НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.852

До захисту допущено
Завідувач кафедри ММСА
_____ О.Л.Тимощук
«___» _____ 2021 р.

Магістерська дисертація

на здобуття ступеня магістра
за спеціальністю 122 «Комп'ютерні науки»
на тему: «Оцінка успішності компанії на основі інформації про неї з використанням аналізу тональності тексту»

Виконав:
студент II курсу, групи КА-03мп
Петелєв Євгеній Русланович _____

Керівник:
доцент кафедри ММСА, к.т.н, доц.
Дідковська М.В. _____

Рецензент:
доцент кафедри програмного забезпечення
комп'ютерних систем КПІ ім. Ігоря Сікорського,
к.т.н, доц. Заболотня Т.М. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ
2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 122 «Комп'ютерні науки»
Освітньо-професійна програма — «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

Завідувач кафедри ММСА

_____ О.Л. Тимощук

«___» _____ 2021 р.

ЗАВДАННЯ

на магістерську дисертацію студенту Петелєву Євгенію Руслановичу

1. **Тема дисертації**: «Оцінка успішності компанії на основі інформації про неї з використанням аналізу тональності тексту», науковий керівник дисертації Дідковська Марина Віталіївна, доцент, кандидат технічних наук, затверджені наказом по університету від «02» листопада № 3651-с

2. **Термін подання студентом дисертації**: 15 грудня 2021 р.

3. **Об'єкт дослідження**: прогнозування оцінки успішності компанії.

4. **Предмет дослідження**: використання аналізу тональності тексту для оцінки успішності компанії.

5. Перелік завдань, які потрібно розробити:
1) Ознайомитися з технічною літературою за темою роботи;
2) Дослідження актуальності обраної теми;
3) Огляд методів для оцінки успішності компанії;
4) Робота з вхідними даними;
5) Запуск різноманітних експериментів;
6) Проведення аналізу результатів оцінки успішності компаній;
7) Проведення аналізу ринкових можливостей запуску стартап-проекту;

8) Підготовка ілюстративного матеріалу;
9) Оформлення пояснювальної записки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
1) Use Case діаграма.
2) Наукова новизна результатів.

7. Дата видачі завдання: 01 вересня 2021 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Отримання завдання на магістерську дисертацію	08.09.2021 – 10.09.2021	
2.	Огляд технічної літератури за темою	11.09.2021 – 30.09.2021	
3.	Дослідження актуальності обраної теми	01.10.2021 – 08.10.2021	
4.	Огляд методів для оцінки успішності компанії	08.10.2021 – 15.10.2021	
5.	Збір вхідних даних	16.10.2021 – 21.10.2021	
6.	Виконання обчислювальних експериментів	22.10.2021 – 29.10.2021	
7.	Аналіз результатів оцінки успішності компанії	30.10.2021 – 04.11.2021	
8.	Проведення аналізу ринкових можливостей запуску стартап-проекту	05.11.2021 – 11.11.2021	
9.	Підготовка ілюстративного матеріалу	12.11.2021 – 19.11.2021	
10.	Оформлення пояснювальної записки	20.11.2021 – 26.11.2021	

Студент Є.Р. Петелєв

Науковий керівник дисертації М.В. Дідковська

РЕФЕРАТ

Магістерська дисертація: 154 с., 24 рис., 22 табл., 1 додаток та 27 джерел.

Об'єктом дослідження є методи для прогнозування оцінки успішності компанії.

Мета роботи – розробка моделі та програмного продукту для оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту. В даній роботі було проаналізовано існуючі методи, які використовуються для оцінки успішності компанії.

Результати роботи:

– запропоновано підхід оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту;
– створено програмний продукт, який використовуючи нову модель оцінки вартості компанії, зумів покращити результати класичних методів оцінки.

Новизна роботи:

– стрімкий розвиток компаній на ринку та прагнення людей моделювати та прогнозувати їх поведінку;
– збільшення інформаційного потоку, який нас оточує та прагнення людей автоматизувати аналіз інформації;
– розроблено підхід для оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту.

Результати даної роботи рекомендується використовувати при розробці системи для оцінки успішності компанії.

УСПІШНІСТЬ КОМПАНІЇ, ВАРТІСТЬ КОМПАНІЇ, ВАРТІСТЬ АКЦІЙ, ФОНДОВИЙ РИНОК, ЧАСОВИЙ РЯД, АВТОРОГРЕСІЯ, НЕЙРОННІ МЕРЕЖІ, АНАЛІЗ ТОНАЛЬНОСТІ ТЕКСТУ, МЕТОД ОПОРНИХ ВЕКТОРІВ.

ABSTRACT

Master`s thesis: 154 p., 24 fig., 22 tables, 1 appendix and 27 sources.

The object of the study is methods for forecasting the evaluation of the company's success.

The purpose of the work is to develop a model and software product to assess the company's success based on information about it using the analysis of the text's tone. This paper analyzes the existing methods used to assess the company's success.

Work results:

– the approach of estimation of success of the company based on the information on it with the use of the analysis of tonality of the text is offered;
– a software product was created, which, using a new model of the company's value, managed to improve the results of classical valuation methods.

The novelty of work:

– sustainable development of companies in the market and the desire of people to model and predict their behavior;
– increasing the flow of information that surrounds us and the desire of people to automate the analysis of information;
– developed an approach to assess the company's success based on information about it using the analysis of the text's tone.

It is recommended to use the results of this work to develop a system for assessing the company's success.

COMPANY SUCCESS, COMPANY VALUE, VALUE STOCKS, STOCK MARKET, TIME SERIES, AUTOROGRESSION, NEURAL NETWORKS, SENTIMENT ANALYSYS, SUPPORT VECTOR MACHINE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЧР – часовий ряд

АР – авторегресія

MAE – Mean Absolute Error

MSE – Mean Squared Error

MA – Moving Average

EMA – Exponential Moving Average

ANN– Artificial Neural Networks

ARMA – Autoregressive moving average

ARIMA – Autoregressive integrated moving average

NLP – Natural Language Processing

SVM – Support Vector Machine

ВСТУП

В сучасному світі існує величезна кількість компаній. Кожного дня на ринку з’являються та зникають сотні компаній. Цікаво, чому одним компаніям вдається все, а інші не виживають? На сьогоднішній день кожен з нас може купити цінні папери тієї чи іншої компанії, проте як не прогадати? Нам, як інвесторам, важливо оцінити бізнес на сьогоднішній день, щоб розуміти, які в нього перспективи на найближче майбутнє.

Ми розуміємо, що оцінка успішності компанії є надзвичайно важливою для нас, інвесторів, проте залежить від багатьох факторів. Ми не можемо зазирнути в майбутнє та знати точно, що буде з акціями компанії, чи варто їх купувати. Ціна акцій постійно коливається, вона може як впасти, так і піднятися відносно ціни, за якою вона була куплена. Передбачення поведінки фінансового ринку є одним з найважчих та найцікавіших завдань в економіці. Задля того, щоб передбачити поведінку ринку варто врахувати багато різних факторів, такі як фізичні, психологічні та інші поведінки. Що ж, можемо зробити висновок, що ціни на акції є дуже нестійкими, а тому їх дуже важко передбачити.

Задача прогнозування ціни акцій є актуальною для сучасної економіки, оскільки передбачення точної вартості акцій пов’язано з отриманням прибутку компаній або особистого капіталу. Точне прогнозування вартості активів на біржі також дозволить зменшити інвестиційний ризик та захистити інвестиційні прибутки від волатильності ринку.

З іншого боку ми живемо в інформаційному суспільстві, де нас оточує величезна кількість різноманітної інформації. Ми бачимо, як ті, чи інші висловлювання популярних людей впливають як на ціну акцій, так і вартість компанії взагалом. Чого тільки варті твіти CEO Tesla та SpaceX Ілона Маска, де після кожного його посту ціни на акції різних компаній просто злітають до

космосу. Ми бачимо, як люди стають мільйонерами, а компанії єдинорогами і це все завдяки тому, що про них говорять і люди інтуїтивно інвестують в них. Що ж, можемо зробити висновок, що дійсно на майбутнє компанії впливає інформаційний простір в якому вона перебуває. Проте будь-яку інформацію потрібно обробити та проаналізувати, а це займає час. Також не потрібно забувати, що кожна людина може розуміти інформацію по своєму, а тому необхідно автоматизувати процес аналізу інформації. Для цієї задачі ми будемо використовувати аналіз тональності тексту, який допоможе нам класифікувати інформацію як позитивну та негативну.

Емоційно-смисловий аналіз тексту є надзвичайно перспективним напрямком, оскільки середньостатичний користувач може зіткнутися з труднощами із пошуком необхідної йому інформації в Інтернеті. Такі моменти і посприяли розвитку галузі автоматичного аналізу тональності текстів.

Отже, метою роботи є розробка моделі та програмного продукту для оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту.

Для досягнення даної мети необхідно розв’язати наступні задачі:

– розглянути існуючі методи та підходи для оцінки успішності компнії;
– проаналізувати та порівняти між собою існуючі підходи для задачі прогнозування вартості акцій;
– проаналізувати методи аналізу тональності тексту;
– розробити модель оцінки вартості компанії на основі інформації про неї з використанням аналізу тональності тексту;
– створити програмний продукт, який використовуючи нову модель оцінки вартості компанії, зумів покращити результати класичних методів оцінки;
– розробити маркетинговий план стартап-проекту для виходу його на ринок.

Актуальність теми обумовлена тим, що в наші дні оцінка вартості компанії є важливою складовою сучасних економічних відносин. Основною стратегією підприємств найчастіше являється підвищення вартості капіталу.

Об'єктом дослідження є методи для прогнозування оцінки успішності компанії.

Предметом дослідження є використання аналізу тональності тексту для оцінки успішності компанії.

Що стосується практичного результату даної магістерської дисертації, то ми поставили за мету розробити модель для оцінки вартості компанії на основі інформації про неї з використанням аналізу тональності тексту, а також створити програмний продукт, який на основі даних компанії з фондового ринку та інформації про неї з інформаційних джерел, та використанням моделі оцінює її успішність, а також прогнозує майбутнє.

Магістерська дисертація складається з 4 розділів. В першому розділі розглядається поняття «успішність компанії». Ми приймаємо, що під «успішністю» розуміємо її вартість. Далі ми розглядаємо підходи для оцінки вартості компанії, та приймаємо, що під її вартістю ми розуміємо ринкову варстіть, а тому далі будемо працювати тільки з публічними компаніями. В першому розділі ми зрозуміли, що маємо справу з задачею прогнозування вартості акцій компаній. В другому розділі ми розглянули класичні методи прогнозування вартості компанії (авторегресія), а також методи з використанням штучних нейронних мереж. Також ми ознайомилися з задачею аналізу тональності тексту та існуючими методами. Третій розділ присвячений детальному опису архітектури програмного продукту та аналізу результатів роботи моделі. В четвертому розділі ми розробили маркетинговий план стартап проекту для виходу його на ринок.

РОЗДІЛ 1 ОЦІНКА УСПІШНОСТІ КОМПАНІЇ

Словосполучення «успішність компанії» є досить неоднозначним. Головна мета створення будь-якої комерційної структури є отримання прибутку. Можна стверджувати, що успішність компанії прямо пропорційна величині цього прибутку. Також я чудово розумію, що успішність це не тільки про гроші, це репутація компанія, наскільки працівники щасливі, яку проблему вирішує ця компанія та ін. Проте я також розумію, що в рамках магістерської дисертації мені необхідно буде виміряти й порівняти успішність, а зрозуміти, як сторонні фактори можуть вплинути на неї. Що ж, ми приймаємо, що під успішністю компанії ми будемо розуміти її вартість. Чим більше вартість компанії, тим вона успішніша.

Далі розберемося, що таке вартість компанії та як її оцінити. Важливо, що ми будемо використовувати ті компанії, які представлені на фондовому ринку, так як саме вони цікаві нам, як інвесторам. Ми не будемо розглядати не публічні компанії, оскільки інформація щодо їх вартості є закрита. Тому сконцентруємось на тих компаніях, чиї дані є відкритими і ми можемо використати їх історичні дані як стартову точку.

1.1 Аналіз існуючих методів для оцінки вартості компанії

Що ж, визначення вартості компанії є дуже важливим завданням у галузі корпоративного управління. Розуміння вартості компанії дає можливість оцінити її рівень успішності та конкурентоспроможності на ринку. Цікаво, що оцінюють вартість компанії з конкретною метою, а саме обчислення ціни

продажу, отримання кредиту чи інше. Вибір мети оцінки вартості зумовлює й вибір конкретного методу для визначення вартості компанії.

Існує безліч методів до оцінювання вартості компанії, але все ще дане питання є відкритим. Наприклад, під час оцінки вартості підприємств в сучасному світі більшість із розроблених підходів не використовуються або використовуються частково. Саме так формується вартість компанії, яка не є актуальною до ринкових умов.

Оцінка вартості компанії відбувається при купівлі, продажі або ліквідуванні компанії. Також для того, щоб компанія отримала фінансування, їй необхідно надати повну інформацію про вартість своїх активів. Зауважимо, що для оцінки бізнесу фахівець аналізує ринок, а також використовує фінансові звіти, моделі грошових потоків.

Не можливо оцінити бізнес одним підходом. Для того, щоб це зробити необхідно декілька підходів, такі як оцінка доходів, активів і ринку. Сама оцінка є складним процесом. Оцінка бізнесу може викликати різноманітні проблеми, такі як загальні інтереси власника та інших акціонерів. Задля того, щоб правильно оцінити угоду, нам необхідно з’ясувати, який підхід найбільше відкликається для нашої компанії.

Існують три основні підходи оцінки бізнесу, які можна поділити на наступні групи.

1.1.1 Оцінка за доходами

Даний підхід визначає вартість компанії на основі її доходу. Включає в себе:

1. Метод капіталізації доходів, який визначає потік доходу та перетворює його в поточну вартість шляхом застосування норми капіталізації, що являє собою спрощений коефіцієнт дисконтування.
2. Метод дисконтованих грошових потоків, який визначає, що вартість підприємства дорівнює вартості чистих активів, сумі вартостей всіх його активів, як матеріальних, так і нематеріальних, за вирахуванням зобов’язань.
3. Методи капіталізації дивідендів, які оцінюють компаній, що представлені на фондовому ринку.

Перевагами даного підходу компанії є те, що у нього з'являється можливість розрахувати вартість компанії виходячи з якості поточних фінансових показників та можливості отримання майбутніх доходів.

Водночас є ряд факторів, що ускладнюють застосування даного підходу. В основі більшості з них лежить проблема прогнозування ключових елементів оцінки, як то: складання прогнозу на середньо- або довгостроковий період. Прогнозування вартості непрофільних і нефункціонуючих активів, а також власного оборотного капіталу для внесення коригувальних поправок; обчислення норми віддачі капіталу, особливо визначення альтернативної вартості капіталу та інші.

1.1.2 Оцінка за витратами

Особливості та переваги витратного підходу полягають в тому, шо можливо оцінити балансову вартість господарюючого суб'єкта незалежно від прибутковості її діяльності. Найчастіше це обумовлено наявністю повної інформації для розрахунків, а також використанням класичних витратних

методів оцінки вартості компанії. Проте, коли діяльність підприємства починає «набирати обертів», вважається некоректним використовувати методи оцінки, засновані тільки на ретроспективних даних, без урахування перспектив розвитку і майбутніх грошових потоків. Методи оцінки за витратами: чистої балансової вартості, чистої вартості матеріальних активів, вартості заміщення та відносної вартості. Перевагами даного підходу є його просте застосування, а також те, що він дає реальну ринкову оцінку матеріальних активів. Недоліками підходу є те, що він не відображає потенційні прибутки та не дає оцінювання нематеріальних активів.

1.1.3 Порівняльна оцінка

Даний підхід дозволяє визначити вартість бізнесу, порівнюючи його з аналогами на ринку. Фахівець зосередиться на методі порівняльних транзакцій. Також він оцінюватиме конкурентні продажі підприємств, щоб оцінити економічні показники виручки та прибутку. Даний підхід добре працює з компаніями, які торгують публічно, тобто представлені на фондових рунках. В таких компаніях інформація про доходи відкрита, та є доступною кожному. Популярні методи даного підходу: галузеві коефіцієнти, порівняння продажів, ринок капіталів.

Перевагою даного підходу є те, що він є самим простим у використанні та статистично обґрунтованим, так як заснований на сучасній ринковій вартості існуючих об'єктів оцінки. Завдяки цьому ми отримуємо актуальні та реальні дані, які є придатними для застосування в інших підходах. Даний підхід є найпопулярнішим та найпростішим у використанні.

Одним з важливих обмежень в застосуванні даного підходу є обов'язкова наявність відкритого та активного ринку.

Оскільки ми сконцентрувалися на тих компаніх, які представлені на фондовому ринку, так як інформація її фінансових показників є відкритою, то ми можемо оцінити вартість компанії як її ринкову капіталізацію. Далі ми детально розглянемо, що являє собою цей показник та як оцінити ринкову капіталізацію компанії.

1.2 Ринкова капіталізація компанії

Ринкова капіталізація – показник діяльності корпорацій. Він розраховується шляхом множенням кількості акцій, випущених корпорацією, на їх ринкову ціну. Важливо, що оцінка діяльності корпорацій може мати певні суб'єктивні риси, які позначається на ціні акцій. Також, даний показник є одним з найпридатніших засобів рейтингоутворення компанії як форми організації підприємства. Цікаво, що величина ринкової капіталізації, а також зростання даної величини є характеристиками успішності компанії.

Розглянемо конкретний приклад. Уявимо, що компанія має 20 мільйонів акцій, що коштують по 200 доларів. Відповідно її ринкова капіталізація складає 4 мільярди доларів. Інвестори використовують даний показник для визначення розміру компанії, а не використання показників продажів. Також при купівлі компанії її ринкова капіталізація використовується для визначення того, чи вигідна вартість покупцеві.

Завдання оцінки вартості компанії є надзвичайно важливим, проте його його важко швидко, а найголовніше точно встановити. Завдяки ринковій капіталізації ми можемо швидко та просто оцінити вартість компанії шляхом

екстраполяції, як ринок оцінює компанію, яка представлена на фондовому ринку. Для того, щоб розрахувати ринкову капіталізацію необхідно помножити ціну акції на кількість доступних акцій.

Оскільки розмір компанії є одним із основних факторів, в яких зацікавлені інвестори, використання ринкової капіталізації є надзвичайно важливим для показу розміру компанії.

Ринкова капіталізація компанії встановлюється шляхом первинного публічного розміщення - IPO. Перед первинним публічним розміщенням компанія, яка має на меті вийти на біржу, залучає інвестиційний банк для використання методів оцінки вартості компанії та визначенням кількості акцій та її ціною.

Далі компанія виходіть на біржу та починає продавати на ній свої акції. Якщо є високий попит на акції компанії, то її ціна зросте. Також інвестори, які продають акції можуть знизити її ціну, якщо вони вважають, що потенціал зростання компанії в майбутньому не виглядає добре. Для такого випадку ринкова капіталізація стає оцінкою вартості компанії в реальному часі. Що ж, основною перевагою ринкової капіталізації є простота та ефективність даної метрики. Інвестори, спираючись на дану метрики, зможить урізноманітнити та збалансувати власні портфелі задля того, щоб отримати прибуток в майбутньому.

Основною відмінністю між ринковою капіталізацією та вартістю компанії є те, що ринкова капіталізація відображає лише вартість власного капіталу компанії. В свою чергу, вартість компанії відображає загальну суму капіталу, тобто й борг, якщо він є. Інвестори використовують вартість компанії як приблизну оцінку вартості її придбання.

Отже, ми розуміємо, що акції компанії є динамічними, тому ми не можемо знати по якій ціні вони будут торгуватися. Також ми бачимо зв’язок ціни на

акцію компанії та її ринковою капіталізацією. Що ж, наша задача зводиться до того, щоб прогнозувати вартість акцій компанії.

1.3 Задача прогнозування вартості акцій компаній

Цікаво, що за всю історію багато людей намагалися заробити на фінансових ринках через прогнозування майбутньої ціни акцій, товарів, курсів валют, опціонів, тощо. Розроблені методики можна поділити на три категорії.

1.3.1 Фундаметальний аналіз

Фундаментальний аналіз має на меті аналіз результатів діяльності компанії для визначення її ціни акцій. Даний метод використовує базову ринкову інформацію для прогнозування майбутнього активу. Наприклад, інвестори можуть аналізувати таку інформацію, як дохід, попит, операційна маржа, конкуренція інших компаній та ін. Фундаментальний аналіз тісно пов'язаний з економічними та політичними факторами, які в свою чергу можуть вплинути на майбутні ціни фінансового активу. Перевагою даного методу є його систематичний підхід, а також здатність передбачати зміни до того, як вони з'являтьс. Інвестори та аналітики аналізують і вивчають компанії, також вони порівнюсть їх з поточною економічною ситуацією. Недоліком є те, що все важче й важче формалізувати ці знання для автоматизації оцінки вартості компанії. Даний метод частіше за все є суб'єктивною оцінкою. В даній роботі фундаментальний аналіз не розглядається.

1.3.2 Технічний аналіз

Технічний аналіз має на меті вивчення історичних цін і закономірностей для прогнозування майбутньої ціни. Даний метод широко використовується у світі фінансів. Аналітик вивчає графіки для виявлення тенденцій та прогнозування майбутніх рухів активів завдяки використанням статистики цін та обсягів продажу. Важливо, що тенденції базуються на питаннях попиту та пропозиції, які мають помітні закономірності або циклічні. Існують технічні індикатори, які можна використовувати як вхідні дані для нейронних мереж, такі як ковзне середнє. Технічний аналіз спирається на припущенні, що історія є циклічною та повторюється, тому майбутній ринковий напрямок може бути визначений шляхом вивчення минулих цін. Даний підхід має широке використання, проте й він має свої недоліки. Технічний аналіз є дуже суб’єктивним, так як кожен аналітик інтерпретує графіки по своєму та має власну думку щодо кожної з компаній, а тому оцінки у всіх будуть різними.

1.3.3 Технологічний аналіз

Технологічний аналіз має на меті вивчення та використання прогнозування часових рядів для проблеми передбачення коливань цін на фінансовому ринку.

Класичний аналіз часових рядів базується на методах експоненційного згладжування, моделі авторегресії ковзного середнього, інтегрованої моделі авторегресії ковзного середнього. Дані методи є дуже популярними та широко використовуються на сучасний ринках. Що цікавого, так це те, що за останні

роки розробляються нові методи інтелектуального аналізу даних з використанням нейронних мереж, глибинного навчання. Також активно розробляється та досліджується гібридний підхід поєднання класичних методів з методами інтелектуального аналізу даних. Даний аналіз є надзвичайно перспективним, завдяки здатністю нейронних мереж вивчати нелінійні та хаотичні системи. Це дає нам можливість перевершити традиційний аналіз та інші методи передбачення коливань цін на фінансовому ринку.

З кожним роком з'являється все більше й більше методів для прогнозування вартості акцій. Було розглянуто і проаналізовано безліч методів як окремо, так і їх комбінацій, проте жоден з них не є настільки успішним, щоб стати провідним методом. Дана робота має на меті проаналізувати класичні та сучасні методи прогнозування вартості акцій компанії, а також запропонувати новий метод, який працював би краще та був точнішим.

1.4 Поняття часового ряду

Розглянемо, що являє собою часовий ряд. Нехай y_1, y_2, ...y_T значення спостережень за економічним процесом протягом T періодів. Дана послідовність є числовими значеннями, кожен з яких має індекс, який залежить від номера періоду, за який він спостерігався. Послідовність, що записана зі зростанням індексу, називається часовим рядом $\{Y_T\}$. Що ж, дамо означення, що часовим рядом є послідовність даних дискретного часу.

Особливістю часових рядів є те, що дуже важливим є спосіб впорядкування даних в часі, оскільки спостереження не є незалежними, тоді як більшість інших статистичних теорій стосується випадкових вибірок незалежних спостережень.

Будь-який часовий ряд можна представити за допомогою чотирьох компонентів такі як тренд, сезонні коливання, циклічні зміни та нерегулярні фактори. Побудова часового ряду полягає у його розкладі на компоненти, кожен з яких аналізується власними методами:

$$y_t = tr_t + s_t + c_t + r_t \tag{1.1}$$

Тренд tr_t – загальна тенденція при різнонаправленому русі, визначена загальною спрямованістю змін показників часового ряду. Тренд може показувати зростання або зниження значень часових рядів, Що ж, аналіз часового ряду починається з виділення тренду.

Серед чинників, що визначають регулярні коливання ряду, розрізняють сезоні та циклічні.

Сезонні коливання s_t , мають періодичний чи близький до нього характер. Це короткочасні рухи, що відбуваються в даних через сезонні фактори. Наприклад, ціни на певні продукти влітку вищі, ніж взимку.

Циклічні коливання c_t схожі на сезонні, але виявляються на триваліших інтервалах часу. Пояснити циклічні коливання можна дією тривалих циклів економічної, демографічної чи іншої природи.

Тренд, сезонна і циклічна компоненти є не випадковими, тому їх називають систематичними або детермінованими компонентами часового ряду. Одним з параметрів, від яких залежить детермінований компонент часового ряду є час.

Нерегулярні коливання r_t це раптові зміни, що відбуваються в часових рядах і які навряд чи будуть повторені. Ці коливання є складовими часових рядів, які не можна пояснити тенденціями, сезонними або циклічними рухами. Ці варіації іноді називають залишковими або випадковими компонентами. Вони

можуть спричинити постійну зміну тенденцій, сезонних і циклічних коливань протягом майбутнього періоду.

Представлення часового ряду за формулою (1.1) називається адитивною моделлю представлення часового ряду.

На сьогоднішній день існує багато різноманітних методів прогнозування економічної інформації, які ми розглянемо в другому розділі. Модель часових рядів враховує минулу поведінку даної змінної і використовує цю інформацію для прогнозування її майбутньої поведінки. Важливо, що під час прогнозування часових рядів аналізуються лише дані спостережень без додаткової інформації та без впливу зовнішніх сил. Основний напрямок розвитку теорії часових рядів є моделювання і аналіз процесу r_t .

1.5 Метрики для аналізу результату

Для аналізу проблеми прогнозування цін використовуються статистичні та нестатистичні метрики.

Нестатистичні показники включають річний прибуток конкретної моделі, а також коефіцієнт попадання чи кількість раз, коли модель правильно передбачила, що ринок буде йти вгору чи вниз.

Статистичні показники включають середню абсолютну похибку (MAE), середньоквадратичну помилку (MSE), середньоквадратичний квадрат (RMSE), середню квадратична помилку передбачення (MSPE) та коефіцієнт кореляціїю.

1.5.1 Середня абсолютна помилка (MAE)

Середня абсолютна є різницею між фактичним значенням та прогнозом. Якщо y_t – прогноз значення часового ряду у періоді t , тоді метрика задається формулою:

$$MAE = \sum_t \left| y_t - \hat{y}_t \right| .$$

Важливо, що MAE є лінійною оцінкою, а тому всі індивідуальні відмінності зважуються однаково в середньому.

1.5.2 Середньоквадратична помилка (MSE)

Середньоквадратична помилка є середньою квадратичною різниці між прогнозом і відповідним спостережуваним значенням. MSE надає відносно високу вагу великим похибкам, оскільки помилки підносяться до квадрату перед тим, як вони усереднюються,. Маємо, що MSE доцільно використовувати, коли великі помилки небажані.

Припустимо, що y_t прогноз значення часового ряду у періоді t , тоді метрика задається формулою:

$$MSE = \sum_t (y_t - \hat{y}_t)^2 .$$

Метрики MSE та MAE можуть набувати значення від 0 до ∞. Дані метрики не враховують напрямок помилок. Отже, чим менше значення може приймати приймає показник, тим точнішим є наш прогноз.

Висновки до розділу 1

У першому розділі було розглянуто поняття «успішність компанії». Ми виділили, що критерієм розвитку підприємства є його вартість, оскільки вона є результуючою характеристикою фінансових показників діяльності компанії. Далі ми будемо розглядати компанії, які представлені на фондових ринках, тому для нас є актуальним поняття ринкової капіталізації компанії.

Основою ринкової капіталізації компанії є акції компанії, тому ми звузили нашу задачу до задачі прогнозування вартості акцій компаній. Ми розглянули існуючі підходи до прогнозування фінансових даних: базові принципи фундаментального, технічного та технологічного аналізу. Було обрано технологічний аналіз для поточного дослідження.

Також було розглянуто поняття часового ряду, та детально описано що він собою являє. Нашим завданням є побудова прогнозу вартості акцій, а тому доцільно використовувати метрики для їх оцінювання.

РОЗДІЛ 2 АНАЛІЗ МЕТОДІВ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ

2.1 Класичні математичні методи прогнозування вартості акцій

Розглянемо класичні методи прогнозування вартості акцій. Далі ми ознайомимося більш детально з методами згладжування та авторегресійними методами.

2.1.1 Методи згладжування

Згладжування це один із найважливіших важливий і найпоширеніших методів прогнозування фінансових ринків. Найчастіше, різні методи згладжування мають за основу концепціію ковзних середніх. Як результат, зменшує вплив випадкового компонента в часовому ряді. Основна гіпотеза методів згладжування полягає у тому, що коливання минулих значень це є випадкові відхилення від деякої плавної кривої. І що цікаво, ця крива може бути екстрапольована для прогнозування.

2.1.1.1 Просте ковзне середнє

Ковзне середнє являє собою інструмент згладжування часових рядів, який використовують для відображення змін на біржі. Важливо, що набір значень, що усереднюють, безупинно рухається в часі. Даний метод відображає тенденцію зміни цін, а також згладжує їхні несуттєві коливання. Оскільки

ковзне середнє є середнім значенням цін у минулому, графіки ковзних середніх “відстають” від поточних змін у часовому ряді. На ринках з чітко вираженим трендом ковзні середні показують добрий результат, проте на ринках, де немає чітко вираженої тенденції, метод дає великі похибки. Формально метод описується так:

$$\hat{y}_t = \frac{1}{N} \sum_{i=1}^{N} y_{k-i},$$

де N - число попередніх моментів часу,

y_{k-i} - реальні значення показника в момент часу t_{k-i}.

