НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Статистична модель прогнозування світового ринку нафти»

Виконав:
студент IV курсу, групи КА-66
Сніжко Андрій Сергійович

_____

Керівник:
доцент, д.т.н. Недашківська Н.І.

_____

Консультант з економічного розділу:
доцент Шевчук О.А.

_____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є.

_____

Рецензент:
доцент кафедри СП ІПСА, к.т.н. Гіоргізова-Гай В.Ш.

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«25» травня 2020 р.

ЗАВДАННЯ
на дипломну роботу студенту
Сніжку Андрію Сергійовичу

1. Тема роботи «Статистична модель прогнозування світового ринку нафти», керівник роботи Недашківська Надія Іванівна, доцент кафедри ММСА, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи: вибірка цін на нафту з 1986 року.

4. Зміст роботи: аналітичний та історичний огляд ринку нафти, теоретичні відомості та основи регресійного аналізу, основи інтелектуального аналізу даних з теми дерев рішень, прогнозування ціни нафти на основі вхідних даних, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) дерева рішень, алгоритм розбиття «CART», ансамбль дерев рішень, принцип роботи програмного продукту, результати моделей.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А.		

7. Дата видачі завдання 27 березня 2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	13.04.2020	
2.	Підготовка першого розділу.	21.04.2020	
3.	Підготовка другого розділу.	1.05.2020	
4.	Розробка програмного продукту.	19.05.2020	
5.	Підготовка третього розділу	25.05.2020	
6.	Підготовка економічної частини	28.05.2020	
7.	Оформлення розділів відповідно до нормоконтролю.	01.06.2020	
8.	Підготовка презентації доповіді.	30.05.2020	
9.	Оформлення дипломної роботи.	02.06.2020	

Студент Андрій СНІЖКО

Керівник Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота містить: 125 с., 7 табл., 53 рис., 2 дод. та 25 джерел.

РЕГРЕСІЯ, ДЕРЕВА РІШЕНЬ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, МАШИННЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ ЦІНИ, РИНОК НАФТИ.

Об’єктом дослідження є вибірка цін на нафту з 1986 року.

Предметом дослідження є методи інтелектуального аналізу даних на основі регресії з використанням дерев рішень.

Програмною мовою була обрана Python.

В даній роботі проведено дослідження ринку нафти. Для побудови моделей були використані дерева рішень та методи машинного навчання. Прогнозування було виконано на основі даних про зміну цін на нафту.

При виконанні роботи було встановлено два методи, що дають найкращі результати які достатньо близькі до реальних. Напрямок розвитку роботи є в розширенні функціоналу та зменшенні похибки прогнозування ціни. Планується додати аналіз ринку палива та встановити залежність між цінами на паливо та нафту.

ABSTRACT

Thesis: 125 p., 7 tabl., 53 fig., 2 add. And 25 references.

REGRESSION, DECISION TREES, INTELLIGENT DATA ANALYSIS, MACHINE LEARNING, PRICE FORECAST, OIL MARKET.

The object of research is the selection of oil prices since 1986.

The subject of research covers methods of intelligent data analysis based on regression using decision trees.

Programming language Python.

This work is dedicated to the analysis of the oil market. Decision trees and machine learning methods were used to build the models. The forecast was conducted using the oil prices data. The prospects of development of the work depend on the expansion of the functionality and reduction the drawbacks in the price forecasting. It is planned to add the fuel market analysis to it and monitor the dependence between the fuel and oil prices.

ЗМІСТ

ВСТУП 8
ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 10
РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД РИНКУ НАФТИ 11
1.1 Історія формування ринку нафти 11
1.2 Вплив ОПЕК на ринок нафти 17
1.3 Формування ціни на нафту 19
1.4 Модель для прогнозування ціни на нафту 24
1.5 Висновки 25
РОЗДІЛ 2 РЕГРЕСІЙНИЙ АНАЛІЗ. ТЕОРІЯ ТА ВІДОМОСТІ 26
2.1 Вступ 26
2.2 Види регресії 28
2.2.1 Лінійна регресія 28
2.2.2 Поліноміальна регресія 30
2.2.3 Логістична регресія 32
2.2.4 Квантільна регресія 33
2.2.5 Гребнева регресія 34
2.2.6 Метод регресії «Лассо» 35
2.2.7 Регресія «ElasticNet» 37
2.2.8 Байєсова регресія 37
2.2.9 Логічна регресія 38
2.3 Підсумки 39
2.4 Дерева Рішень «Decision Trees» 39
2.4.1 Основні терміни та поняття 41
2.4.2 Представлення алгоритму у вигляді дерева 41
2.4.3 Метод побудови дерева рішень. Алгоритм розбиття 44
2.4.4 Метод побудови дерева рішень. Критерій зупинки 47
2.5 Ансамбль дерев рішень 48
2.5.1 Random Forest 50
2.5.2 Extremely Randomized Trees 52
2.5.3 Gradient Boosting 52
2.6 Висновки 53
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ 54
3.1 Вступ 54
3.2 Бібліотеки 54

3.3 Експерименти 55
Далі, на таблиці 3.1, наведено таблицю значень метрик моделей. 55
3.4 Висновки 70
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ 71
4.1 Постановка задачі 71
4.2 Обґрунтування функцій програмного продукту 71
4.3 Економічний аналіз варіантів розробки 78
4.5 Висновки до розділу 82
ВИСНОВКИ 84
ПЕРЕЛІК ПОСИЛАНЬ 85
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ 88
ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 107

ВСТУП

Виробництво та споживання нафтових ресурсів, що прийшли на початку століття на заміну вугіллю та деревині, зростає з кожним роком. Нині контроль над паливно-енергетичними ресурсами має великий вплив на економічний та політичний стан країни на світовій арені в цілому.

Першими нафтопродуктами, що користувалися великим попитом на ринку були керосин та масло для ламп. З початку, знаходження нафти не було чимось важливим, а навіть навпаки – чимось неприємним, адже люди проводили роботи по видобутку води або солі. Згодом, дослідивши цю речовину люди прийшли до висновку, що видобуток цієї корисної копалини матиме дуже важливий вплив на розвиток людства в цілому. Видобуток нафти бере початок у 1847 році, коли в Азербайджані була пробурена перша свердловина. Вже через дванадцять років у США почався розвиток справжньої індустрії.

Прийнято вважати, що сучасна нафтова ера бере свій початок у 1901 році, в той день, коли у південно-східному Техасі була пробурена перша в світі комерційна свердловина здатна для масового виробництва. Ділянка Spindletop видобувала понад 100000 барелів нафти в день, що було більшим ніж вся нафтовидобувна індустрія в Сполучених Штатах Америки разом узята.

Нафта є товаром, що має світовий попит. Від неї залежить велика кількість індустрій, а отже коливання на її ціну можуть сильно вплинути на економіку країн. Два основних чинники, що впливають на ціну нафти:

- настрій на ринку;
- попит та пропозиція;

Ключовим фактором при визначенні цін на нафту є настрій на ринку. Щоб зрозуміти, що це таке, треба зрозуміти для себе два ключових моменти. Просте переконання в тому, що попит на нафту різко зросте в якийсь момент в майбутньому, може привести до різкого зростання цін на нафту в даний

час, оскільки спекулянти і хеджери однаково скуповують ф'ючерсні контракти на нафту. З цього випливає і зворотне. Просте переконання в тому, що попит на нафту в якийсь момент в майбутньому зменшиться, може привести до різкого зниження цін в даний час, оскільки нафтові ф'ючерси продаються (можливо, також продаються по коротким цінами), що означає, що ціни можуть залежати лише від ринкової психології. Також неможна забувати про різни ризики, та чинники на світову економіку. Наприклад, на весні 2020 року ціни на нафту впали через спалах пандемії COVID-19. А це, в свою чергу, привело до того, що обсяги виробництва нафти прийшлося знизити до рекордного мінімуму за останні 20 років.

Концепція попиту та пропозиції виглядає досить простою. При збільшенні попиту ціна повинна зростати, при збільшенні пропозиції – спадати. Але на справді все відбувається трішки інакше. Ціну на нафту встановлює ф’ючерсний ринок. Ф’ючерсний контракт – зобов’язуюча угода для покупця і продавця, що дає право на купівлю нафти по вказаній ціні за барель в заздалегідь визначену дату в майбутньому. За цим контрактом обидві сторони мають виконати свої зобов’язання до визначеної дати. Є два типи ф’ючерсних трейдерів:

- спекулянти
- хеджери

Спекулянт – людина, що не збирається купувати нафту. Її ціль – заробити гроші на вгадуванні майбутнього напрямку ціни на ринку. В свою чергу прикладом хеджера може бути якась авіакомпанія, що хоче захистити себе від потенційного зростання цін.

Задача прогнозування цін на ринку нафту є досить цікавою та актуальною досить довгий час. На мою думку метод регресійного аналізу через побудову дерев рішень якнайкраще підходить для вирішення цієї задачі. Отже дана атестаційна бакалаврська робота присвячена темі інтелектуального аналізу даних, а саме дослідженню дерев рішень на прикладі прогнозування цін на ринку нафти.

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ОПЕК – організація країн-експортерів нафти;

ФВА – функціонально-вартісний аналіз;

ПП – програмний продукт;

МН – машинне навчання;

СПМ – середньоквадратична помилка моделі.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД РИНКУ НАФТИ

1.1 Історія формування ринку нафти

Протягом всієї історії людства енергія була ключовим фактором підвищення рівня життя. Щоб вижити в аграрну епоху, люди спалювали дрова для тепла і приготування їжі. Крім використання в якості будівельного матеріалу, дерево залишалося головним світовим паливом протягом століть. Винахід першого сучасного парового двигуна на початку 18-го століття ознаменувало перехід від аграрної до індустріальної економіки. Парові двигуни могли працювати на деревині або вугіллі, але вугілля швидко стало кращим паливом, що дозволило значно збільшити масштаби індустріалізації. Півтонни вугілля виробляли в чотири рази більше енергії, ніж така ж кількість деревини, і були дешевші у виробництві і, незважаючи на велику масу, легше розподілялися. Вугільні паровози значно скоротили час і вартість внутрішніх перевезень, в той час як пароплави перетнули океани. Машини, що працюють на вугіллі, дозволили домогтися проривів в продуктивності при одночасному зниженні фізичної праці. З настанням 20-го століття екологічні проблеми і нові технології привели до ще одного переходу джерела енергії з вугілля на нафту.

Перша нафта була фактично виявлена китайцями в 600 році до нашої ери і транспортувалася в трубопроводах з бамбука. Проте, відкриття полковником Дрейком нафти в Пенсільванії в 1859 році і відкриття родовища Spindletop в Техасі в 1901 році заклали основу для нової нафтової економіки. Нафта була набагато більш технологічною і гнучкою, ніж вугілля. Крім того, гас, який був спочатку очищено від нафти, забезпечив надійну і відносно недорогу альтернативу «вугільним масел» і китовим масел для заправки ламп. Більшість інших продуктів були викинуті. З технологічним проривом 20-го століття нафту стала кращим джерелом енергії. Ключовими факторами цієї трансформації були електрична лампочка і автомобіль. Володіння

автомобілями і попит на електроенергію росли в геометричній прогресії, а разом з ними і попит на нафту. До 1919 року продажі бензину перевищили продажі гасу. Нафтові кораблі, вантажівки і танки, а також військові літаки в Першій світовій війні довели роль нафти як не тільки стратегічного джерела енергії, а й найважливішого військового активу. До 1920-х років природний газ, який видобувається разом з нафтою, спалювався (або спалювався) в якості побічного продукту. Згодом газ став використовуватися в якості палива для промислового і житлового опалення та енергетики. Коли його цінність була усвідомлена, природний газ став цінним продуктом сам по собі. Перша нафта була фактично виявлена китайцями в 600 році до нашої ери. і транспортується в трубопроводах з бамбука[1]. Проте, відкриття полковника Дрейка нафти в Пенсільванії в 1859 році і відкриття Spindletop в Техасі в 1901 році заклали основу для нової нафтової економіки. Нафта була набагато більш технологічною і гнучкою, ніж вугілля. Крім того, гас, який був спочатку очищено від нафти, забезпечив надійну і відносно недорогу альтернативу «вугільним масел» і китовим масел для заправки ламп. Більшість інших продуктів були викинуті. З технологічним проривом 20-го століття нафту стала кращим джерелом енергії. Ключовими факторами цієї трансформації були електрична лампочка і автомобіль. Володіння автомобілями і попит на електроенергію росли в геометричній прогресії, а разом з ними і попит на нафту. До 1919 року продажі бензину перевищили продажі гасу. Нафтові кораблі, вантажівки і танки, а також військові літаки в Першій світовій війні довели роль нафти як не тільки стратегічного джерела енергії, а й найважливішого військового активу. До 1920-х років природний газ, який видобувається разом з нафтою, спалювався (або спалювався) в якості побічного продукту. Згодом газ став використовуватися в якості палива для промислового і житлового опалення та енергетики. Коли його цінність була усвідомлена, природний газ став цінним продуктом сам по собі[2].

Джон Д. Рокфеллер, який почав свою кар'єру в нафтопереробці, став першим «бароном» галузі в 1865 році, коли він заснував Standard Oil Company. До 1879 року Standard Oil контролювала не тільки 90% переробних потужностей Америки, але і її трубопроводи і системи збору. До кінця 19-го століття домінування Standard Oil розширилася і включає розвідку, видобуток і маркетинг. Сьогодні ExxonMobil є наступником Standard Oil. Поки Рокфеллер будував свою імперію в США, сім'ї Нобеля і Ротшильдів боролися за контроль над видобутком і переробкою нафтових багатств Росії. У пошуках глобальної транспортної мережі для продажу свого гасу Ротшильди замовили перші нафтові танкери у британського трейдера Маркуса Самуеля. Перший з цих танкерів був названий Murex, за типом морської раковини, і став флагманом Shell Transport and Trading, який Самуїл сформував у 1897 році. Компанія Royal Dutch Petroleum розпочала свою діяльність в Голландській Ост-Індії в кінці 1800-х років, а до 1892 році об'єднала виробничі, конвеєрні та переробні виробництва[1]. У 1907 році Royal Dutch і Shell Transport and Trading домовилися про створення Royal Dutch Shell Group. Також в 1907 році відкриття нафти в Ірані колишнім британським золотодобытчиком і близькосхідним шахом призвело до створення англо-перської нафтової компанії. Британське уряд придбав 51% компанії в 1914 році, щоб забезпечити достатню кількість нафти для Королівського флоту в роки, що передували Першій світовій війні. Компанія стала British Petroleum в 1954 році і в даний час є BP. Сьогодні ці три компанії - ExxonMobil, Shell і BP - вважаються початковими «супер-мажорами». В Сполучених Штатах у 1901 році відкриття родовища Spindletop в Техасі призвело до появи таких компаній, як Gulf Oil, Texaco та інших. Домінування Сполучених Штатів у цю епоху було проілюстровано тим фактом, що незалежно від того, де в світі видобувалася нафта, її ціна була фіксованою і дорівнює ціні в Мексиканській затоці. Починаючи з Першої світової війни, нафта стала стратегічним джерелом енергії і величезним геополітичним призом. У 1930-х роках Gulf Oil, BP, Chevron

Texaco і були залучені в концесії, які зробили великі відкриття в Кувейті, Саудівській Аравії та Лівії. На основі цих відкриттів було сформовано картель з семи компаній, які контролювали світової нафтогазовий бізнес протягом більшої частини двадцятого століття. Відомі як Сім Сестер, вони включали в себе: Exxon (спочатку Standard Oil), Royal Dutch / Shell, BP, Mobil, Texaco, Gulf і Chevron. Джон Д. Рокфеллер, який почав свою кар'єру в нафтопереробці, став першим «бароном» галузі в 1865 році, коли він заснував Standard Oil Company. До 1879 року Standard Oil контролювала не тільки 90% переробних потужностей Америки, але і її трубопроводи і системи збору. До кінця 19-го століття домінування Standard Oil розширилася і включає розвідку, видобуток і маркетинг. Сьогодні ExxonMobil є наступником Standard Oil. Поки Рокфеллер будував свою імперію в США, сім'ї Нобеля і Ротшильдів боролися за контроль над видобутком і переробкою нафтових багатств Росії. У пошуках глобальної транспортної мережі для продажу свого гасу Ротшильди замовили перші нафтові танкери у британського трейдера Маркуса Самуеля. Перший з цих танкерів був названий Murex, за типом морської раковини, і став флагманом Shell Transport and Trading, який Самуїл сформував у 1897 році. Компанія Royal Dutch Petroleum розпочала свою діяльність в Голландській Ост-Індії в кінці 1800-х років, а до 1892 році об'єднала виробничі, конвеєрні та переробні виробництва. У 1907 році Royal Dutch і Shell Transport and Trading домовилися про створення Royal Dutch Shell Group[3]. Також в 1907 році відкриття нафти в Ірані колишнім британським золотодобытчиком і близькосхідним шахом призвело до створення англо-перської нафтової компанії. Британське уряд придбав 51% компанії в 1914 році, щоб забезпечити достатню кількість нафти для Королівського флоту в роки, що передували Першій світовій війні. Компанія стала British Petroleum в 1954 році і в даний час є BP. Сьогодні ці три компанії - ExxonMobil, Shell і BP - вважаються початковими «супер-мажорами». В Сполучених Штатах у 1901 році відкриття родовища Spindletop в Техасі призвело до появи таких

компаній, як Gulf Oil, Texaco та інших. Домінування Сполучених Штатів у цю епоху було проілюстровано тим фактом, що незалежно від того, де в світі видобувалася нафта, її ціна була фіксованою і дорівнює ціні в Мексиканській затоці. Починаючи з Першої світової війни, нафта стала стратегічним джерелом енергії і величезним геополітичним призом. У 1930-х роках Gulf Oil, BP, Chevron Texaco і були залучені в концесії, які зробили великі відкриття в Кувейті, Саудівській Аравії та Лівії. На основі цих відкриттів було сформовано картель з семи компаній, які контролювали світової нафтогазовий бізнес протягом більшої частини двадцятого століття. Відомі як Сім Сестер, вони включали в себе: Exxon (спочатку Standard Oil), Royal Dutch / Shell, BP, Mobil, Texaco, Gulf і Chevron.

Починаючи з 1950-х років, відбулися численні зрушення, в результаті яких контроль над видобутком нафти та газу і ціноутворенням перемістився з «Великої нафти» і країн-споживачів нафти в країни-виробники нафти. Уряди багатьох нафтовидобувних країн, особливо на Близькому Сході і в Південній Америці, розглядали інтегровані нафтові компанії (МНК), що працюють там, як інструменти своїх країн походження (зазвичай США або європейських країн). Як з економічних, так і з геополітичних причин лідери країн-виробників почали стверджувати свою владу в контролі за запасами нафти і газу своїх країн (і пов'язаного з ними багатства). Щоб позначити свою знову знайдену влада, в 1960 році уряду Венесуели, Саудівської Аравії, Кувейту, Іраку та Ірану створили Організацію країн-експортерів нафти (ОПЕК) з метою ведення переговорів з МНК з питань видобутку нафти, цін на нафту і майбутні концесійні права. ОПЕК справила незначний вплив протягом першого десятиліття свого існування. Ситуація змінилася на початку 1970-х років із злиттям зростаючого попиту на енергоносії, переглядом умов ведення бізнесу в Лівії Муаммаром Каддафі та арабо-ізраїльською війною. Ця діаграма, частка ОПЕК у світових запасах нафти, ілюструє широту і розподіл запасів нафти ОПЕК. ОПЕК являє собою значну політичну і економічну силу. За їх оцінками, 81% запасів нафти в світі належить їх

членам. Зазначимо, що Саудівська Аравія володіє більшістю резервів ОПЕК, за нею слідують Іран і Венесуела. За межами ОПЕК є й інші великі запаси нафти, включаючи Північне море (контрольоване Великобританією, Норвегією, Данією, Німеччиною, Нідерландами), канадські нафтові піски і глибоководні запаси за межами Бразилії і в Мексиканській затоці. ОПЕК, що базується у Відні, була створена в першу чергу у відповідь на зусилля західних нафтових компаній щодо зниження цін на нафту. ОПЕК дозволяє нафтовидобувних країн гарантувати свої доходи шляхом координації політики та цін. Членство в ОПЕК дає країні престиж в очах світової спільноти. Історично склалося так, що США розглядали ОПЕК як загрозу для поставок дешевої енергії, оскільки картель собі на втіху може встановлювати високі ціни на нафту на світовому ринку. Крім того, нинішня політика США щодо зниження залежності від близькосхідної нафти, на якій домінує ОПЕК, може створити дипломатичні проблеми з тими країнами, які пов'язані з інтересами США. Застереження з участю ОПЕК полягає в тому, що країни-члени не можуть встановлювати індивідуальні квоти на видобуток. Це може бути проблематично, тому що політичні інтереси і економічні міркування сильно розрізняються в різних країнах. В кінцевому рахунку, сила ОПЕК не тільки змістила контроль виробництва і ціноутворення з західних МНК в країни-виробники, а й ознаменувала початок сьогоднішньої ери Національної нафтової компанії (ННК).

З технологічним проривом 20-го століття нафту стала кращим джерелом енергії. Ключовими факторами цієї трансформації були електрична лампочка і автомобіль. Володіння автомобілями і попит на електроенергію росли в геометричній прогресії, а разом з ними і попит на нафту. Сила ОПЕК не тільки змістила контроль виробництва і ціноутворення з західних МНК в країни-виробники, але також ознаменувала початок сьогоднішньої ери Національної нафтової компанії (ННК). В даний час ННК контролюють 77% обсягу світового видобутку нафти і газу, ніж раніше домінуючі МНК. Нова ера нетрадиційних нафти і газу зміщує владу від

ОПЕК та інших експортерів, оскільки країни дивляться на внутрішнє виробництво і енергетичну незалежність. Технологічні прориви в гідророзриві пласта, горизонтальному бурінні і глибоководному видобутку відкривають потенціал для величезних запасів в нових областях[4].

1.2 Вплив ОПЕК на ринок нафти

Багато з найбільших нафтовидобувних країн світу входять в картель, відомий як Організація країн-експортерів нафти (ОПЕК). Сьогодні членами ОПЕК є: Алжир, Ангола, Еквадор, Іран, Ірак, Кувейт, Лівія, Нігерія, Катар, Саудівська Аравія, Об'єднані Арабські Емірати і Венесуела. Станом на 2016 рік ОПЕК вступила в союз з іншими провідними країнами-експортерами нафти, що не входять в ОПЕК, щоб сформувати більш потужну структуру, що має описовий прізвисько ОПЕК + (ОПЕК Плюс). Мета полягає в тому, щоб встановити контроль над ціною дорогоцінного викопного палива, відомого як сира нафта. ОПЕК + контролює понад 50 відсотків світових поставок нафти і близько 90 відсотків доведених запасів нафти. У короткостроковій перспективі ОПЕК + робить істотний вплив на ціну нафти. У довгостроковій перспективі його здатність впливати на ціну на нафту знижується, в першу чергу тому, що окремі країни мають інші стимули, ніж ОПЕК + в цілому.

