie R2 dla poszczególnych okresów i modeli

|  |  |  |  |
| --- | --- | --- | --- |
| Okres | LSTM | XGBoost | Random Forest |
| 2016-2017 | 0.843 | 0.718 | 0.721 |
| 2017-2018 | 0.875 | 0.691 | 0.863 |
| 2018-2019 | 0.598 | 0.674 | 0.823 |
| Pełny zbiór | 0.902 | 0.844 | 0.823 |

Wyniki wskazują, że skuteczność każdego z modeli jest uzależniona od ich architektury i podejścia do przetwarzania danych. Model LSTM, specjalizujący się w analizie danych sekwencyjnych, osiąga najwyższe wartości R2 na pełnym zbiorze danych, uzyskując wynik 0.902. Taki rezultat sugeruje, że sieć ta skutecznie identyfikuje długoterminowe trendy. Najlepsze wyniki LSTM są zauważalne w okresach stabilnych, takie jak lata 2017-2018, gdzie osiągnął R2 na poziomie 0.875. Najmniej jednak, w bardziej zmiennych okresach, takich jak 2018-2019, skuteczność modelu spada (R2 = 0.598), co wskazuje na jego ograniczenia w przewidywaniu krótkoterminowych wahań. Model XGBoost, oparty na gradientowym wspomaganiu drzew decyzyjnych, wykazuje bardziej spójne wyniki różnych przedziałach czasowych. Jego zdolność do korygowania wyników wielu drzew decyzyjnych pozwala na efektywne odzwroowanie ogólnych trendów. Wartości R2 dla XGBoost wynoszą od 0.674 do 0.844, co potwierdza, że model dobrze radzi sobie z globalnymi wzorcami, ale jego uniwersalność nie dorównuje LSTM w analizie szeregów czasowych. Model Random Forest, również oparty na drzewach decyzyjnych, osiąga solidne wyniki, szczególnie w stabilnych okresach, takich jak 2016-2017 (R2 = 0.721). Jego najwyższy wynik spośród wszystkich modeli został odnotowany w latach 2018-2019 (R2= 0.823), co może wynikać z odporności modelu na zakłócenia w danych. Jednak brak mechanizmów sekwencyjnych również ogranicza zdolność modelu do odwzorowania relacji czasowych.

1. Podsumowanie i wnioski

Realizacja projektu i osiągnięcie jego finalnej wersji wymagały znacznego zaangażowania, zarówno pod względem czasu, jak i rozwijania umiejętności. Prace nad aplikacją, obejmującą różnorodne moduły jak kalkulator opcji, analiza sentymentu i wizualizacja danych finansowych, stanowiły ambitne wyzwanie. Wymagały one nie tylko zaawansowanej wiedzy programistycznej, ale również zrozumienia kluczowych zagadnień finansowych i statystycznych, co okazało się istotne podczas implantacji kalkulatora opcji. Moduł ten, bazujący na modelu Blacka-Scholesa, był najbardziej wymagającą częścią projektu. Konieczność zrozumienia mechanizmu działania opcji, wraz z uwzględnieniem parametrów takich jak zmienność implikowana czy stopa wolna od ryzyka, wymagały dogłębnej analizy. Prace nad aplikacją jako całością obejmowały integrację wielu technologii, takich jak PyQt5 do budowy interfejsu, yFinance oraz zewnętrzne API do pobierania danych rynkowych oraz narzędzi analitycznych do przetwarzania danych. Każdy z modułów wnosił inne wyzwania, jednak ukończenie kalkulatora opcji przyniosło szczególną satysfakcję, jako że stanowi najbardziej zaawansowany element systemu. Projekt zakończył się sukcesem, spełniając założenia funkcjonalne oraz pozwalając na zdobycie nowych umiejętności.

7. Literatura

[01] **Tiwari S., Bharadwaj A., Gupta S.**: „Stock price prediction using data analytics”, *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 2017, pp. 1-5.

[02] **Sethi R.**: „Proposal on Financial Computing Algorithm and Analysis”, *International Journal of Computer Applications*, vol. 125, 2015, s. 36-40.

[03] **Black F., Scholes M.**: „The Pricing of Options and Corporate Liabilities”, *Journal of Political Economy*, vol. 81, 1973, pp. 637-654.

[04] **Taylor J.**: „Sentiment Analysis in Financial Markets Using Twitter Data”, *Journal of Financial Data Science*, vol. 9, 2020, pp. 123-139.

[05] **Garcia F.**: „Building GUI with PyQt5 for Data-Driven Applications”, *Software Development Journal*, vol. 6, 2022, pp. 66-78.

[06] **Walker D.**: „Integrating yFinance and Finnhub for Comprehensive Market Data Analysis”, *International Review of Financial Data Science*, vol. 3, 2021, pp. 88-100.

[07] **Johnson P., Taylor A.**: „Developing GUI Applications Using PyQt5”, *Applied Software Engineering Journal*, vol. 9, 2020, pp. 55-70.

[08] **Nguyen L., Zhao K.**: „Leveraging Python Libraries in Financial Data Analysis”, *Journal of Financial Programming*, vol. 8, 2022, pp. 101-120.

[09] **Patel R., Kumar S.**: „Historical and Real-Time Data Analysis Using yFinance and Finnhub APIs”, *International Journal of Market Analytics*, vol. 11, 2021, pp. 89-103.

[10] **Kumar S., Patel R.**: „Building Financial Analysis Modules Using yFinance”, *Journal of Financial Engineering*, vol. 7, 2021, pp. 88-105.

[11] **Smith J., Brown A.**: „Dividend Analysis for Modern Investors”, *Journal of Corporate Finance Studies*, vol. 14, 2020, pp. 45-62.