2.1.1.2 Зважене ковзне середнє

Даний метод розширює ідею простого ковзного середнє, оскільки кожен компонент часового ряду зрівноважується відповідними ваговими коефіцієнтами, тобто:

$$\hat{y}_t = \frac{1}{N} \sum_{i=1}^{N} w_i y_{k-i},$$

де N - число попередніх моментів часу,

y_{k-i} - реальні значення показника в момент часу t_{k-i},

w_i - ваговий коефіцієнт для i того компоненту ряду.

Перевагою зважених коефіцієнтів є насамперед те, що отримана оцінка тренду є гладшою. Значення спостережень можуть бути зважені за допомогою коефіцієнтів, замість того, щоб кожне вхідне спостереження швидко змінювало значення середнього.

2.1.1.2 Експоненціальне ковзне середнє

Даний метод бере за основу ідею зваженого ковзного середнього. У порівнянні з простим ковзним середнім, експоненціальне ковзне середнє більший акцент робить на останнії точках даних і результуючих усереднених значень, які є ближчими до фактичних спостережень за набором даних. Цікаво, що вагові коефіцієнти зменшуються експоненціально, а тому акцент падає на останні спостереження. Припускається, що в цьому процесі не існує сезонних коливань та систематичних трендів, або щоб вони були визначені та вилучені.

Процес експоненціально зваженого ковзного середнього заданого часового ряду визначається як:

$$\hat{y}_t = \alpha y_t + (1-\alpha) y_{t-1},$$

де α - постійний коефіцієнт згладжування від 0 до 1,

y_t - значення в період часу *t*,

y_t - значення EMA прогнозу в будьякий період часу *t*.

Використовуючи послідовно формулу експоненційно зваженого середнього до всього ряду, можна записати наступним чином:

$$\hat{y}_t = \alpha(y_t + (1-\alpha)y_{t-1} + (1-\alpha)^2 y_{t-2} + \ldots + (1-\alpha)^k y_{t-k}) + (1-\alpha)^{k+1} y_{t-(k+1)}$$

для всіх $k \in \{0, 1, 2, ..\}$.

Маємо, що вага кожного спостереження $y_{t\ i}$ визначається як $\alpha(1\ \alpha)^{i}$. З кожним моментом часу зважене середнє оцінює все більшу кількість спостережень. Як результат, відповідні ваги зменшуються в геометричній прогресії.

У експоненціальному згладжуванні, є один або більше параметрів згладжування, які потрібно визначити. Дані параметри впливають на вибір вагів для зважування спостережень.

Коефіцієнт α набуває значення від 0 до 1. Якщо α близько до 1, то згладжування мале, і y_t приблизно дорівнює y_t . Це прийнятно у тому випадку, якщо очікуються дуже великі зміни в значенні середнього. У випадку, коли значення α близько до 0, прогноз дає дуже згладжені оцінки середнього значення, і не враховує останні спостереження.

2.1.1.3 Подвійне експоненціальне ковзне середнє

Даний метод не спрацьовує добре, якщо в даних наявний чітко виражений тренд. Для таких ситуацій було розроблено кілька інших методів, такі як подвійне експоненціальне згладжування чи експоненціальне згладжування другого порядку, що на практиці є рекурсивним застосуванням експоненційного фільтра два рази. Важливо, що основна ідея подвійного експоненціального згладжування полягає в тому, щоб враховувати можливість ряду мати тренд. Кожен набір даних часових рядів може бути розкладений на його складові, які є трендом tr_t , сезонною компонентою s_t та нерегулярними коливаннями r_t .

Для того, щоб висловити це з математичної точки зори, маємо три рівняння:

$$a_t = \alpha y_t + (1-\alpha)(a_{t-1} + b_{t-1})$$
$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$
$$\widehat{y_{t+h}} = a_t + hb_t,$$

де $0 < \alpha < 1$ - коефіцієнт згладжування даних,

$0 < \beta < 1$ - коефіцієнт згладжування тренду.

2.1.1.4 Потрійне експоненціальне ковзне середнє

Даний метод застосовує експоненціальне згладжування тричі. Цікаво, що модель використовується для даних, в яких може бути наявний як тренд, так і сезонність.

Потрійне експоненціальне середнє задається трьома параметрами, які згладжують модель, а також рівняння для коригування сезонної компоненти.

$$a_t = \alpha(y_t - s_{t-L}) + (1-\alpha)(a_{t-1} + b_{t-1})$$
$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$
$$s_t = \gamma(y_t - a_t) + (1-\gamma)s_{t-L}$$
$$\widehat{y_{t+h}} = a_t + hb_t + s_{t+1+(h-1)\ mod\ L},$$

де $0 < \alpha < 1$ коефіцієнт згладжування даних,

$0 < \beta < 1$ - коефіцієнт згладжування тренду,

$0 < \gamma < 1$ - коефіцієнт згладжування сезонної компоненти,

L - довжина сезонного циклу,

H - кількість кроків прогнозування.

2.1.2 Моделі авторегресії

Модель авторегресії є одним з найбільш ефективних інструментів для прогнозування майбутніх значень часового ряду. Дана модель є надзвичайно важливою, адже вона має такі переваги як гнучкість, а також здатність описувати всі особливості стаціонарних часових рядів. Цікаво, що авторегресивні частини даної моделі описують впливання послідовних спостереджень в часі один на одного. Частини ковзних середніх захоплюють деякі можливі неспостережувані потрясіння. На практиці, модель авторегресії дозволяють моделювати різні явища які оточують нас.

2.1.2.1 Проста модель авторегресії

Дана модель показує нам, що значення вихідної змінної лінійно залежить тільки від власних попередніх значень, а також від випадкового шуму.
Просту авторегресійну модель порядку *p* можна позначити як *AR(p)*. Математично модель визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + e_n \,,$$

де a_i - параметри моделі,

e_n - білий шум.

Найпростішим процесом *AR*, який не має залежності між компонентами є *AR(0)*. Цікаво, що на значення прогнозу впливає шум або похибка.

Для випадку *AR(1)* з коефіцієнтом a_1 , маємо, що тільки попереднє значення часового ряду впливає на прогноз.

Якщо ж a_1 близький до 0, то процес буде виглядати як білий шум. Якщо a_1 за модулем буде наближатися до 1, то вихідне значення буде отримувати більший внесок від попередніх значень ряду у порівнянні з шумом.

2.1.2.2 Авторегресія з ковзним середнім (ARMA)

Дана модель характеризує стохастичний процес за допомогою двох компонентів – авторегресії та ковзного середнього.

Цікаво, що частина авторегресії передбачає регресування змінної на власні минулі значення. Частина ковзнього середнього включає моделювання похибки представленої як лінійна комбінація похибок, що відбуваються в минулому.

Модель з p авторегресійними компонентами та q компонентами для ковзного середнього можна позначати як *ARMA(p, q)*. Математично модель визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n ,$$

де a_i, b_i - параметри моделі,

e_n білий шум.

Модель авторегресії з ковзним середнім доцільно застосовувати у випадку, коли система є функцією ряду неспостережуваних потрясінь, а також

власної поведінки. Наприклад, ціни на акції можуть суттєво змінитись за наявності фундаментальної інформації, а також продемонструвати ефект повернення до середнього через певні дії учасників ринку.

2.1.2.3 Авторегресія з інтегрованим ковзним середнім (ARIMA)

Дана модель базується на моделі авторегресії з ковзним середнім. Модель ARIMA є більш складною та складається з трьох компонентів:

1. Авторегресії, яка використовує залежний зв'язок між состереженнями із затримкою в часі.
2. Інтегрованої компоненти, яка має на меті зробити часовий ряд стаціонарним, а тому даний компонент диференціює вхідні спостереження.
3. Ковзного середнього, який використовує залежний зв'язок між спостереженням і відхиленням від ковзного середнього минулих спостережень.

Важливо, що кожен з цих компонентів задає параметри моделі. Модель авторегресії з інтегрованим ковзним середнім можна позначити як *ARIMA(p,d,q)*, де:

– p - кількість лагових спостережень, включених в модель;
– d - кількість разів, коли було застосовано диференціювання до вхідних спостережень;
– q розмір вікна ковзного середнього.

Задля того, щоб застосувати модель ARIMA ми використовуємо формули моделі ARMA з однією відмінністю – на вхід замість y_t подається $\Delta y_t = y_t y_y 1$.

Ми можемо використовувати значення 0 для кожного з параметрів. Таким чином, ми вкажемо на відсутність відповідного компоненту моделі. Модель авторегресії з інтегрованим ковзнім середнім може бути налаштована для виконання функцій моделі авторегресії з ковзнім середнім, або ж простої авторегресії чи ковзного середнього.

Також не потрібно забувати, що моделі авторегресії і ковзного середнього являються лінійними моделями, що працюють на стаціонарних часових рядах. Модель інтегрованої компоненти є процедурою попередньої обробки даних, задля того, щоб звести ряд до стаціонарного.

2.1.2.4 Знаходження параметрів моделі

Значення параметрів p (кількість лагових спостережень) та q (розмір вікна ковзного середнього) можуть бути знайдені за допомогою побудови часткових автокореляційних функцій для оцінки p та q. Для того, щоб це зробити використовують інформаційний критерій Акайке (AIC), який шукає компроміс між точністю та простотою моделі, при оцінці кількості інформації, втраченої моделлю. Як результат, моделі, які показали себе краще при використанні меншої кількості функцій, отримають кращу оцінку, ніж моделі, що використовують більше функцій. Математично критерій визначається як:

$$AIC = 2k - 2ln(\hat{L})$$

де k - кількість оцінених параметрів у моделі,

L - максимальне значення функції правдоподібності для моделі.

Що ж, модель із мінімальним значенням інформаційного критерію Акайке вважається ліпшою. Можемо зробити висновок, що AIC допомогає оцінити точність моделі, але також “штрафує” за збільшення кількості оцінюваних параметрів. Даний штраф перешкоджає перенавчанню моделі, так як збільшення кількості параметрів у моделі завжди покращує її точність прогнозування на тестових даних. Даний критерій реалізовано у відповідних бібліотеках та фреймворках, а тому у сучасних мовах для аналізу та прогнозуванні, все зроблено для нас, щоб ми коритсувалися цим.

2.2 Методи машинного навчання для прогнозування вартості акцій за допомогою нейронних мереж

Окрім класичних математичних методів для прогнозування вартості акцій існують такой й методи машинного навчання для прогнозування вартості акцій за допомогою нейронних мереж. Цікаво, що для вирішення складних проблем прогнозування часових рядів нейронні мережі маєть переваги у порівнянні з класичними методами. Тому використанням нейронних мереж є популярним та перспективним напрямком для прогнозування вартості акцій на фондовому ринку.

Нейронні мережі мають можливість нелінійного моделювання, без будь-якого припущення про розподіл часового ряду. Кожна модель формується на основі даних адаптивно. Саме тому, штучні нейронні мережі керуються даними, а за своєю природою вони є самоадаптивними. Далі ми розглянемо використання нейронних мереж на фінансових ринках, їх вхідних та вихідних даних, архітектури мережі.

2.2.1 Загальні відомості про нейронні мережі

Структура штучної нейронної мережі (ШНМ) ґрунтується на сукупності з'єднаних вузлів нейронів, що називають штучними нейронами. Кожне з’єднання передає сигнал від одного нейрона до іншого. Дана структура схожа на структуру біологічних нейронів у головному мозку людини.

Модель штучної нейронної мережі складається з наступних типів шарів: вхідний, прихований та вихідний шар. Усі ці шари з'єднані ациклічними зв'язками. Цікаво, що ШНМ можуть мати більше одного прихованого шару. Розглянемо тришарову архітектуру найпростішої ШНМ, яку зображено на рисунку 2.1

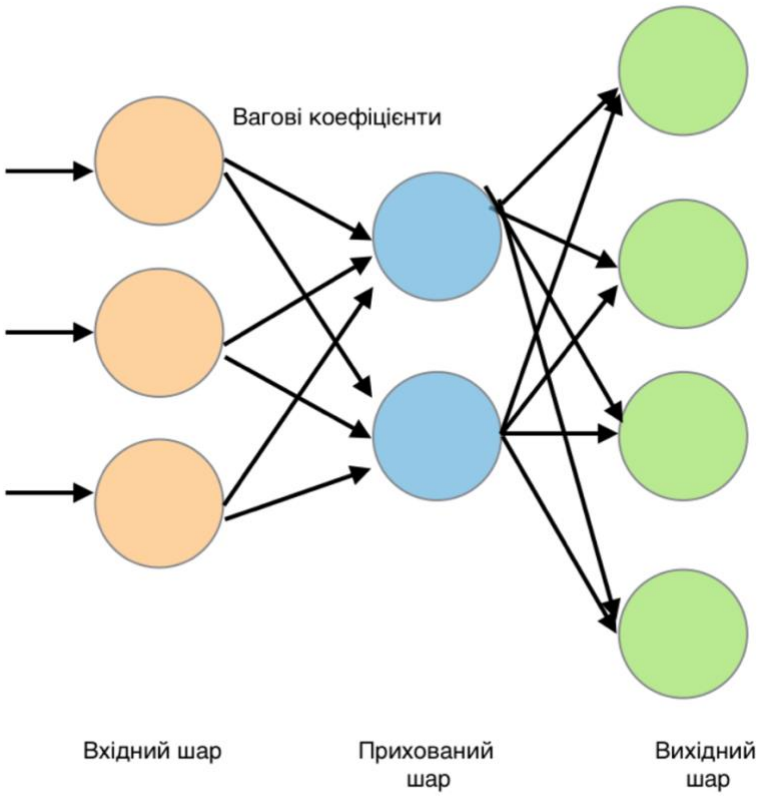

Рисунок 2.1 – Схема нейронної мережі

Вихідне значення нейронної мережі математично задається так:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-i}) + \varepsilon_t, \ \forall t,$$

де p – кількість вхідних змінних,

q – кількість прихованих вузлів,

α_j та β_{ij} – вагові коефіцієнти,

ε_t – випадковий шум.

Розглянемо варіанти використання різних функцій активації g, які можна використовувати на практиці:

1. Сигмоїдна функція.

$$f(x) = \frac{1}{1+e^{-x}}$$

Дана функція є нелінійною. Також вона є однією з найпоширених функцій активацій для навчання нейронних мереж. Сигмоїдну функцію ще називають логістичною функцією.

2. Гіперболічний тангенс.

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Дана функція показує кращий результат у порівнянні з сигмоїдною функцією для багатошарових нейронних мереж. Проте вона також має свої недоліки, а саме не вирішує проблему зникаючого градієнта. Що стосується переваг функції гіперболічного тангенсу, то це те, що вона центрована відносно нуля. Дана особливість допомагає в процесі зворотного поширення помилки. Дана функція активації є менш популярною, чим сигмоїдна функція.

3. Softmax функція.

$$f(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Дана функція використовується для обчислення ймовірності розподілу вектора дійсних чисел. Softmax функція набуває значення діапазоном від 0 до 1. Дана функція активно використовується в методах машинного навчання, а саме задачі класифікації.

4. ReLU функція.

$$f(x) = max(0, x)$$

Дана функція вважається найуспішнішою при виборі функції активації, а також широко застосовується на практиці. ReLU функція показує кращі результати в глибокому навчанні у порівнянні з сигмоїдною функцією та гіперболічним тангенсом. Дана функція активації майже лінійна, а тому їй характерні властивості лінійних моделей. Що ж, основною перевагою функції ReLU полягає в тому, що вона за своєю природою є простою, а тому не потребує обчислення експоненти та ділення. Як результат, вона виконується надзвичайно швидко, що дозволяє нам економити багато часу.

Варто зазначити, існує величезна кількість варіацій для функції активації. В даній роботи ми розглянемо найбільш поширені. Саме такі функції активації ми навели вище.

Навчання нейронної мережі відбувається за рахунок мінімізації функції похибки при реалізації методу зворотного поширення помилки. Даною цільовою функцією є середньоквадратична функція помилки. Наше завдання знайти такі значення ваг нейронної мережі, щоб мінімізувати значення цільової функції.

Що ж, розглянемо найбільш популярні методи оптимізації:

1. Метод градієнтного спуску.

$$x_{t+1} = x_t - \alpha \cdot f'(x_t)$$

Основна ідея даного методу полягає в тому, що для мінімізації функції здійснюються етапи, пропорційні протилежному значенню градієнту. Основним параметром методу градієнтного спуску є швидкість α. Якщо значення параметра α є великим, то можливо робити й більші кроки для пошуку мінімуму, про є недолік - це ризик перейти найнижчу точку. Якщо ж швидкість навчання є малою, то алгоритм буде впевнено рухатися в напрямку негативного градієнта. В цьому випадку реалізація може зайняти багато часу.

2. Метод стохастичного градієнтного спуску.

Даний метод є видом градієнтного спуску, при якому він обробляє 1 елемент для навчання на кожну ітерацію. Що цікаво, так це те, що параметри оновлюються завжду, навіть після однієї ітерації. Така особливість дає змогу оптимізувати нашу цільову функцію швидше, ніж звичайний градієнтний спуск. Серед недоліків можна виділити, що якщо кількість даних для навчання дуже велика, то й кількість ітерацій буде відповідно будет великою.

3. Метод градієнтного спуску міні-серіями.

Даний метод є видом градієнтного спуску, який працює швидше, ніж класичний та стохастичний градієнтні спуски. Припустимо, що існує m вхідних даних, тоді за одну ітерацію методу градієнтного спуску міні-серіями буде оброблятись $b < m$ елементів. Можемо зробити висновок, що якщо кількість навчальних даних велика, то вона обробляється в менших групах на один раз. Перевагою даного методу серед інших є те, що він чудово працює для великої кількість даних для навчання, при цьому оптимізує цільову функцію за меншу кількість ітерацій.

4. Градієнтний спуск з моментом.

Даний метод допомагає прискорити спуск у відповідному напрямку. Також градієнтний спуск з моментом гасить коливання, наближаючись до локального мінімума. Дану особливість можна реалізувати з додаванням до вектора поточного оновлення частку вектора оновлення з попереднього кроку:

$$v_t = \gamma v_{t-1} + (1 - \beta) \cdot dW(x_t)$$
$$W = W - \alpha v_t ,$$

де v і dW аналогічні прискоренням і швидкості,

α - швидкістю навчання,

β - зберігається при 0.9

5. Метод Адама

Даний метод є одним із найпопулярніших алгоритмів, який націлений на оптимізації спуску градієнтів. Таку популярність йому принесло те, що він є обчислювально ефективним, а також має дуже мало вимог до пам'яті. Оптимізатор Адама розраховує адаптивну швидкість навчання для кожного з параметрів з оцінок першого і другого моментів градієнтів.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_1) g_t^2 ,$$

де m_t та v_t оцінки першого та другого моменту відповідно.

Відповідні параметри оновлюються за наступними правилами, які математично визначаються як:

$$\hat{m}_t = \frac{m_t}{1-\beta_1{}^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2{}^t}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t}+\varepsilon} .$$

2.2.2 Вхідні дані для побудови нейронної мережі

Важливим фактором для для побудови штучної нейронної мережі є визначення даних, на яких вона буде тренуватись. В даній роботі мета нейронної мережі це спрогнозувати ціну акції, задля того, щоб прийняти рішення, чи необхідно інвестувати в дану компанію чи ні. Ми сподіваємось, що компанія буде успішною, а тому ціни на її акції дорожчатимуть. Наше завдання у визначенні вхідних даних та індикаторів, які будуть використані для побудови моделі. Також нам необхідно зібрати достатню кількості навчальних даних для навчання нейронної мережі.

Вхідними даними можуть бути як ціни акцій щодня чи їх щоденну зміну, як технічні індикатори, наприклад, ковзні середні, або фундаментальні показники, такі як внутрішня вартість акцій.

Чудовим прикладом такої штучної нейронної мережі з великою кількістю параметрів на вході є JSEsystem, яка займалась моделюванням поведінки Йоганнесбургзької фондової біржі. Дана нейронна мережа мала близько 60 різних показників, які використовувалися даною мережею для розуміння того, що коїться на ринку.

Значення цих вхідних параметрів можна виділити в наступні класи:

– фундаментальні – прибутковість, ціна;

– технічні – ковзні середні тощо;
– економічні – експорт, імпорт;
– процентні ставки;
– ціна на золото та курси валют;
– індекси JSE – ринкові індекси для різних секторів, такі як золото;
– міжнародні індекси – промисловий індекс Доу-Джонса.

Цікаво, що хоч дана штучна нейронна мережа й приймала на вхід близько 60 різних параметрів, проте багато з них виявились надлишковими. В процесі тренування було відкинути близько 20 змінних, також зменшено кількість прихованих вузлів від 21 до 14. Як результат, зросла продуктивність системи та зменшився час навчання моделі.

Визначення та вибір вхідних даних є першим етапом у навчанні штучної нейронної мережі. Другим етапом є обробка та розділення вхідних даних задля того, щоб мережа могла вчитися без перетренування. В даній дипломній роботі для того, задля того, щоб мати об’єктивну порівняльну характеристику для класичних методів та методів машинного навчання, за вхідні дані було обрано попередні значення вартості акцій. Проте в моделі, яка була запропонована нами ми на вхід подаємо кількість позитивних та негативних коментарів. Таким чином ми розширили кількість вхідних параметрів для нашої нейронної мережі. Детальніше про це в наступних розділах.

2.2.3 Нейронна мережа прямого поширення

Нейронна мережа прямого поширення являє собою таку мережу, в якій сигнали поширюються в одному напрямку. Сигнал поширюється починаючи від вхідного шару нейронів і до вихідного шару, проходячи через приховані шари.

Особливістю мереж такого типу є те, що в них немає зворотних зв’язків. Іншим типом нейронних мереж зі зворотними зв’язками є рекурентні нейронні мережі. Спільним між різними видами перцептронів є те, що всі вони є нейромережами з прямим поширенням сигналу. Проте відрізняються вони в основному кількістю шарів, методом навчання, а також функцією активації, які ми розглянули раніше.

2.2.4 Згорткова нейронна мережа (CNN)

Згорткова нейронна мережа (ЗНМ) відноситься до класу глибинних штучних нейронних мереж прямого поширення. Задля того, щоб визначити дану модель, існує декілька теорій. Дана нейронна мережа складається з шарів входу, виходу та прихованих шарів. Приховані шари складаються зі згорткових шарів, повноз'єднаних шарів, агрегувальних шарів та шарів нормалізації.

Основною перевагою згорткової нейронної мережі сереж інших нейронних мереж є використання згорткових шарів. Це зроблено з метою виявлення ознак мережі, що дозволяє тренувати нейронну мережу без важкої попередньої обробки. В даній роботі було застосовано згорткову мережу до часових рядів вартості акцій компанії.

Архітектуру класичної згорткової нейронної мережі можна представити на рисунку 2.2:

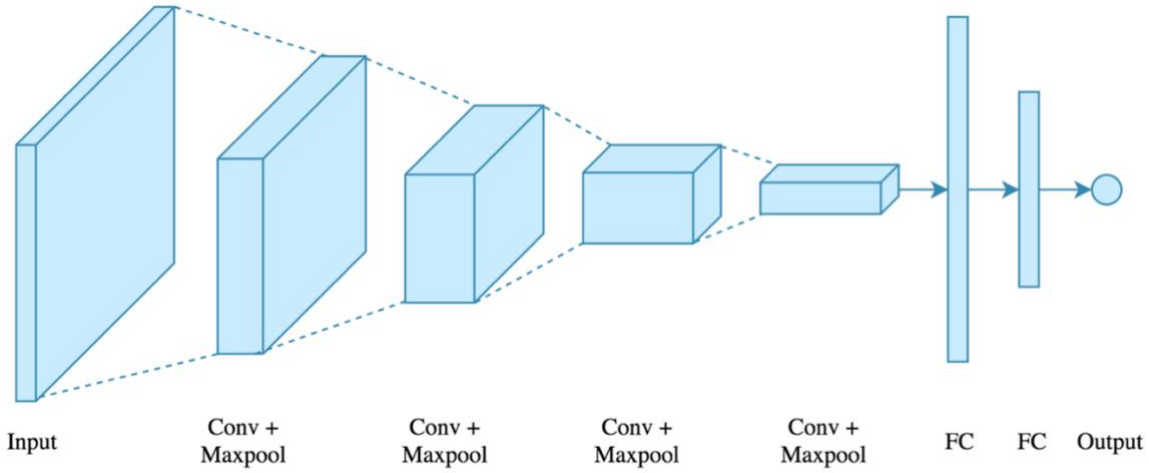

Рисунок 2.2 – Архітектура згорткової нейронної мережі

2.2.5 Рекурентна нейронна мережа (RNN)

Рекурентна нейронна мережа (РНН) являє собою нейронну мережу, нейрони якої передають сигнали зворотного зв'язку один одному. Основна ідея даної нейронної мережі полягає у використанні послідовної інформації. Особливістю рекурентної нейронної мережі є те, що для обробки довільних послідовностей входів, вона використовує внутрішню пам'ять.

В класичній нейронній мережі ми стверджуємо, що всі входи і виходи є незалежними один від одного. Проте таке твердження для багатьох завдань є не найкращим варіантом. Дана нейронна мережа називається рекурентною оскільки вона виконує одну і ту ж дію для кожного елемента послідовності. Вихідні дані в такій мережі залежать від попередніх обчислень. Рекурентні нейронні мережі цілком влаштовують нас для прогнозування вартості акцій компанії, так як майбутні кроки можуть залежати від минулих.

2.3 Аналіз тональності тексту для прогнозування вартості акцій

Використання нейронних мереж для задачі прогнозування вартості акцій є надзвичайно перспективним напрямком для нас, адже вони дають гарні результати. Проте, що якщо ми на вхід крім ціни акцій додамо ще декілька параметрів, які окрім ціни акцій для кожного дня будуть показувати ще й кількість позитивних та негативних новин та згадок щодо компанії. Ми раніше згадували, що живемо в інформаційному суспільстві і нас оточує багато інформації. Як щодо того, щоб спробувати використовувати цю інформацію для прогнозування вартості компаній? Насамперед, нам необхідно зрозуміти, яка інформація є позитивною, а яка негативною, для цього ми використаємо аналіз тональності тексту. Далі ми детально розглянемо як він працює та як саме ми зможемо його використати для нашої задачі.

2.3.1 Загальні відомості про аналіз тональності тексту

Аналіз тональності тексту відноситься до класу методів аналізу тексту в комп'ютерній лінгвістиці. Основна його задача - автоматизоване виявлення в текстах емоційно забарвленої лексики і емоційної оцінки авторів.

Даний клас методів займається аналізом людських думок, емоцій, та формуванням оцінки певних подій. Важливо, що думки є суб’єктивними, на відміну від фактів, які ми сприймаємо об’єктивно. Задля того, щоб оцінити тональність тексту необхідно врахувати думки різних людей, а не тільки погляд однієї людини. Думка однієї людини – це її власний погляд на ситуацію, а тому вона є суб’єктивною і цього не достатньо для його застосування. Так як

кожного дня кількість інформації в Інтернеті зростає, люди постійно висловлюють свої думки будь то написання відгуків чи коментарів, існує потреба в автоматизованому аналізі тексту. Завдяки емоційної тональності окремих слів з тексту ми можемо визначити тональність всього тексту. Ми будемо класифікувати емоційне забарвлення як позитивне, негативне чи нейтральне. Наприклад, слова «добре», «прекрасно» можуть бути промарковані як позитивні. Інша група слів, такі як «погано», «жахливо» позначаються як негативні. Що ж, тональність – це інструмент для класифікації та вилучення емоційно-смислової лексики з тексту.

Отже, основна задача аналізу тональності тексту є виявлення його емоційного забарвлення, яке може бути позитивним, негативним чи нейтральним. Що цікаво, так це те, що текст який ми аналізуємо має суб'єктивну думку. Основна задача сентименатального аналізу тексту це те, щоб оцінка метода співпала з оцінкою автора. Більш детально ми розглянемо це в наступних розділах.

2.3.2 Проблеми при аналізі тональності тексту за словником

Слова емоційного забарвлення є одними з найбільш важливими індикаторами сентименту. Зазвичай ми використовуєто ці слова для вираження позитивної чи негативної думки. Наприклад, «добре», «прекрасно» – вживаються для вираження позитивної думки. Інша група слів, такі як «погано», «жахливо» – для вираження негативної думки. Що цікаво, так це те, що окрім слів, існують фрази та крилаті вислови. Вони є також важливим інструментарієм аналізу тональності тексту, так як несуть позитивну чи

негативну думку. Вони разом формують «лексичний словник», тобто такий набір виразів, кожний елемент якого має певну думку.

Використання слів та виразів є досить не ефективним, навіть незважаючи на те, що вони є дуже важливими при аналізі тональності тексту.

Що ж, ознайомимося з основними проблемами, які виникають при аналізі тональності тексту:

– наявність неологізмів та помилок у словах, в цьому випадку використання словникових методів є недоцільним;
– протилежний відтінок слів при використанні в іншому контексті;
– використання слова, яке має емоційне забарвлення саме по собі, в реченні може мати нейтральний сентимен; таке може виникнути в питальних та умовних реченнях;
– речення, що містять сарказм є дуже складними для аналізу; проблема використання сарказму на сьогоднішній день є дуже популярною.

2.3.3 Методи попередньої обробки даних

Якість кінцевого результату залежить від того, яким форматом документ буде поданий для класифікатора. Також на результат впливає вибір набору характеристик для складання вектору ознак.

Вектор ознак являє собою алгебраїчну модель подання текстів:

$$d_j = (w_{1j}, w_{2j}, \dots, w_{nj}),$$

де d_j – векторне представлення документу j, w_{1j} – вага терміну i в документі j, n – кількість термінів.

Одним з найпоширеніших способів подання документа в задачах аналізу тексту – це у вигляді набору слів або набору N-грам. Наприклад, речення «Я обожнюю цей день» можна представити у вигляді набору уніграм (Я, обожнюю, цей, день) або біграм (Я обожнюю, обожнюю цей, цей день).