Як картель, країни-члени ОПЕК + колективно домовляються про те, скільки нафти добувати, що безпосередньо впливає на готові постачання сирої нафти на світовий ринок у будь-який момент часу. В результаті ОПЕК + чинить вплив на світову ринкову ціну на нафту і, зрозуміло, прагне підтримувати її відносно високою, щоб максимізувати прибутковість.

Наприклад, якщо країни ОПЕК + були незадоволені ціною на нафту, у їх інтересах було скоротити поставки нафти, щоб ціни зростали. Однак жодна окрема країна насправді не хоче скорочувати пропозицію, оскільки це буде означати скорочення доходів. В ідеалі вони хочуть, щоб ціна на нафту

зросла, у той час як вони збільшують пропозицію, щоб доходи також зростали. Але це не динаміка ринку. Таким чином, обіцянка ОПЕК + скоротити поставки викликає негайний стрибок цін на нафту. З часом ціна повертається до рівня, зазвичай нижче, коли пропозиція істотно не скорочується або попит не коригується[5].

І навпаки, ОПЕК + може прийняти рішення про збільшення пропозиції. Наприклад, 21 червня 2018 року ОПЕК + зустрілася у Відні і оголосила, що буде збільшувати пропозицію. Основною причиною цього було компенсувати вкрай низьку видобуток з боку іншого члена ОПЕК + Венесуела. Саудівська Аравія і Росія, які є двома найбільшими експортерами у світі та здатними збільшити виробництво, є великими прихильниками збільшення поставок, оскільки це збільшить їхні доходи. Інші країни, які не можуть нарощувати виробництво або тому, що працюють на повну потужність, або іншим чином не мають права (Іран - санкції), були б проти цього.

Зрештою, сили попиту і пропозиції визначають рівновагу цін, хоча оголошення ОПЕК + можуть тимчасово вплинути на ціну нафти, змінивши очікування. Як приклад можна привести зміну очікувань ОПЕК +, коли її частка у світовому видобутку нафти знизиться, а нова видобуток буде надходити з зовнішніх країн, таких як США і Канада. У березні 2020 року Саудівська Аравія, початковий член ОПЕК, а також найбільший експортер і надзвичайно впливовий на світовому ринку нафти, і Росія, другий провідний експортер і, можливо, другий за важливістю гравець в недавно сформованої ОПЕК +, не вдалося домовитися про скорочення видобутку для стабілізації цін на нафту. Саудівська Аравія відповіла різким збільшенням виробництва. Цей раптове зростання пропозиції стався в той час, коли світовий попит на нафту різко впав, оскільки світ зіткнувся з пандемією COVID-19.

В результаті ринок, який є кінцевим арбітром ціни, переважив прагнення ОПЕК + стабілізувати ціну на нафту на більш високому рівні, ніж диктували закони попиту і пропозиції. Крім підтвердження того, що ринкові

сили могутніші, ніж будь-який картель, особливо на вільних ринках, цей епізод також підтвердив припущення про те, що програми окремих країн матимуть пріоритет перед планами картелів. Нафта Brent Crude Oil станом на 14 квітня 2020 року стоїть близько 30 доларів за барель, що є на рівні ціни 2004 року, в той час як нафта WTI Crude Oil коштує 20,5 долара за барель, що не спостерігалося з 2002 року[4].

1.3 Формування ціни на нафту

Ціни на нафту контролюються трейдерами, які роблять ставки на ф'ючерсні контракти на нафту на товарному ринку. Саме тому ціни на нафту змінюються щодня. Все залежить від того, як пройшла торгівля в той день. Трейдери засновують свої заявки на сприйнятті попиту і пропозиції. Інші організації можуть впливати на рішення учасників торгів своїми діями. Ці впливові особи включають уряд США і Організацію країн-експортерів нафти. На рис. 1.1 відображено ціни на нафту за останні 40 років.

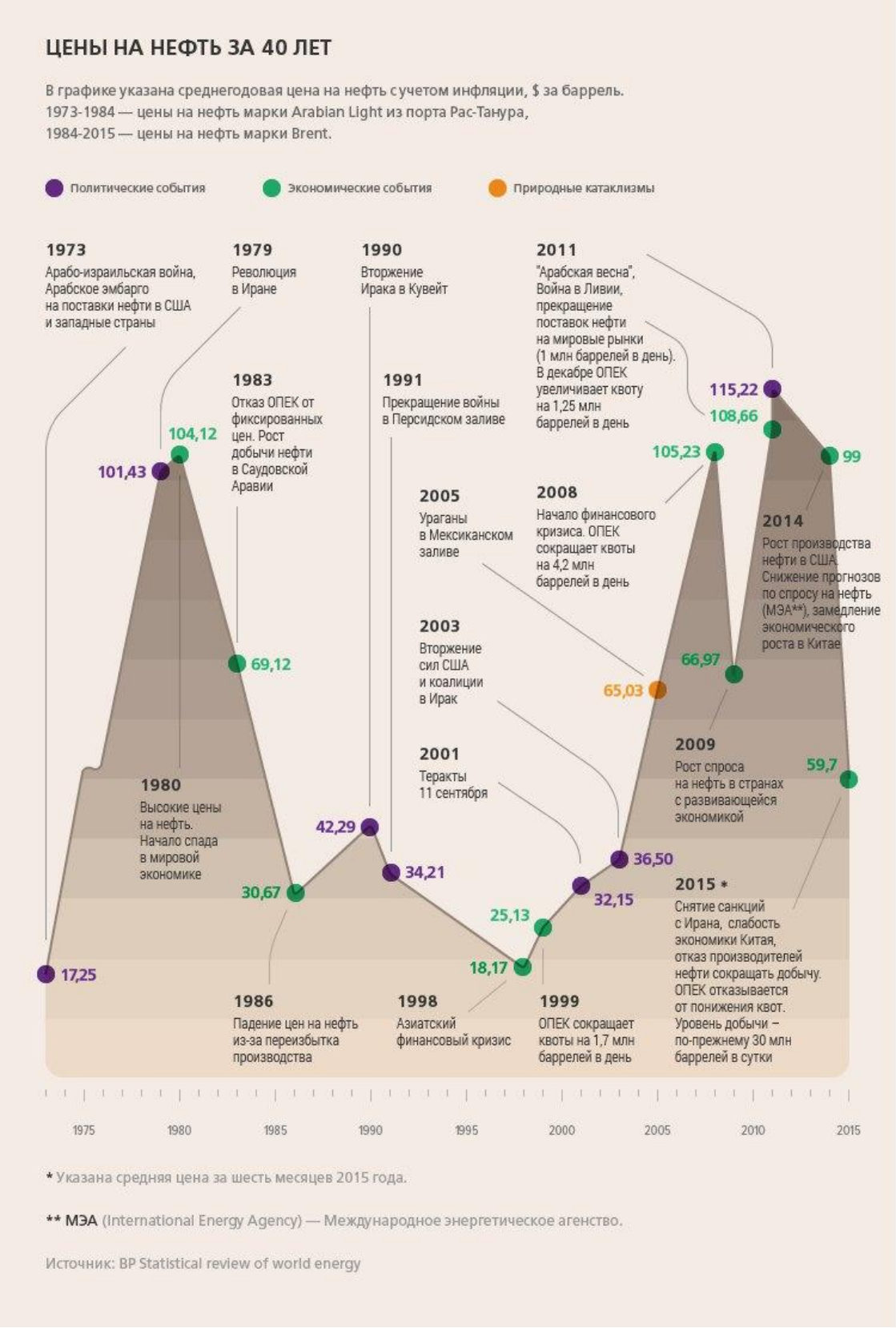


Рисунок 1.1 – Ціни на нафту за 40 років

Існує декілька основних факторів, що сприяють ціноутворенню на ринку нафти[6], а саме:

- попит та пропозиція;
- комерційні запаси нафти;
- кількість бурових вишок;
- непередбачувані події;

Сильне економічне зростання і промислове виробництво, як правило, стимулюють попит на нафту, що відбивається в зміні структури попиту країн, що не входять в ОЕСР, які в останні роки швидко росли. За даними Управління енергетичної інформації США, «споживання нафти в країнах Організації економічного співробітництва і розвитку (ОЕСР) знизилося в період з 2000 по 2010 рік, в той час як споживання нафти поза країнами ОЕСР збільшилася більш ніж на 40%. Китай, Індія і Саудівська Аравія мали найбільше зростання споживання нафти серед країн, що не входять в ОЕСР, за цей період ».

Показник комерційних запасів нафти відображає, кількість реальних запасів нафти по відношенню до прогнозованої заздалегідь кількості. Щосереди оприлюднюється офіційна статистика від AAEI. Збільшення запасів щодо прогнозних даних означає, що або надлишкова кількість нафти вироблялася, або споживання було слабкішим за очікуване(рис. 1.2).

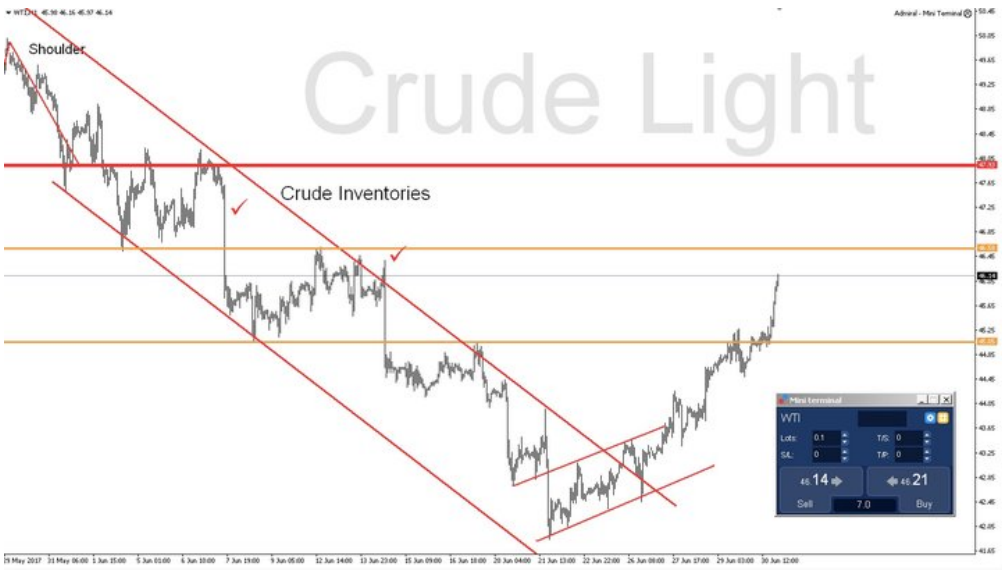


Рисунок 1.2 - Статистика комерційних запасів нафти

Оскільки США є досить сильним гравцем на ринку нафту протягом останніх років, тому кількість бурових вишок є досить важливим фактором, що впливає на формування цін. Якщо бути точнішим, саме публікація статистики мало впливає на ціни на даний момент часу, але тренди та тенденції в цих даних є досить важливими. Вважається, що зростання веж рано чи пізно неодмінно призведе до збільшення виробництва. Більше того, терміни стають меншими, а кореляція стає все більшою. Дані доступні для будь-кого кожної п'ятниці на офіційному веб-сайті Baker & Hughes[7]. Кількість бурових вишок показана на рис. 1.3.

Rotary Rig Count
6/30/2017

BAKER HUGHES

Location	Week	+/-	Week Ago	+/-	Year Ago
Land	915	0	915	507	408
Inland Waters	4	0	4	0	4
Offshore	21	-1	22	2	19
United States Total	940	-1	941	509	431
Gulf Of Mexico	21	0	21	3	18
Canada	189	19	170	113	76
North America	1129	18	1111	622	507
U.S. Breakout Information	**This Week**	**+/-**	**Last Week**	**+/-**	**Year Ago**
Oil	756	-2	758	415	341
Gas	184	1	183	95	89
Miscellaneous	0	0	0	-1	1
Directional	71	-1	72	33	38
Horizontal	792	0	792	460	332
Vertical	77	0	77	16	61
Canada Breakout Information	**This Week**	**+/-**	**Last Week**	**+/-**	**Year Ago**
Oil	112	14	98	77	35
Gas	77	5	72	37	40
Miscellaneous	0	0	0	-1	1

Рисунок 1.3 - Кількість бурових вишок

Природні і техногенні катастрофи можуть вплинути на ціни на нафту, якщо вони матимуть глобальні наслідки.

COVID-19. У січні 2020 року багато уряди почали обмежувати пересування по країні і закривати підприємства, щоб зупинити пандемію коронавіруса. Попит на нафту почав падати. У першому кварталі 2020 року

споживання нафти в середньому становило 94,4 млн барелів на добу, що на 5,6 млн барелів на добу більше, ніж у попередньому році. Падіння попиту ускладнилося через надлишок пропозиції. 6 березня Росія оголосила, що збільшить видобуток в квітні. Щоб зберегти частку ринку, ОПЕК оголосила, що також збільшить видобуток. Коли складські приміщення стали заповнені, ціни впали в мінус. Торговці були готові платити гроші за доставку нафти, так як місця для її зберігання просто не було. 12 квітня 2020 року ОПЕК і Росія домовилися про зниження видобутку, щоб підтримати ціни. Цього було недостатньо для переконання трейдерів, що пропозиція не перевищить попит. На рис. 1.4 відображена від'ємна ціна на нафту. Станом на 20 квітня 2020 року ціна за барель нафти впала до -$ 36,98.

Рисунок 1.4 - Графік цін на нафту.

«Катріна» - ураган 5 категорії, який обрушився на Луїзіану 29 серпня 2005 року. У період з 29 серпня по 5 вересня середня ціна на звичайний

бензин в США зросла на 46 центів до 3,07 долара за галон. Це був найбільший щотижневий стрибок цін за всю історію. Ураган «Катріна» торкнувся 25% видобутку сирої нафти в США. Це відбулося слідом за ураганом «Рита». Сукупний вплив двох ураганів скоротив надходження сирої нафти на нафтопереробні заводи до 11,7 млн барелів на день протягом тижня. Це був найнижчий середній показник з березня 1987 року[8]. Ціни на нафту у 2005 році відображені на рис. 1.5.

Рисунок 1.5 - Графік цін на нафту 2005 рік.

1.4 Модель для прогнозування ціни на нафту

Багато методів прогнозування були розроблені для прогнозування ціни Crude Oil, включаючи традиційні моделі економетрії та підходи машинного навчання, що можуть забезпечити більш точне прогнозування, ніж моделі економетрії, але часто інтерпретувати їх економічно набагато важче. Чим популярнішим стає машинне навчання, тим більш воно розвивається, а дерева прийняття рішень стають основним інструментом в дослідженнях багатьох галузей. Більш того, на відміну від інших моделей машинного навчання, які вважаються «чорними коробками» через труднощі з економічною інтерпретацією, моделі дерев рішень інтерпретується в теорії

(Loh, 2014 року). У цьому дослідженні. Дерево рішень є привабливим методом машинного навчання завдяки своїй ефективності, надійність і відносно проста структура (Quinlan, 1986). Згідно J.R.Quinlan (1992), помітну перевага дерев рішень полягає в тому, що воно дозволяє легко інтерпретувати результати після того, як зроблено прогноз[9].

1.5 Висновки

Прогнозування цін на нафту є досить важкою задачею через високу волотильність. Для її виконання я ознайомився з істотною кількістю різноплановою літератури, а саме: економічною, технічною та історичною. Розглянув історичні відомості про нафту, ознайомився з формуванням ринку, ціни та принципами торгівлі. Проаналізувавши інформацію я прийняв рішення, що для виконання цієї задачі найкраще всього скористатися методами регресійного аналізу.

РОЗДІЛ 2 РЕГРЕСІЙНИЙ АНАЛІЗ. ТЕОРІЯ ТА ВІДОМОСТІ

2.1 Вступ

У статистичному моделюванні регресійний аналіз являє собою набір статистичних процесів для оцінки взаємозв'язків між залежною змінною (часто позначається «вихідної змінної») і однієї або декількома незалежними змінними (часто позначаються «предикторами», «коваріатамі» або «ознаками»). Перш за все треба повернутися в самий початок і зрозуміти для себе, що таке термін «регресія».

Регресія – одностороння стохастична залежність, що встановлює відповідність між випадковими змінними. На відміну від суто функціональної залежності $y = f(x)$, коли кожне значення незалежної змінної x відповідає одному визначеному значенню y, у випадку регресії одне і те ж значення x може відповідати різним значенням y[10]. Приклад показано на рисунку 2.1.

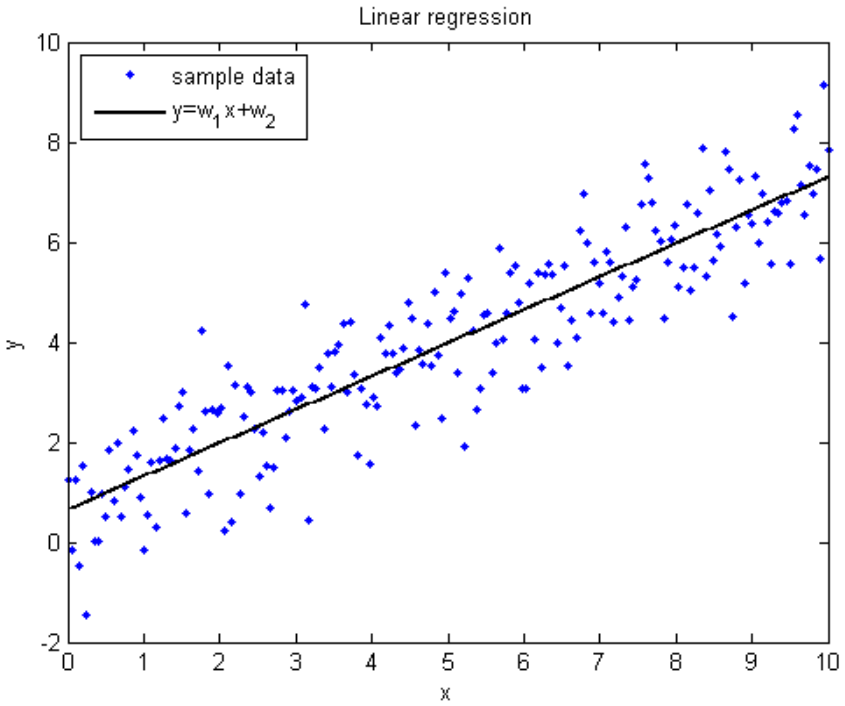


Рисунок 2.1 - Лінійна регресія

Перелік основник термінів:

- викид – результат вимірювання, що виділяється із загальної статистики; припустимо, що в наборі даних є спостереження, яке має дуже високе або дуже низьке значення в порівнянні з іншими спостереженнями в наборі даних, тобто воно не відноситься до сукупності, таке спостереження називається викидом;
- мультиколінеарність – коли незалежні змінні сильно корелюють один з одним, змінні називаються мультиколінеарними; багато методів регресії припускають, що мультиколінеарність не повинна бути наявна в наборі даних. Це тому, що це викликає проблема в ранжируванні змінних в залежності від їх важливості, також це ускладнює роботу з вибору найбільш важливої незалежної змінної (фактора);
- гетероскедастичність – термін, який означає, що ранжування залежної змінної відрізняється від незалежної змінної;
- перенавчання – ситуація, коли алгоритм гарно працює на тренувальній вибірці, але показує поганий результат на тестовій;
- недонавчання – коли статистична модель або алгоритм машинного навчання не можуть схопити тенденцію, що лежить в основі даних;

Приклад зображено на рис. 2.2.

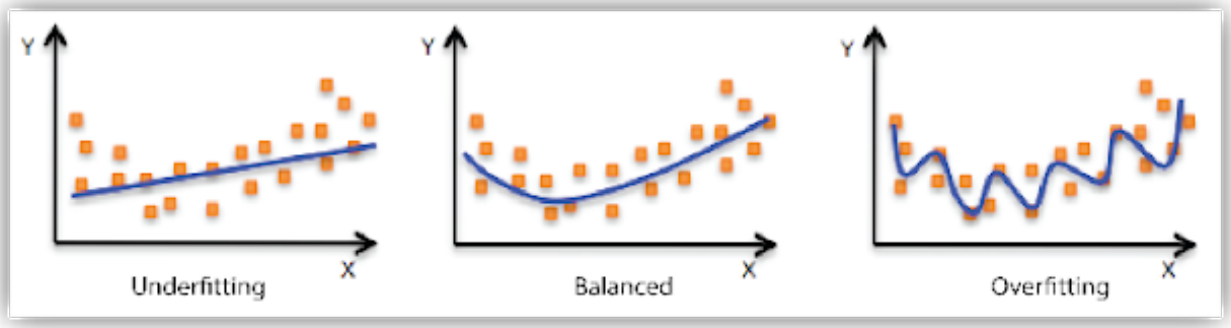


Рисунок 2.2 - Недонавчання, баланс, перенавчання

2.2 Види регресії

2.2.1 Лінійна регресія

Лінійна регресія - найпростіша форма регресії. Це метод, в якому залежна змінна є безперервною за своєю природою. Передбачається, що взаємозв'язок між залежною і незалежною змінними носить лінійний характер. На рис. 2.3 можемо помітити, що даний графік являє собою якось лінійну залежність між пробігом і переміщенням автомобілів.

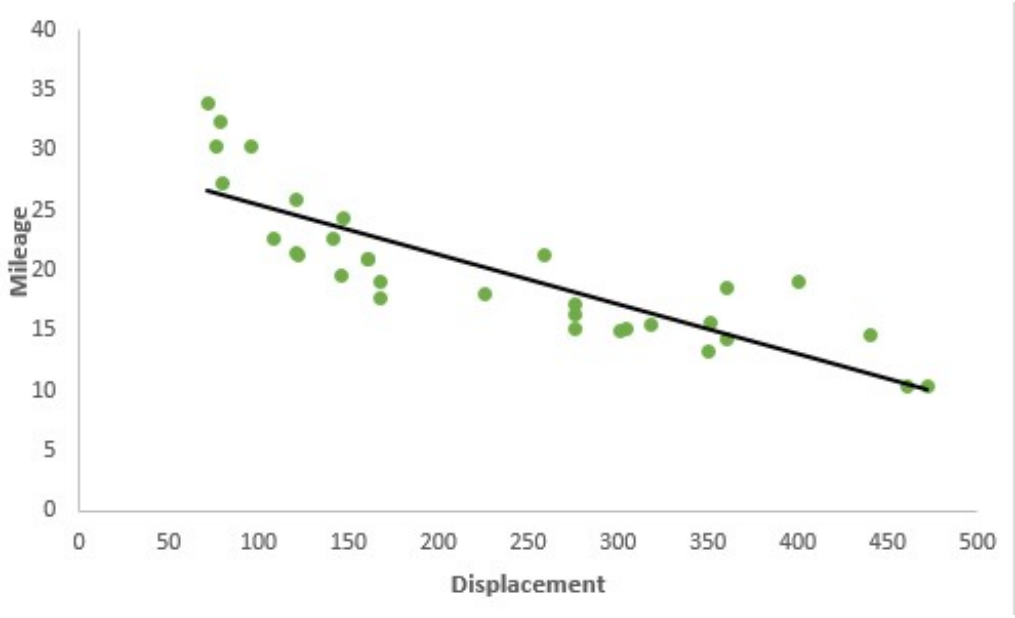


Рисунок 2.3 - Регресія з лінійною залежністю

Лінійна регресія використовується для аналізу та прогнозування. Лінійна регресія - це лінійний підхід для моделювання взаємозв'язку між критерієм або скалярним відгуком і множинними предикторами. Лінійна регресія фокусується на умовному розподілі ймовірності відповіді з урахуванням значень «предикаторів». Для лінійної регресії існує небезпека перенавчання. Приклад на рис. 2.4.