Цікаво що, найкращі результати показує використання уніграм та біграм, а ось використання N-грам більш високих порядків призводить вже до втрати ефективності. Що ж, на практиці, багато хто оцінює результати із застосуванням уніграм, біграм та їх комбінації, наприклад - Я, обожнюю, цей, день, Я обожнюю, обожнюю цей, цей день. Використання уніграми може бути більш доцільним за використання біграм в залежності від типу даних. Інколи комбінація уніграм і біграм дозволяє покращити результат.

Іншим способом представлення тексту є використання символьних N-грам. Речення «Я обожнюю цей день» може бути подане у вигляді наступних 5-символьних N-грам: «я обо», «обожн», божню», «ожнюю», і т.д. Даний спосіб може здатися більш примітивним за попередні, оскільки набір символів фіксованої довжини не здається інформативним, проте цей може давати результати навіть краще ніж N-грами слів, проте в деяких випадках.

Список випадків, в яких N-грами можуть бути корисними:

- мови зі змінною морфологією – наприклад, слова можуть мати багато різних варіацій, при цьому корінь слова залишається незмінним.
- велика кількісті орфографічних помилок – сукупність символів у тексті з помилками буде майже таким же, як з набором символів у тексті без помилок;

2.3.4 Підходи до класифікації тональності

Для аналізу тональності тексту існують різні методи. Вони відрізняються між собою за точністю та швидкістю. Серед популярних методів можемо виділити методи машинного навчання з/без учителя, а також методи, які основані на словниках та правилах. Методи машинного навчання застосовуються для побудови системи з автоматичним аналізом тексту. Цікаво, що методи машинного навчання без учителя, дають нижчу точність, аніж методи машинного навчання з учителем. Що стосується методів, що базуються на словниках та правилах, то вони дають непогану точність. Недоліком даних методів є те, що вони є залежними від предметної області.

В даній роботі ми будемо розглядати аналіз тональності тексту з підходом, заснованим на методах машинного навчання з учителем. Даний метод дозволяє отримати необхідну нам точність, а також простота його реалізації.

Задача аналізу тональності тексту формулюється як задача класифікації тексту на два класи – позитивний та негативний. Основними даними для навчання та тестування, є коментарі, дописи та відгуки. Що ж, можемо сказати, що задача аналізу тональності зводиться до задачі текстової класифікації. Основою текстової класифікація є розподіл тексту за різними темами, наприклад спорт, наука та ін. Для даної класифікації ключовими компонентами є ключові слова, що відносяться до певної теми. В аналізі тональності тексту емоційно забарвлені слова, які характеризують позитивну чи негативну оцінку є більш важливими. Приклад слів – добре, чудово, погано, жахливо.

Отже, можемо зробити висновок, що задача аналізу тональності тексту є задачею текстової класифікації, а тому для її вирішення можна використовувати методи, що застосовується для текстової класифікації. До таких методів належить метод опорних векторів, наївного Баєса та максимум ентропії.

2.3.5 Оцінка якості аналізу тональності

Якість і точність моделі аналізу тональності тексту можна оцінити наступним чином: наскільки результат оцінки тексту є близьким до думки людини щодо емоційної оцінки досліджуваного тексту. Далі ми розглянемо такий показник як точність:

$$P = \frac{M}{N}$$

де M – правильно розпізнані думки,

N – загальна кількість думок знайдених системою.

Маємо, що, точність виражає кількість досліджуваних текстів та речень в оцінці яких думка моделі аналізу тональності збіглось з думкою експерта. Цікаво, що експерти зазвичай погоджуються в оцінках тональності тексту в 80% випадків, тому ми будемо вважати це еталоном. Можемо зробити висновок, що модель з точністю визначення тональності тексту 70-80% робить це як людина, і це дійсно чудовий результат.

Висновки до розділу 2

У розділі 2 було розглянуто традиційні методи прогнозування вартості акцій, а саме методи згладжування та методи авторегресії. Також було розглянуто параметри, які впливають на побудову прогнозу кожним методом.

Окрім використання класичних математичних методів для прогнозування ціни акцій, ми також розглянули методи машинного навчання з використанням нейронних мереж для вирішення цієї задачі. Останні дослідження використання нейронних мереж в задачах прогнозування доводять доцільність та перспективність використання цих методів. Ми розглянули математичні основи нейронних мереж, а саме нейронних мереж прямого поширення, згорткових нейронних мереж та рекурентних нейронних мереж. Важливо правильно обрати вхідні дані. В даній роботі в якості вхідних даних брались значення ціни акціій, а також кількість позитивних та негативних коментарів щодо компанії.

Також ми ознайомилися із загальної задачі аналізу тональності. Далі ми розглянули основні терміни, що використовуються при формулюванні даної проблеми. Проаналізували, з якими основними проблемами зіштовхуються дослідники при виконанні аналізу тональності тексту.

Отже, завдяки використанню аналізу тональності тексту ми зможемо ми зможемо аналізувати інформацію щодо певної компанії з простору інтернету. Ми будемо розуміти, яка кількість інформації була позитивною щодо, компанії, а яка негативною. Далі, ми зможемо побудувати таку модель, яка зможемо прогнозувати вартість ціни акції з використанням не тільки цін за минулий період, а я з використанням інформаційних новин щодо компанії. Ми будемо використовувати методи машинного навчання з учителем для аналізу тональності новин.

РОЗДІЛ 3 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

В цьому розділі ми розглянемо практичне застосування методів для задачі прогнозування вартості акцій компанії.

3.1 Обґрунтування вибору платформи та мови реалізації

Програмний продукт був реалізований у вигляді мобільного застосунку для платформи iOS – мобільна операційна система від компанії Apple. iOS заснований на концепції використання жестів для взаємодії з користувацьким інтерфейсом. Дана концепція призначена для прямого контакту користувача з екраном пристрою. Що цікаво, що дана операційна система є надзвичайно швидкою та зручною у користуванні, безпечною та надійною адже вона працює тільки на пристроях iPhone. З кожним роком дана операційна система стає все швидшою та швидшою, а тому відкриває нові шляхи для створення продуктів інженерами.

Чому була вибрана мобільна платформа? Відповідь дуже проста. З кожним роком кількість користувачів, які користуються мобільним телефоном зростає. Люди все частіше й частіше користуються програмами на мобільних девайсах замість десктопних. Отже, вибір мобільної платформи допоможе застосунку бути максимально зручним для користувача.

Інше питання, чому було обрано саме iOS, а не Android, наприклад. Так я розумію, що користувачів на Android набагато більше, у відсотковому відношенні близько 80% усіх девайсів працюють на Android. Проте якщо порівняти дохід Play Market та App Store, то ситуація тут виглядає навпаки,

адже користувачі iPhone є більш лояльними та частіше купують додатки, аніж власники Android телефонів. Для MVP продукту, вибір iOS користувачів є гарним стартом. Більшість відомих нам застосунків спершу з'явилися саме в App Store, тому ми й обрали користувачів iOS як цільову аудиторію.

Що стосується вибори мови реалізації, то ми обрали Swift, для написання програми на iOS. Невелика частина проекту написана на іншій мові програмування для пристроїв Apple – Objective-C. Swift являє собою багатопарадигмову компільовану мова програмування. Дана мова була розроблена компанією Apple з метою заміни в майбутньому застарілої мови Objective-C, яка з кожним роком стає все більше втрачає популярність серед розробників.

Дана мова успадкувала найкращі елементи мов C і Objective-C, а тому синтаксис зрозумілий для розробників. Використання Swift значно збільшує надійність і безпеку коду, також він є стійкішим до помилкового коду. Програми, які написані даною мовою, компілюються в машинний код. В свою чергу, це забезпечує високу швидкодію. Є інформація від інженерів Apple, що код мови Swift виконується швидше в 1.3 рази чим код на Objective-C. Важливо, що ця мова йде в ногу з сучасними мовами програмування, а тому має такі сучасні методи програмування, як замикання, дженерики, лямбда-вирази, кортежі, операції над колекціями, а також елементи функціонального програмування.

Основним застосуванням даної мови є розробка користувацьких застосунків для існуючих платформ компанії Apple: iOS, macOS, tvOS та watchOS. Важливо, що Swift сумісний з Objective-C, а тому застосунок, який написано мовою Swift може працювати з кодом на Objective-C.

Отже, з платформою та мовою реалізації мобільного застосунку ми розібралися, проте необхідно ще розглянути нашу модель.

Модель прогнозування оцінки вартості компанії була реалізована мовою

Python. Дана мова є високорівневою, інтерпретованою та об'єктно-орієнтованою мовою зі строгою динамічною типізацією. Вона є привабливою для швидкої розробки програм, адже містить структури даних високого рівня та динамічне зв'язування.

Ця мова підтримує модулі та пакети модулів, які позитивно сприяють модульності та повторному використанню коду. Важливо, що сам інтерпретатор Python та її стандартні бібліотеки доступні на всіх основних платформах. В даній мові підтримуються наступні парадигми програмування: функціональна, об'єктно-орієнтована та процедурна.

Python має ефективний підхід до об'єктно-орієнтованого програмування, проте він є надзвичайно простим та зручним для програмістів. Що ж, саме завдяки динамічній обробці типів, сучасному та елегантному синтаксису та своїй інтерпретованості, дана мова програмування є ідеальною для швидкої розробки прикладних програм у багатьох галузях на більшості платформ, а також для написання різноманітних скриптів та моделей, що ми власне і будемо робити.

Отже, наша модель буде реалізовано мовою Python. Це ідеальна мова для таких кейсів, тому що вона моє багато бібліотек та фреймворків для роботи з даними – візуалізація, обробка, трансформація, читання з файлу, запис в файл та ін. З цією задачею чудово справиться бібліотека обробки даних Python Pandas.

Дана бібліотека забезпечує виразні структури даних призначені для спрощення роботи з різними типами даних. Бібліотека є простою та інтуїтивно зрозумілою. Pandas є однією з основниї високорівневих бібліотек для практичного аналізу даних в Python. З кожним роком її можливості стають все більшими й більшими, а тому вона є надзвичайно популярною серед програмістів.

Pandas добре справляється з різними типами даних:

– табличні дані як в таблиці SQL;

– впорядковані і невпорядковані часові ряди;
– дані довільної матриці з рядками і стовпцями;
– будь-яка форма наборів статистичних даних.

В якості бібліотеки для побудови моделі на мові Python було обрано Scikit-learn. Вона містить багато корисних класів та методів для нейронних мереж, лінійної регресії, дерев рішень тощо. За допомогою Scikit-learn можна швидко та якісно побудувати будь-яку потрібну модель для задач прогнозування, класифікації, тощо. Scikit-learn – одна з найкорисніших бібліотек для машинного навчання в Python. Вона побудована на бібліотеках NumPy, SciPy і Matplotlib. Scikit-learn містить багато зручних інструментів для роботи з методами машинного навчання і статистичного моделювання, зокрема з задачами класифікації, кластеризації та регресії.

Що ж, з технологіями реалізації моделі ми також розібралися, проте як використовувати модель, написану мовою Python в iOS застосунку, який компілюється мовою Swift. Вихід є, ми будемо використовувати фреймворк Core ML від Apple для роботи з моделью. Це є надзвичайно зручно для інженерів, адже всю логіку роботи бере на себе Apple. Все, що нам потрібно, так це реалізувати інтерфейс моделі, яку може використовувати будь-який застосунок на iOS.

І останнє, в якості системи контролю версій під час створення застосунку оберемо Git. Git — розподілена система керування версіями файлів для спільної роботи. Дана система є надзвичайно ефективною, надійною та продуктивною. Вона надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Важливо, що для забезпечення стійкості до змін та цілісності історії використовуються криптографічні методи. Що ж, з вибором платформи та мови реалізації ми закінчили. Рухаємося далі.

3.2 Аналіз вимог користувача до роботи програмного продукту

Що ж, основною функцією нашого програмного продукту є прогнозування успішності вибраної користувачем компанії. Для користувача необхідно щоб все виглядало максимально інтуїтивно та зрозуміло. Я думаю, що всі ми хоч раз відкривали додатки для трейдерів, де з усіх сторін мільйон цифр та багато незрозумілих функцій. Це є приклад того, що не потрібно робити.

При запуску застосунку користувач буде бачити тільки список з компаніями. Він може знайти необхідну йому по пошуку, чи вибрав зі списку. Після цього він ознайомиться з фінансовими показниками даної компанії. При необхідності, він може ознайомитися з історичними даними компанії за певний період часу. Далі він натискає на кнопку прогнозування та отримає прогноз успішності компанії. Більш детально як це працює я розповім в наступному розділі. Для користувача необхідно щоб все виглядало максимально просто. Також на останньому екрані він може ознайомитися з останніми новинами щодо компанії, а також її оцінкою аналізу тональності тексту.

Важливо, оскільки це є мобільний застосунок, то пошук компанії, а також її прогноз повинен відбуватися швидко. Іншим моментом є те, що застосунок повинен працювати ефективно, та не використовувати всю пам’ять телефону. Що ж, ми маємо обмеження як по часу запиту, так і ефективності цих запитів.

Також важливо, щоб користувач розумів, що програмний продукт працює справно, в протилежному випадку, а саме у разі виникнення проблеми, необхідно сповістити користувача, що трапилося.

Можемо ознайомитися більш детально з цими вимогами з use case діаграми на рисунку 3.1:

Рисунок 3.1 – Use Case діаграма системи прогнозування успішності компанії

Отже, з аналізом вимог користувача до роботи програмного продукту ми розібралися. Перейдемо до архітектури програмного продукту.

3.3 Аналіз архітектури програмного продукту

Наш програмний продукт реалізований як мобільний застосунок iOS. В якості архітектури додатку я обрав Model-View-ViewModel. Даний шаблон проектування відокремлює розробку графічного інтерфейсу від розробки бізнес логіки, тобто моделі. Іншими словами, MVVM відокремлює представлення від моделі. Модель представлення відповідає за перетворення даних для їх подальшого використання, а тому, можемо зробити висновок, що вона більше схожа на модель, ніж на представлення та містить в собі логіку відображення даних. Архітектура Model-View-ViewModel є набагато стійкішою до масштабування ніж Model-View-Controller. Для навігації я обрав паттерн

програмування Coordinator, який є надзвичайно елегантним та здатним до масштабування. З діаграмою класів даного проекту ви можете ознайомитися на рисунку 3.2:

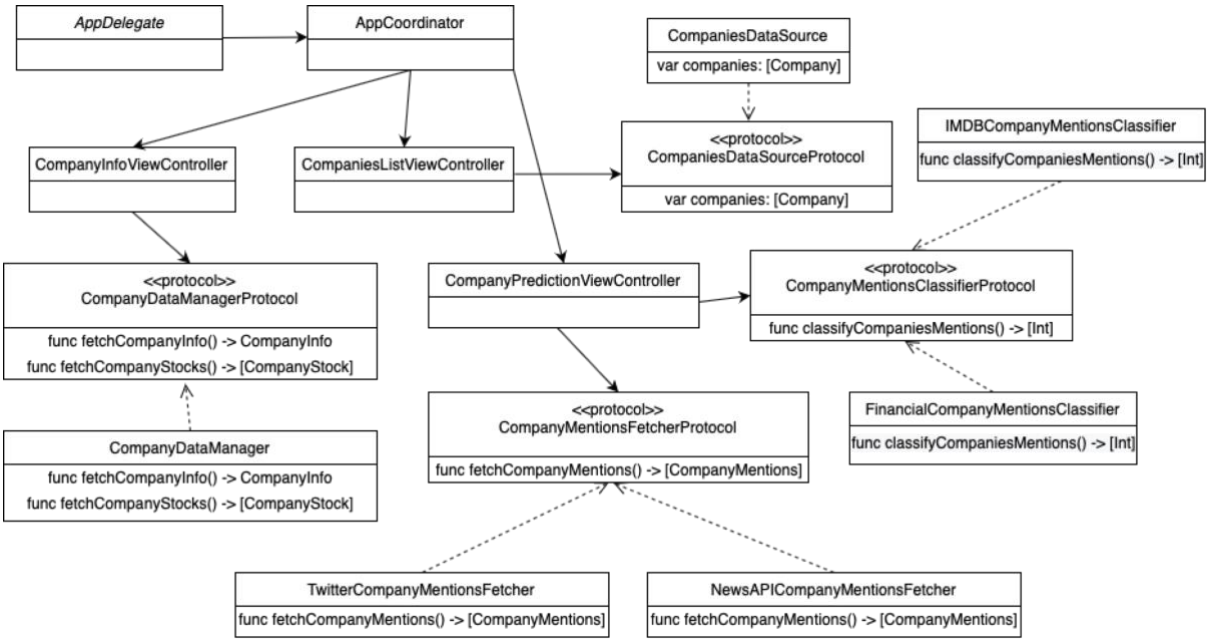

Рисунок 3.2 – Діаграма класів проекту

Ми бичимо, що точкою входу є класс *AppDelegate*. Далі йде *AppCoordinator*, який містить в собі основні екрани, такі як *CompaniesListViewController*, *CompanyInfoViewController* та *CompanyPredictionViewController*. В даних класах міститься логіка для показу списку компаній, логіка запитів щодо основної фінансової інформації про компанію, ціни на її акції, а також останні новини про компанію. Також представлена логіка для аналізу тональності тексту. Зі структурою проекту в середовищі Xcode ви можете ознайомитися рисунку 3.3:

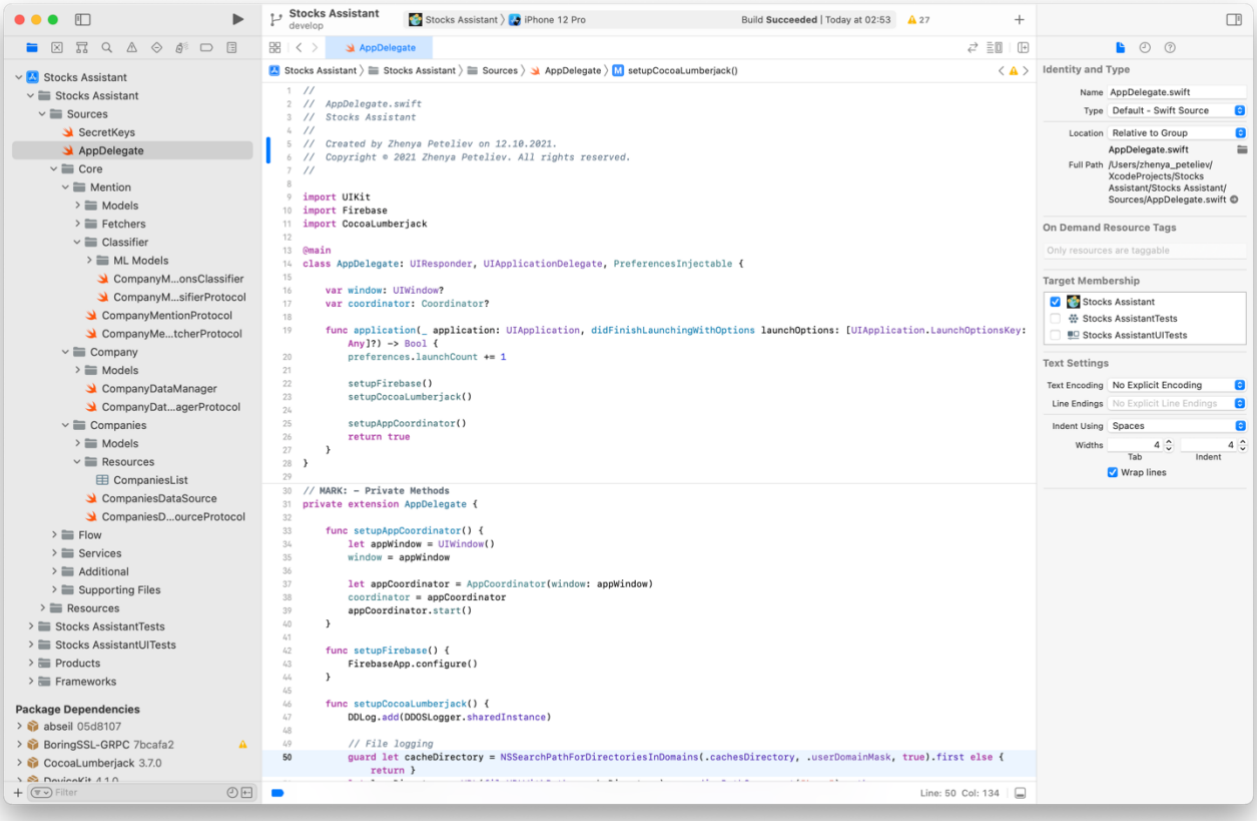

Рисунок 3.3 – Структура проекту в середовищі Xcode

Що стосується моделі прогнозування успішності компанії, то вона має наступний вигляд. Архітектура нейромережі на вхід отримує ціни акцій, а також кількість позитивних та негативних новин. Ціни на акції компанії та інформацію про неї ми будемо брати із сервісу Alpha Vantage, який дозволяє нам отримати інформацію в зручному для нас форматі JSON. Alpha Vantage – веб сервіс, який безкоштовно представляє дані про компанії, які представлені на фондовому ринку. Після того, як ми отримали інформацію від даного сервісу про необхідну нам компанію, ми можемо сформувати необхідний нам датасет ціни акцій. З прикладом такої інформації ми можемо ознайомитися на рисунку 3.4:

```
"Meta Data": {
    "1. Information": "Daily Prices (open, high, low, close) and Volumes",
    "2. Symbol": "IBM",
    "3. Last Refreshed": "2021-12-06",
    "4. Output Size": "Compact",
    "5. Time Zone": "US/Eastern"
},
"Time Series (Daily)": {
    "2021-12-06": {
        "1. open": "119.4000",
        "2. high": "121.1500",
        "3. low": "119.4000",
        "4. close": "119.9100",
        "5. volume": "4785560"
    },
    "2021-12-03": {
        "1. open": "117.3600",
        "2. high": "119.3600",
        "3. low": "117.3600",
        "4. close": "118.8400",
        "5. volume": "6630139"
    },
    "2021-12-02": {
        "1. open": "117.3700",
        "2. high": "117.9800",
        "3. low": "116.5600",
        "4. close": "116.9000",
        "5. volume": "5267149"
    },
```


Рисунок 3.4 – Інформація про компанію

Що стосується самої моделі, то ми використали за основу Feed Forward Neural Network. На вхід цієї моделі подаються ціни на акції компанії, а також кількість позитивних та негативних коментарів і згадувань про компанію за певний період. Отже, на вхід ми маємо ціни акцій, а також кількість коментарів, а на вихід прогнозовану ціну акцій на певний період в майбутньому. З архітектурою нашої моделі по прогнозування ціни на акції, ми можемо ознайомитися на рисунку 3.5:

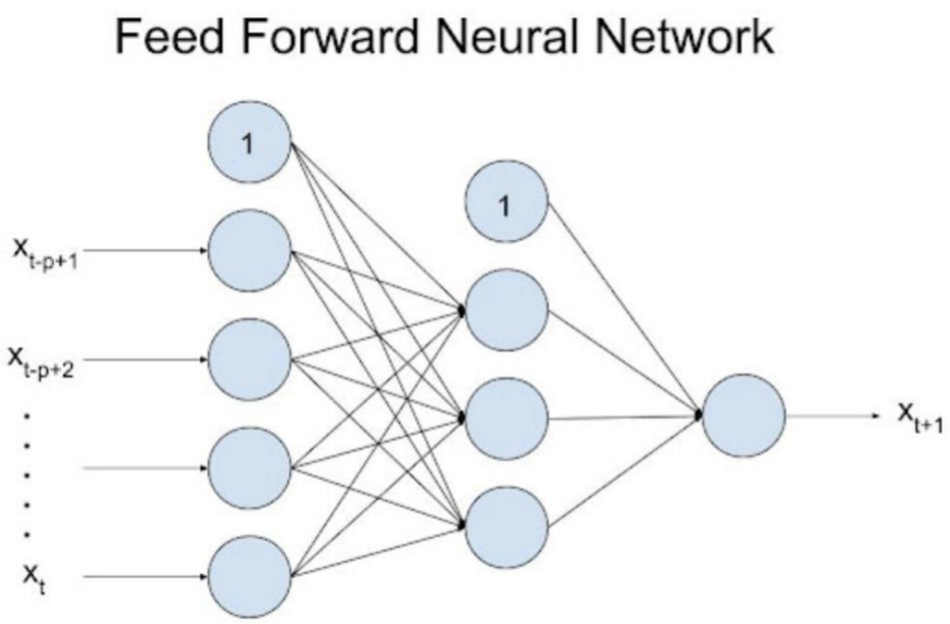

Рисунок 3.5 – Архітектура нейромережі

Що ж, ми натренували та згенерували необхідні нам моделі для прогнозування вартості акцій компанії, а також інтегрували їх у застосунок. Що стосується аналізу тональності тексту, то я скористався готовими моделями, базуються методах на методах машинного навчання з учителем. Дані для формування датасету та навчання цих моделей я взяв з Internet Movie Database, інший з даних фінансового сектора, тобто більш професійного напрямку.

Наступне питання, звідку брати інформацію про компанію, щоб аналізувати її тональность та що ми будемо мати на виході. Інформацію ми беремо із двох джерел, перше – це соціальна мережа Twitter, її ще називають мережею мікроблогів. Ми шукаємо там по ключовим тегам компанії, такі як нікнейм, назва акцій.

Друге джерело новин щодо компанії, це сервіс новин NewsAPI. Це агрегатор новин, який можна легко використовувати як веб сервіс для пошуку актуальних новин. Він дозволяє отримати ключову інформацію, а також має

посилання на джерело. Також він дозволяє отримати інформацію в зручному для нас форматі JSON. Приклад новини із сервісу NewsAPI ви можете бачити на рисунку 3.6:

Top business headlines in the US right now

GET https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=API_KEY


```
{
    status: "ok",
    totalResults: 70,
  - articles: [
      - {
          - source: {
                id: null,
                name: "Investing.com"
            },
            author: "Reuters",
            title: "Asia lags as US, Euro stock futures rally By Reuters - Investing.com",
            description: "Asia lags as U.S., Euro stock futures rally",
            url: https://www.investing.com/news/economy/asia-in-cautious-mood-as-omicron-spreads-us-cpi-looms-2700510,
            urlToImage: https://i-invdn-com.investing.com/news/STOCK-EXCHANGE-RUSSIAN-TRADING-SYSTEMS_800x533_L_1414427815.jpg,
            publishedAt: "2021-12-06T02:31:00Z",
            content: "By Wayne Cole\r\nSYDNEY (Reuters) - Asian share markets lagged a bounce in U.S. and European futures on Monday, while bonds surrendered some of their recent gains and oil rallied as Saudi Arabia lifted… [+3689 chars]"
        },
      - {
          - source: {
                id: "reuters",
                name: "Reuters"
            },
            author: null,
            title: "Alibaba overhauls e-commerce businesses, appoints new CFO - Reuters",
            description: "Alibaba Group Holding Ltd <a href=\"https://www.reuters.com/companies/9988.HK\" target=\"_blank\">(9988.HK)</a> said on Monday it was reorganising its international and domestic e-
```


Рисунок 3.6 – Приклад новини із сервісу NewsAPI

Власне промарковані новини ми використовуємо для прогнозування успішності компанії. Маркування відбувається завдяки моделям аналізу тональності тексту. На вхід вони отримують новину, на вихід маркують, чи вона була позитивною, чи негативною. Далі ми формуємо датасет з кількість позитивної та негативної інформації щодо компанії, та об’єднуємо його з

датасетом цін акцій компанії.

В нашому проекті можна провести такі межі. Частина коду відповідає за користувацький інтерфейс, а також взаємодію з користувачем. Також є класи, котрі відповідають за те, щоб отримати останні новини з різних джерел: будь то Twitter, чи NesAPI. Також є класи котрі займаються аналізом тональності цих новин та маркують їх як позитивні, чи негативні. Далі є бізнес логіка, котра відповідає за те, щоб оцінити успішність компанії.

Виглядає це все як велика система, проте для користувача все набагато простіше. Йому не потрібно знати деталі реалізації, йому потрібно тільки отримати прогноз для компанію. Ознайомимося більш детально з посібником користувача в наступному розділі.

3.4 Посібник користувача

В попередньому пункті було описано, як реалізуються функції прогнозування успішності компанії програмно-архітектурному рівні. В даному пункті буде наведено посібник користувача – опис програми з точки зору використання її користувачем.

Отже, після першого запуску програми, користувач бачить базовий онбординг в продукті, де він може ознайомитися що це за продукт та навіщо він йому потрібен. Приклад онбордингу ми можемо побачити на рисунку 3.7:

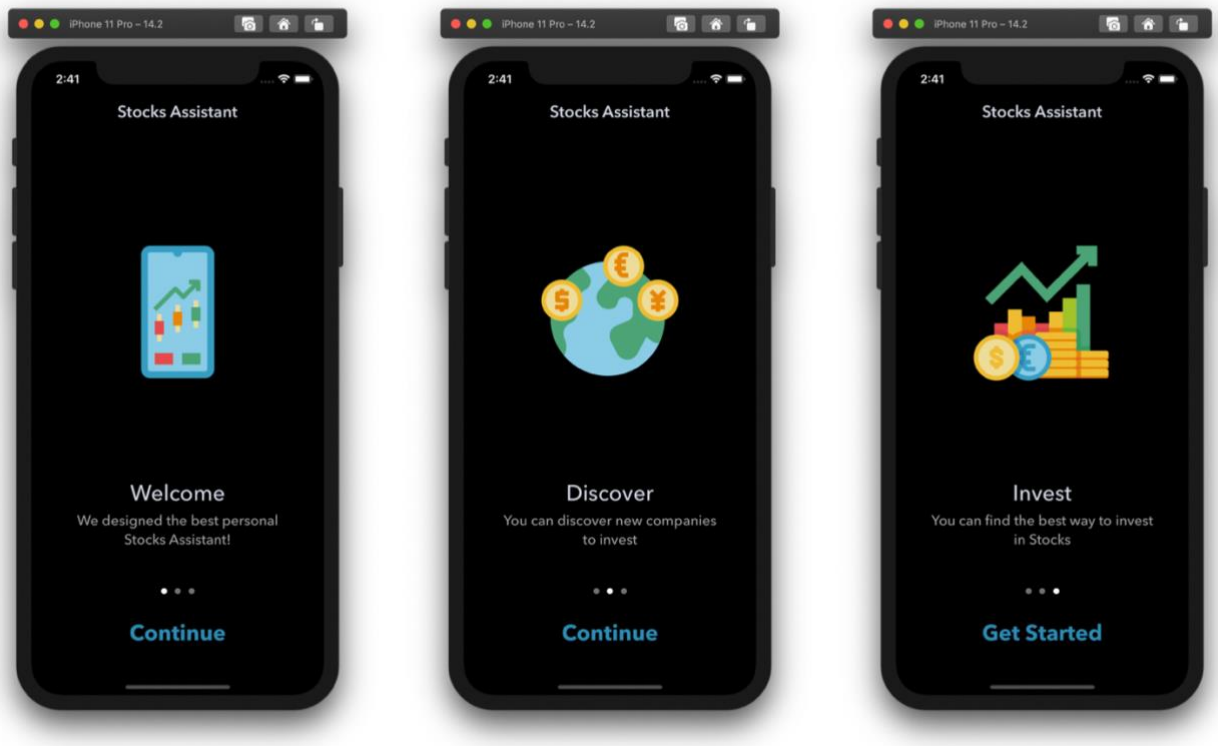

Рисунок 3.7 – Приклад онбордингу в продукті

Далі з’являється екран зі списком компаній. Ми бачимо, що кожна компанія розташована в своїй секції, в залежності від роду її діяльності. Користувач також може скористатися пошуком задля того, щоб знайти необхідну йому компанію. На цьому екрані зображено не тільки назву компанії, а й назву її цінних паперів, та актуальну їх ціну. Ви можете ознакомитися зі списком компаній на рисунку 3.8:

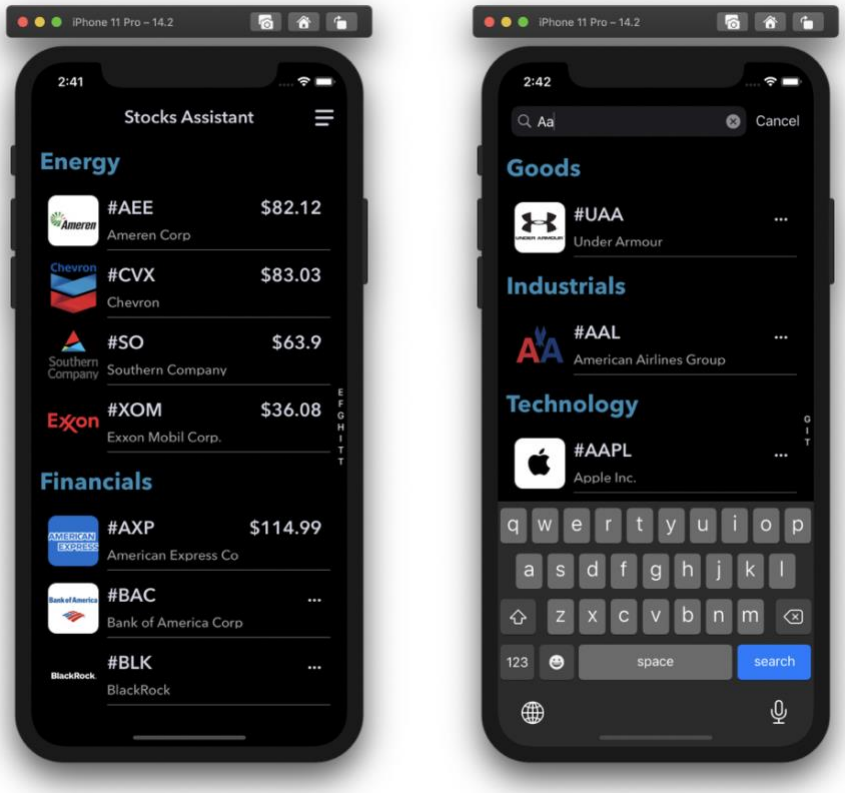

Рисунок 3.8 – Список відкритих компаній

Після того, як користувач вибере компанію, він зможе ознайомитися з загальною інформацією про компанію. Також ви можете переглянути історичну інформацію, як змінювалася ціна на акції даної компанії. Приклад цих екранів ви можете побачити на рисунку 3.9:

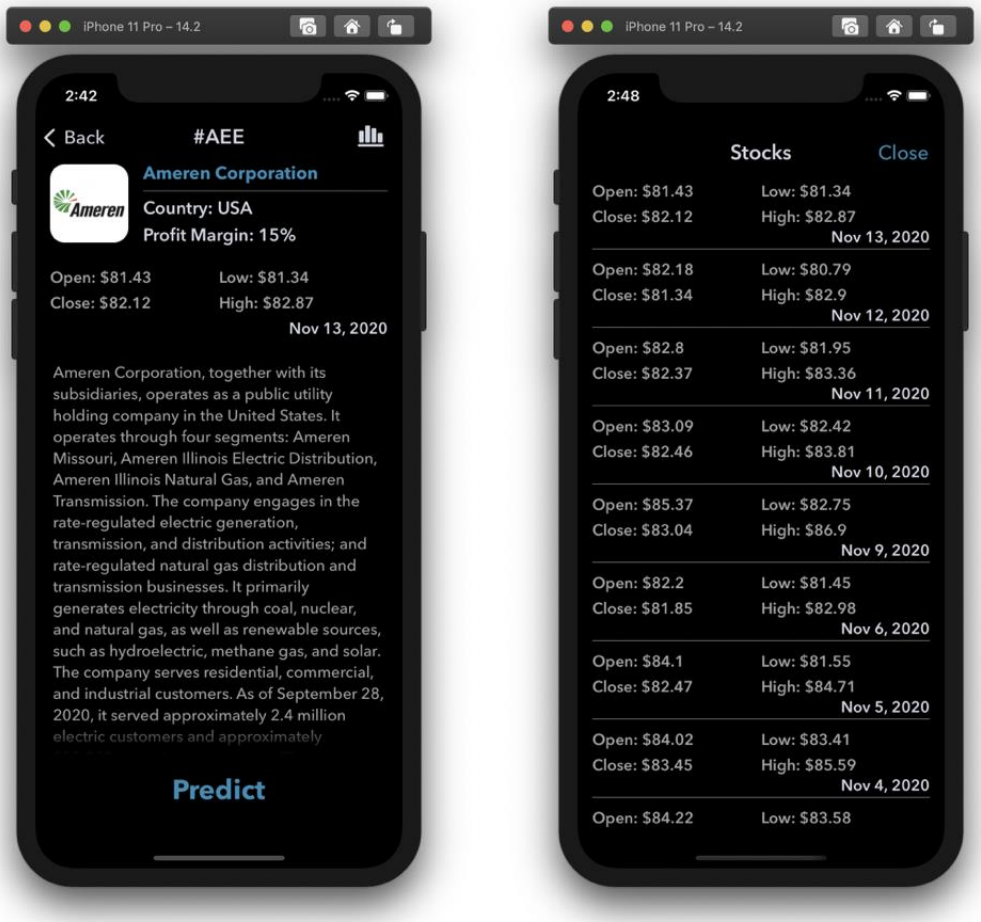

Рисунок 3.9 – Інформація про компанію

Далі користувач може натиснути на кнопку Predict та отримати прогноз успішності компанії. Він бачить значення, яке відображає чи варто інвестувати в обрану компанію чи ні. Ознайомимося з цим екраном на рисунку 3.10:

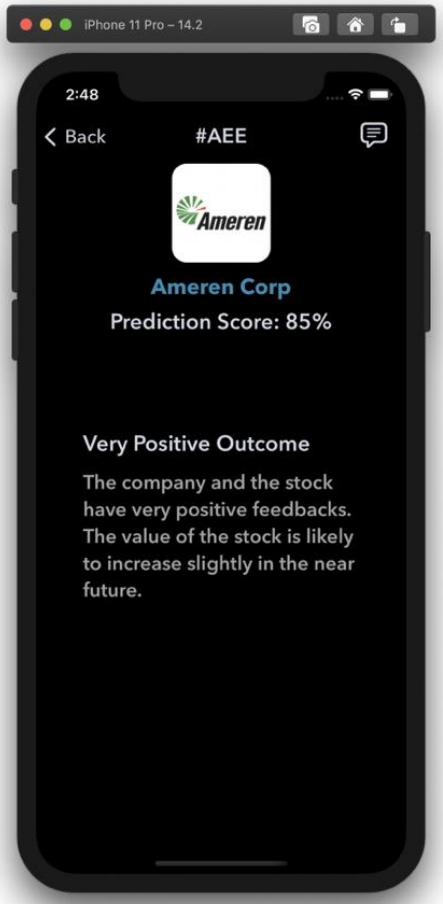

Рисунок 3.10 – Оцінка успішності компанії

Якщо користувач сумнівається в оцінці, то він має змогу власноруч переглянути актуальні новини щодо компанії, а також оцінити наскільки точно модель зуміла проаналізувати тональність тексту. Також якщо він вибере новину зі списку, то зуміє перейти до першоджерела та детально ознайомитися з нею. Ви можете більш детально ознайомитися з цими екранами на рисунку 3.11:

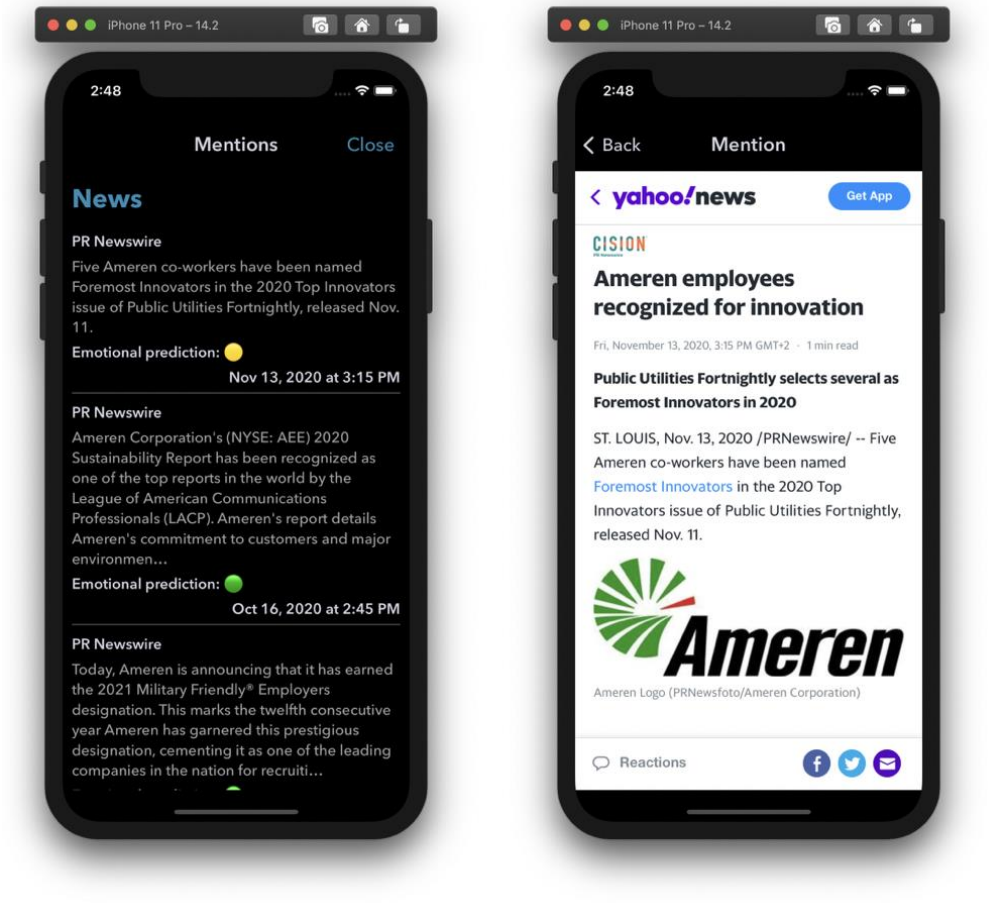

Рисунок 3.11 – Актуальні новини щодо компанії

Отже, ми детально описали як користувач будет взаємодіяти з нашим додатком. Виглядає дійсно все лаконічно та стримано, проте пам’ятаємо скільки всього необхідно було зробити, щоб користувач тільки отримав магічну кнопку Predict. Далі ми розглянемо та проаналізуємо як працює магія прогнозування, а також порівняємо нашу модель з існуючими.

3.5 Аналіз результатів роботи

Що ж, і найцікавіше, перейдемо до результатів роботи нашої моделі. Як працює програма ми ознайомилися в розділах раніше, настав час дійти і до моделі.

Для прикладу, розглянемо як змінювалась ціна на акції компанії Tesla. Ви можете ознайомитися з цим на рисунку 3.12

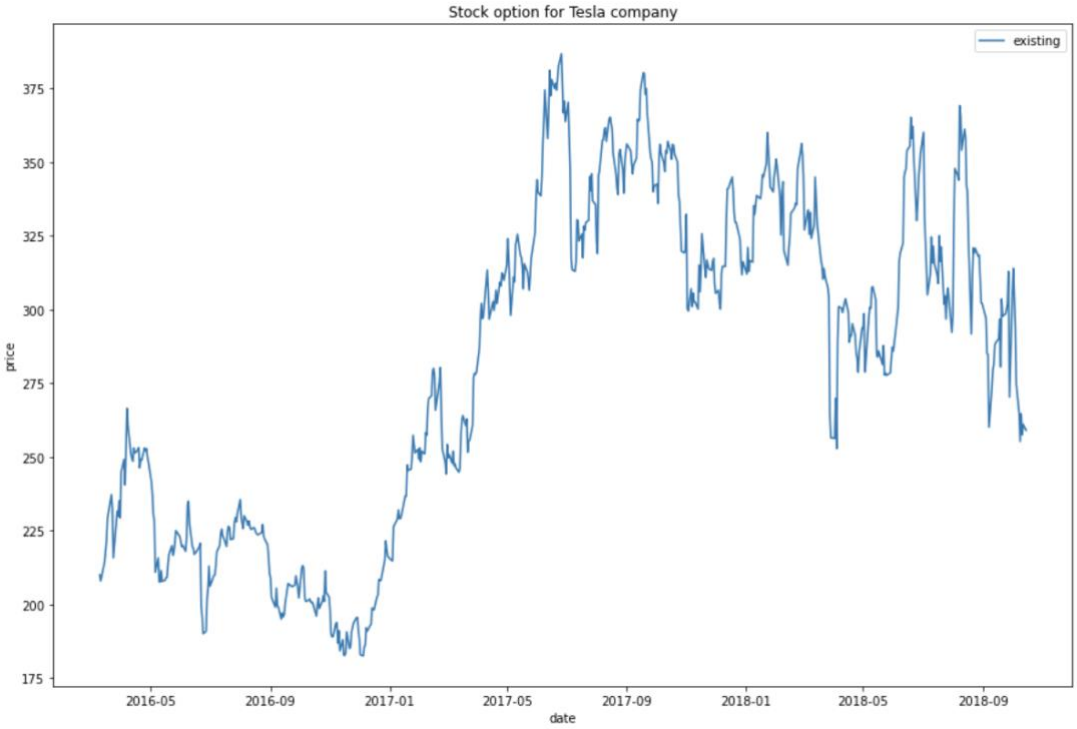

Рисунок 3.12 – Графік зміни ціни на акції компанії Tesla

Далі, сформуємо датасет, в якому окрім цін, ми ще будемо враховувати кількість позитивних та негативних новин щодо компанії. Нагадаю, що новини беремо з двох джерел: соціальна мережа Twitter та сервіс-агрегатор новин NewsAPI. В традиційних методах використовується тільки ціна, щоб робити передбачення, а ми пропонуємо використовувати аналіз тональності тексту, щоб

порахувати кількість позитивних та негативних коментарів було. Ви можете ознайомитися з прикладом даного датасету на рисунку 3.13

date	price	positive_mentions_count	negative_mentions_count
2016-03-10	210.00	2043	909
2016-03-11	207.93	2091	1256
2016-03-14	212.65	2171	1282
2016-03-15	214.27	2306	1284
2016-03-16	218.00	2014	1133
2016-03-17	221.47	1998	1269
2016-03-18	229.10	2314	1123
2016-03-21	235.34	1813	881
2016-03-22	237.21	2718	1486
2016-03-23	232.37	2304	1490
2016-03-24	215.78	2621	927
2016-03-28	231.61	2607	1359
2016-03-29	229.89	2485	1281
2016-03-30	235.09	2121	1352
2016-03-31	229.34	2054	1340

Рисунок 3.13 – Датасет з додатковими параметрами

Також можемо побудувати графік розподілу ціни акцій. Ознайомимося з ним на рисунку 3.14:

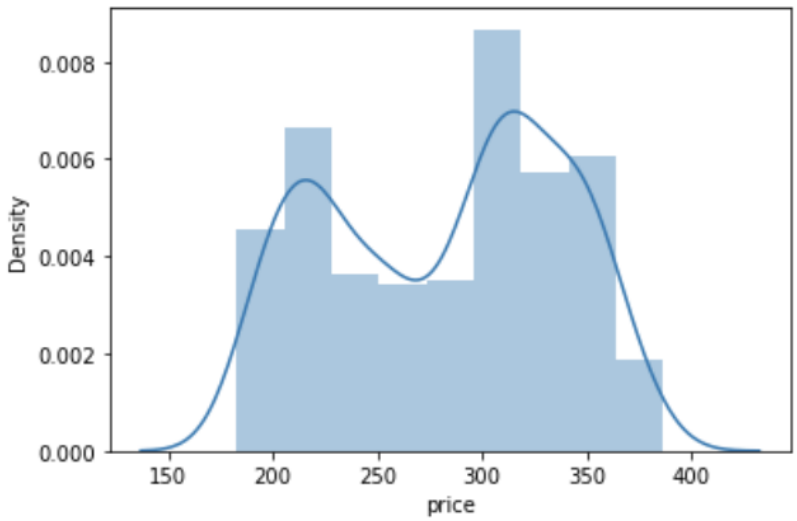

Рисунок 3.14 – Графік розподілу ціни акцій

Далі побудуємо графік залежності кількості позитивних коментарів від ціни акції. Ми можемо його бачити на рисунку 3.15. Проаналізувавши графік, можемо зробити висновок, що не має очевидної лінійної залежності, проте тренд є позитивним. Побудуємо аналогічно графік залежності кількості негативних коментарів від ціни акції, з яким ми можемо ознайомитися на рисунку 3.16

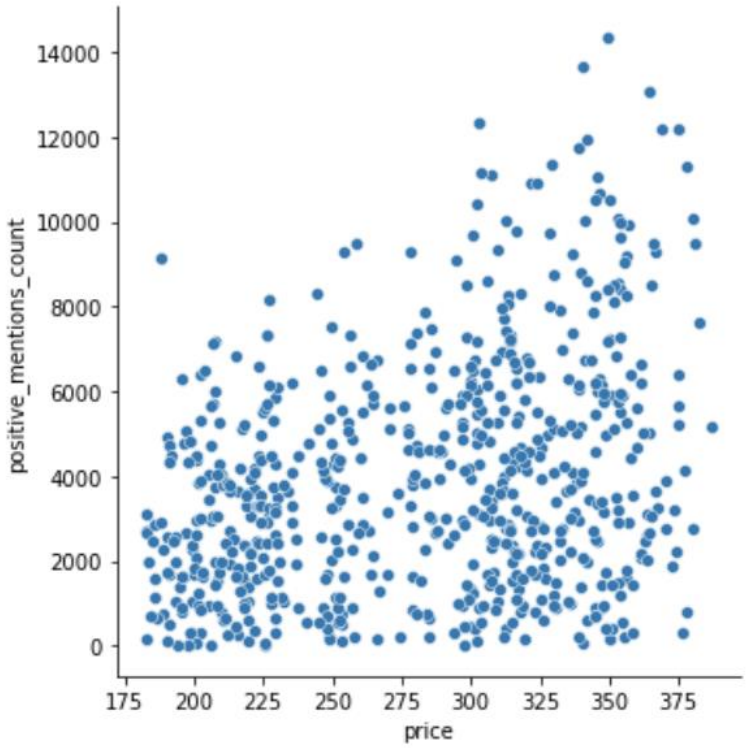

Рисунок 3.15 – Залежність кількості позитивних коментарів від ціни акції

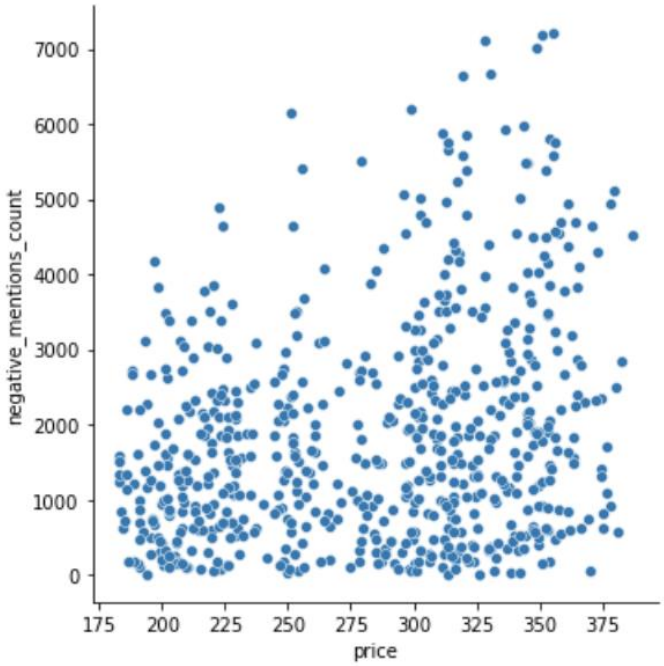

Рисунок 3.16 – Залежність кількості негативних коментарів від ціни акції

Що ж, в нашому експерименті було вирішено розбити часовий ряд на тренувальний та валідаційний. Ознайомимося з ними на рисунку 3.17

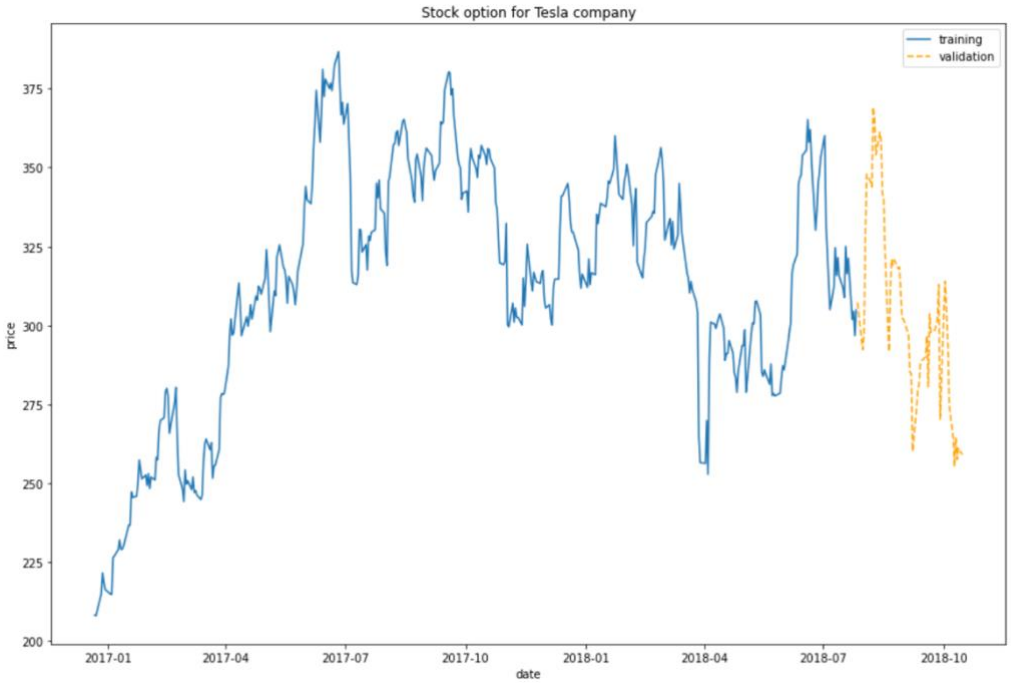

Рисунок 3.17 – Тренувальний та валідаційний часовий ряд

Використаємо класичний статистичний метод авторегресії для прогнозування вартості акцій. Ви можемо порівняти з валідаційним часовим рядом на рисунку 3.18

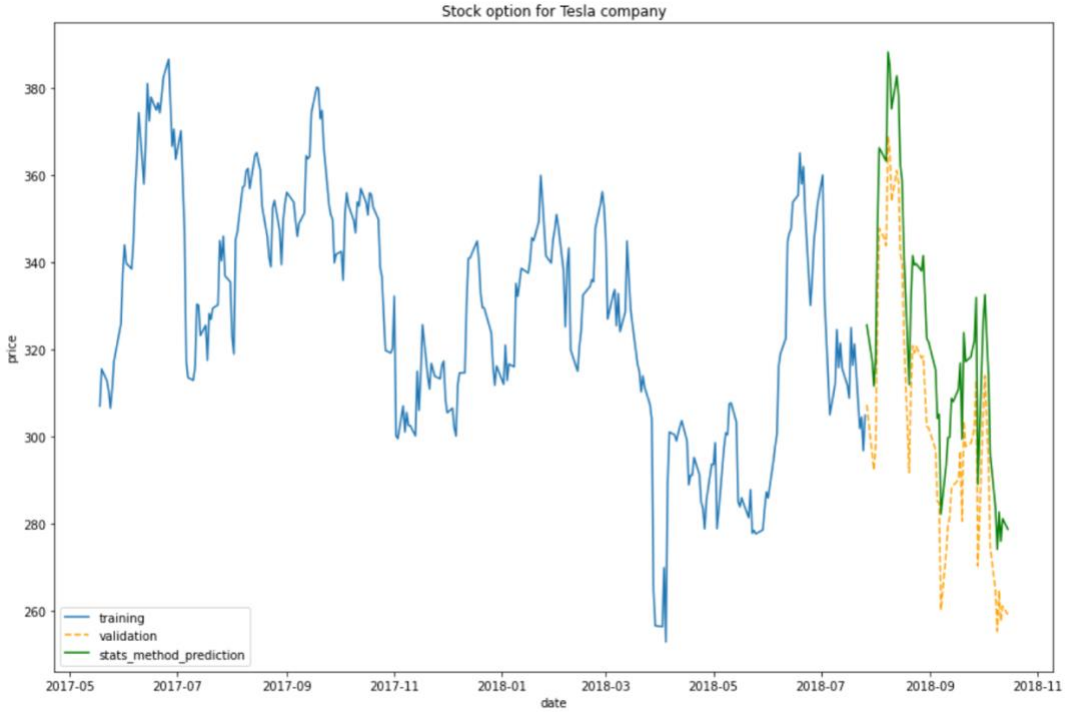

Рисунок 3.18 – Прогнозування вартості акцій статистичним методом

Далі використаємо методи машинного навчання для прогнозування вартості акцій за допомогою нейронних мереж. Результати ми можемо спостерігати на рисунку 3.19

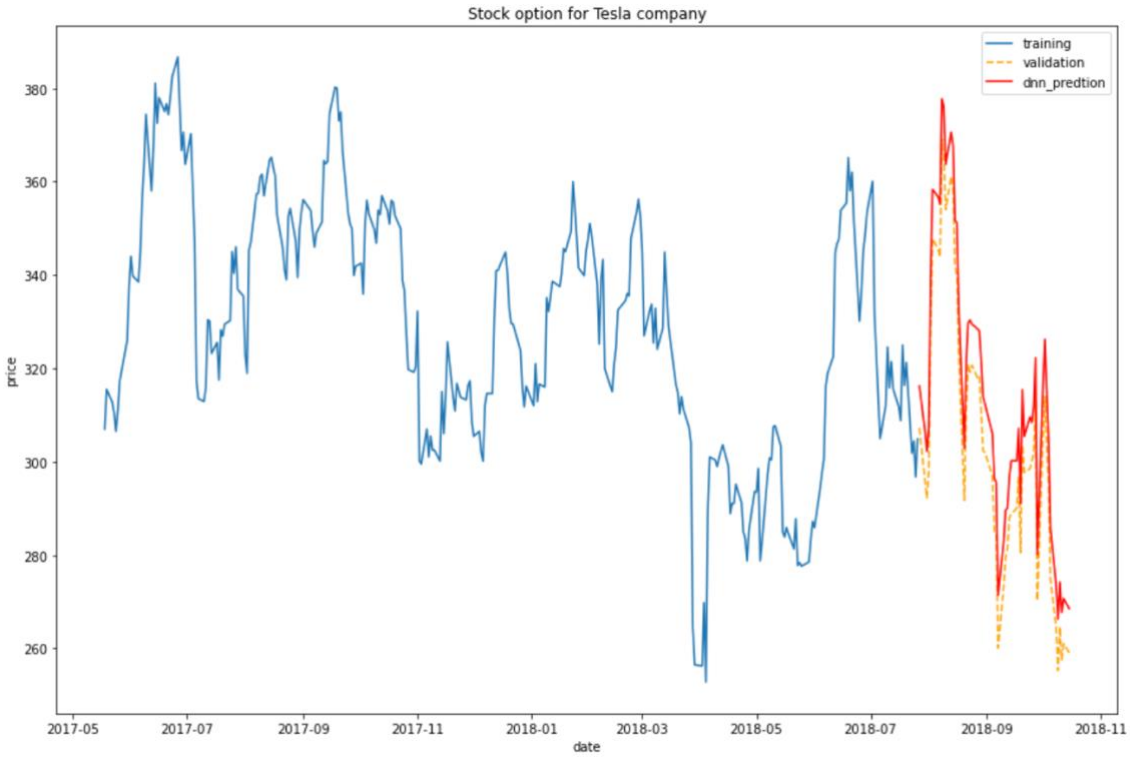

Рисунок 3.19 – Прогнозування вартості акцій за допомогою нейромереж

Ми бачимо, що результат став кращим. Похибка стала меншою. А тепер скористаємося нашим вдосконаленим методом прогнозуванням вартості акцій за допомогою нейронних мереж та семантичного аналізу новин щодо компанії. Результати проілюстровано на рисунку 3.20