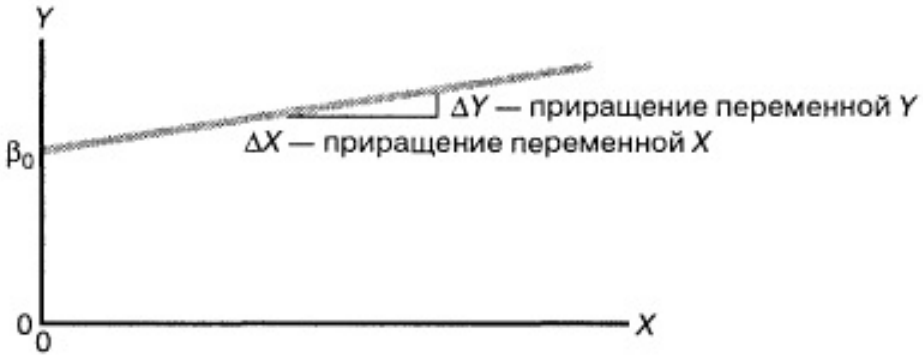


Рисунок 2.4 - Лінійна регресія

Формула простої лінійної регресії:

$$y = f(x, b) + \varepsilon, E(\varepsilon) = 0,$$

(2.1)

де b – параметри моделі;

ε – випадкова помилка моделі.

$f(x, b)$ має вигляд $f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k$,

(2.2)

де b_j - параметри (коефіцієнти) регресії;

x_j - регресори (фактори моделі);

k - кількість факторів моделі.

Основні твердження лінійної регресії

- наявність лінійного відношення між незалежними і залежними змінними;
- не повинно бути ніяких викидів;
- гетероскедастичність;
- випадкові спостереження повинні бути незалежними;
- відсутність мультиколінеарності та автокореляції

Після того, як регресійна модель була підібрана для групи даних, перевірка залишків (відхилення від лінії регресії відповідно до

спостережуваних значень) дозволяє аналітику перевірити обґрунтованість свого припущення про існування лінійної залежності. Побудова залишків на осі Y щодо залежної змінної на осі X виявляє будь-які можливі нелінійні відносини між змінними або може попередити аналітика про наявність прихованих змінних. Приклад на рис. 2.5 показує наявність викидів[11].

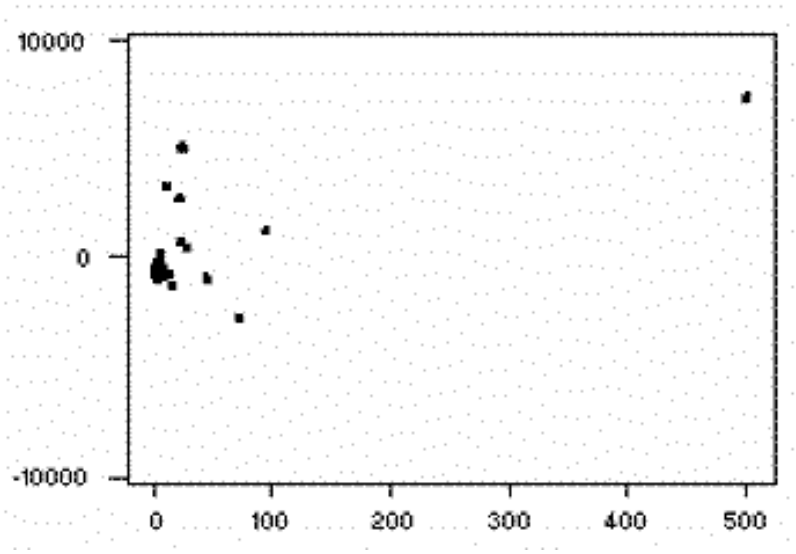


Рисунок 2.5 – Викиди

2.2.2 Поліноміальна регресія

Поліноміальна регресія використовується для криволінійних даних. Поліноміальна регресія відповідає методу найменших квадратів. Мета регресійного аналізу - змоделювати очікуване значення залежної змінної у щодо незалежної змінної x. На наведеному нижче прикладі видно, що червона крива відповідає даним краще, ніж зелена. Отже, в ситуаціях, коли зв'язок між залежною і незалежною змінними здається нелінійним, ми можемо використовувати моделі поліноміальної регресії(рис. 2.6).

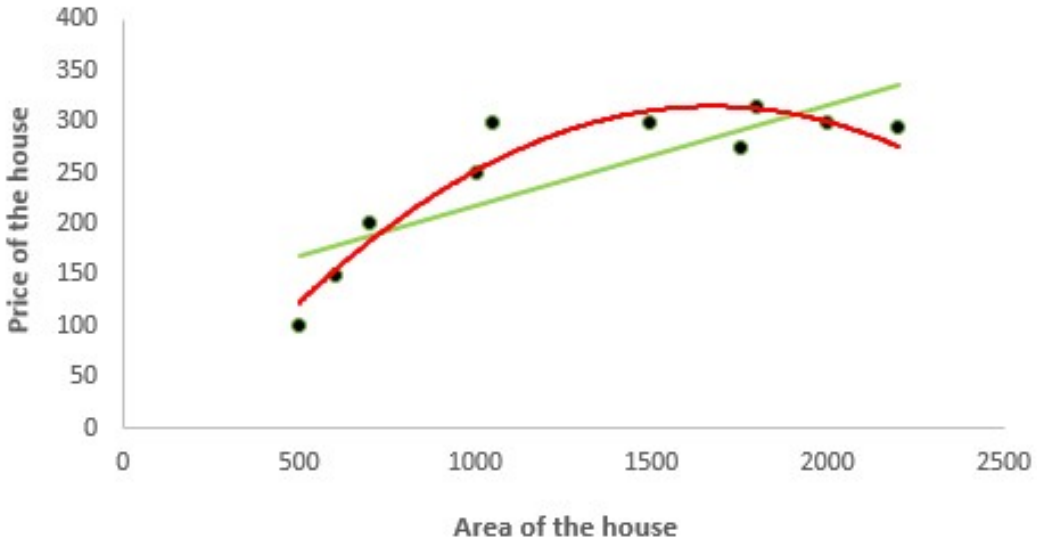


Рисунок 2.6 - Поліноміальна регресія

Загальне рівняння поліноміальної регресії:

$$y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_2 X^k + \varepsilon, \tag{2.3}$$

де $\beta_j -$ параметри полінома;

$X^j -$ змінні, $\varepsilon -$ похибка.

Переваги використання поліноміальної регресії:

- поліном забезпечує найкращу апроксимація між залежною і незалежною змінними;
- підходить для великої кількості функцій;

Недоліки використання поліноміальної регресії:

- наявність одного або двох викидів в даних може серйозно вплинути на результати нелінійного аналізу;
- занадто чутливі до викидів;
- існує менше інструментів перевірки моделей для виявлення викидів в нелінійної регресії, ніж для лінійної регресії[12];

2.2.3 Логістична регресія

У логістичної регресії залежна змінна має двійковий характер (має дві категорії). Незалежні змінні можуть бути безперервними або двійковими. У многочленній логістичній регресії ви можете мати більше двох категорій в залежній змінній. Логістична модель має вигляд:

$$p = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+..+b_kx_k)}}, \tag{2.4}$$

де $p -$ логістична функція;

$\beta^t X -$ вектори-стовпці значень незалежних змінних $X_1, \dots, X_k$ і параметрів (коефіцієнтів регресії);

$\beta_1, \dots, \beta_k -$ дійсні числа[11].

Приклад дивись на рис. 2.7.

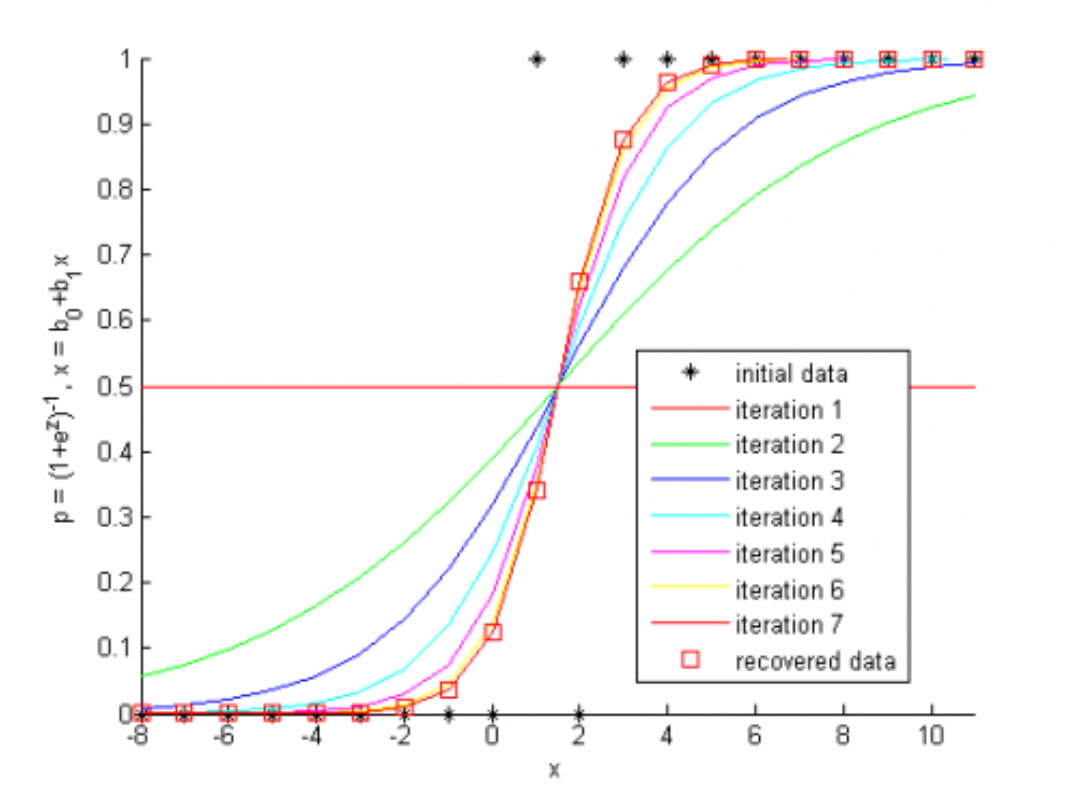


Рисунок 2.7 - Логістична регресія

2.2.4 Квантільна регресія

Квантільна регресія – регресія(прогноз), що спеціально вводить зміщення у результат. Замість пошуку середнього значення передбачуваної змінної, квантільна регресія прагне знайти медіану та будь-які інші кванти (які іноді називають «процентілями»). Як прямий параметр для обчислення точок для обчислення точок оновлення. Квантілі особливо корисні для оптимізації товарних запасів в якості прямого методу для обчислення точки відновлення. Поняття квантільної регресії представляє більш сучасну галузь статистики. Формула моделі регресії має вигляд:

$$Q_\tau(y_i) = \beta_0(\tau) + \beta_1(\tau)x_{i1} + \cdots + \beta_p(\tau)x_{ip},\ i = 1, \dots, n, \tag{2.5}$$

де $\beta_p(\tau)$ – функція залежності від квантіля;

x_{ip} – регресор.

На рисунку 2.8 продемонстровано прогноз росту попиту.

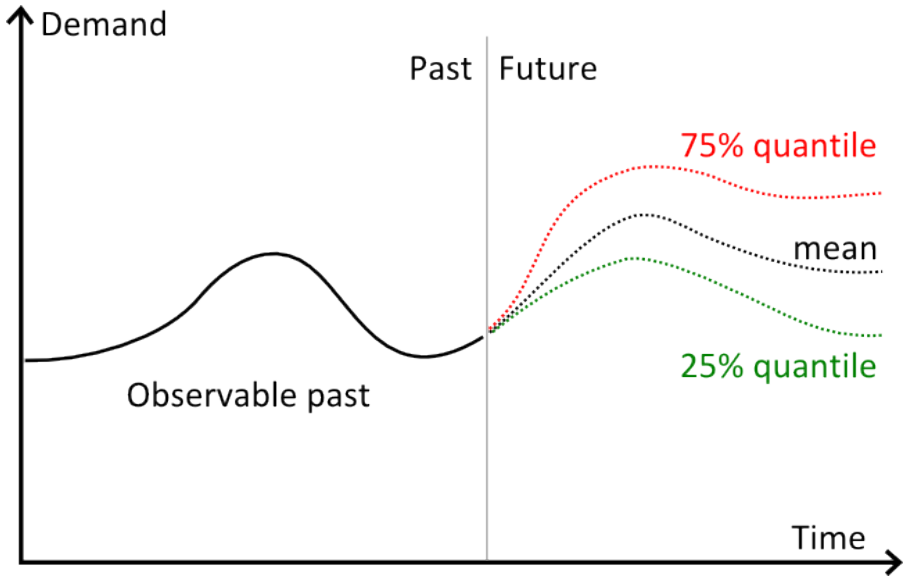


Рисунок 2.8 - Прогнози росту попиту з часом

3 відокремлених прогнозу:

- червоним кольором відображено 75% квантільний прогноз;
- чорним кольором відображений прогноз на основі середніх значень;
- зеленим кольором позначено 25% квантільний прогноз[13];

Переваги цього методу над лінійною регресією:

- дуже корисна коли в даних наявна гетероскедастичність;
- стійкість до викидів;
- розподіл залежної змінної можна описати за допомогою різних квантилів;

2.2.5 Гребнева регресія

Гребнева регресія – один із методів для зменшення розмірності. Найчастіше використовується для боротьби із надмірністю даних, коли незалежні змінні корелюють одна з одною. Регресія дає змогу побудувати модель, коли кількість прогнозованих змінних у наборі перевищує кількість змінних спостережень. Метод Тихонова майже аналогічний до методу гребневої регресії, але він має більший набір даних, а тому він може дати рішення, навіть якщо набір даних містить багато статистичних шумів (незрозумілий розкид в виборці)[14].

Наприклад, для рівняння $Ax = b$ не існує єдиного розв'язку x, тоді ми можемо використати гребеневу регресію:

$$\min||Ax - b||^2 + ||\Gamma x||^2, \tag{2.6}$$

де $\Gamma -$ матриця Тихонова;

$A -$ матриця;

$b -$ вектор.

Гребенева регресія запобігає перенавчанню та недонавчанню шляхом введення штрафної функції $||\Gamma x||^2$, де $\Gamma -$ матриця Тихонова (обумовлена користувачем матриця, яка дозволяє алгоритму віддавати перевагу певним рішенням над іншими). Нижче, на рис. 2.9, наведено приклад.

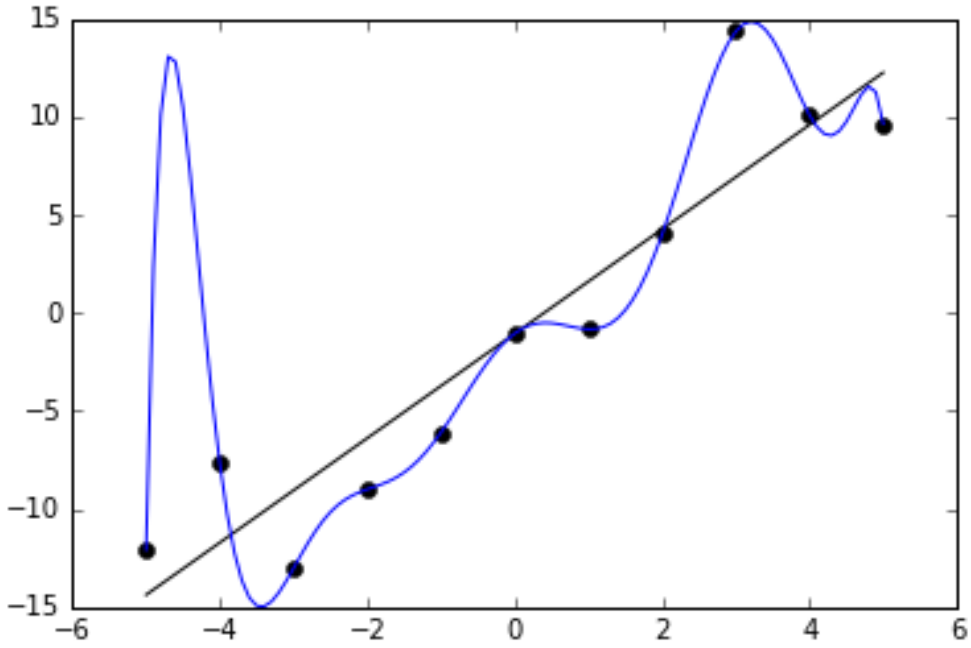


Рисунок 2.9 - Гребнева регресія

Синя крива мінімізує похибку точок даних. Іноді це може привести до перенавчання. Введення матриці Тихонова відображається у чорну криву. Треба розуміти що вона не мінімізує помилки, але захищає від перенавчаннях[15].

2.2.6 Метод регресії «Ласо»

Суть регресії за методом «Ласо» є в тому, що вводиться додатковий додаток в функціонал оптимізації моделі, що часто дозволяє отримувати більш стійке рішення. Умова мінімізації квадратів помилки при оцінці параметрів $\hat{\beta}$ виражається наступною формулою:

$$\hat{\beta} = argmin(\sum_{i=1}^{n}(y_j - \sum_{j=1}^{m}\beta_j x_{ij})^2 + \lambda|\beta|),$$

(2.7)

де $\lambda -$ параметр регуляризації.

Що має сенс штрафу за складність, при цьому досягається певний компроміс між помилкою регресії і розмірністю використовуваного простору ознак, вираженого сумою абсолютних значень коефіцієнтів.

Регресія «Ласо» відноситься до регуляризації L1, яка додає штраф, який дорівнює абсолютному значенню величини коефіцієнтів. Цей тип регуляризації може привести до розрідженої моделі з декількома коефіцієнтами. Деякі коефіцієнти можуть стати нульовими і будуть виключені з моделі. Великі штрафи призводять до того, що значення коефіцієнтів ближче до нуля, що ідеально підходить для створення більш простих моделей. З іншого боку, регуляризація L2 (наприклад, гребенева регресія) не призводить до виключення коефіцієнтів або розріджених моделей. Це робить методом «Ласо» набагато простішим для інтерпретації.

Якщо порівняти регресію за методом «Ласо», та гребеневу регресію, неможливо відповісти яка з них краща. Обидві ці регресії добре працюють з мультиколінеарністю. Найкраще за все обирати метод відповідно до початкового набору даних[16]. Приклад порівняння наведено на рис. 2.10.

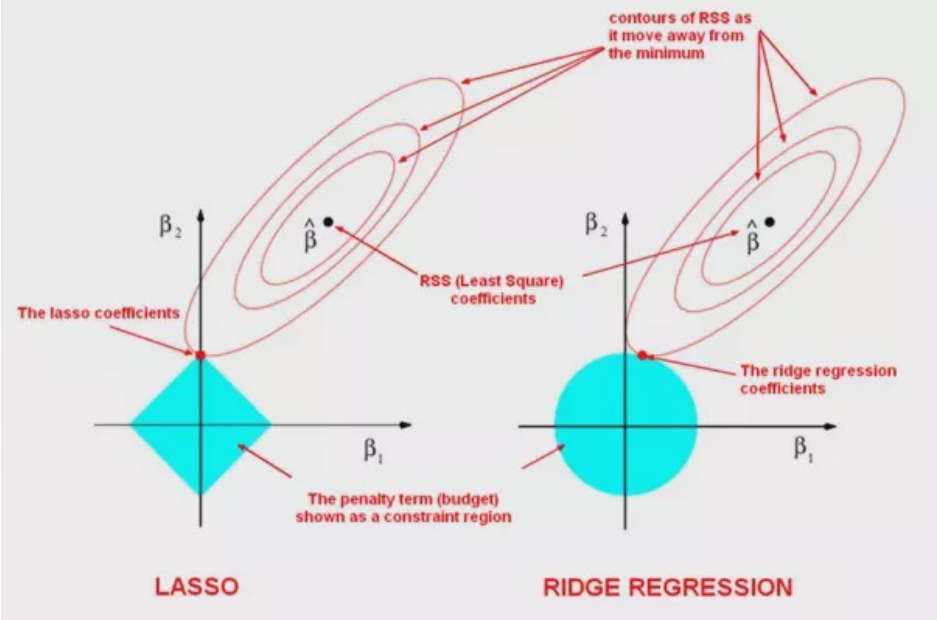


Рисунок 2.10 - Порівняння регресій

2.2.7 Регресія «ElasticNet»

Еластична сітка - це гібрид методів регресії «Лассо» і «Гребнева». Він тренується з L1 і L2 регуляризаціями і бере найкраще. Еластична сітка корисна, коли є кілька взаємопов'язаних функцій. Метод «Лассо» ймовірно, обере один з них випадковим чином, в той час як еластична сітка, швидше за все, вибере обидва. Виходячи з цього завдання оптимізації можна побачити на формулі (2.8).

$$\hat{\beta} = (\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta_1\|) \tag{2.8}$$

Практична перевага вибору компромісу між «Лассо» і «Гребневой» полягає в тому, що воно дозволяє «ElasticNet» успадковувати частину стабільності «Гребневой» регресії при обертанні[17].

2.2.8 Байєсова регресія

Байєсова регресія схожа на гребневу регресію, проте заснована на тому припущенні, що в даних шум (помилка) розподілений нормально - відповідно, передбачається, що загальне розуміння про структуру даних вже є, і це дає можливість отримувати більш точну модель (у порівнянні з лінійною регресією точно). Але у реальному житті, особливо коли ми працюємо з великим набором даних, початкові знання про дані не можуть похвалитися точністю, тому припущення будується на підставі пов'язаних величин, тобто воно штучно за своєю суттю - і це суттєвий недолік даного типу регресії. Формула для вирішення задачі має наступний вигляд:

$$\tau = y(x, w) + \varepsilon,$$

(2.9)

де $\tau -$ спостережувана змінна;

$\varepsilon \sim N(0, \sigma^2) -$ нормально розподілена помилка.

2.2.9 Логічна регресія

Логічна регресія – вид регресії, що використовується у випадку, коли змінні двійкові. Загалом, взаємодії між змінними враховуються не часто, ці взаємодії зазвичай залишаються простими (не більше двох-тристоронніх взаємодій). Але часто, особливо коли все змінні є двійковими, їх взаємодія є причиною відмінностей у відповідях.