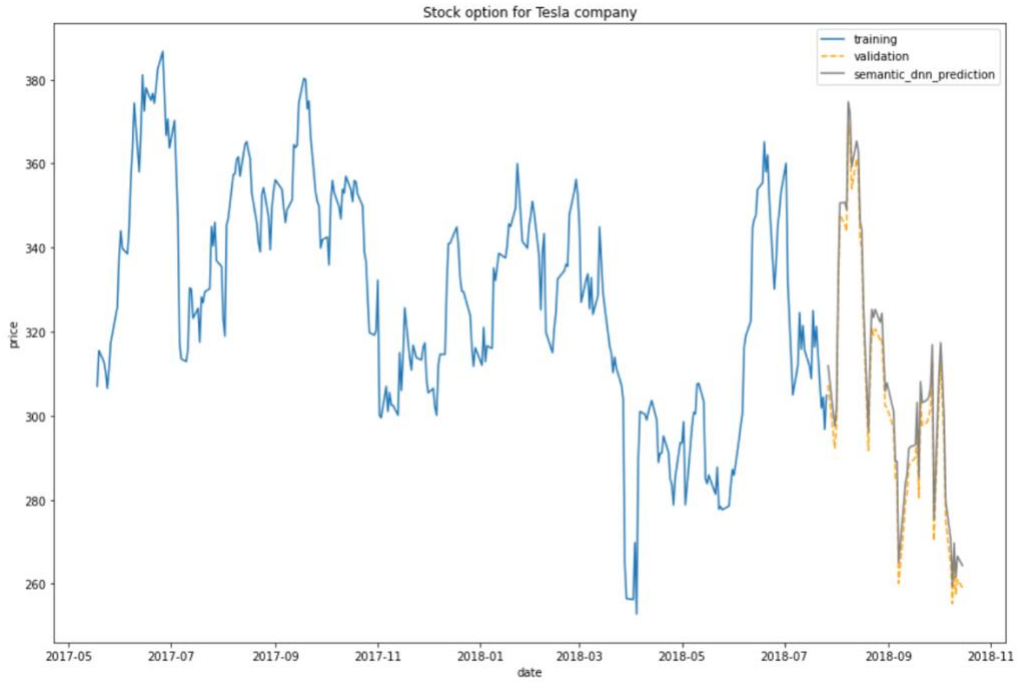

Рисунок 3.20 – Порівняння методів прогнозування вартості акцій

Проаналізувавши графіки, ми бачимо, що прогнозування вартості акцій компанії за допомогою нейромереж та семантичного аналізу новин показало себе найточніше, а тому даний метод працює краще за інші. Побудуємо розподіл по MAE, щоб проілюструвати це. Графік розподілу по MAE для трьох методів зображено на рисунку 3.21

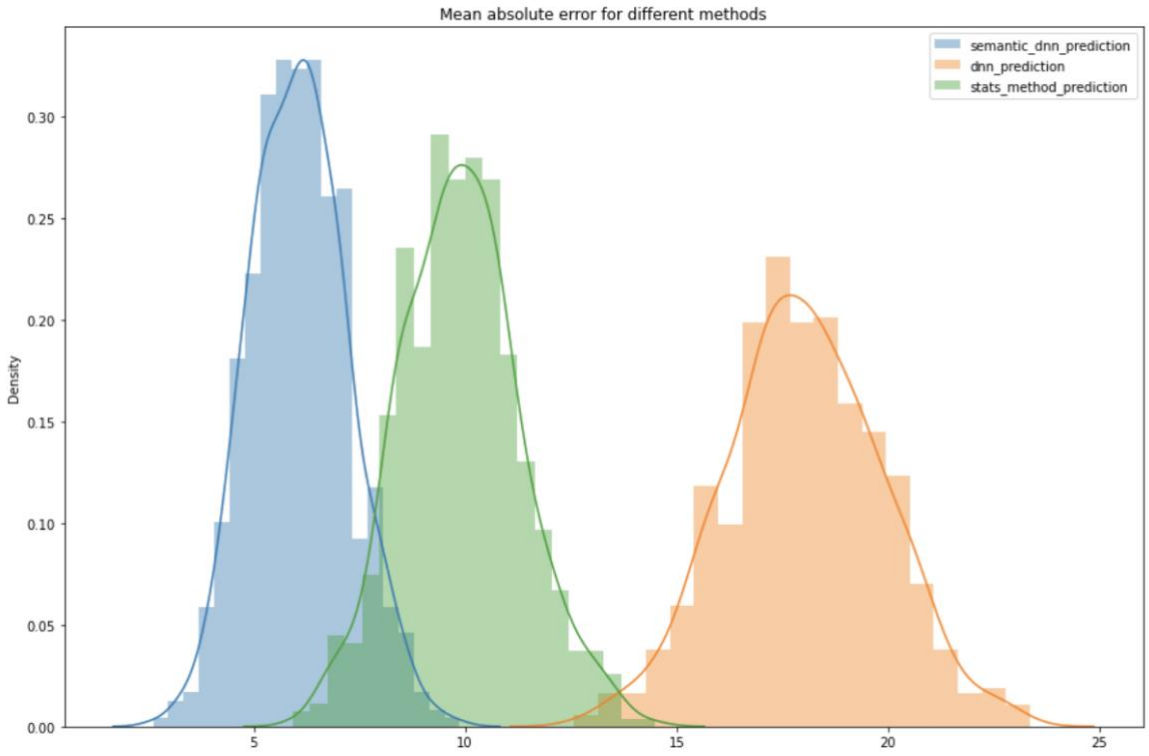

Рисунок 3.21 – Графік розподілу трьох методів по MAE

Отже, ми провели даний експеримент та наочно показали, що використання методів машинного навчання для прогнозування вартості акцій за допомогою нейронних мереж плюс семантичного аналізу новин щодо компанії працює краще за інші методи, а тому він точніше буде прогнозувати успішність компанії.

Висновки до розділу 3

В третьому розділі роботи ми розглянули питання побудови архітектури системи та створення моделі для прогнозування успішності компанії на основі інформації про неї з використанням аналізу тональності тексту.

Що ж, ми розглянули та вирішили наступні питання:

- вибір платформи та мови програмування для реалізації завдання;
- визначення необхідних користувачеві функцій;
- розробка архітектури програми для реалізації визначених функцій;
- розробка керівництва користувача для полегшення використання системи;
- аналіз результатів роботи моделі для перевірки відповідності результатів до теоретичних матеріалів.

Даний розділ є логічним завершенням початої у попередніх розділах теоретичної інформації щодо оцінки успішності компанії. Він ілюструє їх практичну реалізацію у вигляді повноцінної програми.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

У даному розділі я хочу розглянути ключові особливості розробленого мобільного застосунку як стартап-проекту. Проект розглядатиметься як мобільний застосунок для прогнозування успішності компанії.

4.1 Опис ідеї проекту

Основна ідея стартап проекту полягає у виведенні на ринок для комерційного використання розробленого мобільного застосунку для прогнозування успішності компанії. Що ж, проаналізуємо зміст ідеї стартап-проекту, можливі напрямки застосування, а також ключові цінності для користувача застосунку. Ця інформація зібрана в таблиці 4.1.

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Мобільний застосунок для прогнозування успішності компанії.	Прогнозувати успішність компанії для найближчого майбутнього.	Можливість швидко отримати оцінку успішності обраної компанії для найближчого майбутнього.

Далі проведемо аналіз потенційних техніко-економічних переваг ідеї у порівнянни з конкурентами. Результати аналізу зображено в таблиці 4.2.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характери-тики ідеї	Товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конку-рент 1	Конку-рент 2			
1.	Ціна	200$/ рік	1000$/ рік	6000$/ рік			+
2.	Прибутки	30000$/ рік	20000$ / рік	50000$ / рік		+	
3.	Контроль якості	Аналітики та прог-рамісти	Прог-рамісти	Аналі-тики, прог-рамісти та деякі клієнти			+
5.	Постійні витрати	100$/ рік	5000$/ рік	10000$ / рік			+
6.	Змінні витрати	800$ - 1000$/ рік	2000$ - 3000$/ рік	4000$ - 6000$/ рік			+
7.	Патенти на продукти	Відсутні	Патент на кож-ний проект	Деکі-лька патен-тів на проект	+		
8.	Гнучкі ціни	Ціна єдина	Ціна єдина	Ціна варію-ється з року в рік		+	
9.	Законо-давчі обмеження	Немає	Немає	Немає			+

4.2 Технологічний аудит ідеї проекту

Далі, визначимо технологічну здійсненність ідеї проекту за допомогою аналізу таких складових, як технології, за якою буде виготовлено товар згідно ідеї проекту, існування таких технологій, чи їх необхідно розробити / доробити, доступність таких технологій авторам проекту. Результати даного аналізу зображено в таблиці 4.3.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Мобільний застосунок для прогнозування успішності компанії.	Технологія проектування та розробки нейронних мереж Scikit-learn.	Так	Технологія доступна, однак, їй функціонал менший за відповідний у альтернативних технологій.
	Технологія проектування та розробки нейронних мереж TensorFlow, без застосування додаткових модулів.	Так	Дані технології доступні, однак, штатного функціоналу все ще не вистачає для вирішення поставлених задач.
	Технологія проектування та розробки нейронних мереж TensorFlow, із застосуванням бібліотеки Keras.	Так	Дані технології доступні.
Обрана технологія реалізації ідеї проекту: технологія проектування та розробки нейронних мереж TensorFlow, із застосуванням бібліотеки Keras.			

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Далі, проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Результати даного аналізу зображено в таблиці 4.4.

Таблиця 4.4 – Характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	Відсутні
2.	Загальний обсяг продаж, грн/ум.од	500 000
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6.	Середня норма рентабельності в галузі (або по ринку), %	Орієнтовно, 40%

Таким чином, за попереднім оцінюванням, ринок є привабливим для входження.

Надалі визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи. Ці дані зображено в таблиці 4.5.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Можливість швидко отримати оцінку успішності вибраної компанії для найближчого майбутнього.	Малий, середній та великий бізнеси	Немає	Точність і повнота пошуку Час пошуку Оптимальне використання ресурсів

Після визначення потенційних груп клієнтів проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту (таблиця 4.6), та факторів, що йому перешкоджають (таблиця 4.7).

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Відсутність попиту	Бізнес може не оцінити переваги продукту, або ж у цілому відмовитися від оцінки успішності	Акцентувати увагу на клієнтах, що вже скористалися продуктом, якщо такі є, навести інфографіку результативності (очікувану), запропонувати знижку потенційному клієнту в рамках тендеру.
2.	Неточне прогнозування	Неточність оцінки успішності компанії	Розробка і випуск додаткових функціональних можливостей для підвищення ефективності оцінки успішності.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Кобрендінг	Пропозиція від певної компанії, що спеціалізується на прогнозуванні успішності компаній, розробити спільний продукт	Виділення частини штату на реалізацію проекту, підготовка акційних пропозицій по переходу на новий продукт існуючим клієнтам.
2.	Загальні тенденції на ринку	Зростає популярність програм для прогнозуванні успішності компаній	Підготовка акційних пропозицій для нових клієнтів, створення реферальної системи

Надалі проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку. Результати даного аналізу зображені в таблиці 4.8.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Чиста конкуренція	Гравці ринку не мають явних переваг один над одним	Більш вигідні умови на тендерах, агресивний маркетинг
2. Регіональна конкуренція	Гравці ринку – інтернаціональні підприємства	Вихід на ті ринки, які ще не зайняті конкурентами
3. Внутрішньогалузева конкуренція	Гравці ринку знаходяться в одній галузі – розробці ПЗ	
4. Товарно-видова конкуренція	Усі продукти гравців ринку мають одне призначення	Розробка простого інтерфейсу, швидкість роботи застосунку та точність оцінки успішності компанії

Продовження таблиці 4.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5. Конкурентні переваги нецінові	Продукти відрізняються гнучкістю, функціоналом (незначно) і надійністю.	У маркетингу неявно порівнювати власний продукт з іншими, робити вигідні цінові пропозиції
6. Марочна конкуренція	Значна увага приділяється бренду, що розробив продукт	Кобрендінг

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі програмних продуктів для проектування двигунів. Результати даного аналізу зображено в таблиці 4.9.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти
	Динаміка галузі, продуктова лінія	Наявність товарних знаків, доступ до ресурсів, патенти на продукти
Висновки:	Конкуренція є, так як ринок набирає обертів, проте великих гравців ще не має	Для входу на ринок необхідно написати бета-версію програмного продукту. На даний момент потенційних конкурентів немає.

Далі визначимо та обґрунтуємо фактори конкурентоспроможності, які зображені в таблиці 4.10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Інновації	Інноваційні рішення мають забезпечити перевагу нашим клієнтам над конкурентами
2.	Функціонал	Функціонал повинен покривати вирішення необхідних задач клієнтів
3.	Цінова політика	Вартість продукту відіграє велику роль при виборі системи клієнтом
4.	Ресурсоємність	Великі затрати технічних ресурсів можуть спровокувати необхідність залучення додаткових коштів

За визначеними факторами конкурентоспроможності проведемо аналіз сильних та слабких сторін стартап-проекту. Результати даного аналізу зображено в таблиці 4.11.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін додатку

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим застосунком						
			–3	–2	–1	0	+1	+2	+3
1	Інновації	18			+				
2	Функціонал	12	+						
3	Цінова політика	16			+				
4	Ресурсоємність	15			+				

Наступним проведемо SWOT-аналіз на основі сильних і слабких сторін проекту, а також виділених загроз і можливостей. SWOT-матриця зображено в таблиці 4.12.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: розумна цінова політика, функціонал забезпечує рішення більшості задач клієнта	Слабкі сторони: відсутність співпраці з інноваційними центрами
Можливості: впровадження інноваційних рішень, оптимізація роботи продукту	Загрози: неточність результатів

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтований оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Дані альтернативи зображено в таблиці 4.13.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Відкриття офісу для прийняття замовлень та продажу запропонованого програмного продукту	Середня	10-12 місяців
2.	Створення власного Internet-ресурсу	Середня	4-6 місяців
3.	Створення власного мобільного застосунку	Висока	6-8 місяці

З визначених альтернатив доцільно обрати альтернативу 3 («Створення власного мобільного застосунку») як таку, що має найбільшу ймовірність отримання ресурсів, і при цьому несуттєво поступається за термінами реалізації альтернативі з найбільш стислими строками.

4.4 Розроблення ринкової стратегії проекту

Задля того, щоб розробити ринкову стратегію нам необхідно описати цільові груп потенційних споживачів, які можна побачити в таблиці 4.14.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Інтенсив-ність конкуренції в сегменті	Простота входу у сегмент
1.	Малий бізнес	Готові	Середня	Середня
2.	Середній бізнес	Середня	Слабка	Середня
3.	Великий бізнес	Середня	Слабка	Середня

Було обрано цільову групу підприємств групи малого бізнеса. Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку, яку ображено в таблиці 4.15.

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспро можні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
	Розробка та створення додаткових функціональних модулів	Таргетні пропозиції	Відсутність аналогічних до новостворених функціональних модулів у конкурентів	Розробка та удосконалення прогнозування успішності компаній.

Наступним кроком є вибір стратегії конкурентної поведінки, яку зображено в таблиці 4.16.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохід-цем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Він не є «першопрохідцем», проте аналогів саме з такою реалізацією ідеї не існує	Можливі обидва варіанти	Стандартні функціональні модулі будуть виконувати схожі функції.	Унікальна цінова політика, функціональні інновації, сучасні технології.

Далі ми розробимо стратегію позиціонування, що полягає у формуванні ринкової позиції, за яким споживачі будуть ідентифікувати торгівельну марку/проект. Її зображено в таблиці 4.17.

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспро-можні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Можливість швидко отримати оцінку успішності вибраної компанії для найближчого майбутнього.	Розробка та удосконалення існуючих модулів на основі потреб ринку та інформації від клієнтів.	Спеціалізовані рішення, мобільний додаток.	Прогнозування, оцінка, аналіз.

4.5 Розроблення маркетингової програми стартап-проекту

Що ж, сформуємо маркетингову концепцію товару, яку отримає споживач. В таблиці 4.18 зображено результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Отримати оцінку успішності компанії для найближчого майбутнього.	Можливість швидко отримати оцінку успішності обраної компанії для найближчого майбутнього.	Прогнозування оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту.

Надалі розробимо трирівневу маркетингову модель товару: уточнимо ідею продукту, його фізичні складові, особливості процесу його надання. Дана модель зображена в таблиці 4.19.

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Програмний продукт - мобільний iOS застосунок для прогнозування оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту.
II. Товар у реальному виконанні	Властивості / характеристики: 1. Можливість вибрати компанію зі списку 2. Можливість вибору дати для прогнозування успішності. 3. Отримання оцінки успішності компанії Якість: програмний продукт пройшов всі етапі тестування та готовий до використання.

Продовження таблиці 4.19

Рівні товару	Сутність та складові
II. Товар у реальному виконанні	Властивості / характеристики: 1. Можливість вибрати компанію зі списку 2. Можливість вибору дати для прогнозування успішності компанії в майбутньому. 3. Отримання оцінки успішності компанії
	Якість: програмний продукт пройшов всі етапі тестування та готовий до використання.
	Пакування: App Store, маркетплейс для iOS застосунків
	Марка: назва організації-розробника «Peteliev Inc.», назва товару «Companies Assistant».
III. Товар із підкріпленням	До продажу: застосунок в App Store, де користувач зможе влосноруч його встановити. Після продажу: Швидкодія, ефективність, точність, легкість у користуванні

Важливо, що код закритий та захищений від можливості декомпіляції..

Наступним ми визначимо цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів. Аналіз проводився експертним методом і його результати зображено в таблиці 4.20.

Таблиця 4.20 – Визначення меж встановлення цін

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
-	200$/міс	Рівень доходів підприємств надзвичайно високий	10-500$/міс

Далі визначимо оптимальну систему збуту, в межах якого приймається рішення. Дану систему зображено в таблиці 4.21.

Таблиця 4.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Таргетні пропозиції для користувачів / органіка з інтернету / SEO	Презентації функціоналу	-	-

Розробимо концепцію маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів. Дану концепцію зображено в таблиці 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомлення	Концепція рекламного звернення
Клієнт намагається зрозуміти оцінку компанії в найближчом у майбутньому .	Мережа Інтернет, соціальні мережі, E-mail.	Унікальність ПЗ, можливість вибору компанії для прогнозування успішості.	Продемонструва -ти унікальність данного ПЗ та показати що та чи інша компанія покаже оцінку близьку до прогнозованої.	Показати можливість за не велику ціну збільшити прибуток свого портфелю акцій за допомогою прогнозуванн я успішності компаній.

Висновки до розділу 4

Отже, в даному розділі було виконано перший етап розроблення стартап-проекту, а саме маркетинговий аналіз.

Фондовий ринок є однією з найважливіших сфер ринкової економіки, оскільки він надає компаніям доступ до капіталу, дозволяючи інвесторам купувати акції в компанії. Купуючи акції, інвестори можуть отримати прибуток від своїх інвестицій, проте не всі вони успішно отримують його. Це відбувається тому, що ціна акцій постійно коливається, і в будь-який момент вона може впасти нижче ціни, за якою вона була куплена. Саме тому передбачення того, як буде поводитись фінансовий ринок, є одним з найважчих та найцікавіших завдань в економіці.

Ми можемо зробити висновок, що ідея стартапу на сьогоднішній день є надзвичайно актуальною, адже всі прагнуть отримувати інвестиції зі своїх акцій, а тому правильно сформувати портфель компаній для інвестування є надзвичайно важливим.

З огляду на потенційну групу клієнтів, а саме, людей, які займаються інвестуванням в компанії, купуючи їх акції на ринку цінних паперів, та ріст даного ринку є великі перспективи впровадження даного програмного забезпечення.

Для ринкової реалізації проекту доцільно обрати таку альтернативу впровадження: створення мобільного застосунку та розповсюдження його серед користувачів. Доцільно почати з аудиторії США, так як вона є для нас тестовою, адже користувачі є надзвичайно активними, а потім додавати країни Європи.

ВИСНОВКИ

Оцінка успішності компанії є надзвичайно популярним та перспективним напрямком в сучасному світі, адже допоможе як моделювати, так і прогнозувати повіденку компаній, а отже і ситуацію на ринку. В даній магістерській роботі вирішувалась задача оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тектсу.

Під час роботи було проаналізовано існуючі методи, які використовуються для оцінки успішності компанії. Ми прийняли до уваги, що під успішністю компанії будемо розуміти її ринкову вартість, а тому нашою задачею було прогнозування вартості акцій компанії. Було проаналізовано основні підходи для даної задачі. Ми розглянули як класичні методи, наприклад авторегресія, так і математичні методи методи машинного навчання, такі як нейронні мережі, для прогнозування вартості акцій.

Окрім того, було розроблено модель оцінки вартості компанії на основі інформації про неї з використанням аналізу тональності тексту. Як результат, було створено програмний продукт, який використовуючи нову модель оцінки вартості компанії, зумів покращив результати класичних методів оцінки.

Нагадаю, що метою даної роботи була розробка моделі для оцінки успішності компанії на основі інформації про неї з використанням аналізу тональності тексту. Ця мета була успішно досягнута, результати роботи моделі ми порівняли з класичними методами та показали, що вона працює краще, а тому оцінка успішності компанії є точнішою.

Об'єктом дослідження даної роботи є методи для прогнозування оцінки успішності компанії. Предметом дослідження є використання аналізу тональності тексту для оцінки успішності компанії.

Ключовим моментом є те, що ми показали, як можливо покращити оцінку

успішності компанії завдяки використання аналізу тональності тексту, а саме аналіз інформації про компанію. Ми дійсно зуміли простежити тенденцію, що чим більше є позитивних новин щодо компанії, тим її акції дорожчають, а тому вартість компанія також зростає. Це дійсно круто, адже похибка для нашої моделі є меншою у порівнянні з класичними методами, а тому наша модель працює точніше. Важливо, що модель з кожним днем стає все розумнішою та розумнішою, а тому працює точніше.

Що стосується практичного результату даної магістерської дисертації, то ми поставили за мету розробити модель для оцінки вартості компанії на основі інформації про неї з використанням аналізу тональності тексту, а також створити програмний продукт, який на основі даних компанії з фондового ринку та інформації про неї з інформаційних джерел, та використанням моделі оцінює її успішність, а також прогнозує майбутнє.

Дану магістерську дисертацію рекомендується покращити взявши до уваги більшу кількість інформаційних джерел, розширити базу факторів, які впливають на успішність компанії, а також додати вагові коефіцієнти для різних джерел новин.

Також в майбутньому планується додати підтримку інших платформ, та зробити можливість додавати власну оцінку компанії, таким чином ви можете оцінити наскільки ваші прогнози будуть збігатися з реальністю.

ПЕРЕЛІК ПОСИЛАНЬ

1. Adhikari R. An Introductory Study on Time Series Modeling and Forecasting. *Journal of Artificial Intelligence.* 2009. Vol. 42, No. 5. P.856–874.
2. Бідюк П.І. Аналіз часових рядів: навчальний посібник. Київ: Політехніка, 2010. 317 с.
3. Nwankpa C., Ijomah W. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *Journal of Artificial Intelligence*. 2018. Vol. 22, No. 3. P.2–20.
4. Yoon Y., G. Swales Applying Artificial Neural Networks to Investment Analysis. London: Taylor & Francis, 2011. 80 p.
5. Володин С. Н. Прогнозирование динамики курсовых стоимостей акций фондового рынка с применением резонансных систем искусственного интеллекта: дис. ... канд. техн. наук: 01.05.03 / Высшая школа экономики. Москва, 2017. 113 с.
6. Jaybhay K. Stock Market Prediction Model by Combining Numeric and News Textual Mining. *International Journal of Computer Applications*. 2012. Vol. 6, No. 19. P. 18–20.
7. Di Persio L. Artificial Neural Networks architectures for stock price prediction: comparisons and applications. *INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING*. 2016. Vol. 31, No. 10. P. 404–405.
8. Robert J. Van Eyden The Application of Neural Networks in the Forecasting of Share Prices. New York: Finance and Technology Publishing, 1996. 326 p.
9. Lawrence R. Using Neural Networks to Forecast Stock Market Prices. New York; London: Oxford University Press, 1997. 326 p.
10. Гавва В.Н. Потенціал підприємства: формування та оцінювання: Навч. посіб. Київ: Центр навчальної літератури, 2004. 224 с.

11. Єгерєв И.А. Стоимость бизнеса: Искусство управления: Учебное пособие. Москва: Дело, 2003. 480 с.
12. Paulo-Santos António, Ramos Carlos, Marques Nuno C. Determining the Polarity of Words through a Common Online Dictionary. *Proceedings by 15th Portuguese Conference on Artificial Intelligence*, EPIA. Lisbon, Portugal, 2011. P. 649-663.
13. Романишин М., Романюк А. Тональний словник української мови на основі сентимент-анотованого корпусу. *Українське мовознавство*. 2013. Вип. 43. С. 63-74.
14. Thelwall M., Buckley K., Paltoglou G. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*. 2010. Vol. 17, No. 61. P.2544–2558.
15. He Yulan, Zhou Deyu. Self-training from labeled features for sentiment analysis. ScienceDirect. 2014. URL: https://www.sciencedirect.com/science/article/abs/pii/S0306457310000932 (Last accessed: 21.09.2021).
16. Rita McCue, Jonathan Schler. The importance of neutral examples for learning sentiment. 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.9735 (Last accessed: 26.09.2021).
17. Too much information. The Economist. URL: https://www.economist.com/node/18895468 (Last accessed: 23.10.2021).
18. Koppel, Moshe. The Importance of Neutral Examples for Learning Sentiment. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.9735&rep=rep1&type =pdf (Last accessed: 16.10.2021).
19. Nozomi Kobayashi, Ryu Iida, Kentaro Inui, Yuji Matsumoto. Opinion Mining on the Web by Extracting Subject-Aspect-Evaluation Relations. 2006. URL:

https://www.aaai.org/Papers/Symposia/Spring/2006/SS-06-03/SS06-03-018.pdf (Last accessed: 06.10.2021).

20. Дописувачі Вікіпедії. iOS. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/IOS (Last accessed: 09.11.2021).
21. Дописувачі Вікіпедії. Swift. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Swift_(мова_програмування) (Last accessed: 09.11.2021).
22. Дописувачі Вікіпедії. Python. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Python (Last accessed: 03.11.2021).
23. Дописувачі Вікіпедії. Pandas. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Pandas (Last accessed: 03.11.2021).
24. Дописувачі Вікіпедії. Scikit-learn. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Scikit-learn (Last accessed: 04.11.2021).
25. Дописувачі Вікіпедії. Git. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Git (Last accessed: 15.11.2021).
26. Дописувачі Вікіпедії. Тональність тексту. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Аналіз_тональності_тексту (Last accessed: 15.11.2021).
27. Дописувачі Вікіпедії. Нейронна мережа прямого поширення. Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/Нейронна_мережа_прямого_поширення (Last accessed: 24.10.2021).

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
struct SecretKeys {
    // Twitter
    //https://twitter.com
    static let twitterKey = "1v7XifCxjsBYVUdxztAEjWihY"
    static let twitterSecretKey = "6DPbtAeaUSVUepCqTpCQg1y4fCesUofH16QhciIp14SaN21Kvc"

    // Alpha Vantage
    //https://www.alphavantage.co
    static let alphaVantageKey1 = "959O90SZGQT7U1ZG"
    static let alphaVantageKey2 = "2LOBLE5OEEC05OX7"

    // News API
    //https://newsapi.org
    static let newsAPIKey = "205fce48b35445b2ba90adc39f4added"

    // App Store
    static let appStoreAppID = "1111"
}


@main
class AppDelegate: UIResponder, UIApplicationDelegate, PreferencesInjectable {
    var window: UIWindow?
    var coordinator: Coordinator?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        preferences.launchCount += 1
        setupFirebase()
        setupCocoaLumberjack()
        setupAppCoordinator()
        return true
```

```
    }
}

//MARK: - Private Methods
private extension AppDelegate {

    func setupAppCoordinator() {
        let appWindow = UIWindow()
        window = appWindow
        let appCoordinator = AppCoordinator(window: appWindow)
        coordinator = appCoordinator
        appCoordinator.start()
    }

    func setupFirebase() {
        FirebaseApp.configure()
    }

    func setupCocoaLumberjack() {
        DDLog.add(DDOSLogger.sharedInstance)

        guard let cacheDirectory = NSSearchPathForDirectoriesInDomains(.cachesDirectory,
.userDomainMask, true).first else { return }
        let logsDirectory = URL(fileURLWithPath: cacheDirectory).appendingPathComponent("Logs").path
        let fileManager = DDLogFileManagerDefault(logsDirectory: logsDirectory)
        fileManager.maximumNumberOfLogFiles = 5
        let fileLogger = DDFileLogger(logFileManager: fileManager)
        fileLogger.maximumFileSize = 1 * 1024 * 1024 // 1MB
        fileLogger.doNotReuseLogFiles = true
        DDLog.add(fileLogger)
    }
}


struct TwitterMentionNetworkModel: CompanyMentionProtocol {
```

```
    let text: String
    let author: String?
    var sourceURL: URL?
    let creationDate: Date?
    var emotionalPrediction: CompanyMentionEmotionalPrediction?
}


struct NewsAPIMentionNetworkModel: CompanyMentionProtocol, Codable {
    enum CodingKeys: String, CodingKey {
        case text = "description"
        case sourceURL = "url"
        case creationDate = "publishedAt"
        case author
    }

    let text: String
    let author: String?
    let sourceURL: URL?
    let creationDate: Date?
    var emotionalPrediction: CompanyMentionEmotionalPrediction?
}


struct NewsAPIMentionsContainerNetworkModel: Codable {
    enum CodingKeys: String, CodingKey {
        case mentions = "articles"
    }
    var mentions: [NewsAPIMentionNetworkModel]
}


final class TwitterCompanyMentionsFetcher: CompanyMentionsFetcherProtocol {

    //MARK: - CompanyMentionsFetcherProtocol
```

```
    func fetchMentions(with searchSource: String, count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void) {
        var companyMentions = [TwitterMentionNetworkModel]()
        swifter.searchTweet(using: searchSource, lang: "en", count: count, tweetMode: .extended, success: {
[weak self] (results, _) in
            guard let self = self else { return }

            for i in 0...count {
                let object = results[i]
                guard
                    let text = object["full_text"].string,
                    let tweetID = object["id_str"].string,
                    let author = object["user"]["name"].string,
                    let accountName = object["user"]["screen_name"].string
                else { continue }

                let sourceStringURL = "https://twitter.com/\(accountName)/status/\(tweetID)"
                let sourceURL = URL(string: sourceStringURL)
                let creationDateString = object["created_at"].string ?? ""
                let creationDate = self.dateFormatter.date(from: creationDateString)
                let companyMention = TwitterMentionNetworkModel(text: text, author: author, sourceURL:
sourceURL, creationDate: creationDate)
                companyMentions.append(companyMention)
            }
            completion(companyMentions)
        }, failure: { error in
            print("Error with fetching data from Twitter API: \(error)")
        })
    }

    func fetchMentions(with searchSources: [String], count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void) {
        var companyMentions = [CompanyMentionProtocol]()
        let dispatchGroup = DispatchGroup()
        for searchSource in searchSources {
```

```
            dispatchGroup.enter()
            fetchMentions(with: searchSource, count: count) { mentions in
                defer { dispatchGroup.leave() }
                companyMentions.append(contentsOf: mentions)
            }
        }
        dispatchGroup.notify(queue: .main) {
            completion(companyMentions)
        }
    }

    //MARK: - Private Properties
    private let swifter = Swifter(consumerKey: SecretKeys.twitterKey, consumerSecret:
SecretKeys.twitterSecretKey)
    private let dateFormatter: DateFormatter = {
        let dateFormatter = DateFormatter()
        dateFormatter.locale = Locale(identifier: "en_US_POSIX")
        dateFormatter.dateFormat = "E MMM dd HH:mm:ss Z yyyy"
        return dateFormatter
    }()
}


final class NewsAPICompanyMentionsFetcher: CompanyMentionsFetcherProtocol {

    //MARK: - CompanyMentionsFetcherProtocol
    func fetchMentions(with searchSource: String, count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void) {
        dataTask?.cancel()

        var urlComponents = self.baseURLComponents
        urlComponents.path = "/v2/everything"
        urlComponents.setQueryItems(with: ["q": searchSource, "apiKey": SecretKeys.newsAPIKey])
        guard let requestUrl = urlComponents.url else { return }
        dataTask = defaultSession.dataTask(with: requestUrl) { [weak self] (data, _, error) in
```

```
            guard let self = self else { return }
            defer { self.dataTask = nil }
            guard let data = data, error == nil else { return }
            let decoder = JSONDecoder()
            decoder.dateDecodingStrategy = .iso8601

            do {
                let companyMentionsContainer = try
decoder.decode(NewsAPIMentionsContainerNetworkModel.self, from: data)
                let mentions = companyMentionsContainer.mentions
                DispatchQueue.main.async {
                    completion(mentions)
                }
            } catch {
                print("Error with decoding data from NewsAPI: \(error.localizedDescription)")
            }
        }
        dataTask?.resume()
    }

    func fetchMentions(with searchSources: [String], count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void) {
        var companiesMentions = [CompanyMentionProtocol]()
        let dispatchGroup = DispatchGroup()
        for searchSource in searchSources {
            dispatchGroup.enter()
            fetchMentions(with: searchSource, count: count) { companyMentions in
                defer { dispatchGroup.leave() }
                companiesMentions.append(contentsOf: companyMentions)
            }
        }
        dispatchGroup.notify(queue: .main) {
            completion(companiesMentions)
        }
    }
```

```
    //MARK: - Private Properties
    private var dataTask: URLSessionDataTask?
    private let defaultSession = URLSession(configuration: .default)

    private let baseURLComponents: URLComponents = {
        var urlComponents = URLComponents()
        urlComponents.scheme = "https"
        urlComponents.host = "newsapi.org"
        return urlComponents
    }()
}




//MARK: - CompanyMentionEmotionalPrediction
enum CompanyMentionEmotionalPrediction: Int {
    case positive = 1
    case neutral = 0
    case negative = -1
}

//MARK: - CompanyMentionProtocol
protocol CompanyMentionProtocol {
    var text: String { get }
    var author: String? { get }
    var sourceURL: URL? { get }
    var creationDate: Date? { get }
    var emotionalPrediction: CompanyMentionEmotionalPrediction? { get set }
}


protocol CompanyMentionsFetcherProtocol {
```

```
    func fetchMentions(with searchSource: String, count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void)
    func fetchMentions(with searchSources: [String], count: Int, completion: @escaping
([CompanyMentionProtocol]) -> Void)
}


final class CompanyMentionsClassifier: CompanyMentionsClassifierProtocol {

    //MARK: - Public Methods
    func classifyMentionsUsingFinancialTweetsModel(_ mentions: [String], completion: @escaping
((positivePredictionScore: Int, predictions: [Int])) -> Void) {
        guard let mentionsClassifierModel = textClassifierFinancialTweetsModel else { return }
        do {
            let inputs = mentions.map(TextClassifierFinancialTweetsInput.init)
            let predictionsOutputs = try mentionsClassifierModel.predictions(inputs: inputs)
            var predictions = [Int]()
            var positivePredictionCount = Double(0)
            for predictionOutput in predictionsOutputs {
                switch predictionOutput.label {
                case "positive":
                    predictions.append(1)
                    positivePredictionCount += 1
                case "neutral":
                    predictions.append(0)
                    positivePredictionCount += 0.5
                case "negative":
                    predictions.append(-1)
                default:
                    continue
                }
            }

            let sourcePositivePredictionScore = positivePredictionCount / Double(predictionsOutputs.count)
            let finalPositivePredictionScore = Int(sourcePositivePredictionScore * 100)
```

```
            DispatchQueue.main.async {
                completion((finalPositivePredictionScore, predictions))
            }
        } catch {
            print("Error with classifying mentions using Financial Tweets model: \(error)")
        }
    }

    func classifyMentionsUsingInternetMovieDatabaseModel(_ mentions: [String], completion: @escaping
((positivePredictionScore: Int, predictions: [Int])) -> Void) {
        guard let mentionsClassifierModel = textClassifierInternetMovieDatabaseModel else { return }
        do {
            let inputs = mentions.map(TextClassifierInternetMovieDatabaseInput.init)
            let predictionsOutputs = try mentionsClassifierModel.predictions(inputs: inputs)
            var predictions = [Int]()
            var positivePredictionCount = 0
            for predictionOutput in predictionsOutputs {
                switch predictionOutput.label {
                case "pos":
                    predictions.append(1)
                    positivePredictionCount += 1
                case "neg":
                    predictions.append(-1)
                default:
                    continue
                }
            }

            let sourcePositivePredictionScore = Double(positivePredictionCount) /
Double(predictionsOutputs.count)
            let finalPositivePredictionScore = Int(sourcePositivePredictionScore * 100)
            DispatchQueue.main.async {
                completion((finalPositivePredictionScore, predictions))
            }
        } catch {
```

```
            print("Error with classifying mentions using Internet Movie Database model: \(error)")
        }
    }

    //MARK: - Private Properties
    private let textClassifierFinancialTweetsModel = try? TextClassifierFinancialTweets(configuration:
MLModelConfiguration())
    private let textClassifierInternetMovieDatabaseModel = try?
TextClassifierInternetMovieDatabase(configuration: MLModelConfiguration())
}




protocol CompanyMentionsClassifierProtocol {
    func classifyMentionsUsingFinancialTweetsModel(_ mentions: [String], completion: @escaping
((positivePredictionScore: Int, predictions: [Int])) -> Void)
    func classifyMentionsUsingInternetMovieDatabaseModel(_ mentions: [String], completion: @escaping
((positivePredictionScore: Int, predictions: [Int])) -> Void)
}



struct CompanyInfoNetworkModel: Codable {
    enum CodingKeys: String, CodingKey {
        case stockSymbol = "Symbol"
        case companyName = "Name"
        case companyDescription = "Description"
        case country = "Country"
        case profitMargin = "ProfitMargin"
        case marketCapitalization = "MarketCapitalization"
    }
    enum CodingError: Error {
        case canNotConvertStringToUInt
        case canNotConvertStringToDouble
    }
```

```
    let stockSymbol: String
    let companyName: String
    let companyDescription: String
    let country: String
    let profitMargin: Double
    let marketCapitalization: UInt

    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)
        country = try container.decode(String.self, forKey: .country)
        stockSymbol = try container.decode(String.self, forKey: .stockSymbol)
        companyName = try container.decode(String.self, forKey: .companyName)
        companyDescription = try container.decode(String.self, forKey: .companyDescription)
        let profitMarginString = try container.decode(String.self, forKey: .profitMargin)
        guard let profitMargin = Double(profitMarginString) else { throw
CodingError.canNotConvertStringToDouble }
        self.profitMargin = profitMargin
        let marketCapitalizationString = try container.decode(String.self, forKey: .marketCapitalization)
        guard let marketCapitalization = UInt(marketCapitalizationString) else { throw
CodingError.canNotConvertStringToUInt }
        self.marketCapitalization = marketCapitalization
    }

    func encode(to encoder: Encoder) throws {
        var container = encoder.container(keyedBy: CodingKeys.self)
        try container.encode(country, forKey: .country)
        try container.encode(stockSymbol, forKey: .stockSymbol)
        try container.encode(companyName, forKey: .companyName)
        try container.encode(companyDescription, forKey: .companyDescription)
        try container.encode("\(profitMargin)", forKey: .profitMargin)
        try container.encode("\(marketCapitalization)", forKey: .marketCapitalization)
    }
}
```

```
struct CompanyStockNetworkModel {
    let openPrice: Double
    let highPrice: Double
    let lowPrice: Double
    let closePrice: Double
    let volume: Int
    let date: Date
}


final class CompanyDataManager: CompanyDataManagerProtocol {

    static let shared = CompanyDataManager()
    private init() { /* empty */}

    //MARK: - CompanyDataManagerProtocol
    func fetchCompanyInfo(from stockSymbol: String, completion: @escaping (CompanyInfoNetworkModel) ->
Void) {
        var urlComponents = self.baseURLComponents
        urlComponents.path = "/query"
        urlComponents.setQueryItems(with: ["function": "OVERVIEW", "symbol": stockSymbol, "apikey":
SecretKeys.alphaVantageKey1])

        guard let requestUrl = urlComponents.url else { return }
        dataTask = defaultSession.dataTask(with: requestUrl) { [weak self] (data, _, error) in
            guard let self = self else { return }
            defer { self.dataTask = nil }
            guard let data = data, error == nil else { return }
            let decoder = JSONDecoder()

            do {
                let companyData = try decoder.decode(CompanyInfoNetworkModel.self, from: data)
                DispatchQueue.main.async {
```

```
                completion(companyData)
            }
        } catch {
            print("Error with decoding data from Alpha Vantage: \(error.localizedDescription)")
        }
    }
    dataTask?.resume()
}

func fetchCompanyStocks(from stockSymbol: String, completion: @escaping
([CompanyStockNetworkModel]) -> Void) {
    var urlComponents = self.baseURLComponents
    urlComponents.path = "/query"
    urlComponents.setQueryItems(with: ["function": "TIME_SERIES_DAILY", "symbol": stockSymbol,
"apikey": SecretKeys.alphaVantageKey2])

    guard let requestUrl = urlComponents.url else { return }
    dataTask = defaultSession.dataTask(with: requestUrl) { [weak self] (data, _, error) in
        guard let self = self else { return }
        defer { self.dataTask = nil }
        guard let data = data, error == nil,
            let rootJSON = try? JSONSerialization.jsonObject(with: data, options: []) as? [String: Any],
            let stocksDict = rootJSON["Time Series (Daily)"] as? [String: Any]
        else {
            print("Error with decoding data from Alpha Vantage for \(stockSymbol)")
            return
        }
        let stocksModels = stocksDict.compactMap { dict -> CompanyStockNetworkModel? in
            guard let innderDict = dict.value as? [String: String],
                let openPriceString = innderDict["1. open"], let openPrice = Double(openPriceString),
                let highPriceString = innderDict["2. high"], let highPrice = Double(highPriceString),
                let lowPriceString = innderDict["3. low"], let lowPrice = Double(lowPriceString),
                let closePriceString = innderDict["4. close"], let closePrice = Double(closePriceString),
                let volumeString = innderDict["5. volume"], let volume = Int(volumeString),
                let date = self.dateFormatter.date(from: dict.key)
```

```
            else { return nil }
            return CompanyStockNetworkModel(openPrice: openPrice, highPrice: highPrice, lowPrice:
lowPrice, closePrice: closePrice, volume: volume, date: date)
        }
        let sortedStocksModels = stocksModels.sorted { $0.date > $1.date }
        DispatchQueue.main.async {
            completion(sortedStocksModels)
        }
    }

    dataTask?.resume()
  }

  func fetchCurrentCompanyStock(from stockSymbol: String, completion: @escaping
(CompanyStockNetworkModel) -> Void) {
    fetchCompanyStocks(from: stockSymbol) { stocks in
        guard let currentStock = stocks.first else { return }
        completion(currentStock)
    }
  }

  //MARK: - Private Properties
  private var dataTask: URLSessionDataTask?
  private let defaultSession = URLSession(configuration: .default)

  private let baseURLComponents: URLComponents = {
    var urlComponents = URLComponents()
    urlComponents.scheme = "https"
    urlComponents.host = "www.alphavantage.co"
    return urlComponents
  }()

  private let dateFormatter: DateFormatter = {
    let dateFormatter = DateFormatter()
    dateFormatter.locale = Locale(identifier: "en_US_POSIX")
```

```
        dateFormatter.dateFormat = "yyyy-MM-dd"
        dateFormatter.timeZone = TimeZone(identifier: "GMT")
        return dateFormatter
    }()
}


protocol CompanyDataManagerProtocol {
    func fetchCompanyInfo(from stockSymbol: String, completion: @escaping (CompanyInfoNetworkModel) ->
Void)
    func fetchCompanyStocks(from stockSymbol: String, completion: @escaping
([CompanyStockNetworkModel]) -> Void)
    func fetchCurrentCompanyStock(from stockSymbol: String, completion: @escaping
(CompanyStockNetworkModel) -> Void)
}


//MARK: - CompanyModel
struct CompanyModel: Codable {
    enum Sector: Codable {
        case industrials
        case healthcare
        case technology
        case telecomMedia
        case goods
        case energy
        case financials
    }

    // ex. Apple Inc.
    let fullName: String
    // ex. #AAPL
    let stockName: String
    // ex. @Apple
    let accountName: String
```

```
    // ex. technology
    let sector: Sector
    // ex. AAPL
    var stockSymbol: String {
        return String(stockName.dropFirst())
    }
    // ex. Apple
    var accountSymbol: String {
        return String(accountName.dropFirst())
    }
    var logo: UIImage {
        return UIImage(named: stockName) ??  imageLiteral(resourceName: "default-company-logo")
    }
}

//MARK: - Department + CaseIterable
extension CompanyModel.Sector: CaseIterable { /* empty */}

//MARK: - Department + RawRepresentable
extension CompanyModel.Sector: RawRepresentable {

    typealias RawValue = String
    init?(rawValue: String) {
        switch rawValue {
        case "industrials": self = .industrials
        case "healthcare": self = .healthcare
        case "technology": self = .technology
        case "telecom-media": self = .telecomMedia
        case "goods": self = .goods
        case "energy": self = .energy
        case "financials": self = .financials
        default: return nil
        }
    }
```

```
    var rawValue: RawValue {
      switch self {
      case .industrials: return "Industrials"
      case .healthcare: return "Healthcare"
      case .technology: return "Technology"
      case .telecomMedia: return "Telecom-Media"
      case .goods: return "Goods"
      case .energy: return "Energy"
      case .financials: return "Financials"
      }
    }
}


final class CompaniesDataSource: CompaniesDataSourceProtocol {
    static let shared = CompaniesDataSource()
    private init() { /* empty */ }
    private(set) lazy var companies: [CompanyModel] = self.getCompaniesFromPlist()

    //MARK: - Private Properties
    private let fileName = "CompaniesList"
    private let fileType = "plist"
}

//MARK: - Private Methods
private extension CompaniesDataSource {

    func getCompaniesFromPlist() -> [CompanyModel] {
        guard let filePath = Bundle.main.path(forResource: fileName, ofType: fileType) else {
            preconditionFailure("[CompaniesDataSource]\(fileName) doesn't exist")
        }
        let fileURL = URL(fileURLWithPath: filePath)
        guard let fileData = try? Data(contentsOf: fileURL) else {
            preconditionFailure("[CompaniesDataSource] Can't read data from \(fileName)")
        }
```

```
        let decoder = PropertyListDecoder()
        guard let companies = try? decoder.decode([CompanyModel].self, from: fileData) else {
            preconditionFailure("[CompaniesDataSource] Can't decode data from \(fileData)")
        }
        return companies
    }
}


protocol CompaniesDataSourceProtocol {
    var companies: [CompanyModel] { get }
}


final class DefaultNavigationController: UINavigationController {
    //MARK: - UINavigationController
    override var prefersStatusBarHidden: Bool {
        return false
    }
    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .lightContent
    }
    override var childForStatusBarHidden: UIViewController? {
        return topViewController
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        setupNavigationBar()
    }
}

//MARK: - Setup Methods
private extension DefaultNavigationController {
```

```
    func setupNavigationBar() {
        navigationBar.tintColor = .navigationBarTint
        navigationBar.barTintColor = .background
        navigationBar.titleTextAttributes = [
            .foregroundColor: UIColor.title1,
            .font: UIFont.navigationControllerTitle
        ]
        navigationBar.barStyle = .black
        navigationBar.setBackgroundImage(UIImage(), for: .default)
        navigationBar.shadowImage = UIImage()
        navigationBar.isTranslucent = false
    }
}


final class ProgressViewController: UIViewController {
    @IBOutlet private weak var titleLabel: UILabel!
    @IBOutlet private weak var activityIndicatorView: UIActivityIndicatorView!

    //MARK: - Initializers
    init(viewModel: ProgressViewModelViewavable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }
```

```
    //MARK: - Public Methods
    func startAnimating() {
        activityIndicatorView.startAnimating()
    }

    func stopAnimating() {
        activityIndicatorView.stopAnimating()
    }

    //MARK: - Private Properties
    private let viewModel: ProgressViewModelViewavable
}

//MARK: - Helper Methods
private extension ProgressViewController {

    func setupViewController() {
        setupLoadingViews()
    }

    func setupLoadingViews() {
        titleLabel.text = viewModel.title
    }
}


private typealias Dependencies = LocalizationInjectable

final class ProgressViewModel: ProgressViewModelViewavable, Dependencies {    enum State {
        case loading
        case predicting
    }

    //MARK: - Public Properties
    var title: String {
```

```
        switch state {
        case .loading:
            return localizationTable.localized(for: "General_LoadingTitle")
        case .predicting:
            return localizationTable.localized(for: "General_PredictingTitle")
        }
    }

    //MARK: - Initializer
    init(state: State) {
        self.state = state
    }

    //MARK: - Private Properties
    private let state: State
    private lazy var localizationTable = localization.tables.other
}


protocol ProgressViewModelViewavable {
    var title: String { get }
}


final class SplashViewController: UIViewController {
    private enum Configuration {
        enum Keyframe1 {
            static let startTime: TimeInterval = 0.0
            static let duration: TimeInterval = 0.3
            static let scale: CGFloat = 1.1
        }
        enum Keyframe2 {
            static let startTime: TimeInterval = 0.35
            static let duration: TimeInterval = 0.6
            static let scale: CGFloat = 0.9
```

```
    }

    static let animationDelay: TimeInterval = 0.3
    static let animationDuration: TimeInterval = 0.9
  }

  //MARK: - Outlets
  @IBOutlet private weak var titleLabel: UILabel!

  //MARK: - Initializers
  init(viewModel: SplashViewModelProtocol) {
    self.viewModel = viewModel
    super.init(nibName: nil, bundle: nil)
  }

  required init?(coder aDecoder: NSCoder) {
    preconditionFailure("init(coder:) has not been implemented")
  }

  //MARK: - UIViewController
  override func viewDidLoad() {
    super.viewDidLoad()
    setupViewController()
  }

  //MARK: - Public Methods
  func runAnimation() {
    UIView.animateKeyframes(withDuration: Configuration.animationDuration, delay:
Configuration.animationDelay, options: [], animations: {
      UIView.addKeyframe(withRelativeStartTime: Configuration.Keyframe1.startTime, relativeDuration:
Configuration.Keyframe1.duration) {
        let scale = Configuration.Keyframe1.scale
        self.titleLabel.transform = CGAffineTransform(scaleX: scale, y: scale)
      }
```

```
            UIView.addKeyframe(withRelativeStartTime: Configuration.Keyframe2.startTime, relativeDuration:
Configuration.Keyframe2.duration) {
                let scale = Configuration.Keyframe2.scale
                self.titleLabel.transform = CGAffineTransform(scaleX: scale, y: scale)
                self.titleLabel.alpha = 0.0
            }
        }, completion: { _ in
            self.viewModel.completeAnimation()
        })
    }

    //MARK: - Private Properties
    private let viewModel: SplashViewModelProtocol
}

//MARK: - Setup Methods
private extension SplashViewController {

    func setupViewController() {
        setupViewsColor()
    }

    func setupViewsColor() {
        titleLabel.textColor = .title1
    }
}


final class SplashViewModel: SplashViewModelProtocol {
    var onCompleteAnimation: (() -> Void)?

    func completeAnimation() {
        onCompleteAnimation?()
    }
}
```

```
protocol SplashViewModelProtocol: class {
    func completeAnimation()
}


final class TutorialRootViewController: UIViewController {
    //MARK: - Outlets
    @IBOutlet private weak var actionlButton: UIButton!

    //MARK: - Initializers
    init(viewModel: TutorialRootViewModelViewable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }

    //MARK: - Actions
    @IBAction func performAction(_ sender: UIButton) {
        touchFeedbackGenerator.impactOccurred()
        viewModel.performAction()
    }

    //MARK: - Private Properties
    private let viewModel: TutorialRootViewModelViewable
```

```
    private let touchFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)
}

//MARK: - Setup Methods
private extension TutorialRootViewController {

    func setupViewController() {
        setupTitle()
        setupStateChanging()
    }

    func setupTitle() {
        title = viewModel.title
    }

    func setupStateChanging() {
        updateViewController()

        viewModel.onStateChanged = { [weak self] in
            self?.updateViewController()
        }
    }

    func updateViewController() {
        actionlButton.setTitle(viewModel.actionButtonTitle, for: .normal)
    }
}


typealias TutorialRootViewModelProtocol = TutorialRootViewModelCoordinatable &
TutorialRootViewModelViewable

final class TutorialRootViewModel: TutorialRootViewModelProtocol, LocalizationInjectable {

    enum State {
```

```
        case `continue`
        case begin
    }

    //MARK: - Public Properties
    var onStateChanged: (() -> Void)?
    var onPerformAction: (() -> Void)?
    var pageViewModels: [TutorialPageViewModelProtocol] = []
    var state: State = .continue {
        didSet {
            onStateChanged?()
        }
    }
    var actionButtonTitle: String {
        switch state {
        case .continue:
            return localizationTable.localized(for: "Tutorial_ContinueButtonTitle")
        case .begin:
            return localizationTable.localized(for: "Tutorial_BeginButtonTitle")
        }
    }

    private(set) lazy var title = localizationTable.localized(for: "Tutorial_Title")

    //MARK: - Initializer
    init() {
        let pageViewModels = [
            TutorialPageViewModel(title: localizationTable.localized(for: "Tutorial_PageTitle_1"), description:
localizationTable.localized(for: "Tutorial_PageDescription_1"), image:  imageLiteral(resourceName: "tutorial-
icon-1")),
            TutorialPageViewModel(title: localizationTable.localized(for: "Tutorial_PageTitle_2"), description:
localizationTable.localized(for: "Tutorial_PageDescription_2"), image:  imageLiteral(resourceName: "tutorial-
icon-2")),
```

```
        TutorialPageViewModel(title: localizationTable.localized(for: "Tutorial_PageTitle_3"), description:
localizationTable.localized(for: "Tutorial_PageDescription_3"), image:  imageLiteral(resourceName: "tutorial-
icon-3"))
      ]
      self.pageViewModels = pageViewModels
   }

   //MARK: - Public Methods
   func performAction() {
      onPerformAction?()
   }

   //MARK: - Private Properties
   private lazy var localizationTable = localization.tables.other
}


protocol TutorialRootViewModelViewable: class {
   var onStateChanged: (() -> Void)? { get set }
   var state: TutorialRootViewModel.State { get }

   var title: String { get }
   var actionButtonTitle: String { get }

   func performAction()
}


protocol TutorialRootViewModelCoordinatable: class {
   var onPerformAction: (() -> Void)? { get set }
   var state: TutorialRootViewModel.State { get set }
   var pageViewModels: [TutorialPageViewModelProtocol] { get }
}
```

```
final class TutorialPageViewController: UIViewController {
    //MARK: - Outlets
    @IBOutlet private weak var titleLabel: UILabel!
    @IBOutlet private weak var descriptionLabel: UILabel!
    @IBOutlet private weak var iconImageView: UIImageView!

    //MARK: - Initializers
    init(viewModel: TutorialPageViewModelProtocol) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }

    //MARK: - Private Properties
    private let viewModel: TutorialPageViewModelProtocol
}

//MARK: - Setup Methods
private extension TutorialPageViewController {

    func setupViewController() {
        setupIcon()
        setupTitle()
        setupDescription()
    }
```

```
    func setupIcon() {
        iconImageView.image = viewModel.image
    }

    func setupTitle() {
        titleLabel.text = viewModel.title
        titleLabel.textColor = .title1
    }

    func setupDescription() {
        descriptionLabel.text = viewModel.description
        descriptionLabel.textColor = .title2
    }
}




final class TutorialPageViewModel: TutorialPageViewModelProtocol {
    let title: String
    let description: String
    let image: UIImage

    init(title: String, description: String, image: UIImage) {
        self.title = title
        self.description = description
        self.image = image
    }
}


protocol TutorialPageViewModelProtocol: class {
    var title: String { get }
    var description: String { get }
    var image: UIImage { get }
```

```
}


final class SettingsViewController: UIViewController {
    private enum Configuration {
        static let tableViewEstimatedRowHeight: CGFloat = 50
        static let tableViewSectionHeaderHeight: CGFloat = 20
    }

    //MARK: - Outlets
    @IBOutlet private weak var tableView: UITableView!