Логічна регресія дозволяє дослідити, як зв’язок між змінними впливає на результат загалом. Приклад моделі логічної регресії:

$$g\big(E(y)\big) = \beta_0 + \sum_{j=1}^{t} \beta_j L_j$$

(2.10)

де $L_j -$ булева функція від змінних X_j.

На рис. 2.11 зображені можливі дії при роботі з логічними деревами[18].

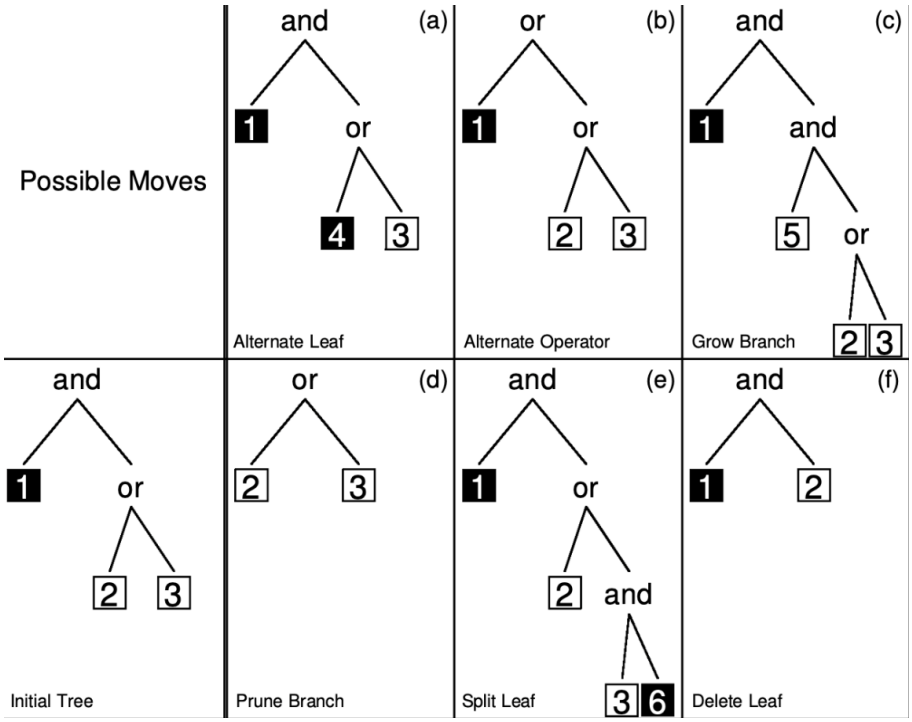


Рисунок 2.11 - Дії з логічними деревами

2.3 Підсумки

На основі інформації, я прийшов до висновку, що виконання моєї роботи потребує використання регресії. Оскільки задача є досить важкою та нелінійною, було вирішено розглянути дерева рішень . Завдяки цьому я зможу коректо обробити всю вхідну інформацію і різні фактори і в подальшому правильно розробити прогноз.

2.4 Дерева Рішень «Decision Trees»

Дерева рішень («Decision Trees») - це статистичний метод, що дозволяє передбачати приналежність спостережень або об'єктів до того чи іншого класу категоріальної залежною змінною або середнє значення кількісної

змінної в залежності від відповідних значень однієї або декількох незалежних змінних.

Дерево має багато аналогій в реальному житті і виявляється, що воно вплинуло на широку область машинного навчання, що охоплює як класифікацію, так і регресію. При аналізі рішень дерево рішень може використовуватися для візуального і явного уявлення рішень і їх прийняття. Як випливає з назви, метод використовує деревоподібну модель рішень. Незважаючи на те, що цей інструмент широко використовується для дослідження стратегії для досягнення певної мети, він також широко використовується в машинному навчанні.

Мета методу дерев рішень – спрогнозувати значення цільової змінної в залежності від відповідних значень незалежних змінних (предикторів, атрибутів). В свою чергу, дерева рішень поділяються на дерева регресії і дерева класифікації.

Якщо мова йде про дерева регресії, прогнозується значення цільової змінної, що залежить від відповідних значень предикторів.

Наприклад, прогнозується ймовірність регістрації на певному сайті в залежності від статі і віку відвідувача. Дерева регресії працюють з кількісною цільовою змінною.

В деревах класифікації все виглядає інакше. Робиться прогноз приналежність об'єкта до тієї чи іншої категорії цільової змінної в залежності від відповідних значень предикторів.

Наприклад, класифікуються добрі і погані учні в залежності від їх оцінок. Дерева класифікації працюють з категоріальної цільової змінної.

Залежність значення цільової змінної від значень предикторів, представляється у вигляді ієрархічної структури - «дерева». Якщо залежна змінна є категоріальної, будується дерево класифікації. Якщо залежна змінна є кількісної, будується дерево регресії.

Також дерева рішень допомагають вирішувати завдання опису об’єктів. Тобто за допомогою дерева, можна замінити велику структуру на більш зручну, маленьку.

2.4.1 Основні терміни та поняття

Для розуміння теорії дерев рішень треба розібратися з основними термінами[19]:

- об’єкт – приклад, шаблон, дослідження;
- атрибут – ознака, незалежна змінна, властивість;
- цільова змінна – залежна змінна, мітка класу;
- вузол – внутрішній вузол дерева, вузол перевірки;
- кореневий вузол – початковий вузол дерева рішень;
- лист – кінцевий вузол дерева, вузол рішення;
- вирішальне правило – умова у вузлі, перевірка;
- ентропія – міра хаотичності інформації.

2.4.2 Представлення алгоритму у вигляді дерева

Для більш простого та чіткого розуміння потрібно розглянути приклад(рис. 2.12).

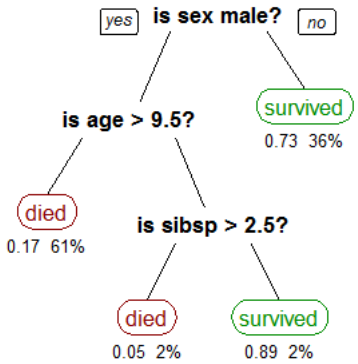


Рисунок 2.12 - Дерево рішень

У наведеній вище моделі використовуються 3 атрибута з набору даних, а саме стать, вік і кількість подружжя або дітей разом.

Дерево рішень намальовано догори ногами з коренем вгорі. На (рис. 2.12) жирний текст чорного кольору є умовою(внутрішнім вузлом), на основі якого дерево розділяється на гілки(ребра). Кінець гілки, яка більше не поділяється, є рішенням(листком), в даному випадку, чи загинув пасажир або вижив, (представлено червоним або зеленим кольором) відповідно.

Хоча в реальному наборі даних буде набагато більше ознак, і це буде просто гілка в набагато більшому дереві, але ви не можете ігнорувати простоту цього алгоритму. Важливість функції ясна, і відносини можуть бути легко відстеженні. Ця методологія більш відома як дерево прийняття рішень на основі даних, а дерево вище називається деревом класифікації, оскільки мета полягає в тому, щоб класифікувати пасажира як вижив або померлого.

Дерева регресії представлені так само, просто вони прогнозують безперервні значення, такі як ціна на нерухомість(рис. 2.13).

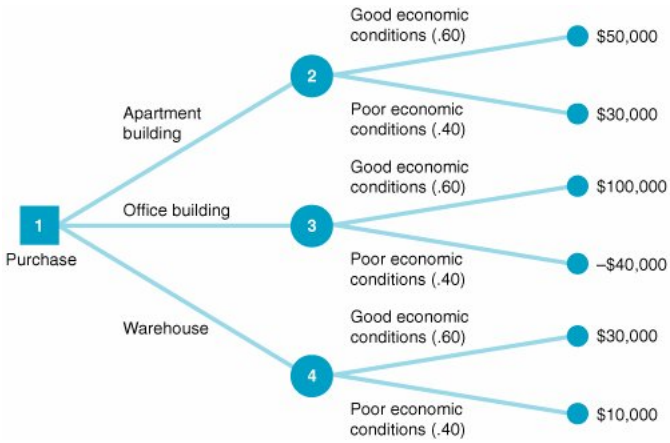


Рисунок 2.13 - Дерево регресії

Загалом, алгоритми дерева рішень називаються «CART» або деревами класифікації і регресії[20].

Алгоритм розбиття «CART» для регресії. Функція вартості «CART», яка мінімізується для задач регресії:

$$J(T, x_h) = \frac{|T_{left}|}{|T|} MSE(T_{left}) + \frac{|T_{right}|}{|T|} MSE(T_{right}) \tag{2.11}$$

$$MSE(T_{left}) = \sum_{k \in T_{left}} \left(\hat{y}_{T_{left}} - y_k\right)^2 \tag{2.12}$$

$$\hat{y}_{T_{left}} = \frac{1}{|T_{left}|} \sum_{k \in T_{left}} y_k \tag{2.13}$$

$$X_h^* = argmin_h J(T, x_h), \tag{2.14}$$

де X_h^* – корінь дерева(піддерева);

MSE – середньоквадратична похибка.

«CART» рекурсивно розбиває множину на підмножини з мінімально можливими значеннями MSE з ваговими коефіцієнтами – відносними розмірами цих підмножин[25].

MSE(mean squared error) – середньоквадратична помилка моделі.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(2.15)

Коефіцієнт детермінації — статистичний показник, в моделі є мірою залежності варіацій залежної змінної від варіацій незалежних змінних. Відображає наскільки спостереження побудованої моделі підтверджують реальні.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

(2.16)

Отже, що ж насправді відбувається в фоновому режимі? Вирощування дерева включає в себе рішення про те, які функції вибрати і які умови використовувати для поділу, а також знати, коли зупинитися. Оскільки дерево зазвичай зростає довільно, вам потрібно його обрізати, щоб воно виглядало красиво. Також великою перевагою цього алгоритму є те, що він – жадібний.

Жадібний алгоритм – алгоритм який на кожному кроці робить локально оптимальний вибір, допускаючи, що результат буде глобально оптимальний.

2.4.3 Метод побудови дерева рішень. Алгоритм розбиття

Даний підрозділ присвячено огляду загального підходу до побудови дерева рішень з використанням алгоритму розбиття.

Ідея:

- побудова дерева зверху-вниз;

- рекурсивне розбиття навчальної вибірки на максимально більш «чисті» підмножини;

- розбиття має бути значущим – класифікувати найбільшу кількість елементів навчальної вибірки.

Якщо множина T містить елементи, які відносяться до різних класів, тоді:

$$T = \bigcup_{i=1}^{q_h} T_i$$

$$T_i \subseteq T : x_h = c_{hi} \tag{2.17}$$

Множини T_i більш «чисті» в порівнянні з T.

$$T \coloneqq T_i, \forall_i = 1, \dots, q_h \tag{2.18}$$

Далі, на рис. 2.14 показано приклад розбиття на множини.

Де грає	Суперник	Температура	Дощ	Перемога
Вдома	Вище	Висока	Так	Ні
Вдома	Вище	Висока	Ні	Так
Вдома	Нижче	Норма	Ні	Так
Вдома	Нижче	Висока	Так	Ні
В гостях	Нижче	Норма	Так	Так
В гостях	Нижче	Висока	Так	Ні
В гостях	Нижче	Висока	Ні	Ні
В гостях	Вище	Норма	Ні	Так

Рисунок 2.14 - Розбиття на множини

Критерії вибору змінної розбиття:

- ентропійні;

- джині.

Нехай множина T складається з n об'єктів, k з яких мають властивість S. Тоді ентропія множини T по відношенню до властивості S:

$$H(T,S) = -\frac{k}{n}\log_2\frac{k}{n} - \frac{n-k}{n}\log_2\frac{n-k}{n} \tag{2.19}$$

$H(T,S)$=1, коли $k = \frac{n}{2}$.

Якщо S може приймати s різних значень, кожне з яких – в k_i, тоді

$$H(T,S) = -\sum_{i=1}^{s}\frac{k_i}{n}\log_2\frac{k_i}{n} \tag{2.20}$$

Далі, на рис. 2.15 наведено приклад розрахунку ентропії.

Де грає	Суперник	Температура	Дощ	Перемога
Вдома	Вище	Висока	Так	Ні
Вдома	Нижче	Норма	Ні	Так
В гостях	Нижче	Норма	Так	Так
В гостях	Нижче	Висока	Так	Ні
Вдома	Вище	Висока	Ні	Так
Вдома	Нижче	Висока	Так	Ні
В гостях	Нижче	Висока	Ні	Ні
В гостях	Вище	Норма	Ні	Так

$$H(T,Victory) = -\frac{4}{8}\log_2\frac{4}{8} - \frac{4}{8}\log_2\frac{4}{8} = 1$$

Рисунок 2.15 - Розрахунок ентропії

Міра забрудненості Джині формула (2.15):

$$G(T_i, V) = 1 - \sum_{k=1}^{S} (p_{i,k})^2, \tag{2.21}$$

де $p_{i,k}$ – доля прикладів класу k серед навчальних прикладів в i – му вузлі.

$G(T_i, V) = 0$, якщо T_i містить приклади одного класу[25].

2.4.4 Метод побудови дерева рішень. Критерій зупинки

Якщо дотримуватися суто теоретичного погляду, то алгоритм зупиниться в той момент, коли усі параметри будуть розподілені по підмножинах T_i. На більш менш великому наборі даних відбудеться проблема перенавчання. Перенавчання – ознака моделі при котрій дерево рішень гарно справляється з навчальною вибіркою, але абсолютно неспроможне працювати на нових даних. Для уникнення цієї проблеми були розроблені наступні підходи:

- рання зупинка;

-зупинка в разі, якщо всі об'єкти в вершині відносяться до одного класу;

- обмеження максимальної глибини дерева;

- завдання мінімально допустимого числа прикладів у вузлі;

Рання зупинка – зупинка алгоритму за критерієм. Тобто, задається певний критерій для навчання, при досягненні якого алгоритм зупиняється, наприклад відсоток правильно розпізнаних даних. В Scikit-Learn клас Decision TreeRegression має різні гіперпараметри.

Обмеження максимальної глибини дерева(max_depth) – зупинка алгоритму в момент досягнення заданої кількості розбиття гілок дерева.

Завдання мінімально допустимого числа прикладів у вузлі(min_samples_split) – обмеження на мінімальну кількість прикладів у

вузлі перш ній його можна буде розщепити. Це використовується для уникнення простих розбиттів в яких правило є невагомими.

Min_samples_leaf – мінімальна кількість прикладів у листковому вузлі.

Max_leaf_nodes – максимальна кількість листкових вузлів.

Max_features – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла[28].

Параметри max_... слід зменшувати, а параметри min_... слід збільшувати для зменшення ризику перенавчання дерева. Приклад наведено на рис. 2.16.

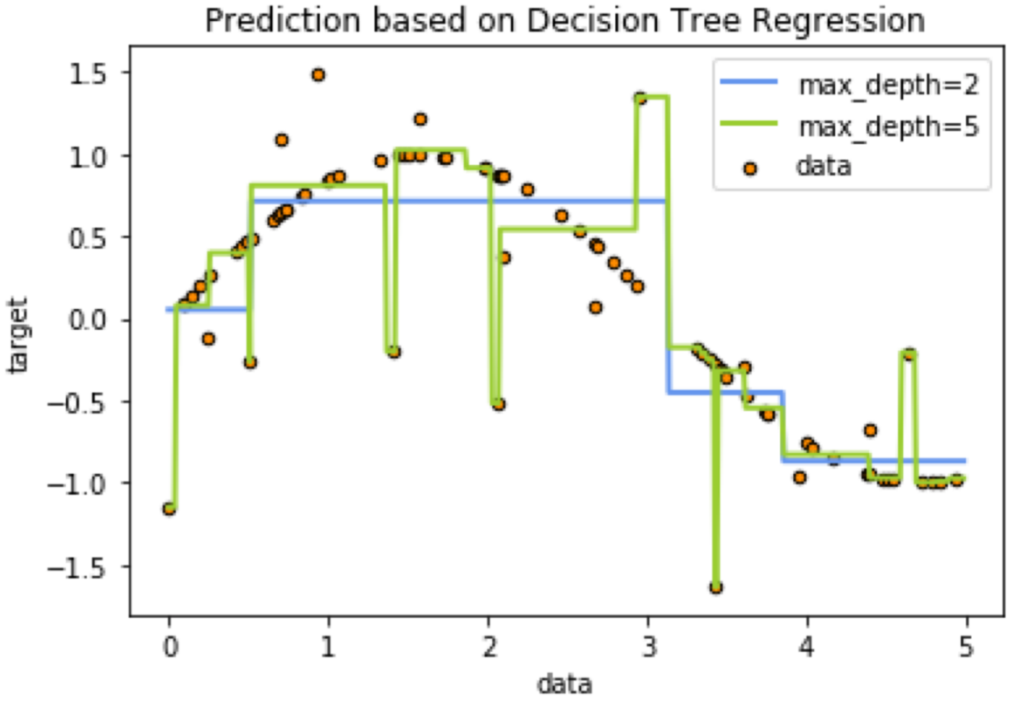


Рисунок 2.16 – Застосування гіперпараметру

2.5 Ансамбль дерев рішень

Давайте уявимо, що ви проводите соціальне опитування серед великої кількості населення, а потім агрегуєте їх відповіді. З великою ймовірністю виявиться, що ця агрегована відповідь буде краща за відповідь експерта у галузі. Це називається колективним розумом. Аналогічно якщо ви агрегуєте

прогнози групи прогнозаторів (таких як класифікатори або регресори), то в більшості випадків отримаєте кращі прогнози, ніж прогноз від найкращого індивідуального прогнозатора. Група прогнозаторів називається ансамблем. Відповідно, прийом носить назву ансамблеве навчання, а алгоритм – ансамблевий метод. На рисунку 2.17 зображено приклад ансамблю дерев рішень.

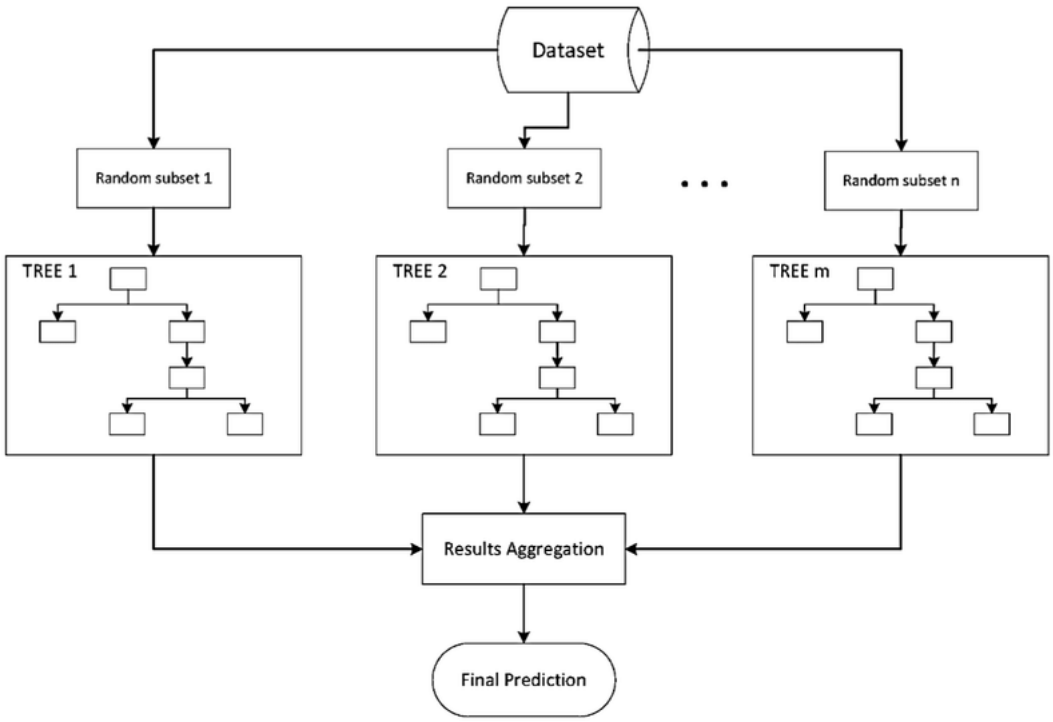


Рисунок 2.17 - Ансамбль дерев рішень

Наприклад, ви можете навчати групу класифікаторів на основі дерев прийняття рішень, використавши для кожного різні випадкові піднабори навчального набору. Для формування прогнозів ви лише отримуєте прогнози всіх індивідуальних дерев і прогнозуєте клас, який став володарем більшості голосів дерев. Такий ансамбль дерев прийняття рішень називається випадковим лісом і, незважаючи на свою простоту, є одним з самих потужних алгоритмів МН, доступних на сьогоднішній день[21]. Самими популярними методами є бегінг(bagging), бустінг(boosting).

Bagging (Bootstrap Aggregation) використовується, коли наша мета - зменшити дисперсію дерева рішень. Тут ідея полягає в тому, щоб створити кілька підмножин даних з навчальної вибірки, обраної випадковим чином. Тепер кожна колекція даних підмножини використовується для навчання їх дерев рішень. В результаті ми отримуємо ансамбль різних моделей. Використовується середнє значення всіх прогнозів для різних дерев, яке є більш надійним, ніж одне дерево рішень.

Boosting це ще один метод ансамблю для створення колекції предикторов. У цьому методі моделі формують ознаки і передають їх наступним моделям для покращення результату. Іншими словами, ми підбираємо послідовні дерева (випадкова вибірка), і на кожному кроці мета полягає в тому, щоб знайти чисту помилку з попереднього дерева[22].

2.5.1 Random Forest

Метод Random Forest можна розглядати як BAGGing з невеликою зміною. Потрібно зробити ще один крок, де на додаток до випадкової підмножини даних, він також приймає випадковий вибір ознак, а не використовує всі ознаки для створення нових дерев. Коли в вас є багато дерев рішень це називається Random Forest.Приклад нижче на (рис. 2.18).