    //MARK: - Initializers
    init(viewModel: SettingsViewModelViewable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }

    //MARK: - Actions
    @objc func closeSettingsAction() {
        viewModel.closeSettings()
    }

    //MARK: - Private Properties
    private let viewModel: SettingsViewModelViewable
```

```
}

//MARK: - UITableViewDataSource
extension SettingsViewController: UITableViewDataSource {

    func numberOfSections(in tableView: UITableView) -> Int {
        return viewModel.sections.count
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return settingsCells(in: section).count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell: SettingTableViewCell = tableView.dequeueReusableCell(for: indexPath)
        let settingCell = self.settingCell(for: indexPath)
        var style: SettingTableViewCell.Style = []

        if indexPath.row == 0 {
            style.formUnion(.top)
        }
        if indexPath.row == settingsCells(in: indexPath.section).count - 1 {
            style.formUnion(.bottom)
        }
        cell.updateContents(title: settingCell.title, icon: settingCell.icon, customRightView:
settingCell.customRightView, customFooterView: settingCell.customFooterView,
useDisclosureIconAsRightView: settingCell.useDisclosureIconAsRightView, style: style)
        return cell
    }

    func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
        let viewForHeader = UIView()
        viewForHeader.backgroundColor = .clear
        return viewForHeader
    }
```

```
}

//MARK: - UITableViewDelegate
extension SettingsViewController: UITableViewDelegate {

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)

        let settingCell = self.settingCell(for: indexPath)
        settingCell.action?()
    }

    func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
        return Configuration.tableViewSectionHeaderHeight
    }
}

//MARK: - Setup Methods
private extension SettingsViewController {

    func setupViewController() {
        setupActions()
        setupTableView()
        setupNavigationBar()
    }

    func setupNavigationBar() {
        title = viewModel.title

        let rightBarButtonItem = UIBarButtonItem(title: viewModel.closeButtonTitle, style: .plain, target: self,
action: #selector(closeSettingsAction))
        navigationItem.rightBarButtonItem = rightBarButtonItem

        let attributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title3, .font:
UIFont.navigationControllerBarButtonTitle]
```

```
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        states.forEach { rightBarButtonItem.setTitleTextAttributes(attributes, for: $0) }
    }

    func setupTableView() {
        tableView.rowHeight = UITableView.automaticDimension
        tableView.estimatedRowHeight = Configuration.tableViewEstimatedRowHeight
        tableView.register(SettingTableViewCell.self)
    }

    func setupActions() {
        let alertController = UIAlertController(title: nil, message: viewModel.featureIsNotAvailableAlertMessage,
preferredStyle: .alert)
        alertController.addAction(UIAlertAction(title: viewModel.featureIsNotAvailableAlertButtonTitle, style:
.cancel))
        viewModel.onOpenAboutApplication = { [weak self] in
            self?.present(alertController, animated: true)
        }
        viewModel.onOpenPrivacyPolicy = { [weak self] in
            self?.present(alertController, animated: true)
        }
        viewModel.onRateOnTheAppStore = { [weak self] _ in
            self?.present(alertController, animated: true)
        }
        viewModel.onShareApplication = { [weak self] _ in
            self?.present(alertController, animated: true)
        }
    }
}

//MARK: - Helper Methods
private extension SettingsViewController {

    func settingCell(for indexPath: IndexPath) -> SettingCell {
        return viewModel.sections[indexPath.section][indexPath.row]
```

```
    }

    func settingsCells(in section: Int) -> [SettingCell] {
        return viewModel.sections[section]
    }
}




private typealias SettingsViewModelProtocol = SettingsViewModelCoordinatable &
SettingsViewModelViewable
private typealias Dependencies = LocalizationInjectable & PreferencesInjectable

final class SettingsViewModel: SettingsViewModelProtocol, Dependencies {
    //MARK: - Public Properties
    var onClose: (() -> Void)?
    var onOpenAboutApplication: (() -> Void)?
    var onOpenPrivacyPolicy: (() -> Void)?
    var onRateOnTheAppStore: ((URL) -> Void)?
    var onShareApplication: ((SharingSource) -> Void)?

    private(set) var sections: [[SettingCell]] = [[]]
    private(set) lazy var title = localizationTable.localized(for: "Settings_Title")
    private(set) lazy var closeButtonTitle = localizationTable.localized(for: "General_CloseButtonTitle")
    private(set) lazy var featureIsNotAvailableAlertMessage = localizationTable.localized(for:
"Settings_FeatureIsNotAvailableAlert_Message")
    private(set) lazy var featureIsNotAvailableAlertButtonTitle = localizationTable.localized(for:
"Settings_FeatureIsNotAvailableAlert_ButtonTitle")

    //MARK: - Initializer
    init() {
        var sections = [
            [
```

```
            SettingCell(icon:  imageLiteral(resourceName: "settings_about_icon"), title:
localizationTable.localized(for: "Settings_SectionAboutAppTitle"), action: { [unowned self] in
self.openAboutApplication() }),
            SettingCell(icon:  imageLiteral(resourceName: "settings_privacy_icon"), title:
localizationTable.localized(for: "Settings_SectionPrivacyPolicyTitle"),action: { [unowned self] in
self.openPrivacyPolicy() })
        ],
        [
            SettingCell(icon:  imageLiteral(resourceName: "settings_rate_icon"), title:
localizationTable.localized(for: "Settings_SectionRateAppTitle"), action: { [unowned self] in
self.rateOnTheAppStore() }),
            SettingCell(icon:  imageLiteral(resourceName: "settings_share_icon"), title:
localizationTable.localized(for: "Settings_SectionShareAppTitle"), action: { [unowned self] in
self.shareApplication() })
        ]
    ]

    let mentionsCountsItems = mentionsCounts.map(String.init)
    let mentionsCountSegmentControl = UISegmentedControl(items: mentionsCountsItems)
    mentionsCountSegmentControl.overrideUserInterfaceStyle = .dark
    mentionsCountSegmentControl.selectedSegmentIndex = selectedMentionsCountSegmentIndex
    mentionsCountSegmentControl.addTarget(self, action: #selector(mentionsCountDidChangeValue(_:)),
for: .valueChanged)

    sections.append([
        SettingCell(title: localizationTable.localized(for: "Settings_SectionMentionsCountTitle"),
customFooterView: mentionsCountSegmentControl, useDisclosureIconAsRightView: false)
    ])
    self.sections = sections
  }

  //MARK: - Public Methods
  func closeSettings() {
    onClose?()
  }
```

```
    //MARK: - Private Properties
    private lazy var localizationTable = localization.tables.other
    private var mentionsCounts: [Int] {
        return PreferencesMentionsCountsProvider.mentionsCounts
    }
}

//MARK: - Actions
private extension SettingsViewModel {

    func rateOnTheAppStore() {
        guard let appReviewURL = URL(string: "itms-
apps://itunes.apple.com/app/id\(SecretKeys.appStoreAppID)?action=write-review&mt=8") else { return }
        onRateOnTheAppStore?(appReviewURL)
    }

    func shareApplication() {
        guard let applicationUrl = URL(string: "...") else { return }
        let sharingSource = SharingSource.option1(applicationUrl: applicationUrl)
        onShareApplication?(sharingSource)
    }

    func openAboutApplication() {
        onOpenAboutApplication?()
    }

    func openPrivacyPolicy() {
        onOpenPrivacyPolicy?()
    }

    @objc func mentionsCountDidChangeValue(_ sender: UISegmentedControl) {
        let mentionsCount = mentionsCounts[sender.selectedSegmentIndex]
        preferences.mentionsCount = mentionsCount
    }
```

```
}

//MARK: - Helper Methods
private extension SettingsViewModel {

    var selectedMentionsCountSegmentIndex: Int {
        return mentionsCounts.firstIndex(of: preferences.mentionsCount) ?? 0
    }
}




protocol SettingsViewModelViewable: class {
    var onOpenAboutApplication: (() -> Void)? { get set }
    var onOpenPrivacyPolicy: (() -> Void)? { get set }
    var onRateOnTheAppStore: ((URL) -> Void)? { get set }
    var onShareApplication: ((SharingSource) -> Void)? { get set }
    var title: String { get }
    var closeButtonTitle: String { get }
    var sections: [[SettingCell]] { get }
    var featureIsNotAvailableAlertMessage: String { get }
    var featureIsNotAvailableAlertButtonTitle: String { get }
    func closeSettings()
}


protocol SettingsViewModelCoordinatable: class {
    var onClose: (() -> Void)? { get set }
}


final class SettingTableViewCell: UITableViewCell, NibLoadableView {
    private enum Configuration {
        static let cornerRadius: CGFloat = 15
```

```
    }

    struct Style: OptionSet {
        let rawValue: Int

        static let top = Style(rawValue: 1 << 0)
        static let regular = Style(rawValue: 1 << 1)
        static let bottom = Style(rawValue: 1 << 2)
    }

    //MARK: - Outlets
    @IBOutlet private weak var roundedView: UIView!
    @IBOutlet private weak var titleLabel: UILabel!
    @IBOutlet private weak var iconImageView: UIImageView!
    @IBOutlet private weak var disclosureImageView: UIImageView!
    @IBOutlet private weak var customFooterViewContainer: UIView!

    //MARK: - Public Methods
    func updateContents(title: String, icon: UIImage?, customRightView: UIView?, customFooterView:
UIView?, useDisclosureIconAsRightView: Bool, style: Style) {
        self.style = style
        self.customRightView = customRightView
        self.customFooterView = customFooterView
        self.useDisclosureIconAsRightView = useDisclosureIconAsRightView

        titleLabel.text = title
        iconImageView.image = icon
        iconImageView.isHidden = icon == nil
        customFooterViewContainer.isHidden = customFooterView == nil

        updateCell(newCustomRightView: customRightView, newCustomFooterView: customFooterView)
        updateLayout()
    }

    override func layoutSubviews() {
```

```
        super.layoutSubviews()
        updateLayout()
    }

    //MARK: - Private Properties
    private var style: Style = .regular
    private var useDisclosureIconAsRightView: Bool = true

    private var customRightView: UIView? = nil {
        didSet {
            oldValue?.removeFromSuperview()
        }
    }
    private var customFooterView: UIView? = nil {
        didSet {
            oldValue?.removeFromSuperview()
        }
    }

    private lazy var maskLayer: CAShapeLayer = {
        let shapeLayer = CAShapeLayer()
        roundedView.layer.mask = shapeLayer
        return shapeLayer
    }()
}

//MARK: - Update Methods
private extension SettingTableViewCell {

    func updateLayout() {
        roundedView.layoutIfNeeded()
        var corners: UIRectCorner = []
        if style.contains(.top) {
            corners.formUnion([.topLeft, .topRight])
        }
```

```
        if style.contains(.bottom) {
            corners.formUnion([.bottomLeft, .bottomRight])
        }
        let cornerRadius = Configuration.cornerRadius
        let maskLayerPath = UIBezierPath(roundedRect: roundedView.bounds, byRoundingCorners: corners,
cornerRadii: CGSize(width: cornerRadius, height: cornerRadius)).cgPath
        maskLayer.path = maskLayerPath
    }

    func updateCell(newCustomRightView: UIView?, newCustomFooterView: UIView?) {
        if let newCustomFooterView = newCustomFooterView {
            customFooterViewContainer.embedView(newCustomFooterView)
        }
        if let newCustomRightView = newCustomRightView {
            disclosureImageView.superview?.addSubview(newCustomRightView)
            disclosureImageView.isHidden = true
            newCustomRightView.translatesAutoresizingMaskIntoConstraints = false
            newCustomRightView.centerYAnchor.constraint(equalTo:
disclosureImageView.centerYAnchor).isActive = true
            newCustomRightView.trailingAnchor.constraint(equalTo:
disclosureImageView.trailingAnchor).isActive = true
        } else {
            disclosureImageView.isHidden = !useDisclosureIconAsRightView
        }
    }
}


struct SettingCell {
    let icon: UIImage?
    let title: String
    let action: (() -> Void)?
    let customRightView: UIView?
    let customFooterView: UIView?
    let useDisclosureIconAsRightView: Bool
```

```
    init(icon: UIImage? = nil, title: String, action: (() -> Void)? = nil, customRightView: UIView? = nil,
customFooterView: UIView? = nil, useDisclosureIconAsRightView: Bool = true) {
        self.icon = icon
        self.title = title
        self.action = action
        self.customRightView = customRightView
        self.customFooterView = customFooterView
        self.useDisclosureIconAsRightView = useDisclosureIconAsRightView
    }
}


final class CompanyTableViewCell: UITableViewCell, NibLoadableView {
    //MARK: - Outlets
    @IBOutlet private weak var fullNameLabel: UILabel!
    @IBOutlet private weak var stockNameLabel: UILabel!
    @IBOutlet private weak var stockPriceLabel: UILabel!
    @IBOutlet override var imageView: UIImageView? {
        get {
            return _imageView
        }
        set {
            _imageView = newValue
        }
    }
    @IBOutlet override var backgroundView: UIView? {
        get {
            return _backgroundView
        }
        set {
            _backgroundView = newValue
        }
    }
```

```
    override func awakeFromNib() {
        super.awakeFromNib()
        setupCell()
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)
        backgroundView?.backgroundColor = isSelected ? UIColor(hex: 0x222226) : UIColor.background
    }

    //MARK: - Public Methods
    func configure(stockName: String, fullName: String, image: UIImage?) {
        stockPriceLabel.text = "..."
        stockNameLabel.text = stockName
        fullNameLabel.text = fullName
        imageView?.image = image
    }

    func updateStockPrice(with stockPrice: String) {
        stockPriceLabel.text = stockPrice
    }

    //MARK: - Private Properties
    private var _imageView: UIImageView?
    private var _backgroundView: UIView?
}

//MARK: - Private Methods
private extension CompanyTableViewCell {

    func setupCell() {
        backgroundView?.clipsToBounds = true
        backgroundView?.layer.cornerRadius = 15
        backgroundView?.layer.maskedCorners = [.layerMinXMinYCorner, .layerMinXMaxYCorner]
    }
```

```
}


final class CompaniesListViewController: UIViewController {
    private enum Configuration {
        static let tableViewRowHeight: CGFloat = 80
        static let tableViewSectionHeaderHeight: CGFloat = 60
    }
    //MARK: - Outlets
    @IBOutlet private weak var tableView: UITableView!

    //MARK: - Initializers
    init(viewModel: CompaniesListViewModelViewable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        deselectRowIfNeeded()
    }

    //MARK: - Actions
    @objc func openSettingsAction() {
        viewModel.openSettings()
```

```
    }

    //MARK: - Private Properties
    private let viewModel: CompaniesListViewModelViewable
    private let searchController = UISearchController(searchResultsController: nil)
    private let selectionFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)
}

//MARK: - UITableViewDataSource
extension CompaniesListViewController: UITableViewDataSource {

    func numberOfSections(in tableView: UITableView) -> Int {
        return sectionsForCurrentState.count
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return companies(in: section).count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let company = self.company(for: indexPath)
        let cell: CompanyTableViewCell = tableView.dequeueReusableCell(for: indexPath)
        cell.configure(stockName: company.stockName, fullName: company.fullName, image: company.logo)
        viewModel.fecthStockPrice(stockSymbol: company.stockSymbol) { stockPrice in
            cell.updateStockPrice(with: "$\(stockPrice)")
        }
        return cell
    }

    func sectionIndexTitles(for tableView: UITableView) -> [String]? {
        return sectionIndexTitlesForCurrentState
    }

    func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
        let companiesSection = sectionsForCurrentState[section]
```

```
        let sectionHeader: CompaniesListSectionHeaderView =
tableView.dequeueReusableHeaderFooterView()
        sectionHeader.configure(title: companiesSection.title)
        return sectionHeader
    }
}

//MARK: - UITableViewDataSource
extension CompaniesListViewController: UITableViewDelegate {

    func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
        return Configuration.tableViewSectionHeaderHeight
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        selectionFeedbackGenerator.impactOccurred()
        let company = self.company(for: indexPath)
        viewModel.openCompanyDashboard(company)
    }
}

//MARK: - UITableViewDataSource
extension CompaniesListViewController: UISearchResultsUpdating {

    func updateSearchResults(for searchController: UISearchController) {
        guard let searchText = searchController.searchBar.text else { return }
        viewModel.filterCompanies(for: searchText) { [weak self] in
            self?.tableView.reloadData()
        }
    }
}

//MARK: - Setup Methods
private extension CompaniesListViewController {
```

```
    func setupViewController() {
        setupTableView()
        setupNavigationBar()
        setupSearchController()
        scrollToFirstRowAndHideSearchBar()
    }

    func setupNavigationBar() {
        title = viewModel.title
        navigationItem.rightBarButtonItem = UIBarButtonItem(image: #imageLiteral(resourceName:
"settings_icon"), style: .plain, target: self, action: #selector(openSettingsAction))
        let backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        navigationItem.backBarButtonItem = backBarButtonItem
        let attributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title1, .font:
UIFont.navigationControllerBackBarButtonTitle]
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        states.forEach { backBarButtonItem.setTitleTextAttributes(attributes, for: $0) }
    }

    func setupTableView() {
        tableView.sectionIndexColor = .title1
        tableView.rowHeight = Configuration.tableViewRowHeight
        tableView.register(CompanyTableViewCell.self)
        tableView.registerForHeaderFooterView(CompaniesListSectionHeaderView.self)
    }

    func setupSearchController() {
        searchController.searchResultsUpdater = self
        searchController.obscuresBackgroundDuringPresentation = false
        searchController.searchBar.placeholder = viewModel.searchPlaceholder
        searchController.searchBar.keyboardAppearance = .dark
        navigationItem.searchController = searchController
        definesPresentationContext = true
    }
}
```

```
//MARK: - Helper Methods
private extension CompaniesListViewController {

    var isFiltering: Bool {
        return searchController.isActive && !isSearchBarEmpty
    }

    var isSearchBarEmpty: Bool {
        return searchController.searchBar.text?.isEmpty ?? true
    }

    var sectionsForCurrentState: [CompaniesModelsSection] {
        return isFiltering ? viewModel.filteredSections : viewModel.sections
    }

    var sectionIndexTitlesForCurrentState: [String] {
        return isFiltering ? viewModel.filteredSectionIndexTitles : viewModel.sectionIndexTitles
    }

    func company(for indexPath: IndexPath) -> CompanyModel {
        return sectionsForCurrentState[indexPath.section].companies[indexPath.row]
    }

    func companies(in section: Int) -> [CompanyModel] {
        return sectionsForCurrentState[section].companies
    }

    func deselectRowIfNeeded() {
        guard let indexPath = tableView.indexPathForSelectedRow else { return }
        tableView.deselectRow(at: indexPath, animated: false)
    }

    func scrollToFirstRowAndHideSearchBar() {
        tableView.scrollToRow(at: IndexPath(row: 0, section: 0), at: .top, animated: false)
```

```
    }
}


private typealias CompaniesListViewModelProcotol = CompaniesListViewModelCoordinatable &
CompaniesListViewModelViewable
private typealias Dependencies = CompaniesDataSourceInjectable & LlocalizationInjectable &
CompanyDataManagerInjectable

final class CompaniesListViewModel: CompaniesListViewModelProcotol, Dependencies {
    //MARK: - Public Properties
    var onOpenSettings: (() -> Void)?
    var onOpenCompanyDashboard: ((CompanyModel) -> Void)?
    var onFetchStockPrice: ((Double) -> Void)?

    private(set) lazy var sections: [CompaniesModelsSection] =
self.makeSortedSectionsFromCompaniesArray()
    private(set) lazy var filteredSections: [CompaniesModelsSection] = []
    private(set) lazy var sectionIndexTitles: [String] = self.makeSectionIndexTitles(from: sections)
    var filteredSectionIndexTitles: [String] {
        return makeSectionIndexTitles(from: filteredSections)
    }

    private(set) lazy var title = localizationTable.localized(for: "CompaniesList_Title")
    private(set) lazy var searchPlaceholder = localizationTable.localized(for:
"CompaniesList_SearchPlaceholder")

    //MARK: - Public Methods
    func openSettings() {
        onOpenSettings?()
    }

    func openCompanyDashboard(_ company: CompanyModel) {
        onOpenCompanyDashboard?(company)
    }
```

```
    func fecthStockPrice(stockSymbol: String, completion: @escaping ((Double) -> Void)) {
        if let currentStockPrice = stocksPricesCache[stockSymbol] {
            completion(currentStockPrice)
            return
        }
        companyDataManager.fetchCurrentCompanyStock(from: stockSymbol) { [weak self] currentStock in
            guard let self = self else { return }
            self.stocksPricesCache[stockSymbol] = currentStock.closePrice
            completion(currentStock.closePrice)
        }
    }

    func filterCompanies(for searchText: String, completion: (() -> Void)) {
        filteredSections = sections
            .map { section -> CompaniesModelsSection in
                let filteredCompanies = section.companies.filter { company in
                    company.fullName.lowercased().contains(searchText.lowercased()) ||
company.stockName.lowercased().contains(searchText.lowercased())
                }
                return (title: section.title, companies: filteredCompanies)
            }
            .filter { section in
                !section.companies.isEmpty
            }

        completion()
    }

    //MARK: - Private Properties
    private var stocksPricesCache = [String: Double]()
    private let sectors = CompanyModel.Sector.allCases
    private lazy var localizationTable = localization.tables.other
}
```

```
//MARK: - Private Methods
private extension CompaniesListViewModel {

    func makeSectionIndexTitles(from sections: [CompaniesModelsSection]) -> [String] {
        return sections.map { String($0.title.prefix(1)) }
    }

    func makeSortedSectionsFromCompaniesArray() -> [CompaniesModelsSection] {
        var sections = [CompaniesModelsSection]()
        let companies = companiesDataSource.companies
        sectors.forEach { sector in
            var tempCompanies = [CompanyModel]()
            companies.forEach { company in
                if company.sector == sector {
                    tempCompanies.append(company)
                }
            }
            tempCompanies = tempCompanies.sorted { $0.stockName < $1.stockName }
            sections.append((title: sector.rawValue, companies: tempCompanies))
        }

        sections = sections.sorted { $0.title < $1.title }
        return sections
    }
}


typealias CompaniesModelsSection = (title: String, companies: [CompanyModel])

protocol CompaniesListViewModelViewable: class {
    var title: String { get }
    var searchPlaceholder: String { get }
    var sections: [CompaniesModelsSection] { get }
    var filteredSections: [CompaniesModelsSection] { get }
    var sectionIndexTitles: [String] { get }
```

```
    var filteredSectionIndexTitles: [String] { get }
    func openSettings()
    func openCompanyDashboard(_ company: CompanyModel)
    func filterCompanies(for searchText: String, completion: (() -> Void))
    func fecthStockPrice(stockSymbol: String, completion: @escaping ((Double) -> Void))
}


protocol CompaniesListViewModelCoordinatable: class {
    var onOpenSettings: (() -> Void)? { get set }
    var onOpenCompanyDashboard: ((CompanyModel) -> Void)? { get set }
}


final class CompanyPredictionViewController: UIViewController {
    //MARK: - Outlets
    @IBOutlet private weak var progressContainer: UIView!
    @IBOutlet private weak var companyNameLabel: UILabel!
    @IBOutlet private weak var companyPredictionScoreLabel: UILabel!
    @IBOutlet private weak var companyPredictionTitleLabel: UILabel!
    @IBOutlet private weak var companyPredictionDescriptionLabel: UILabel!
    @IBOutlet private weak var companyLogoImageView: UIImageView!

    //MARK: - Initializers
    init(viewModel: CompanyPredictionViewModelViewavable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
```

```
        super.viewDidLoad()
        setupViewController()
    }

    //MARK: - Actions
    @objc func openCompanyMentionsListAction() {
        viewModel.openCompanyMentionsList()
    }

    //MARK: - Private Properties
    private let viewModel: CompanyPredictionViewModelViewavable
    private let progressViewController = ProgressViewController(viewModel: ProgressViewModel(state:
.predicting))
}

//MARK: - Setup Methods
private extension CompanyPredictionViewController {

    func setupViewController() {
        setupProgressViews()
        setupNavigationBar()
        viewModel.onUpdate = { [weak self] in
            self?.setupContent()
            self?.updateContent()
        }
    }

    func setupContent() {
        companyNameLabel.text = viewModel.companyFullName
        companyLogoImageView.image = viewModel.companyLogo
        companyPredictionScoreLabel.text = "Prediction Score: \(viewModel.predictionScore)%"
        companyPredictionTitleLabel.text = viewModel.predictionTitle
        companyPredictionDescriptionLabel.text = viewModel.predictionDescription
    }
```

```
    func setupProgressViews() {
        progressContainer.embedView(progressViewController.view)
        progressContainer.isHidden = false
        progressViewController.startAnimating()
    }

    func setupNavigationBar() {
        title = viewModel.title
        let chartImageConfig = UIImage.SymbolConfiguration(pointSize: 25, weight: .medium, scale: .default)
        let chartImage = UIImage(systemName: "text.bubble", withConfiguration: chartImageConfig)
        navigationItem.rightBarButtonItem = UIBarButtonItem(image: chartImage, style: .plain, target: self,
action: #selector(openCompanyMentionsListAction))
        let backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        navigationItem.backBarButtonItem = backBarButtonItem
        let attributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title1, .font:
UIFont.navigationControllerBackBarButtonTitle]
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        states.forEach { backBarButtonItem.setTitleTextAttributes(attributes, for: $0) }
    }
}

//MARK: - Helper Methods
private extension CompanyPredictionViewController {

    func updateContent() {
        hideProgressViews()
    }

    func hideProgressViews() {
        progressContainer.isHidden = true
        progressViewController.stopAnimating()
    }
}
```

```
private typealias CompanyPredictionViewModelProtocol = CompanyPredictionViewModelCoordinatable &
CompanyPredictionViewModelViewavable
private typealias Dependencies = TwitterCompanyMentionsFetcherInjectable &
NewsAPICompanyMentionsFetcherInjectable & PreferencesInjectable &
CompanyMentionsClassifierInjectable & LocalizationInjectable

final class CompanyPredictionViewModel: CompanyPredictionViewModelProtocol, Dependencies {
    //MARK: - Public Properties
    var onUpdate: (() -> Void)?
    var onOpenCompanyMentionsList: ((CompanyModel, [CompanyMentionsSource]) -> Void)?

    var predictionScore: Int {
        return predictionScores.reduce(0, +) / predictionScores.count
    }

    private(set) var mentionsSources: [CompanyMentionsSource] = []
    private(set) lazy var predictionTitle: String = makePredictionTitle()
    private(set) lazy var predictionDescription: String = makePredictionDescription()
    private(set) lazy var title: String = company.stockName
    private(set) lazy var companyLogo: UIImage? = company.logo
    private(set) lazy var companyFullName: String = company.fullName

    //MARK: - Initializer
    init(company: CompanyModel) {
        self.company = company
        fetchCompanyMentions()
    }

    //MARK: - Public Methods
    func openCompanyMentionsList() {
        onOpenCompanyMentionsList?(company, mentionsSources)
    }

    //MARK: - Private Properties
    private let company: CompanyModel
```

```
    private var predictionScores = [Int]()
    private lazy var localizationTable = localization.tables.other
}

//MARK: - Private Methods
private extension CompanyPredictionViewModel {

    func fetchCompanyMentions() {
        let allFetchersDispatchGroup = DispatchGroup()
        let twitterFetchersDispatchGroup = DispatchGroup()
        let mentionsCount = preferences.mentionsCount
        var companyTwitterMentions = [CompanyMentionProtocol]()
        twitterFetchersDispatchGroup.enter()
        twitterCompanyMentionsFetcher.fetchMentions(with: company.accountName, count: mentionsCount) {
[weak self] companyMentions in
self?.companyMentionsClassifier.classifyMentionsUsingInternetMovieDatabaseModel(companyMentions.ma
p { $0.text }) { predictionScore, predictions in
            defer { twitterFetchersDispatchGroup.leave() }
            var newCompanyMentions = [CompanyMentionProtocol]()
            for (var companyMention, prediction) in zip(companyMentions, predictions) {
                companyMention.emotionalPrediction = CompanyMentionEmotionalPrediction(rawValue:
prediction)
                newCompanyMentions.append(companyMention)
            }

            companyTwitterMentions.append(contentsOf: newCompanyMentions)
            self?.predictionScores.append(predictionScore)
        }
    }

    twitterFetchersDispatchGroup.enter()
    twitterCompanyMentionsFetcher.fetchMentions(with: company.stockName, count: mentionsCount) {
[weak self] companyMentions in
self?.companyMentionsClassifier.classifyMentionsUsingFinancialTweetsModel(companyMentions.map {
$0.text }) { predictionScore, predictions in
```

```
            defer { twitterFetchersDispatchGroup.leave() }
            var newCompanyMentions = [CompanyMentionProtocol]()
            for (var companyMention, prediction) in zip(companyMentions, predictions) {
                companyMention.emotionalPrediction = CompanyMentionEmotionalPrediction(rawValue:
prediction)
                newCompanyMentions.append(companyMention)
            }

            companyTwitterMentions.append(contentsOf: newCompanyMentions)
            self?.predictionScores.append(predictionScore)
        }
    }
    allFetchersDispatchGroup.enter()
    twitterFetchersDispatchGroup.notify(queue: .main) { [weak self] in
        defer { allFetchersDispatchGroup.leave() }
        let mentionsSource: CompanyMentionsSource = (sourceName: "Twitter", mentions:
companyTwitterMentions)
        self?.mentionsSources.append(mentionsSource)
    }

    allFetchersDispatchGroup.enter()
    newsAPICompanyMentionsFetcher.fetchMentions(with: company.accountSymbol, count:
mentionsCount) { [weak self] companyMentions in

self?.companyMentionsClassifier.classifyMentionsUsingFinancialTweetsModel(companyMentions.map {
$0.text }) { predictionScore, predictions in
            defer { allFetchersDispatchGroup.leave() }
            var newCompanyMentions = [CompanyMentionProtocol]()
            for (var companyMention, prediction) in zip(companyMentions, predictions) {
                companyMention.emotionalPrediction = CompanyMentionEmotionalPrediction(rawValue:
prediction)
                newCompanyMentions.append(companyMention)
            }
            let mentionsSource: CompanyMentionsSource = (sourceName: "News", mentions:
newCompanyMentions)
```

```
            self?.mentionsSources.append(mentionsSource)
            self?.predictionScores.append(predictionScore)
        }
    }
    allFetchersDispatchGroup.notify(queue: .main) { [weak self] in
        self?.onUpdate?()
    }
}

func makePredictionTitle() -> String {
    switch predictionScore {
    case 0..<20:
        return localizationTable.localized(for: "CompanyPrediction_PredictionTitle_0..20")
    case 20..<40:
        return localizationTable.localized(for: "CompanyPrediction_PredictionTitle_20..40")
    case 40..<60:
        return localizationTable.localized(for: "CompanyPrediction_PredictionTitle_40..60")
    case 60..<80:
        return localizationTable.localized(for: "CompanyPrediction_PredictionTitle_60..80")
    case 80...100:
        return localizationTable.localized(for: "CompanyPrediction_PredictionTitle_80..100")
    default:
        return localizationTable.localized(for: "General_UndefinedTitle")
    }
}

func makePredictionDescription() -> String {
    switch predictionScore {
    case 0..<20:
        return localizationTable.localized(for: "CompanyPrediction_PredictionDescription_0..20")
    case 20..<40:
        return localizationTable.localized(for: "CompanyPrediction_PredictionDescription_20..40")
    case 40..<60:
        return localizationTable.localized(for: "CompanyPrediction_PredictionDescription_40..60")
    case 60..<80:
```

```
            return localizationTable.localized(for: "CompanyPrediction_PredictionDescription_60..80")
        case 80...100:
            return localizationTable.localized(for: "CompanyPrediction_PredictionDescription_80..100")
        default:
            return localizationTable.localized(for: "General_UndefinedTitle")
        }
    }
}


protocol CompanyPredictionViewModelViewavable: class {
    var onUpdate: (() -> Void)? { get set }
    var title: String { get }
    var companyLogo: UIImage? { get }
    var companyFullName: String { get }
    var predictionTitle: String { get }
    var predictionDescription: String { get }
    var predictionScore: Int { get }
    func openCompanyMentionsList()
}


typealias CompanyMentionsSource = (sourceName: String, mentions: [CompanyMentionProtocol])

protocol CompanyPredictionViewModelCoordinatable: class {
    var onOpenCompanyMentionsList: ((CompanyModel, [CompanyMentionsSource]) -> Void)? { get set }
    var mentionsSources: [CompanyMentionsSource] { get }
}


final class CompanyDashboardViewController: UIViewController {
    //MARK: - Outlets
    @IBOutlet private weak var progressContainer: UIView!
    @IBOutlet private weak var descriptionTextOverlayView: UIView!
    @IBOutlet private weak var companyNameLabel: UILabel!
```

```
@IBOutlet private weak var companyCountryLabel: UILabel!
@IBOutlet private weak var companyProfitMarginLabel: UILabel!
@IBOutlet private weak var companyStockDateLabel: UILabel!
@IBOutlet private weak var companyLowStockPriceLabel: UILabel!
@IBOutlet private weak var companyHighStockPriceLabel: UILabel!
@IBOutlet private weak var companyOpenStockPriceLabel: UILabel!
@IBOutlet private weak var companyCloseStockPriceLabel: UILabel!
@IBOutlet private weak var companyDescriptionTextView: UITextView!
@IBOutlet private weak var companyLogoImageView: UIImageView!