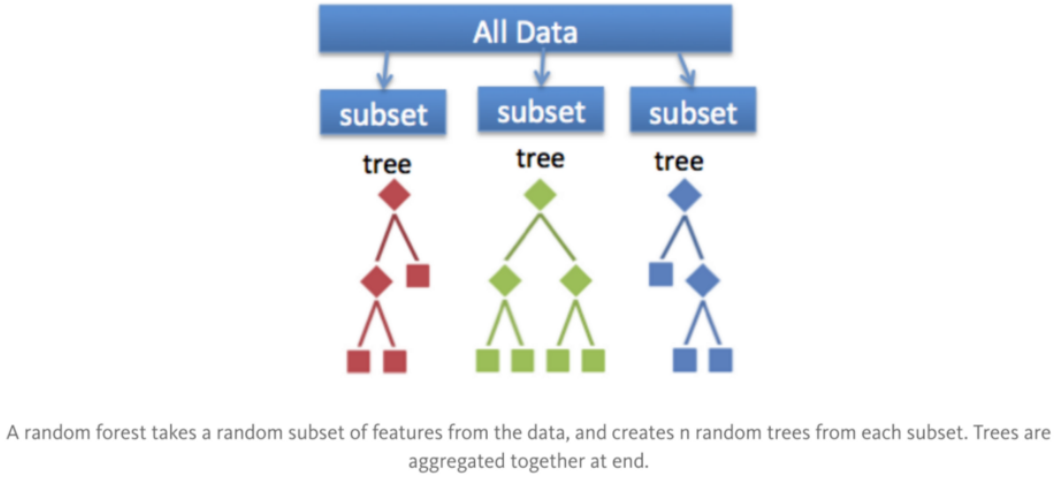


Рисунок 2.18 - Random Forest

Вирішуючи, де ділитися і як приймати рішення, дерева рішень BAGGed мають повний набір функцій на вибір. Хоча завантажені вибірки можуть трохи відрізнятися, дані в більшості випадків будуть розбиватися на одні і ті ж функції в кожній моделі. В порівнянні, моделі випадкового лісу вирішують де поділятися на основі випадкового вибору ознак. Замість того, щоб розділяти схожі ознаки в кожному вузлі, моделі Random Forest реалізують певний рівень диференціації, оскільки кожне дерево буде поділятися на основі різних ознак. Цей рівень диференціації забезпечує більший ансамбль для агрегації, отже, отриманий прогноз буде набагато точнішим[23].

Припустимо, що в початковому наборі даних є N спостережень та M ознак. Алгоритм можна поділити на наступні етапи:

1. Спочатку вибірка з набору навчальних даних береться випадковим чином із повторенням.
2. Підмножини з M об'єктів вибираються випадковим чином, обирається найкраща ознака і використовується для ітеративного поділу вузла.
3. Процес побудови відбувається до вичерпання підвибірки.

4. Остаточний прогноз дається на основі агрегації прогнозів по n кількості дерев.

2.5.2 Extremely Randomized Trees

Extremely Randomized Trees додають ще один крок рандомізації до алгоритму випадкового лісу. Random Forest обчислює оптимальний поділ, щоб забезпечити можливість охоплення ознаки в випадково обраній підмножині, і потім обирає кращу ознаку для поділу. Замість цього ExtraTrees вибере випадковий поділ для кожної ознаки в випадковій підмножині, а потім вибере кращу ознаку для поділу, і порівняє ці випадково вибрані ознаки для поділу. Надзвичайно рандомізовані дерева набагато ефективніше в обчислювальному відношенні, ніж метод випадкового лісу, і їх продуктивність майже завжди на одному рівні. У деяких випадках вони можуть навіть працювати краще[24].

2.5.3 Gradient Boosting

Gradient Boosting - це розширення методу бустінг. Gradient Boosting = градієнтний спуск + бустінг. Він використовує алгоритм градієнтного спуску, який може оптимізувати будь-яку диференційовану функцію втрат. Ансамбль дерев будується один за іншим, а окремі дерева підсумовуються послідовно. Наступне дерево намагається відновити втрату (різницю між фактичними і прогнозними значеннями)[22]. Основна ідея полягає у тому, що увага приділяється випадкам, які трудно передбачити, і модель навчається на попередніх помилках. На кожному етапі виявляється слабкий учень і його прогноз порівнюється з правильною відповіддю, що ми очікуємо. Градієнт перш за все, часна похідна від нашої функції втрат, тому він описує крутизну функції помилок.

Переваги використання техніки підвищення градієнта:

- Підтримує різні функції втрат;
- Добре працює з взаємодіями.
- Недоліки використання техніки підвищення градієнта:
- Схильний до перенавчання;
- Потрібна ретельний підбір гіперпараметрів.

2.6 Висновки

Даний розділ присвячено регресійному аналізу та методам машинного навчання, що його використовують. Сама тема регресійного аналізу є досить великою та застосовується у різних напрямках аналізу. У конкретній роботі РА використовується для прогнозування цін на нафту. Оскільки формування ціни є досить складним, то лінійні методи відпадають одразу. Я обрав підхід з використанням ансамблю дерев рішень. Цей підхід є досить оптимальним, адже завдяки своїм особливостям гарантує досить гарний результат прогнозу. Мною були розглянуті методі беггінг та бустінг. Для реалізації програмного продукту я обрав методи Random Forest, Decision Trees, GradientBoosting, ExtremelyRandomizedTrees. З моєї точки зору вони показали досить гарні результати.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Вступ

Модель для прогнозування цін на нафту була розроблена з використанням теоретичних відомостей, історичних фактів, методів інтелектуального аналізу даних та машинного навчання.

Для створення програмного продукту була обрана мова програмування Python. Вона є досить зручною у використанні, має велику швидкодію, широкий функціонал та спектр можливостей. Завдяки різноманітним бібліотекам та інструментам кінцевий продукт є дуже високої якості, а час на його розробку зменшується у порівнянні з іншими мовами програмування.

Програмний продукт є гнучким, кросплатформеним і може використовуватися на будь-якому комп’ютері з встановленим Python і відповідними бібліотеками.

3.2 Бібліотеки

Використані бібліотеки:

1. pandas – використовується для роботи з даними.
2. numpy – використовується для роботи з багатовимірними масивами та матрицями.
3. sklearn.ensemble – бібліотека для робити з ансамблями дерев рішень.
4. sklearn.tree – бібліотека для роботи з деревами рішень.
5. sklearn.linear_model – бібліотека з лінійними моделями.
6. sklearn.model_slection – бібліотека для роботи з даними.
7. sklearn.metrics – бібліотека функцій оцінки, продуктивності.
8. matplotlib.pyplot – використовується для побудови графіків.

3.3 Експерименти

Далі, на таблиці 3.1, наведено таблицю значень метрик моделей.

Таблиця 3.1 – Метрики моделей

Назва моделі	R-square Value	Mean Squared Error	Explained Variance Score	Mean Absolute Error	Median Absolute Error
Random Forest	0.9600078	46.14370	0.960588	4.770672	2.866699
Extremely Trees	0.98745845	14.4706820	0.987468	2.583831	1.5214999
Gradient Boosting	0.9833593	19.200260	0.983360	3.01084194	2.1799603
Decision Tree Regression	0.8864948	130.964509	0.891784	7.203219	3.6531451
Linear Regression	0.7018832	343.97296	0.7412638	14.553832	12.377887
Hist Gradient Boosting	0.892661	123.84887	0.90026	7.302708	3.703953

На рис. 3.1 – 3.30 наведено результати роботи програми.

```
SCORERANDOMFOREST 0.9861506979332211
SCORE_HISTGRADIENT 0.8383279532148777
SCORE_EXTRATREE 0.9917463829871511
SCOREGRADIENTBOOSTING 0.985413407951858
SCOREDECISIONTREE 0.8819225968027677
SCORELINEAR 0.5906587746235229
```

Рисунок 3.1 – Точність алгоритмів

```
 ***Random Forest Regressor***
R-Square Value: 0.9891189476534826
Mean Squared Error 9.894086251625314
Explained Variance Score 0.9892686149475304
Mean Absolute Error 2.3535088888889084
Median Absolute Error 1.747900000000005
```

Рисунок 3.2 – Оцінки методу Random Forest

```
 ***Extra Tree Model***
R-Square Value: 0.9913393285609078
Mean Squared Error 7.875105043750019
Explained Variance Score 0.9915671008708611
Mean Absolute Error 1.8502986111111133
Median Absolute Error 1.187499999999993
```

Рисунок 3.3 – Оцінки методу ExtremelyTrees

```
 ***Gradient Tree Boosting***
R-Square Value: 0.979191550572670 7
Mean Squared Error 18.921018559613387
Explained Variance Score 0.979224910822909 1
Mean Absolute Error 2.938590306751457
Median Absolute Error 1.865376525542853 2
```

Рисунок 3.4 – Оцінки методу GradientBoosting

```
 ***Decision Tree Regressor***
R-Square Value: 0.950143075615325 3
Mean Squared Error 45.334650950431154
Explained Variance Score 0.950210190725165 7
Mean Absolute Error 4.057910642489286
Median Absolute Error 2.677285714285716 3
```

Рисунок 3.5 – Оцінки методу Decision Tree Regression

```
 ***Linear Regression Model***
R-Square Value: 0.741934053053640 5
Mean Squared Error 234.65806949379123
Explained Variance Score 0.741948680419814 5
Mean Absolute Error 13.008459979142405
Median Absolute Error 11.885118740179676
```

Рисунок 3.6 – Оцінки методу Linear Regression

```
 ***HIST_Gradient Tree Boosting***
R-Square Value: 0.8383279532148777
Mean Squared Error 119.58043644576031
Explained Variance Score 0.8537347440867262
Mean Absolute Error 6.347666459293221
Median Absolute Error 3.1393096111846583
```

Рисунок 3.7 – Оцінка методу HistGradientBoosting

Рисунок 3.8 – Реальні ціни на нафту 2017 рік

Month,Year,Predicted Price
,1,2017,48.525225
,2,2017,49.82722500000016
,3,2017,49.36537499999995
,4,2017,53.39965000000026
,5,2017,57.67912500000028
,6,2017,58.42012499999998
,7,2017,52.463800000000084
,8,2017,46.77847500000013
,9,2017,46.77847500000013
,10,2017,46.77847500000013
0,11,2017,46.77847500000013
1,12,2017,46.77847500000013

Рисунок 3.9 – Прогнозування методом Random Forest

Month,Year,Predicted Price
),1,2017,52.78199999999995
.,2,2017,50.57999999999995
,3,2017,47.82000000000002
,4,2017,54.4499999999999
,5,2017,54.16209999999994
,6,2017,52.371899999999954
,7,2017,50.899999999999984
,8,2017,42.86999999999992
,9,2017,45.38699999999992
),10,2017,49.09949999999993
.0,11,2017,49.657499999999935
.1,12,2017,47.13919999999994

Рисунок 3.10 – Прогнозування ціни методом ExtraTrees

Month,Year,Predicted Price
),1,2017,47.2239317967904
l,2,2017,50.6162531301886
2,3,2017,47.80064194602698
3,4,2017,54.417009060197174
4,5,2017,51.31893791354574
5,6,2017,59.73729084562934
6,7,2017,60.99356580056546
7,8,2017,42.83952020280306
8,9,2017,40.4460626094592
9,10,2017,33.11633756620499
l0,11,2017,28.164322303529495
l1,12,2017,13.916635471499267

Рисунок 3.11 – Прогнозування ціни методом GradientBoosting

Month,Year,Predicted Price
),1,2017,52.556666666666665
,2,2017,52.556666666666665
,3,2017,52.556666666666665
,4,2017,52.556666666666665
,5,2017,52.556666666666665
,6,2017,52.556666666666665
,7,2017,46.885
,8,2017,46.885
,9,2017,46.885
,10,2017,46.885
0,11,2017,46.885
1,12,2017,46.885

Рисунок 3.12 – Прогнозування ціни методом Decision Tree

Month,Year,Predicted Price
),1,2017,91.64301934085688
L,2,2017,92.15656812415
?,3,2017,92.67011690744312
3,4,2017,93.18366569073714
4,5,2017,93.69721447403026
5,6,2017,94.21076325732338
5,7,2017,94.7243120406165
7,8,2017,95.23786082390961
3,9,2017,95.75140960720273
9,10,2017,96.26495839049585
L0,11,2017,96.77850717378897
L1,12,2017,97.292055957083

Рисунок 3.13 – Прогнозування методом лінійної регресії

Month,Year,Predicted Price
),1,2017,81.83569642906033
.,2,2017,85.68668664253124
?,3,2017,85.68668664253124
3,4,2017,85.68668664253124
4,5,2017,86.1968314888789
5,6,2017,84.29408485933438
5,7,2017,84.43837810866194
7,8,2017,90.19647662405247
3,9,2017,87.85429839346364
9,10,2017,87.03905068334353
.0,11,2017,86.37080142223523
.1,12,2017,84.69146707880039

Рисунок 3.14 – Прогнозування методом HistGradient

```
[{'normalize': True}
 {'criterion': 'mse', 'max_depth': None, 'max_features': 'auto', 'min_samples_split': 2, 'n_estimators': 450}
 {'learning_rate': 0.2, 'loss': 'ls', 'max_features': 'auto'}
 {'learning_rate': 0.2, 'loss': 'least_squares'}
 {'criterion': 'friedman_mse', 'splitter': 'best'}
 {'criterion': 'mse', 'max_depth': None, 'n_estimators': 150}]
```