//MARK: - Initializers
init(viewModel: CompanyDashboardViewModelViewable) {
    self.viewModel = viewModel
    super.init(nibName: nil, bundle: nil)
}

required init?(coder aDecoder: NSCoder) {
    preconditionFailure("init(coder:) has not been implemented")
}

//MARK: - UIViewController
override func viewDidLoad() {
    super.viewDidLoad()
    setupViewController()
}

//MARK: - Actions
@IBAction func predictAction(_ sender: UIButton) {
    touchFeedbackGenerator.impactOccurred()
    viewModel.openCompanyPrediction()
}

@objc func openCompanyStocksListAction() {
    viewModel.openCompanyStocksList()
}
```

```
    //MARK: - Private Properties
    private let viewModel: CompanyDashboardViewModelViewable
    private let touchFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)
    private let progressViewController = ProgressViewController(viewModel: ProgressViewModel(state:
.loading))

    private let dateFormatter: DateFormatter = {
        let dateFormatter = DateFormatter()
        dateFormatter.dateStyle = .medium
        dateFormatter.timeStyle = .none
        return dateFormatter
    }()
}

//MARK: - Setup Methods
private extension CompanyDashboardViewController {

    func setupViewController() {
        setupCompanyLogo()
        setupProgressViews()
        setupNavigationBar()
        setupTextViewOverlay()
        fetchContent()
    }

    func setupCompanyLogo() {
        companyLogoImageView.image = viewModel.companyLogo
    }

    func setupProgressViews() {
        progressContainer.embedView(progressViewController.view)
        progressContainer.isHidden = false
        progressViewController.startAnimating()
    }
```

```
    func setupNavigationBar() {
        title = viewModel.title
        let chartImageConfig = UIImage.SymbolConfiguration(pointSize: 25, weight: .medium, scale: .default)
        let chartImage = UIImage(systemName: "chart.bar.xaxis", withConfiguration: chartImageConfig)
        navigationItem.rightBarButtonItem = UIBarButtonItem(image: chartImage, style: .plain, target: self,
action: #selector(openCompanyStocksListAction))
        let backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        navigationItem.backBarButtonItem = backBarButtonItem
        let attributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title1, .font:
UIFont.navigationControllerBackBarButtonTitle]
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        states.forEach { backBarButtonItem.setTitleTextAttributes(attributes, for: $0) }
    }

    func setupTextViewOverlay() {
        let maskLayer = CAGradientLayer()
        maskLayer.colors = [UIColor.clear.cgColor, UIColor.black.cgColor]
        maskLayer.frame = descriptionTextOverlayView.bounds
        descriptionTextOverlayView.layer.mask = maskLayer
    }
}

//MARK: - Update Methods
private extension CompanyDashboardViewController {

    func fetchContent() {
        let dispatchGroup = DispatchGroup()
        dispatchGroup.enter()
        viewModel.fetchCompanyInfo { [weak self] companyInfo in
            defer { dispatchGroup.leave() }
            self?.updateContent(with: companyInfo)
        }
        dispatchGroup.enter()
        viewModel.fetchCompanyStock { [weak self] companyStock in
```

```
            defer { dispatchGroup.leave() }
            self?.updateContent(with: companyStock)
        }
        dispatchGroup.notify(queue: .main) { [weak self] in
            self?.hideLoadingViews()
        }
    }

    func updateContent(with companyInfo: CompanyInfoNetworkModel) {
        companyNameLabel.text = companyInfo.companyName
        companyCountryLabel.text = "Country: \(companyInfo.country)"
        companyProfitMarginLabel.text = "Profit Margin: \(makeProfitMarginReadableValue(from:
companyInfo.profitMargin))%"
        companyDescriptionTextView.text = companyInfo.companyDescription
    }

    func updateContent(with companyStock: CompanyStockNetworkModel) {
        let currentStockDate = dateFormatter.string(from: companyStock.date)
        companyStockDateLabel.text = "\(currentStockDate)"
        companyLowStockPriceLabel.text = "Low: $\(companyStock.lowPrice)"
        companyHighStockPriceLabel.text = "High: $\(companyStock.highPrice)"
        companyOpenStockPriceLabel.text = "Open: $\(companyStock.openPrice)"
        companyCloseStockPriceLabel.text = "Close: $\(companyStock.closePrice)"
    }
}

//MARK: - Helper Methods
private extension CompanyDashboardViewController {

    func makeProfitMarginReadableValue(from profitMargin: Double) -> Int {
        return Int(profitMargin * 100)
    }

    func hideLoadingViews() {
        progressContainer.isHidden = true
```

```
        progressViewController.stopAnimating()
    }
}


private typealias CompanyDashboardViewModelProtocol = CompanyDashboardViewModelCoordinatable &
CompanyDashboardViewModelViewable
private typealias Dependencies = CompanyDataManagerInjectable

final class CompanyDashboardViewModel: CompanyDashboardViewModelProtocol, Dependencies {
    //MARK: - Public Properties
    var onOpenCompanyPrediction: ((CompanyModel) -> Void)?
    var onOpenCompanyStocksList: ((CompanyModel) -> Void)?

    private(set) lazy var title: String = company.stockName
    private(set) lazy var companyLogo: UIImage? = company.logo

    //MARK: - Initializer
    init(company: CompanyModel) {
        self.company = company
    }

    //MARK: - Public Methods
    func openCompanyPrediction() {
        onOpenCompanyPrediction?(company)
    }

    func openCompanyStocksList() {
        onOpenCompanyStocksList?(company)
    }

    func fetchCompanyInfo(completion: @escaping ((CompanyInfoNetworkModel) -> Void)) {
        companyDataManager.fetchCompanyInfo(from: company.stockSymbol) { companyInfo in
            completion(companyInfo)
        }
```

```
    }

    func fetchCompanyStock(completion: @escaping ((CompanyStockNetworkModel) -> Void)) {
        companyDataManager.fetchCurrentCompanyStock(from: company.stockSymbol) { companyStock in
            completion(companyStock)
        }
    }

    //MARK: - Private Properties
    private let company: CompanyModel
}


protocol CompanyDashboardViewModelViewable: class {
    var title: String { get }
    var companyLogo: UIImage? { get }
    func openCompanyPrediction()
    func openCompanyStocksList()
    func fetchCompanyInfo(completion: @escaping ((CompanyInfoNetworkModel) -> Void))
    func fetchCompanyStock(completion: @escaping ((CompanyStockNetworkModel) -> Void))
}


protocol CompanyDashboardViewModelCoordinatable: class {
    var onOpenCompanyPrediction: ((CompanyModel) -> Void)? { get set }
    var onOpenCompanyStocksList: ((CompanyModel) -> Void)? { get set }
}


final class CompanyStocksListViewController: UIViewController {
    private enum Configuration {
        static let tableViewRowHeight: CGFloat = 80
    }
    //MARK: - Outlets
    @IBOutlet private weak var tableView: UITableView!
```

```
    @IBOutlet private weak var progressContainer: UIView!

    //MARK: - Initializers
    init(viewModel: CompanyStocksListViewModelViewavable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
    }

    //MARK: - Actions
    @objc func closeStocksListAction() {
        viewModel.closeStocksList()
    }

    //MARK: - Private Properties
    private let viewModel: CompanyStocksListViewModelViewavable
    private let progressViewController = ProgressViewController(viewModel: ProgressViewModel(state:
.loading))
}

//MARK: - UITableViewDataSource
extension CompanyStocksListViewController: UITableViewDataSource {

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return viewModel.stocks.count
    }
```

```
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let stock = viewModel.stocks[indexPath.row]
        let cell: CompanyStockTableViewCell = tableView.dequeueReusableCell(for: indexPath)
        cell.configure(lowStockPrice: stock.lowPrice, highStockPrice: stock.highPrice, openStockPrice:
stock.openPrice, closeStockPrice: stock.closePrice, stockDate: stock.date)
        return cell
    }
}

//MARK: - Setup Methods
private extension CompanyStocksListViewController {

    func setupViewController() {
        setupTableView()
        setupProgressViews()
        setupNavigationBar()
        viewModel.onUpdate = { [weak self] in
            self?.updateContent()
        }
    }

    func setupTableView() {
        tableView.indicatorStyle = .white
        tableView.rowHeight = Configuration.tableViewRowHeight
        tableView.register(CompanyStockTableViewCell.self)
    }

    func setupProgressViews() {
        progressContainer.embedView(progressViewController.view)
        progressContainer.isHidden = false
        progressViewController.startAnimating()
    }

    func setupNavigationBar() {
```

```
        title = viewModel.title

        let rightBarButtonItem = UIBarButtonItem(title: viewModel.closeButtonTitle, style: .plain, target: self,
action: #selector(closeStocksListAction))
        navigationItem.rightBarButtonItem = rightBarButtonItem

        let attributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title3, .font:
UIFont.navigationControllerBarButtonTitle]
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        states.forEach { rightBarButtonItem.setTitleTextAttributes(attributes, for: $0) }
    }
}

//MARK: - Helper Methods
private extension CompanyStocksListViewController {

    func updateContent() {
        tableView.reloadData()
        hideProgressViews()
    }

    func hideProgressViews() {
        progressContainer.isHidden = true
        progressViewController.stopAnimating()
    }
}


private typealias CompanyStocksListViewModelProtocol = CompanyStocksListViewModelCoordinatable &
CompanyStocksListViewModelViewavable
private typealias Dependencies = CompanyDataManagerInjectable & LocalizationInjectable

final class CompanyStocksListViewModel: CompanyStocksListViewModelProtocol, Dependencies {

    //MARK: - Public Properties
```

```
    var onClose: (() -> Void)?
    var onUpdate: (() -> Void)?
    private(set) var stocks: [CompanyStockNetworkModel] = []
    private(set) lazy var title: String = localizationTable.localized(for: "CompanyStocksList_Title")
    private(set) lazy var closeButtonTitle = localizationTable.localized(for: "General_CloseButtonTitle")

    //MARK: - Initializer
    init(company: CompanyModel) {
        self.company = company
        fetchCompanyStocks()
    }

    //MARK: - Public Methods
    func closeStocksList() {
        onClose?()
    }

    //MARK: - Private Properties
    private let company: CompanyModel
    private lazy var localizationTable = localization.tables.other
}

//MARK: - Private Methods
private extension CompanyStocksListViewModel {

    func fetchCompanyStocks() {
        companyDataManager.fetchCompanyStocks(from: company.stockSymbol) { [weak self] stocks in
            self?.stocks = stocks
            self?.onUpdate?()
        }
    }
}


protocol CompanyStocksListViewModelViewavable: class {
```

```
    var onUpdate: (() -> Void)? { get set }
    var title: String { get }
    var closeButtonTitle: String { get }
    var stocks: [CompanyStockNetworkModel] { get }
    func closeStocksList()
}


protocol CompanyStocksListViewModelCoordinatable: class {
    var onClose: (() -> Void)? { get set }
}


final class CompanyMentionsListViewController: UIViewController {
    private enum Configuration {
        static let tableViewEstimatedRowHeight: CGFloat = 60
        static let tableViewSectionHeaderHeight: CGFloat = 60
    }
    //MARK: - Outlets
    @IBOutlet private weak var tableView: UITableView!

    //MARK: - Initializers
    init(viewModel: CompanyMentionsListViewModelViewable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
```

```
    }

    //MARK: - Actions
    @objc func closeMentionsListAction() {
        viewModel.closeMentionsList()
    }

    //MARK: - Private Properties
    private let viewModel: CompanyMentionsListViewModelViewable
    private let selectionFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)
}

//MARK: - UITableViewDataSource
extension CompanyMentionsListViewController: UITableViewDataSource {

    func numberOfSections(in tableView: UITableView) -> Int {
        return viewModel.sections.count
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return mentions(in: section).count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let mention = self.mention(for: indexPath)
        let cell: CompanyMentionTableViewCell = tableView.dequeueReusableCell(for: indexPath)
        cell.configure(author: mention.author, mention: mention.text, creationDate: mention.creationDate,
emotionalPrediction: mention.emotionalPrediction)
        return cell
    }

    func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
        let mentionsSection = viewModel.sections[section]
        let sectionHeader: CompanyMentionListSectionHeaderView =
tableView.dequeueReusableHeaderFooterView()
```

```
        sectionHeader.configure(title: mentionsSection.title)
        return sectionHeader
    }
}

//MARK: - UITableViewDataSource
extension CompanyMentionsListViewController: UITableViewDelegate {

    func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
        return Configuration.tableViewSectionHeaderHeight
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        selectionFeedbackGenerator.impactOccurred()
        guard let mentionURL = mention(for: indexPath).sourceURL else { return }
        viewModel.openMentionSource(mentionURL)
    }
}

//MARK: - Setup Methods
private extension CompanyMentionsListViewController {

    func setupViewController() {
        setupTableView()
        setupNavigationBar()
    }

    func setupTableView() {
        tableView.indicatorStyle = .white
        tableView.rowHeight = UITableView.automaticDimension
        tableView.estimatedRowHeight = Configuration.tableViewEstimatedRowHeight
        tableView.register(CompanyMentionTableViewCell.self)
    }

    func setupNavigationBar() {
```

```
        title = viewModel.title
        let states: [UIControl.State] = [.normal, .highlighted, .focused]
        let rightBarButtonItem = UIBarButtonItem(title: viewModel.closeButtonTitle, style: .plain, target: self,
action: #selector(closeMentionsListAction))
        navigationItem.rightBarButtonItem = rightBarButtonItem
        let rightBarButtonItemAttributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title3, .font:
UIFont.navigationControllerBarButtonTitle]
        states.forEach { rightBarButtonItem.setTitleTextAttributes(rightBarButtonItemAttributes, for: $0) }
        let backBarButtonItem = UIBarButtonItem(title: "Back", style: .plain, target: nil, action: nil)
        navigationItem.backBarButtonItem = backBarButtonItem
        let backBarButtonItemAttributes: [NSAttributedString.Key: Any] = [.foregroundColor: UIColor.title1, .font:
UIFont.navigationControllerBackBarButtonTitle]
        states.forEach { backBarButtonItem.setTitleTextAttributes(backBarButtonItemAttributes, for: $0) }
    }
}

//MARK: - Helper Methods
private extension CompanyMentionsListViewController {

    func mention(for indexPath: IndexPath) -> CompanyMentionProtocol {
        return viewModel.sections[indexPath.section].mentions[indexPath.row]
    }

    func mentions(in section: Int) -> [CompanyMentionProtocol] {
        return viewModel.sections[section].mentions
    }
}


private typealias CompanyMentionsListViewModelProtocol = CompanyMentionsListViewModelCoordinatable
& CompanyMentionsListViewModelViewable
private typealias Dependencies = LocalizationInjectable

final class CompanyMentionsListViewModel: CompanyMentionsListViewModelProtocol, Dependencies {
    //MARK: - Public Properties
```

```
    var onClose: (() -> Void)?
    var onOpenMentionSource: ((URL) -> Void)?

    private(set) lazy var sections: [CompanyMentionsSection] =
makeMentionsSectionsFromMentionsSources()
    private(set) lazy var title: String = localizationTable.localized(for: "CompanyMentionsList_Title")
    private(set) lazy var closeButtonTitle = localizationTable.localized(for: "General_CloseButtonTitle")

    //MARK: - Initializer
    init(company: CompanyModel, mentionsSources: [CompanyMentionsSource]) {
        self.company = company
        self.mentionsSources = mentionsSources
    }

    //MARK: - Public Methods
    func closeMentionsList() {
        onClose?()
    }

    func openMentionSource(_ mentionURL: URL) {
        onOpenMentionSource?(mentionURL)
    }

    //MARK: - Private Properties
    private let company: CompanyModel
    private let mentionsSources: [CompanyMentionsSource]
    private lazy var localizationTable = localization.tables.other
}

//MARK: - Private Methods
private extension CompanyMentionsListViewModel {

    func makeMentionsSectionsFromMentionsSources() -> [CompanyMentionsSection] {
        let sections: [CompanyMentionsSection] = mentionsSources.map { (title: $0.sourceName, mentions:
$0.mentions) }
```

```
        return sections
    }
}


typealias CompanyMentionsSection = (title: String, mentions: [CompanyMentionProtocol])

protocol CompanyMentionsListViewModelViewable: class {
    var title: String { get }
    var closeButtonTitle: String { get }
    var sections: [CompanyMentionsSection] { get }
    func closeMentionsList()
    func openMentionSource(_ mentionURL: URL)
}


protocol CompanyMentionsListViewModelCoordinatable: class {
    var onClose: (() -> Void)? { get set }
    var onOpenMentionSource: ((URL) -> Void)? { get set }
}


final class CompanyMentionSourceViewController: UIViewController {
    //MARK: - Outlets
    @IBOutlet private weak var webView: WKWebView!
    @IBOutlet private weak var progressContainer: UIView!

    //MARK: - Initializers
    init(viewModel: CompanyMentionSourceViewModelViewable) {
        self.viewModel = viewModel
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
```

```
    }

    //MARK: - UIViewController
    override func viewDidLoad() {
        super.viewDidLoad()
        setupViewController()
        loadWebViewContent()
    }

    //MARK: - Private Properties
    private let viewModel: CompanyMentionSourceViewModelViewable
    private let progressViewController = ProgressViewController(viewModel: ProgressViewModel(state:
.loading))
}

//MARK: - Setup Methods
private extension CompanyMentionSourceViewController {

    func setupViewController() {
        setupWebView()
        setupProgressViews()
        setupNavigationBar()
    }

    func setupWebView() {
        webView.navigationDelegate = self
    }

    func setupProgressViews() {
        progressContainer.embedView(progressViewController.view)
        progressContainer.isHidden = false
        progressViewController.startAnimating()
    }

    func setupNavigationBar() {
```

```
        title = viewModel.title
    }
}

//MARK: - WKNavigationDelegate
extension CompanyMentionSourceViewController: WKNavigationDelegate {

    func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
        hideLoadingViews()
    }
}

//MARK: - WebView Methods
private extension CompanyMentionSourceViewController {

    func loadWebViewContent() {
        let urlRequest = URLRequest(url: viewModel.mentionURL)
        webView.load(urlRequest)
    }
}

//MARK: - Helper Methods
private extension CompanyMentionSourceViewController {

    func hideLoadingViews() {
        progressContainer.isHidden = true
        progressViewController.stopAnimating()
    }
}


private typealias CompanyMentionSourceViewModelProtocol =
CompanyMentionSourceViewModelCoordinatable & CompanyMentionSourceViewModelViewable
private typealias Dependencies = LocalizationInjectable
```

```
final class CompanyMentionSourceViewModel: CompanyMentionSourceViewModelProtocol, Dependencies {
    //MARK: - Public Properties
    let mentionURL: URL
    private(set) lazy var title: String = localizationTable.localized(for: "CompanyMentionSource_Title")

    //MARK: - Initializer
    init(mentionURL: URL) {
        self.mentionURL = mentionURL
    }

    //MARK: - Private Properties
    private lazy var localizationTable = localization.tables.other
}


protocol CompanyMentionSourceViewModelViewable {
    var title: String { get }
    var mentionURL: URL { get }
}


protocol CompanyMentionSourceViewModelCoordinatable {
    /* empty */
}


protocol Coordinator {
    func start()
}

protocol NavigationCoordinator: Coordinator {
    var navigationController: UINavigationController { get }
}

protocol PageCoordinator: Coordinator {
```

```
    var pageViewController: UIPageViewController { get }
}


private typealias Dependencies = PreferencesInjectable

final class AppCoordinator: Coordinator, Dependencies {
    enum ChildCoordinator {
        case main
        case tutorial
    }

    //MARK: - Initializer
    init(window: UIWindow) {
        self.window = window
        window.rootViewController = navigationController
        window.makeKeyAndVisible()
    }

    //MARK: - Public Methods
    func start() {
        if isFirstLaunch {
            startTutorialCoordinator()
        } else {
            startMainCoordinator()
        }
        guard let rootView = window.rootViewController?.view else { return }
        runSplashAnimation(from: rootView)
    }

    //MARK: - Private Properties
    private let window: UIWindow
    private let navigationController = DefaultNavigationController()
    private var childCoordinators = [ChildCoordinator: Coordinator]()
}
```

```
//MARK: - Start Coordinators
private extension AppCoordinator {

    func startTutorialCoordinator() {
        let coordinator = (childCoordinators[.tutorial] as? TutorialCoordinator) ??
TutorialCoordinator(navigationController: navigationController)
        coordinator.onComplete = { [unowned self] in
            self.childCoordinators[.tutorial] = nil
            self.startMainCoordinator()
        }
        childCoordinators[.tutorial] = coordinator
        coordinator.start()
    }

    func startMainCoordinator() {
        let coordinator = (childCoordinators[.main] as? MainCoordinator) ??
MainCoordinator(navigationController: navigationController)
        childCoordinators[.main] = coordinator
        coordinator.start()
    }
}

//MARK: - Helpers
private extension AppCoordinator {

    var isFirstLaunch: Bool {
        return preferences.launchCount == 1
    }

    func runSplashAnimation(from view: UIView) {
        let splashViewModel = SplashViewModel()
        let splashViewController = SplashViewController(viewModel: splashViewModel)
        splashViewController.view.frame = view.bounds
```

```
        view.addSubview(splashViewController.view)
        view.bringSubviewToFront(splashViewController.view)

        splashViewController.runAnimation()
        splashViewModel.onCompleteAnimation = { [weak splashViewController] in
            splashViewController?.view.removeFromSuperview()
        }
    }
}


final class MainCoordinator: NavigationCoordinator {
    //MARK: - Public Properties
    var onComplete: (() -> Void)?
    let navigationController: UINavigationController

    //MARK: - Initializer
    init(navigationController: UINavigationController) {
        self.navigationController = navigationController
    }

    //MARK: - Public Methods
    func start() {
        presentCompaniesListScreen()
    }

    //MARK: - Private Properties
    private var companyMentionsListNavigationController: UINavigationController?
}

//MARK: - Present Screens
private extension MainCoordinator {

    func presentCompaniesListScreen() {
        let companiesListViewModel = CompaniesListViewModel()
```

```
        companiesListViewModel.onOpenCompanyDashboard = { [weak self] company in
            self?.presentCompanyDashboardScreen(with: company)
        }
        companiesListViewModel.onOpenSettings = { [weak self] in
            self?.presentSettingsScreen()
        }

        let companiesListViewController = CompaniesListViewController(viewModel: companiesListViewModel)
        navigationController.setViewControllers([companiesListViewController], animated: true)
    }

    func presentCompanyDashboardScreen(with company: CompanyModel) {
        let companyDashboardViewModel = CompanyDashboardViewModel(company: company)
        companyDashboardViewModel.onOpenCompanyPrediction = { [weak self] company in
            self?.presentCompanyPredictionScreen(with: company)
        }
        companyDashboardViewModel.onOpenCompanyStocksList = { [weak self] company in
            self?.presentCompanyStocksListScreen(with: company)
        }

        let companyDashboardViewController = CompanyDashboardViewController(viewModel:
companyDashboardViewModel)
        navigationController.pushViewController(companyDashboardViewController, animated: true)
    }

    func presentSettingsScreen() {
        let settingsViewModel = SettingsViewModel()
        settingsViewModel.onClose = { [weak self] in
            self?.navigationController.dismiss(animated: true)
        }
        let settingsViewController = SettingsViewController(viewModel: settingsViewModel)
        let innerNavigationController = DefaultNavigationController(rootViewController: settingsViewController)
        navigationController.present(innerNavigationController, animated: true)
    }
```

```
    func presentCompanyPredictionScreen(with company: CompanyModel) {
        let companyPredictionViewModel = CompanyPredictionViewModel(company: company)
        companyPredictionViewModel.onOpenCompanyMentionsList = { [weak self] company,
mentionsSources in
            self?.presentCompanyMentionsListScreen(with: company, mentionsSources: mentionsSources)
        }
        let companyPredictionViewController = CompanyPredictionViewController(viewModel:
companyPredictionViewModel)
        navigationController.pushViewController(companyPredictionViewController, animated: true)
    }

    func presentCompanyStocksListScreen(with company: CompanyModel) {
        let companyStocksListViewModel = CompanyStocksListViewModel(company: company)
        companyStocksListViewModel.onClose = { [weak self] in
            self?.navigationController.dismiss(animated: true)
        }
        let companyStocksListViewController = CompanyStocksListViewController(viewModel:
companyStocksListViewModel)
        let innerNavigationController = DefaultNavigationController(rootViewController:
companyStocksListViewController)
        navigationController.present(innerNavigationController, animated: true)
    }

    func presentCompanyMentionsListScreen(with company: CompanyModel, mentionsSources:
[CompanyMentionsSource]) {
        let companyMentionsListViewModel = CompanyMentionsListViewModel(company: company,
mentionsSources: mentionsSources)
        companyMentionsListViewModel.onOpenMentionSource = { [weak self] mentionURL in
            self?.presentCompanyMentionSourceScreen(with: mentionURL)
        }
        companyMentionsListViewModel.onClose = { [weak self] in
            self?.navigationController.dismiss(animated: true)
        }
        let companyMentionsListViewController = CompanyMentionsListViewController(viewModel:
companyMentionsListViewModel)
```

```
        let companyMentionsListNavigationController = DefaultNavigationController(rootViewController:
companyMentionsListViewController)
        self.companyMentionsListNavigationController = companyMentionsListNavigationController
        navigationController.present(companyMentionsListNavigationController, animated: true)
    }

    func presentCompanyMentionSourceScreen(with mentionURL: URL) {
        guard let companyMentionsListNavigationController = companyMentionsListNavigationController else {
return }
        let companyMentionSourceViewModel = CompanyMentionSourceViewModel(mentionURL:
mentionURL)
        let companyMentionSourceViewController = CompanyMentionSourceViewController(viewModel:
companyMentionSourceViewModel)
companyMentionsListNavigationController.pushViewController(companyMentionSourceViewController,
animated: true)
    }
}


final class TutorialCoordinator: NSObject, NavigationCoordinator, PageCoordinator {
    //MARK: - Public Properties
    var onComplete: (() -> Void)?
    let pageViewController: UIPageViewController
    let navigationController: UINavigationController

    //MARK: - Initializer
    init(navigationController: UINavigationController) {
        let pageViewController = UIPageViewController(transitionStyle: .scroll, navigationOrientation:
.horizontal, options: nil)
        self.pageViewController = pageViewController
        let tutorialRootViewModel = TutorialRootViewModel()
        self.tutorialRootViewModel = tutorialRootViewModel
        self.navigationController = navigationController
        super.init()
```

```
        setupPageControl()
        setupPageViewController()
        setupTutorialRootViewModel()
    }

    //MARK: - Public Methods
    func start() {
        presentTutorialScreen()
    }

    //MARK: - Private Properties
    private let tutorialRootViewModel: TutorialRootViewModelProtocol
    private var currentPageViewControllerIndex: Int = 0
    private lazy var pageViewControllersCount: Int = pageViewControllers.count
    private lazy var pageViewControllers: [UIViewController] =
tutorialRootViewModel.pageViewModels.map(TutorialPageViewController.init)
}

//MARK: - Present Methods
private extension TutorialCoordinator {

    func presentTutorialScreen() {
        let rootViewController = TutorialRootViewController(viewModel: tutorialRootViewModel)
        navigationController.setViewControllers([rootViewController], animated: true)
        rootViewController.addChild(pageViewController)
        rootViewController.view.embedView(pageViewController.view, insets: UIEdgeInsets(top: 100, left: 0,
bottom: 120, right: 0))
        guard let firstPage = pageViewControllers.first else { return }
        pageViewController.setViewControllers([firstPage], direction: .forward, animated: true, completion: nil)
    }
}

//MARK: - UIPageViewControllerDataSource
extension TutorialCoordinator: UIPageViewControllerDataSource {
```

```
    func pageViewController(_ pageViewController: UIPageViewController, viewControllerBefore
viewController: UIViewController) -> UIViewController? {
        guard let pageViewControllerIndex = pageViewControllers.firstIndex(of: viewController) else { return nil }
        let previousIndex = pageViewControllerIndex - 1
        guard previousIndex >= 0, pageViewControllers.count > previousIndex else { return nil }
        return pageViewControllers[previousIndex]
    }

    func pageViewController(_ pageViewController: UIPageViewController, viewControllerAfter viewController:
UIViewController) -> UIViewController? {
        guard let pageViewControllerIndex = pageViewControllers.firstIndex(of: viewController) else { return nil }
        let nextIndex = pageViewControllerIndex + 1
        let pageViewControllersCount = pageViewControllers.count
        guard pageViewControllersCount != nextIndex, pageViewControllersCount > nextIndex else { return nil }
        return pageViewControllers[nextIndex]
    }

    func presentationCount(for pageViewController: UIPageViewController) -> Int {
        return pageViewControllers.count
    }

    func presentationIndex(for pageViewController: UIPageViewController) -> Int {
        return currentPageViewControllerIndex
    }
}

//MARK: - UIPageViewControllerDelegate
extension TutorialCoordinator: UIPageViewControllerDelegate {

    func pageViewController(_ pageViewController: UIPageViewController, didFinishAnimating finished: Bool,
previousViewControllers: [UIViewController], transitionCompleted completed: Bool) {
        guard
            completed,
            let currentPageViewController = pageViewController.viewControllers?.first,
```

```
        let currentPageViewControllerIndex = pageViewControllers.firstIndex(of: currentPageViewController)
else { return }
        self.currentPageViewControllerIndex = currentPageViewControllerIndex
        updateTutorialRootViewModelState()
    }
}

//MARK: - Private Methods
private extension TutorialCoordinator {

    func setupPageControl() {
        let pageControl = UIPageControl.appearance(whenContainedInInstancesOf:
[TutorialRootViewController.self])
        pageControl.currentPage = currentPageViewControllerIndex
        pageControl.numberOfPages = pageViewControllers.count
    }

    func setupPageViewController() {
        pageViewController.delegate = self
        pageViewController.dataSource = self
    }

    func setupTutorialRootViewModel() {
        tutorialRootViewModel.onPerformAction = { [weak self] in
            guard let self = self else { return }
            if let nextViewController = self.pageViewController.nextPageViewController, let
nextViewControllerIndex = self.pageViewControllers.firstIndex(of: nextViewController) {
                self.currentPageViewControllerIndex = nextViewControllerIndex
                self.updateTutorialRootViewModelState()
                self.pageViewController.goToNextPage()
            } else {
                self.onComplete?()
            }
        }
    }
```

```
    func updateTutorialRootViewModelState() {
        tutorialRootViewModel.state = (currentPageViewControllerIndex == pageViewControllersCount - 1) ?
.begin : .continue
    }
}
```