Рисунок 3.15 – Використання GridSearch для пошуку оптимальних параметрів

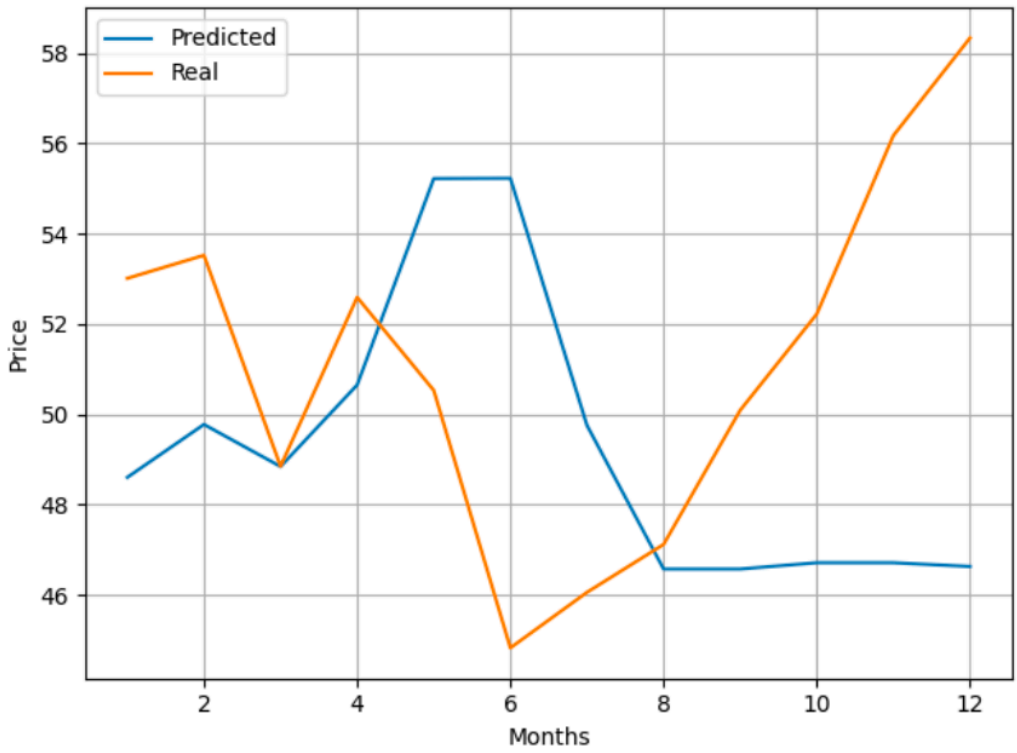


Рисунок 3.16 – Порівняння цін Random Forest та реальними

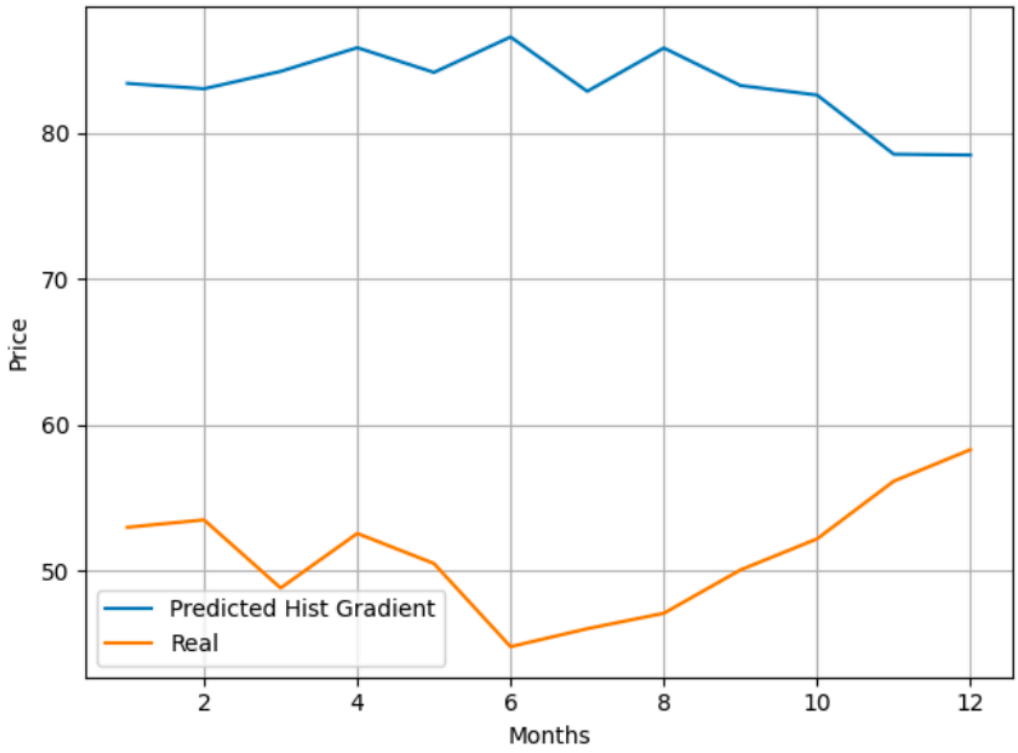


Рисунок 3.17 – Порівняння цін

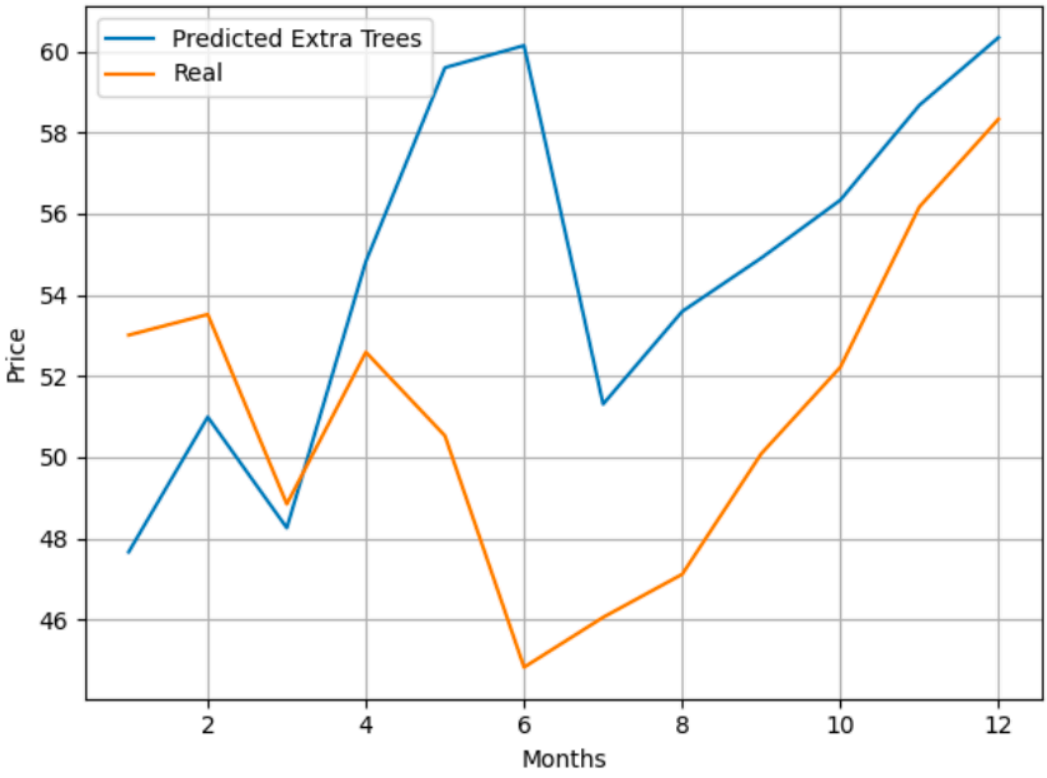


Рисунок 3.18 – Порівняння цін

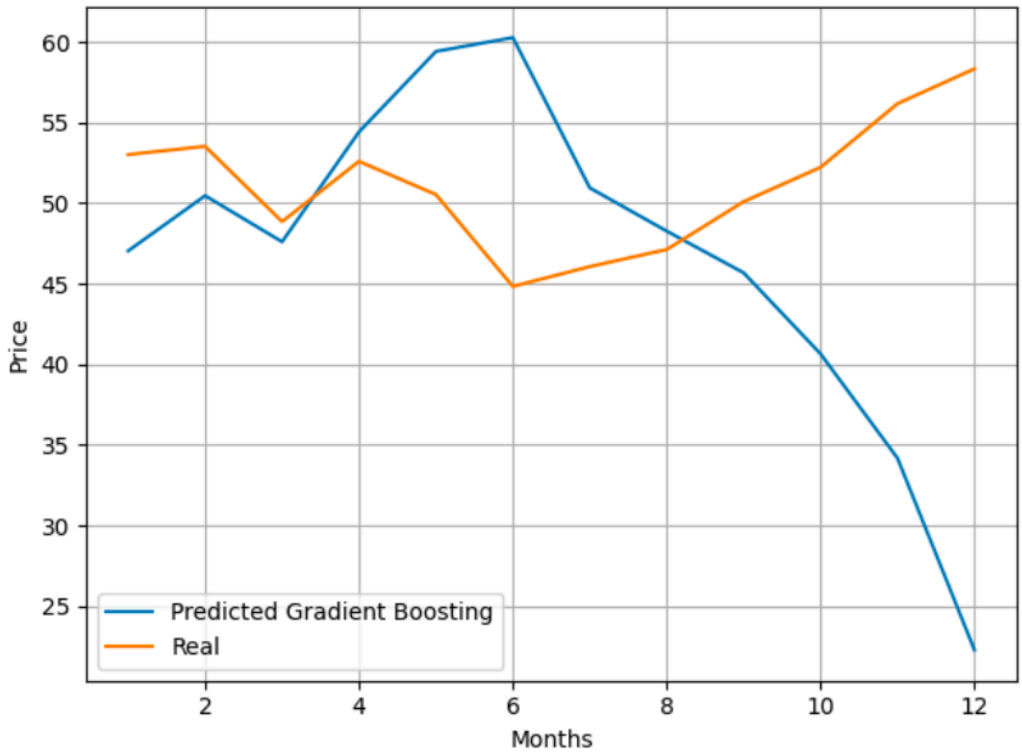


Рисунок 3.19 – Порівняння цін

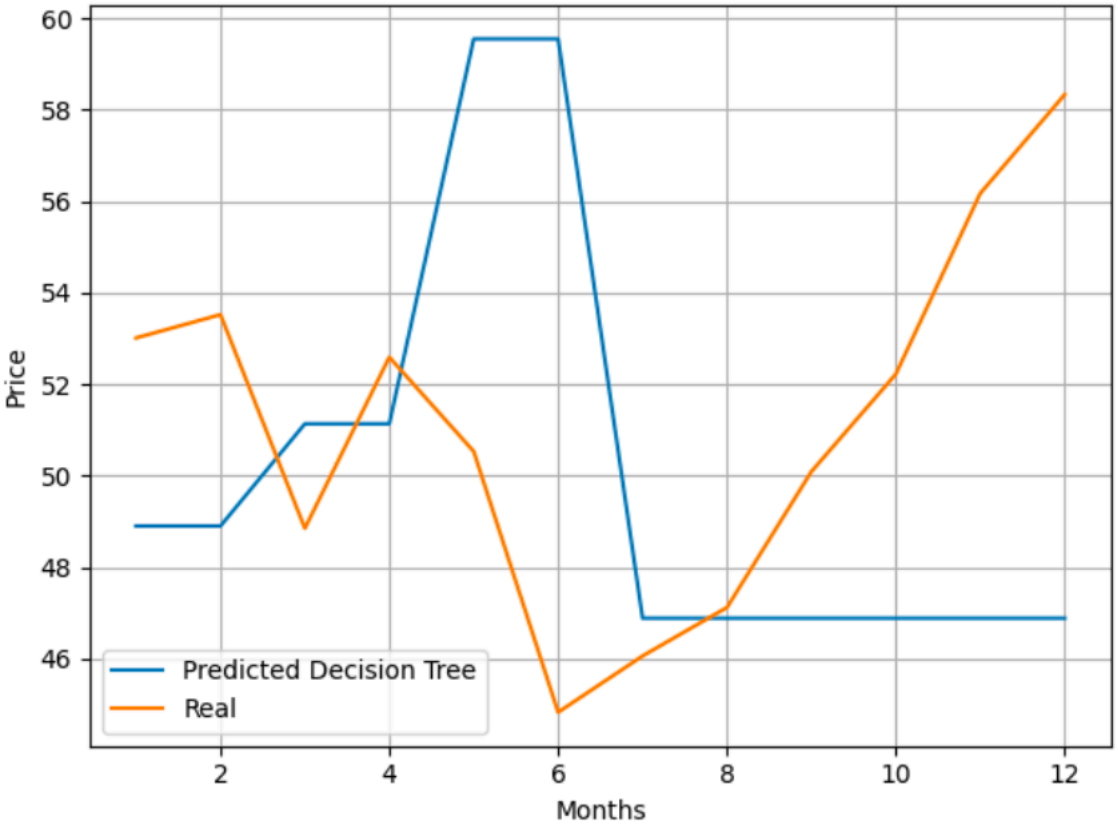


Рисунок 3.20 – Порівняння цін

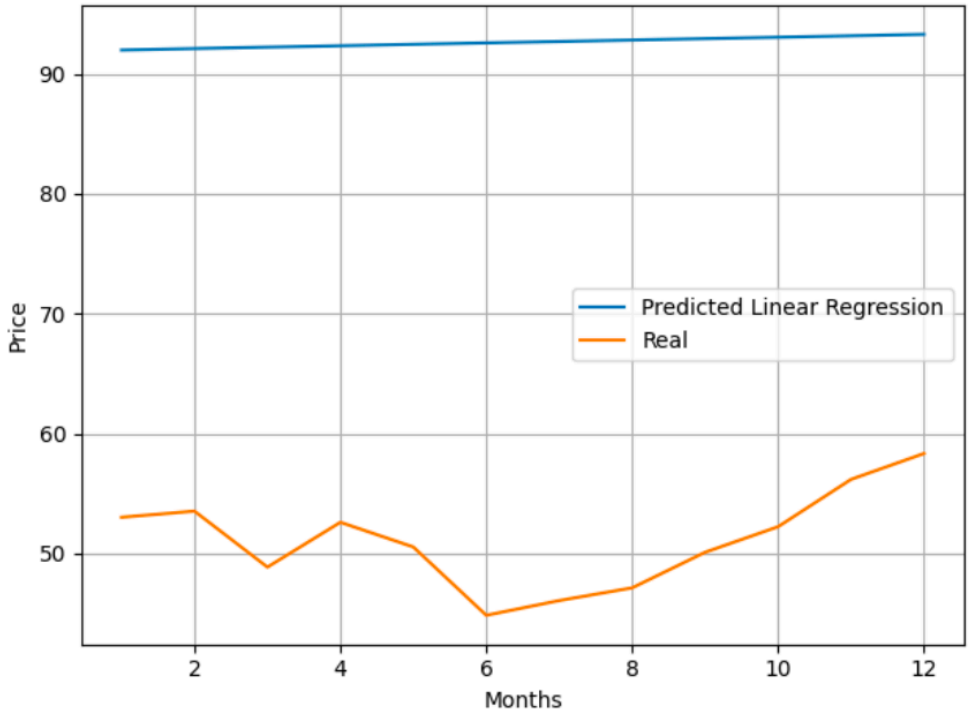


Рисунок 3.21 – Порівняння цін

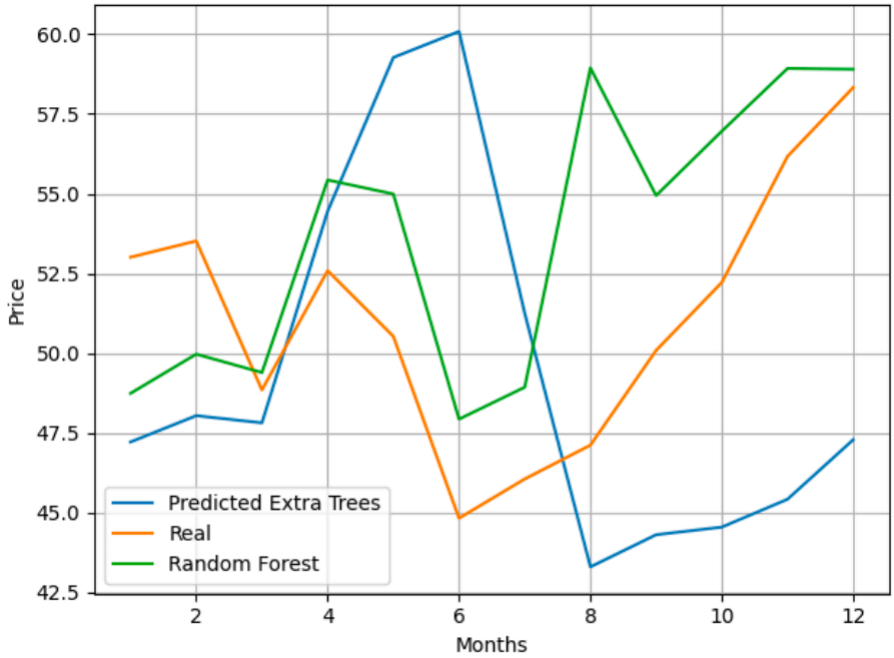


Рисунок 3.22 – Порівняння цін

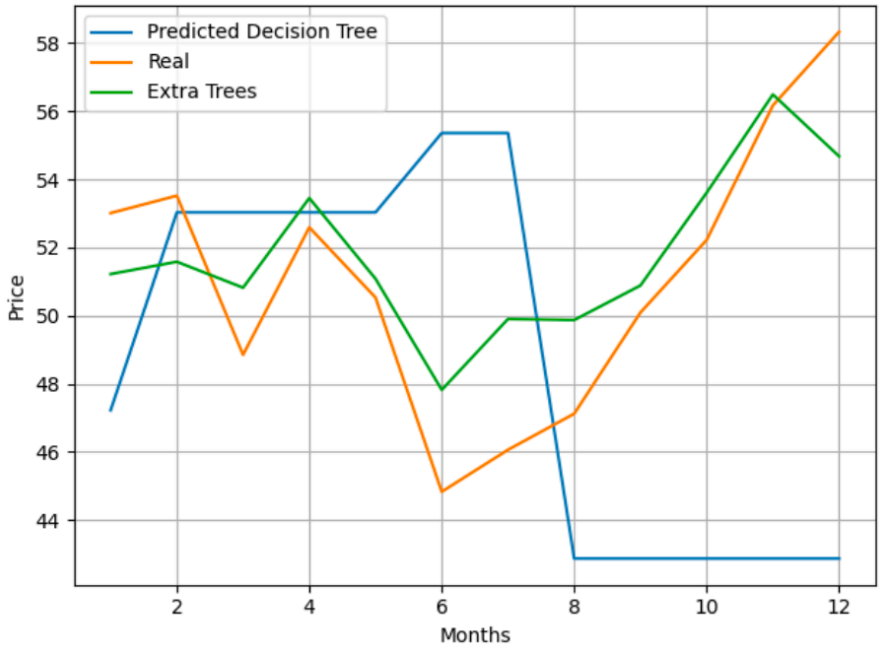


Рисунок 3.23 – Порівняння цін

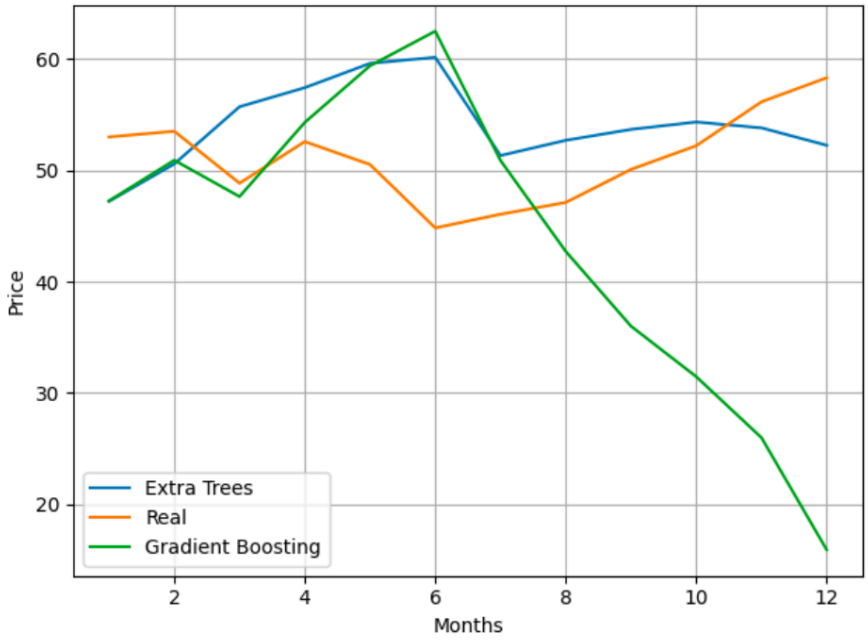


Рисунок 3.24 – Порівняння цін

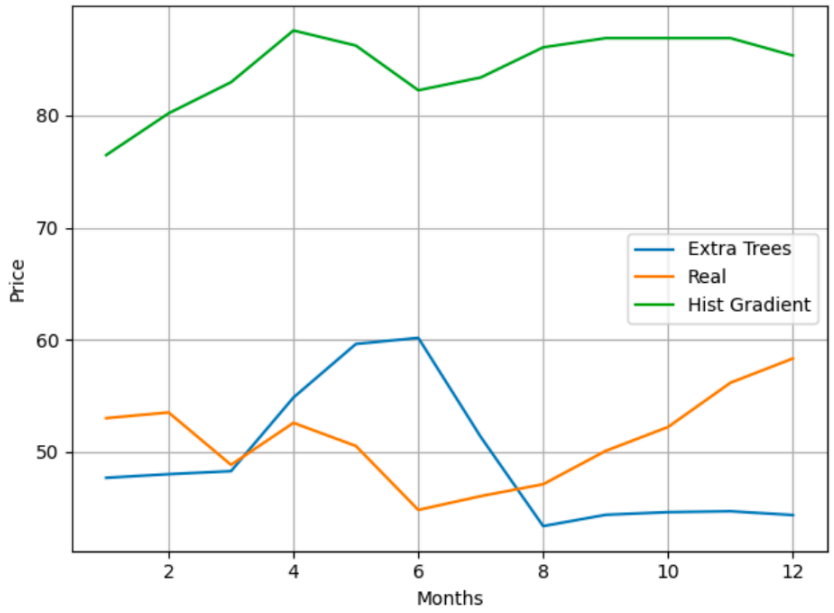


Рисунок 3.25 – Порівняння цін

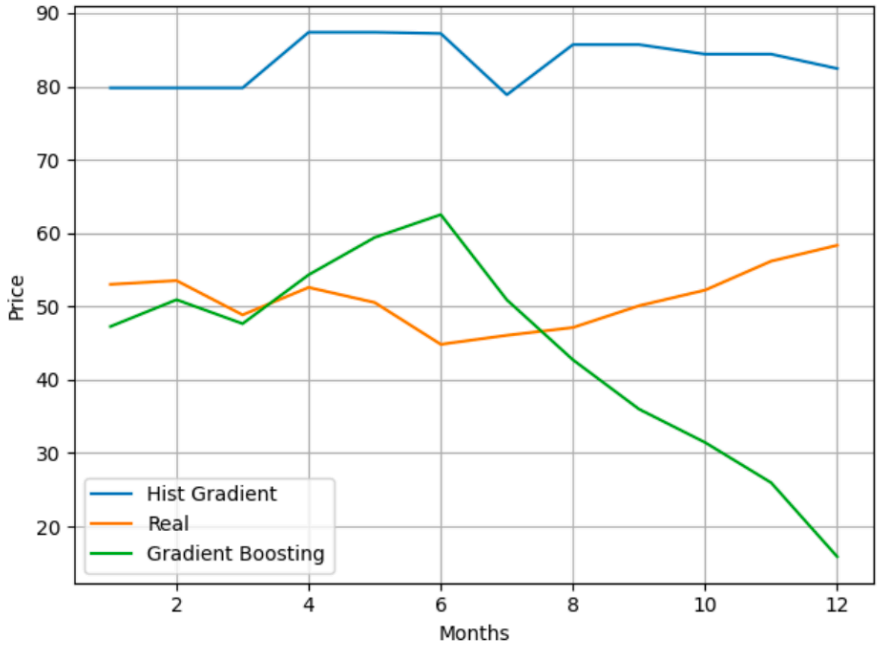


Рисунок 3.26 – Порівняння цін

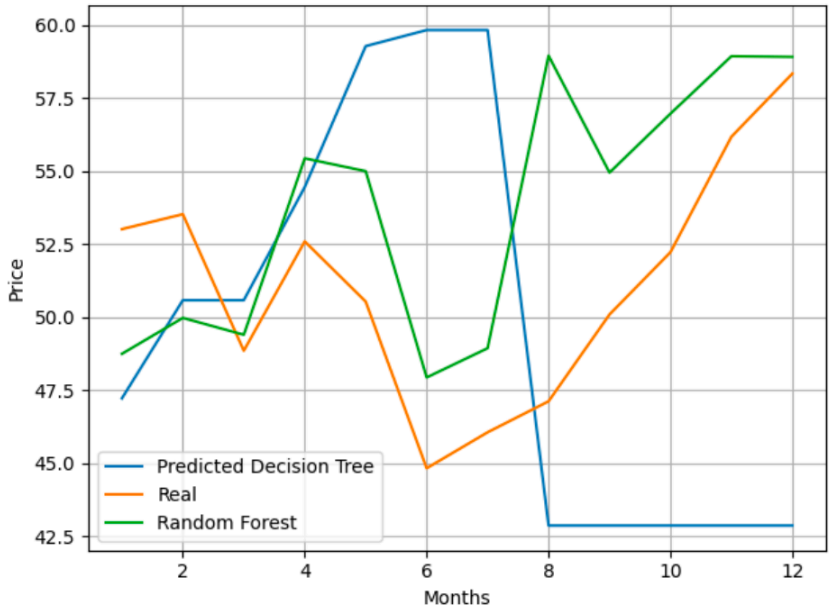


Рисунок 3.27 – Порівняння цін

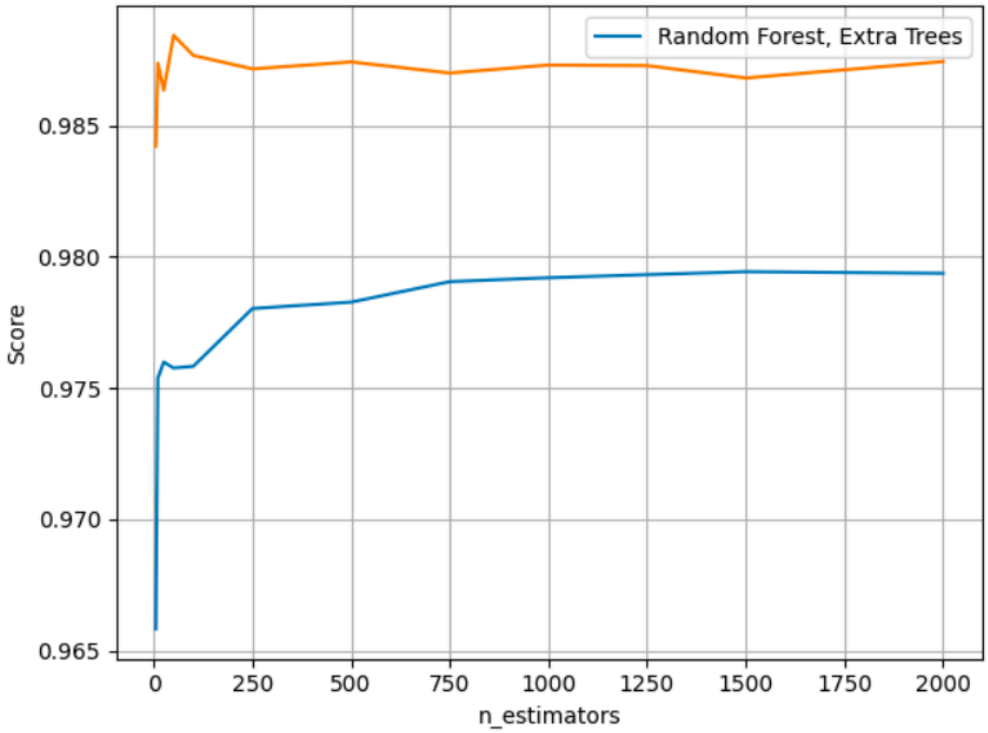


Рисунок 3.28 – Порівняння точності для Random Forest та Extra Trees

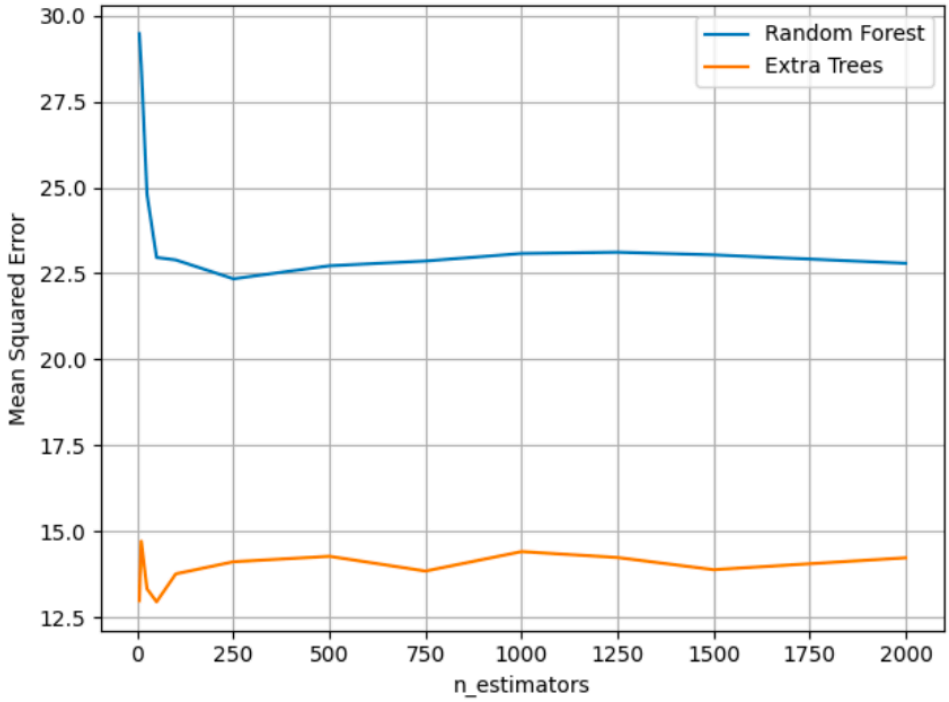


Рисунок 3.29 – Порівняння Mean Squared Error

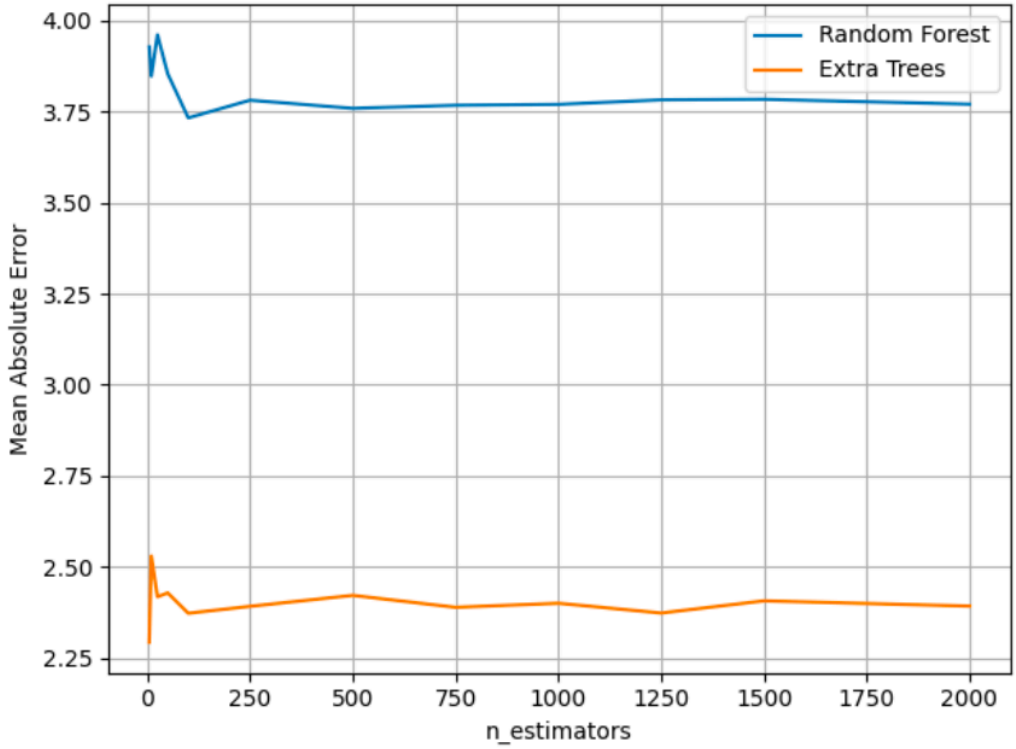


Рисунок 3.30 – Порівняння Mean Absolute Error

3.4 Висновки

В даному розділі показі експерименти при роботі з програмою. На мою думку, реалізовані алгоритми показали себе досить непогано. Результати досить різні і залежать від методу та його параметрів. Використовуючи GridSearch вдалося підібрати вдалі параметри, що позитивно вплинули на результат роботи програмного продукту. Краще всього показали себе такі методи як ExtraTrees та Random Forest, в окремих випадках різниця спрогнозованих результатів з реальними не перевищує 5%. На мою думку поставлена задача була виконана, програмний продукт є закінченим та максимально оптимізованим.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі

Спроектувати програмний продукт для прогнозування цін на нафту. ПП може бути використаний на будь-якій платформі.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує дані за допомогою методів технічного аналізу. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

- F_1 – вибір мови програмування;
- F_2 – використання готових бібліотек;
- F_3 – середовище розробки;

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

a) мова програмування Python;

b) мова програмування C#

Функція F_2:

a) використання готової бібліотеки;

b) написання алгоритмів роботи з даними вручну;

Функція F_3:

a) Python notebook;

b) IDE Pycharm;

Варіанти реалізації основних функцій наведені у морфологічній карті системи (дивись рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій.

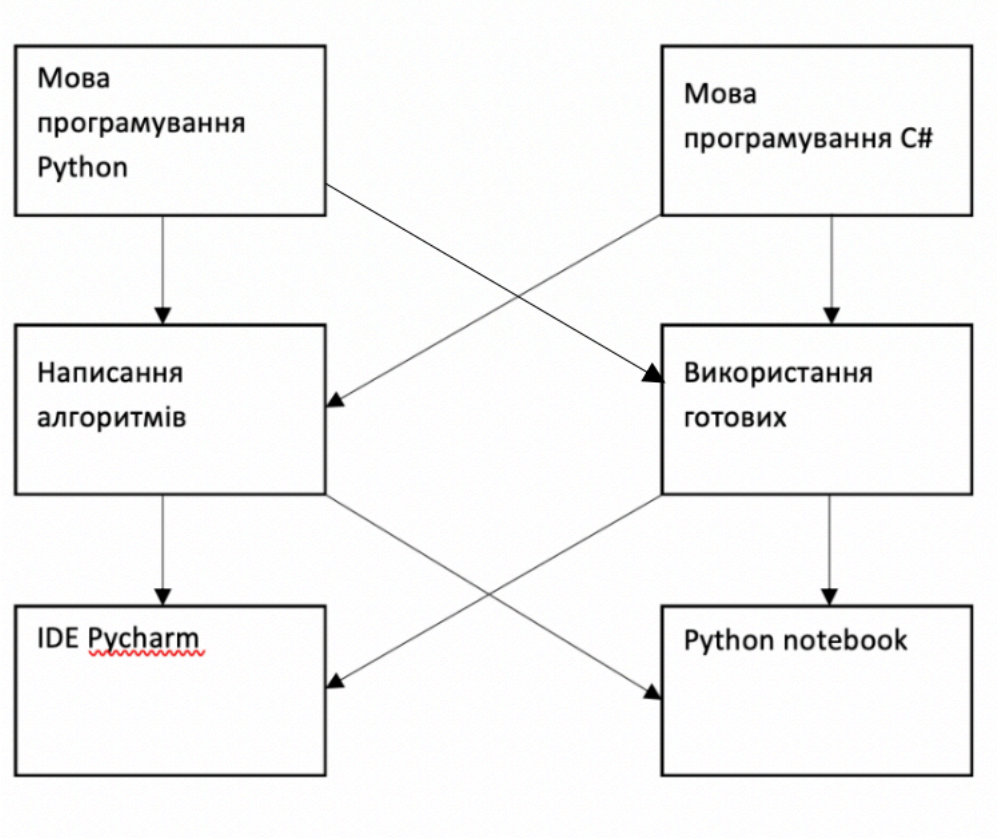


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Далі, на таблиці 4.1, наведено приклад позитивно-негативної матриці.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Займає менше часу при написанні коду	Динамічна типізація
	Б	Код швидко виконується, кросплатформений	Займає більше часу при написанні коду
F_2	А	Легкість реалізації, економія часу	Меньша гнучкість
	Б	Оптимально для власних продуктів	Затрачений час, можливі помилки
F_3	А	Широкий вибір можливостей	Відсутність відладки коду
	Б	Присутня можливість відладки коду	Необхідна додаткова інсталяція

Функція F_1:

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант Б має бути відкинутий.

Функція *F2:*

Оскільки реалізація існуючих бібліотек не визиває сумніву в їх якості, то варіант б має бути відкинутий

Функція *F3:*

Середа розробки не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти А та Б гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1)F1а – F2а – F3а

2)F1а – F2а – F3б

Для характеристики ПП будемо використовувати наступні параметри:

- X1- параметр функції F1
- X2 – парметр функції F2
- X3, X4 – парметри функції F3

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту. Інформація наведена у табл. 4.2.

Табл. 4.2 – Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Гірші	Середнє	Кращі
Швидкодія мови програмування, с	X1	17	7	3
Об’єм пам’яті для збереження даних , кб	X2	260	140	35
Час навчання моделі, хв	X3	350	230	70
Об’єм програмного коду, строк	X4	3800	1900	800

За даними таблиці 4.2 будуються графічні характеристики параметрів – (рис. 4.2 – рис. 4.5).

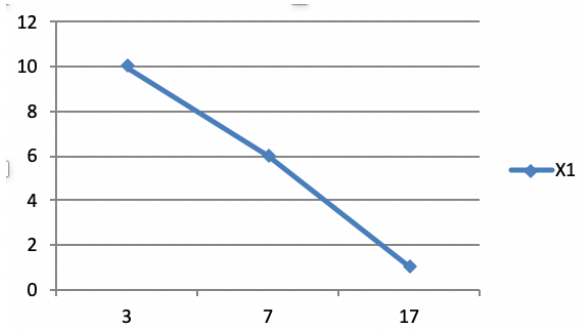


Рисунок 4.2 – X1, швидкодія мови програмування

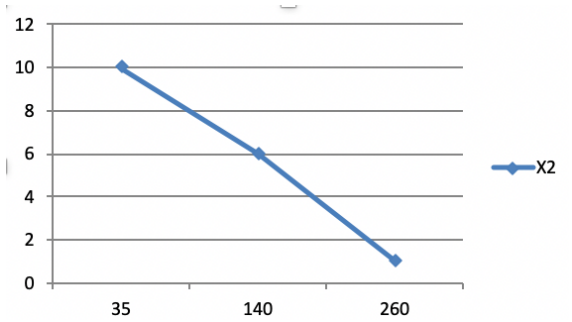


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

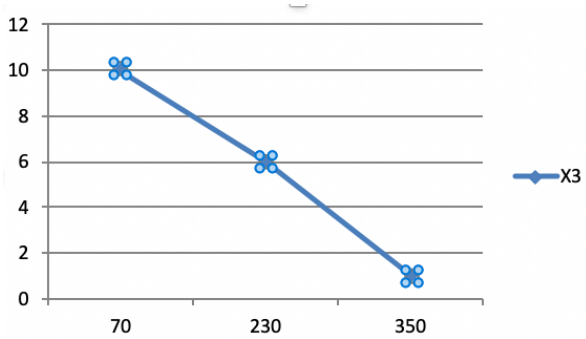


Рисунок 4.4 – X3, час обробки даних алгоритмом

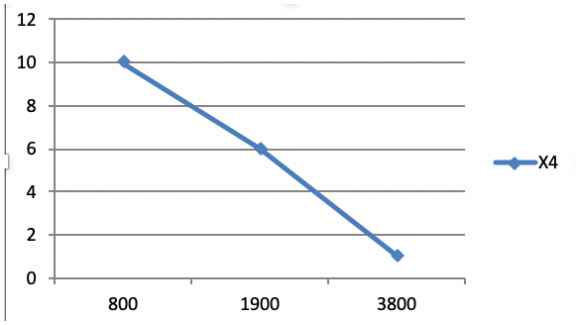


Рис. 4.5 Значення параметру X4- об’єм програмного коду, строк

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей.

У таблиці 4.3 наведено результати ранжування параметрів.

Таблиця 4.3 – Результати ранжування параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	ВідхиленняΔ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	1	2	2	1	1	2	1	10	-7,5	56,25
X2	2	1	1	2	2	1	2	11	-6,5	42,25
X3	3	4	3	4	3	3	3	23	5,5	30,25
X4	4	3	4	3	4	4	4	26	8,5	72,25
Разом	10	10	10	10	10	10	10	70	0	201

Найменший ранг – 1, найбільший ранг – 4.

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 201}{49 * (64 - 4)} = 0{,}8204 > W_k = 0{,}67$$

У таблиці 4.4 наведено попарне зрівняння параметрів.

Табл. 4.4 – Попарне зрівняння параметрів

Параметри	**Експерти**							**Кінцева оцінка**	**Числове значення**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
X1 та X2	<	>	>	<	<	>	<	<	1.5
X1 та X3	<	<	<	<	<	<	<	<	0.5
X1 та X4	<	<	<	<	<	<	<	<	0.5
X2 та X3	<	<	<	<	<	<	<	<	0.5
X2 та X4	<	<	<	<	<	<	<	<	0.5
X3 та X4	<	>	<	>	<	<	<	<	1.5

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Розрахунок вагомості параметрів наведено в табл. (4.5)

Таблиця 4.5 Розрахунок вагомості параметрів

Параметри	Параметри X_i				Перший крок		Другий крок	
	X1	X2	X3	X4	b_i	K_{bi}	b_i	K_{bi}
X1	1	0,5	0,5	0,5	2,5	0,15625	9,25	0,157
X2	1,5	1	0,5	0,5	3,5	0,21875	12,25	0,207
X3	1,5	1,5	1	0,5	4,5	0,28125	16,25	0,275
X4	1,5	1,5	1,5	1	5,5	0,34375	21,25	0,361
Загалом:					16	1	59	1

Як видно з таблиці (4.5), різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Залишаються наступні варіанти:

1)F1(а) => F2(а) => F3(а)

2)F1(а) => F2(а) => F3(б)

У таблиці 4.6 наведено розрахунок показників рівня якості варіантів Реалізації основних функцій ПП.

Таблиця 4.6 Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основна функція	Варіант реалізації	Абсолютне значення параметру	Бальна оцінка параметру –	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	а)X1	5	8	0,157	1,256
F2	а)X2	140	6	0,207	1,242
F3	а)X3	230	6	0,275	1,65
	а)X4	1900	6	0,361	2,166
	б)X3	150	8	0,275	2,2
	б)X4	1350	8	0,361	2,888

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \cdots + K_{\text{ТУ}}[F_{zk}]$$

$$K_{я1} = 1{,}256 + 1{,}242 + 1{,}65 + 2{,}166 = 6{,}314$$

$$K_{я2} = 1{,}256 + 1{,}242 + 2{,}2 + 2{,}888 = 7{,}586$$

Оскільки варіант 2 має найбільший коефіцієнт якості, він є найкращим.

4.3 Економічний аналіз варіантів розробки

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе три окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

При цьому два варіанти мають різні додаткове завдання:

3.1. Реалізація методів аналізу. Для першого варіанту.

3.2. Створення нового методу аналізу. Для другого варіанту.

Завдання 1 за ступенем новизни відноситься до групи В, ступінь складності 1 трудомісткість дорівнює: $T_Р = 43$ людино-днів завдання. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П = 0,81$Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ} = 0.8$. Тоді, трудомісткість програмування першого завдання дорівнює:

$$T_1 = 43 * 0,81 * 0,8 = 27,86 \text{ людино-днів}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання використовується алгоритм другої групи складності, степінь новизни Б, тобто

$T_р = 27$ людино-днів, $К_п = 0,9$, $К_{ск} = 1$, $К_{ст} = 0,8$:

$$T_2 = 27 \cdot 0,9 \cdot 0,8 = 19,44 \text{ людино-днів}$$

Для завдання три(а) використовується алгоритм третьої групи складності, степінь новизни Г тобто

$T_р = 64$ людино-днів, $К_{ск} = 1$, $К_{ст} = 0,6$, $К_П = 1,021$

$$T_{(а)} = 8 \cdot 0,6 \cdot 0,36 = 39,21 \text{ людино-днів}$$

Для завдання три(Б) використовується алгоритм першої групи складності, степінь новизни Б, тобто

$T_р = 8$ людино-днів, $К_{ск} = 1$, $К_{ст} = 0,6$, $К_П = 0,36$

$T_{З(а)} = 8 \cdot 0,6 \cdot 0,36 = 1,728$ людино-днів

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (27,86 + 19,44 + 39,21) \cdot 8 = 692,08$ людино-годин

$T_{II} = (27,86 + 19,44 + 1,728) \cdot 8 = 392,22$ людино-годин

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь один програміст з окладом 14000 грн., один дата аналітик з окладом 18000грн. Визначимо зарплату за годину за формулою:

$$C_ч = (14000 + 18000)/(2 * 8 * 22) = 90,90грн$$

$$С_{зп} = 90,90 \cdot 692,08 \cdot 1,2 = 75492,09 \text{ грн.}$$

$$С_{зп} = 90,90 \cdot 392,22 \cdot 1,2 = 42783,36 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{від} = С_{зп} \cdot 0.22 = 75492,09 \cdot 0.22 = 16608,26 \text{ грн.}$$

$$С_{від} = С_{зп} \cdot 0.22 = 42783,36 \cdot 0.22 = 9412,33 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години (СМ).

Так як одна ЕОМ обслуговує одного програміста з окладом 14000 грн., та датасєнтиста з окладом 18000 грн. з коефіцієнтом занятості 0,2 то для однієї машини отримаємо:

$$С_г = 12 * 14000 * 0,6 + 12 * 18000 * 0,6 = 230400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{зп} = С_г \cdot (1 + К_з) = 230400 \cdot (1 + 0.2) = 276480 \text{ грн.}$$

Відрахування на соціальний внесок 22% :

$$С_{ВІД} = 276480 * 0,22 = 60825,6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 60000 грн.

$$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1,15 * 0,25 * 60000 = 17250 \text{ грн.}$$

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} * Ц_{ПР} * К_Р = 1,15 * 60000 * 0,05 = 3450 \text{ грн.}$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (365 - 142 - 16) * 8 * 0,8 = 1324,8 \text{ год}$$

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = 1324,8 * 0,156 * 0,6 * 1,53 = 189,72 \text{ грн.}$$

Накладні витрати розраховуємо за формулою:

$$С_Н = 60000 * 0,67 = 40200 \text{ грн.}$$

Тоді, річні експлуатаційні витрати розраховуємо за формулою:

$$С_{ЕКС} = 276480 + 60825,6 + 17250 + 3450 + 189,72 + 40200 = 398395,32 \text{ грн.}$$

Тоді собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{398395,32}{1324,8} = 300,72 \text{ грн/год}$$

Враховуючи, що всі роботи ведуться на ЕОМ, витрати на оплату машинного часу:

$$С_м = 300,72 * 692,08 = 208122,298 \text{ грн.}$$

$$С_м = 300,72 * 392,22 = 117948,39 \text{ грн.}$$

Накладні витрати відповідно:

$$С_Н = 75492,09 * 0,67 = 50579,70 \text{ гн.}$$

$$С_Н = 42783,36 * 0,67 = 32684,85 \text{ грн.}$$

Розрахуємо повну вартість розробки:

$$С_{ПП} = 75492,09 + 16608,26 + 208122,298 + 50579,70 = 350799,35 \text{ грн.}$$

$$С_{ПП} = 42783,36 + 9412,33 + 117948,39 + 32684,85 = 202828,93 \text{ грн.}$$

4.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою

$$K_{я1} = 1,256 + 1,242 + 1,65 + 2,166 = 6,314$$

$$K_{я2} = 1,256 + 1,242 + 2,2 + 2,888 = 7,586$$

$$К_{ТЕР1} = \frac{6,314}{350799,35} = 1,8 * 10^{-5}$$

$$К_{ТЕР2} = \frac{7,586}{202828,93} = 3,7 * 10^{-5}$$

4.5 Висновки до розділу

Отже враховуючи всі дослідження, що описані вище, можна сказати, що другий варіант реалізації є найбільш оптимальним зі сторони якісно-економічної оцінки. Його коефіцієнт техніко-економічного рівня складає $3,7 * 10^{-5}$

Цей варіант реалізації програмного продукту має такі параметри: мова програмування Python/використання готових бібліотек/середовище розробки IDE Pycharm.

ВИСНОВКИ

В даній роботи були побудовані різні моделі для прогнозування цін на нафту. Були досліджені методи регресійного аналізу, а саме дерева рішень. Була опрацьована література, розглянуті історичні відомості про формування ринку нафти. На основі теоретичних відомостей був обраний метод регресійного аналізу , адже він якнайкраще підходить для вирішення задачі дипломної роботи. Було обрано метод дерев рішень у його різних варіаціях. Побудовані моделі були порівняні з реальними показниками на ринку. У процесі виконання роботи, було досліджено обрані моделі, виявлено залежність точності моделі від її гіперпараметрів. Завдяки обраним моделям було створено програмний продукт який дає гарні результати при тестуванні, в окремих випадках похибка між результатами програмного продукту та реальними не перевищує декількох відсотків. Після аналізу результатів програмного продукту були обрані дві моделі, що дають найкращий результат у порівнянні з іншими.

ПЕРЕЛІК ПОСИЛАНЬ

1.Reng L., History of Oil, 14.02.2020. URL: https://www.ektinteractive.com/history-of-oil/ (Last accessed: 02.04.2020).

2.Buisseret D., Francaviglia R., Saxon G., Graves J., Historic Texas from the Air 2009, p.163

3.Lamar K. International Directory of Company Histories. Journal of Saint James. 1991. Vol. 4, No. 1. P. 192-194.

4.Lioudis N., OPEC's Influence on Global Oil Prices, 21.04.2020. URL: https://www.investopedia.com/ask/answers/060415/how-much-influence-does-opec-have-global-price-oil.asp (Last accessed: 17.04.2020).

5.Dr. Edmund M. Daukoru, Oil market stability: the role of OPEC 8 September, 14.02.2006. URL: https://www.opec.org/opec_web/en/press_room/994.htm (Last acessed: 23.04.2020).

6.Amadeo K., What Affects Oil Prices? Three Critical Factors, 23.04.2020. URL: https://www.thebalance.com/how-are-oil-prices-determined-3305650 (Last accessed: 02.05.2020).

7.Bajpai P., Top Factors That Affect the Price of Oil 21.04.2020. URL: https://www.investopedia.com/articles/investing/072515/top-factors-reports-affect-price-oil.asp (Last accessed 27.04.2020).

8.Kosakowski P., What determines oil prices?, 21.04.2020. URL: https://www.investopedia.com/articles/economics/08/determining-oil-prices.asp (Last accessed: 09.05.2020).

9.Chen E., He J., Crude Oil Price Prediction with Decision Tree Based Regression Approach, 5.01.2019. URL: https://scholarworks.lib.csusb.edu/jitim/ (Last accessed: 05.05.2020).

10.Foley B., What is Regression Analysis and Why Should I Use It?, 14.02.2018. URL: https://www.surveygizmo.com/resources/blog/regression-analysis/ (Last accessed: 29.04.2020).

11.Bhalla D., 15 Types of regression in data science, 25.03.2018. URL: https://www.listendata.com/2018/03/regression-analysis.html (Last accessed: 14.05.2020).

12.Pant A., Introduction to Linear Regression and Polynomial Regression, 13.01.2019. URL: https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb (Last accessed: 12.05.2020).

13.Верморель Ж., Квантильная регрессия, 03.02.2012. URL: https://www.lokad.com/ru/квантильная-регрессия-(временные-ряды)-определение (дата звернення: 14.05.2020).

14.Stephaniem K., Ridge Regression: Simple Definition, 29.07.2017. URL: https://www.statisticshowto.com/ridge-regression/ (Last accessed: 14.05.2020).

15.Padmanabha A., Ridge Regression, 18.05.2016. URL: https://brilliant.org/wiki/ridge-regression/ (Last accessed: 16.05.2020).

16.Stephaniem K., Lasso Regression: Simple Definition, 24.09.2015. URL: https://www.statisticshowto.com/lasso-regression/ (Last accessed: 16.05.2016).

17.Sunil R., 7 Regression Techniques you should know!, 14.08.2015. URL: https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/ (Last accessed: 17.05.2020).

18.Ruczinski I., Kooperberg C., Leblanc M. Logic Regression. Journal of Computational and Graphical Statistics. 2005. Vol. 12, No. 3. P. 475-482.

19.Шахиди А., Деревья решений: общие принципы, 4.12.2019. URL: https://loginom.ru/blog/decision-tree-p1 (дата звернення: 18.05.2020).

20.Gupta P., Decision Trees in Machine Learning, 17.05.2017. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (Last accessed: 19.05.2020).

21.Geron A. Applied machine-learning using Scikit-learn and TensorFlow. 2018. Vol. 1. P. 239.

22.Nagpal A., Decision Tree Ensembles- Bagging and Boosting, 17.10.2017. URL: https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9 (Last accessed: 19.05.2020).

23.Lutins E., Ensemble Methods in Machine Learning: What are They and Why Use Them?, 02.08.2017. URL: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f (Last accesed: 20.05.2020).

24.Russel M., Decision Tree Ensemble Methods, 15.08.2017. URL: https://medium.com/@rnbrown/decision-tree-ensemble-methods-6a89181b7083 (Last accessed: 20.05.2020)

25.Недашківська Н. І., Конспект лекцій з дисципліни «Моделі і методи інтелектуального аналізу даних», 2019.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, ShuffleSplit
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingRegressor
import matplotlib.pyplot as plt

from sklearn.metrics import r2_score
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error
from sklearn.metrics import mean_squared_error

from sklearn.model_selection import GridSearchCV

# Setup paths for the input data
path = "PET_PRI_SPT_S1_M.xls"
# Path to save the test results of predicted  prices
```

```
test_path = "testResults.csv"


# Array of Dataframes input
data = []


for i in range(2):
    if (i != 0):
        data.append(pd.read_excel(path, sheet_name=i, skiprows=[0, 1]))


# result storing average value and price change per year
resultdf = []


# result storing average value and price change per year
resultdf_monthly = []


# Types of Data Available
# types = ['Crude Oil', 'Conventional Gasoline', 'RBOB Regular Gasoline', 'Heating Oil', 'Diesel Fuel',
#          'Kerosene Type Jet Fuel', 'Propane']
types = ['Çrude Oil', 'Conventional Gasoline']


#### This method applies the initial analysis of Finding Monthly and Yearly Change in Price
def analyze(data2, ind):
    monthList = []
    yearList = []
    monthDiff = []
    nextData = []
    row_iterator = data2.iterrows()
```

```
str_cols = data2.columns[data2.dtypes == object]
data2[str_cols] = data2[str_cols].fillna('.')
data2.fillna(0, inplace=True)

for index, row in row_iterator:
    current_date = row['Date']
    monthList.append(current_date.month)
    yearList.append(current_date.year)
    if (index == len(data2) - 1):
        nextData.append(data2.iloc[index][1])
        monthDiff.append(0)
    else:
        nextData.append(data2.iloc[index + 1][1])
        monthDiff.append(data2.iloc[index + 1][1] - row[1])

data2['Month'] = monthList
data2['Year'] = yearList
data2['Monthly Change in Price'] = monthDiff

# generate map from keys
map_month = dict.fromkeys(data2.Month.unique())

for month in data2.Month.unique():
    df1 = data2.loc[data2['Month'] == month]
    print('*******')
    print('For month', month)
    print(df1.iloc[:, 1])
```

```
    mean_mnth = np.asarray(df1.iloc[:, 1]).mean()
    print('mean:', mean_mnth)
    map_month[month] = mean_mnth

yearls_m = list(map_month.keys())
print(map_month)

meanls_m = list(map_month.values())

res = pd.DataFrame(np.column_stack([yearls_m, meanls_m]),
                   columns=['Month', 'Mean Price of ' + str(types[ind])])

resultdf_monthly.append(res)

# generate map from keys
map_ = dict.fromkeys(data2.Year.unique())

for yr in data2.Year.unique():
    df = data2.loc[data2['Year'] == yr]
    mean = np.asarray(df.iloc[:, 1]).mean()
    map_[yr] = mean

yearls = list(map_.keys())

meanls = list(map_.values())

result = pd.DataFrame(np.column_stack([yearls, meanls]),
                      columns=['Year', 'Mean Price of ' + str(types[ind])])
```

```
    yearDiff = []
    for index, row in result.iterrows():
        if (index == len(result) - 1):
            yearDiff.append(0)
        else:
            yearDiff.append(result.iloc[index + 1][1] - row[1])

    result['Yearly Change in Price'] = yearDiff

    resultdf.append(result)

# def using_gridsearchcv(file,imp_attr):
#
#     train, test = train_test_split(file, test_size=0.2)
#     targ = train[list(target_col)]
#     algoritmy = {
#         'linear_regression' : {
#             'model': LinearRegression(),
#             'parameters': {
#                 'normalize': [True, False]
#             }
#         },
#         'extra_tree' : {
#             'model': ExtraTreesRegressor(),
#             'parameters': {
#                 'n_estimators': [i for i in range(50, 1500, 50)],
#                 'min_samples_split': [2, 3, 4, 5],
```

```
#             'criterion': ['mse','mae'],
#             'max_features': ['auto', 'sqrt', 'log2'],
#             'max_depth': [None, 5, 8]
#         }
#     },
#     'gradient_boosting': {
#         'model': GradientBoostingRegressor(),
#         'parameters': {
#             'learning_rate': [0.1, 0.2, 0.3],
#             ## 'subsample': [1, 2, 3],
#             'max_features': ['auto', 'sqrt', 'log2'],
#             'loss': ['ls', 'lad']
#         }
#     },
#     'hist_gradient_boosting': {
#         'model': HistGradientBoostingRegressor(),
#         'parameters': {
#             'learning_rate': [0.1, 0.2, 0.3],
#             ## 'subsample': [1, 2, 3],
#             'loss': ['least_squares', 'least_absolute_deviation']
#         }
#     },
#     'decision_tree': {
#         'model': DecisionTreeRegressor(),
#         'parameters': {
#             'criterion': ['mse', 'friedman_mse'],
#             'splitter': ['best', 'random']
#         },
```

```
#         },
#         'random_forest': {
#             'model': RandomForestRegressor(),
#             'parameters': {
#                 'n_estimators': [i for i in range(50, 1500, 50)],
#                 'max_depth': [5, 8, None],
#                 'criterion': ['mse']
#             }
#         }
#     }
#     scores = []
#     cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
#     for model_name, config in algoritmy.items():
#         gs = GridSearchCV(config['model'], config['parameters'], cv=cv,
return_train_score=False)
#         print("Fit...", config['model'])
#         gs.fit(train[list(imp_attr)], targ.values.ravel())
#         scores.append({
#             'model': model_name,
#             'best_score': gs.best_score_,
#             'best_parameters': gs.best_params_
#         })


#     return (pd.DataFrame(scores, columns=['model', 'best_score', 'best_parameters']).head())



### Method where different machine learning models are trained and comparitive study is done
def trainModels(file, imp_attr):
```

```
    # split into train and test
    train, test = train_test_split(file, test_size=0.2)


    # Removing the target/predictor from the train data
    targ = train[list(target_col)]


    ### Random Forest Model
    rand_forest_model = RandomForestRegressor(n_estimators=500, max_features=2,
oob_score=True, random_state=115, criterion='mse', max_depth=None)
    rand_forest_model.fit(train[list(imp_attr)], targ.values.ravel())
    print('SCORERANDOMFOREST', rand_forest_model.score(test[list(imp_attr)],
test[list(target_col)]))


    ### HistGradientBoosting
    hs_boosting_model= HistGradientBoostingRegressor(learning_rate=0.2, loss='least_squares')
    hs_boosting_model.fit(train[list(imp_attr)], targ.values.ravel())
    print('SCORE_HISTGRADIENT', hs_boosting_model.score(test[list(imp_attr)],
test[list(target_col)]))
    ###EXtraTreeRedression Model
    extra_tree_model = ExtraTreesRegressor(criterion='mae', max_depth=None,
max_features='auto', min_samples_split=2)
    extra_tree_model.fit(train[list(imp_attr)], targ.values.ravel())
    print('SCORE_EXTRATREE', extra_tree_model.score(test[list(imp_attr)],
test[list(target_col)]))


    ### Gradient Boosting
    gr_boosting_model = GradientBoostingRegressor(n_estimators=500, learning_rate=0.2,
subsample=1, max_features='auto', loss='ls')
    gr_boosting_model.fit(train[list(imp_attr)], targ.values.ravel())
    print('SCOREGRADIENTBOOSTING', gr_boosting_model.score(test[list(imp_attr)],
test[list(target_col)]))
```

```
    ### Decision Tree Model

    decision_tree_model = DecisionTreeRegressor(max_depth=4, criterion='mse',
splitter='random')

    decision_tree_model.fit(train[list(imp_attr)], targ)

    print('SCOREDECISIONTREE', decision_tree_model.score(test[list(imp_attr)],
test[list(target_col)]))


    ### Linear Regression Model

    linear_model = LinearRegression(normalize=True)

    linear_model.fit(train[list(imp_attr)], targ)

    print('SCORELINEAR', linear_model.score(test[list(imp_attr)], test[list(target_col)]))

    return rand_forest_model, hs_boosting_model, extra_tree_model, gr_boosting_model,
decision_tree_model, linear_model, test




#### Method which tests the given model and Prints out the statistics regarding each of them

def testNEvalModels(test, rf_model, hs_model, et_model, gd_model,  dt_model, lm_model,
imp_attr):

    print('\n Evaluation Staistics:')

    ### Evaluating Random Forest

    print("\n ***Random Forest Regressor***")

    # Evaluation metric: r square

    r2 = r2_score(test[list(target_col)], rf_model.predict(test[list(imp_attr)]))

    print("R-Square Value:", r2)


    # extracting the test target values and convert to float

    true_vals = test[list(target_col)].values

    true_vals_flt = true_vals.astype(np.float)
```

```
prediction = rf_model.predict(test[list(imp_attr)])

# reshaping the array is required to convert it into numpy array
aa = prediction.reshape(-1, 1)

mean_squared_error(true_vals_flt, prediction)

mse = np.mean((true_vals_flt - aa) ** 2)
print("Mean Squared Error", mse)

print("Explained Variance Score", explained_variance_score(true_vals_flt, prediction))
print("Mean Absolute Error", mean_absolute_error(true_vals_flt, prediction))
print("Median Absolute Error", median_absolute_error(true_vals_flt, prediction))
### Evaluating HIST_Gradient Method
print("\n ***HIST_Gradient Tree Boosting***")
# Evaluation metric: r square
r2_hs = r2_score(test[list(target_col)], hs_model.predict(test[list(imp_attr)]))
print("R-Square Value:", r2_hs)

# extracting the test target values and convert to float
true_vals = test[list(target_col)].values
true_vals_flt = true_vals.astype(np.float)

prediction = hs_model.predict(test[list(imp_attr)])

# reshaping the array is required to convert it into numpy array
aa = prediction.reshape(-1, 1)
```

```
mean_squared_error(true_vals_flt, prediction)

mse = np.mean((true_vals_flt - aa) ** 2)
print("Mean Squared Error", mse)

print("Explained Variance Score", explained_variance_score(true_vals_flt, prediction))
print("Mean Absolute Error", mean_absolute_error(true_vals_flt, prediction))
print("Median Absolute Error", median_absolute_error(true_vals_flt, prediction))
### Evaluating Extra Tree Model
print("\n ***Extra Tree Model***")
# Evaluation metric: r square
r2_et = r2_score(test[list(target_col)], et_model.predict(test[list(imp_attr)]))
print("R-Square Value:", r2_et)

# extracting the test target values and convert to float
true_vals = test[list(target_col)].values
true_vals_flt = true_vals.astype(np.float)

prediction = et_model.predict(test[list(imp_attr)])

# reshaping the array is required to convert it into numpy array
aa = prediction.reshape(-1, 1)

mean_squared_error(true_vals_flt, prediction)

mse = np.mean((true_vals_flt - aa) ** 2)
print("Mean Squared Error", mse)
```

```
print("Explained Variance Score", explained_variance_score(true_vals_flt, prediction))
print("Mean Absolute Error", mean_absolute_error(true_vals_flt, prediction))
print("Median Absolute Error", median_absolute_error(true_vals_flt, prediction))
### Evaluating Gradient Method
print("\n ***Gradient Tree Boosting***")
# Evaluation metric: r square
r2_gd = r2_score(test[list(target_col)], gd_model.predict(test[list(imp_attr)]))
print("R-Square Value:", r2_gd)

# extracting the test target values and convert to float
true_vals = test[list(target_col)].values
true_vals_flt = true_vals.astype(np.float)

prediction = gd_model.predict(test[list(imp_attr)])

# reshaping the array is required to convert it into numpy array
aa = prediction.reshape(-1, 1)

mean_squared_error(true_vals_flt, prediction)

mse = np.mean((true_vals_flt - aa) ** 2)
print("Mean Squared Error", mse)

print("Explained Variance Score", explained_variance_score(true_vals_flt, prediction))
print("Mean Absolute Error", mean_absolute_error(true_vals_flt, prediction))
print("Median Absolute Error", median_absolute_error(true_vals_flt, prediction))
### Evaluating Decision tree
```

```
    print("\n ***Decision Tree Regressor***")
    y_2 = dt_model.predict(test[list(imp_attr)])
    r2_dt = r2_score(true_vals_flt, y_2)
    print("R-Square Value:", r2_dt)

    print("Mean Squared Error", mean_squared_error(true_vals_flt, y_2))

    print("Explained Variance Score", explained_variance_score(true_vals_flt, y_2))
    print("Mean Absolute Error", mean_absolute_error(true_vals_flt, y_2))
    print("Median Absolute Error", median_absolute_error(true_vals_flt, y_2))

    ### Evaluating Linear Model
    print("\n ***Linear Regression Model***")
    pred_lm = lm_model.predict(test[list(imp_attr)])

    r2_lm = r2_score(true_vals_flt, pred_lm)

    print("R-Square Value:", r2_lm)

    print("Mean Squared Error", mean_squared_error(true_vals_flt, pred_lm))

    print("Explained Variance Score", explained_variance_score(true_vals_flt, pred_lm))
    print("Mean Absolute Error", mean_absolute_error(true_vals_flt, pred_lm))
    print("Median Absolute Error", median_absolute_error(true_vals_flt, pred_lm))

### ESTIMATORY NA GRAPHIKI
N_EST = [5, 10, 25, 50, 100, 250, 500, 750, 1000, 1250, 1500, 2000]
def ForGraphs(file, imp_attr):
```

```
    train, test = train_test_split(file, test_size=0.2)
    targ = train[list(target_col)]
    SCORE = []
    MAE = []
    MSE = []
    MEDAE = []

    MODELS_NAME = ["RANDOM FOREST", "EXTRA TREES"]
    for i in N_EST:
        MODELS = [
            RandomForestRegressor(n_estimators=i, max_features=2, oob_score=True,
random_state=115, criterion='mse',
                                  max_depth=None),
            ExtraTreesRegressor(n_estimators=i, criterion='mae', max_depth=None,
max_features='auto',
                                min_samples_split=2)
        ]
        for j, k in zip(MODELS, MODELS_NAME):
            print("\n ***", k, "***", "N_est=", i)
            model = j
            model.fit(train[list(imp_attr)], targ.values.ravel())
            print(model.score(test[list(imp_attr)], test[list(target_col)].values.ravel()))
            print(model.score(test[list(imp_attr)], test[list(target_col)].values.ravel()))

            # Evaluation metric: r square
            r2 = r2_score(test[list(target_col)], model.predict(test[list(imp_attr)]))
            SCORE.append(r2)
            print("R-Square Value:", r2)
```

```
        # extracting the test target values and convert to float
        true_vals = test[list(target_col)].values
        true_vals_flt = true_vals.astype(np.float)

        prediction = model.predict(test[list(imp_attr)])

        # reshaping the array is required to convert it into numpy array
        aa = prediction.reshape(-1, 1)

        mean_squared_error(true_vals_flt, prediction)

        mse = np.mean((true_vals_flt - aa) ** 2)
        MSE.append(mse)
        MAE.append(mean_absolute_error(true_vals_flt, prediction))
        MEDAE.append(median_absolute_error(true_vals_flt, prediction))
        print("Mean Squared Error", mse)
        print("MSE:", mean_squared_error(true_vals_flt, prediction))
        print("Explained Variance Score", explained_variance_score(true_vals_flt, prediction))
        print("Mean Absolute Error", mean_absolute_error(true_vals_flt, prediction))
        print("Median Absolute Error", median_absolute_error(true_vals_flt, prediction))
    return MSE, MAE, MEDAE , SCORE
#### The chosen model is then applied to predict the future crude oil prices
def applyModel(model, test):
    # Apply selected model to test data
    pred = model.predict(test[list(imp_attr)])

    # save predicted into target column in test
    test_dtm['Predicted Price'] = pred
```

```
    # save df to file
    test_dtm.to_csv(test_path)



# Create a Excel Writer Object to Write Yearly analysis Price Change
writer = pd.ExcelWriter('output_y.xlsx')
writer_m = pd.ExcelWriter('output_m.xlsx')

for i in range(len(data)):
    analyze(data[i], i)

# Storing resultant yearly analysis
for i in range(len(resultdf)):
    resultdf[i].to_excel(writer, 'data' + str(i))
writer.save()

# Storing resultant yearly analysis
for i in range(len(resultdf_monthly)):
    resultdf_monthly[i].to_excel(writer_m, 'data' + str(i))
writer_m.save()

##### Code to predict the next 6 months Prices

# Crude oil data being taken as input
train_data = data[0]
target_col = ["Cushing, OK WTI Spot Price FOB (Dollars per Barrel)"]
```

```
# features selected on which the model would be based on
imp_attr = ['Month', 'Year']

# Training different models and choosing the best one for prediction
# using_gridsearchcv(train_data, imp_attr)
rf, hs, et, gd, dt, lm, test = trainModels(train_data, imp_attr)
testNEvalModels(test, rf, hs, et, gd, dt, lm, imp_attr)

# create test data for next 12 months
test_dtm = {}
test_dtm['Month'] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
test_dtm['Year'] = [2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017]
test_dtm = pd.DataFrame.from_dict(test_dtm)
real_p = [53.01,53.52,48.85,52.59,50.53,44.83,46.06,47.12,50.09,52.22,56.17,58.33]

# As per the Evaluation Statistics we get to know that Random Forest
# is performing better than other considered models, so we apply rf to test data
applyModel(lm, test_dtm)
# fig, ax = plt.subplots()
# ax.plot(tuple(test_dtm['Month']), tuple(test_dtm['Predicted Price']))
# ax.plot(tuple(test_dtm['Month']), tuple(real_p))
# ax.grid()
# ax.set_xlabel('Months')
# ax.set_ylabel('Price')
# ax.legend(['Predicted Linear Regression', 'Real'])
# plt.show()

MSE, MAE, MEDAE, SCORE = ForGraphs(train_data, imp_attr)
```

```
RF_SCORE = [v for k,v in enumerate(SCORE) if not k%2]
ET_SCORE = [v for k,v in enumerate(SCORE) if k%2]

RF_MSE = [v for k,v in enumerate(MSE) if not k%2]
ET_MSE = [v for k,v in enumerate(MSE) if k%2]

RF_MAE = [v for k,v in enumerate(MAE) if not k%2]
ET_MAE = [v for k,v in enumerate(MAE) if k%2]

RF_MEDAE = [v for k,v in enumerate(MAE) if not k%2]
ET_MEDAE = [v for k,v in enumerate(MAE) if k%2]
#
# print(SCORE)
# print(RF_SCORE)
# print(ET_SCORE)
# print(MSE)
# print(MAE)
# print(MEDAE)
#
# fig, ax = plt.subplots()
# ax.plot(tuple(N_EST), tuple(RF_SCORE))
# ax.grid()
# ax.set_xlabel('n_estimators')
# ax.set_ylabel('Score')
# ax.legend(['Random Forest'])
# plt.show()
#
# fig, ax = plt.subplots()
```

```
# ax.plot(tuple(N_EST), tuple(ET_SCORE))
# ax.grid()
# ax.set_xlabel('n_estimators')
# ax.set_ylabel('Score')
# ax.legend(['Extra Trees'])
# plt.show()
#
#
fig, ax = plt.subplots()
ax.plot(tuple(N_EST), tuple(RF_MAE))
ax.plot(tuple(N_EST), tuple(ET_MAE))
ax.grid()
ax.set_xlabel('n_estimators')
ax.set_ylabel('Mean Absolute Error')
ax.legend(['Random Forest', 'Extra Trees'])
plt.show()

# print('DATA',data[0])
print("Yearly Change in Price is stored in file named: output_y.xlsx")
print("Monthly Change in Price is stored in file named: output_m.xlsx")
print("The resultant Predicted Values are stored in a File named: testResults.csv")
```

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Прогнозування цін на нафту

Виконав: Сніжко Андрій Сергійович

Науковий керівний:
к.т.н. доцент Недашківська Надія Іванівна

Постановка задачі

- Провести дослідження ринку нафти, ознайомитися з історичними та теоретичними відомостями, що впливають на формування ринку нафти
- Проаналізувати вхідні дані, обрати метод дослідження, продумати модель для прогнозування ціни
- Застосувати обраний метод на модель, розробити програмний продукт

Актуальність роботи

Виробництво та споживання нафтових ресурсів, що прийшли на початку століття на заміну вугіллю та деревині, зростає з кожним роком. Нині контроль над паливно-енергетичними ресурсами має великий вплив на економічний та політичний стан країни на світовій арені в цілому. Прогнозування цін на нафту є досить важкою але дуже необхідною задачею, адже нафта є основою економіки багатьох країн.

Об’єкт дослідження

Об’єкт дослідження – вибірка цін на нафту з 1986 року

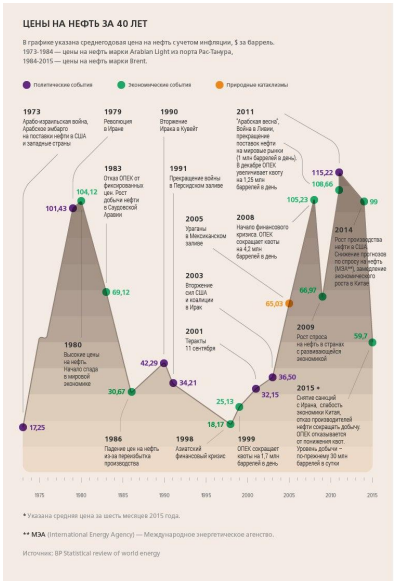
Предмет дослідження

Предмет дослідження – методи інтелектуального аналізу даних на основі регресії з використанням дере рішень

Мета роботи

Розробка програмного продукту та побудування моделі для прогнозування цін на нафту

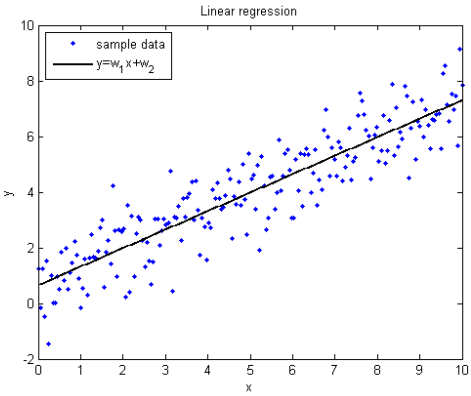
Ціни на нафту за останні 40 років



Рекордне падінні цін через COVID-19

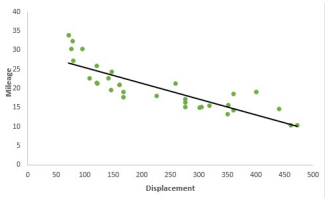
Поняття регресії

- Регресія – одностороння стохастична залежність, що встановлює відповідність між випадковими змінними. На відміну від суто функціональної залежності $y = f(x)$, коли кожне значення незалежної змінної x відповідає одному визначеному значенню y, у випадку регресії одне і те ж значення x може відповідати різним значенням y

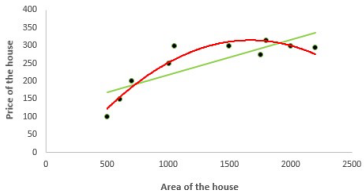


Види регресії

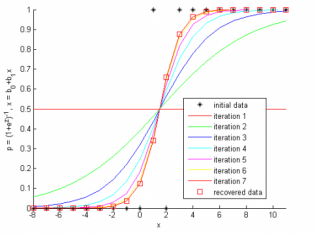
- лінійна
- поліноміальна
- логістична
- Квантільна
- гребнева
- ласо
- еластична сітка
- байесова
- логічна



Лінійна регресія - найпростіша форма регресії. Це метод, в якому залежна змінна є безперервною за своєю природою . Формула простої лінійної регресії:
$y = f(x, b) + \varepsilon, E(\varepsilon) = 0,$
де b – параметри моделі, ε – випадкова помилка моделі.
$f(x, b)$ має вигляд $f(x, b) = b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k$,
де b_j - параметри (коефіцієнти) регресії, x_j - регресори (фактори моделі), k - кількість факторів моделі.

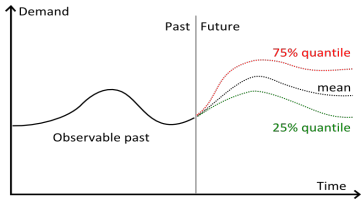


Поліноміальна регресія використовується для криволінійних даних. Поліноміальна регресія відповідає методу найменших квадратів. Мета регресійного аналізу - змоделювати очікуване значення залежної змінної y щодо незалежної змінної x. На рисунку видно, що червона крива відповідає даним краще, ніж зелена. Отже, в ситуаціях, коли зв'язок між залежною і незалежною змінними здається нелінійним, ми можемо використовувати моделі поліноміальної регресії. Загальне рівняння поліноміальної регресії:
$y = \beta_0 + \beta_1X + \beta_2X^2 + \cdots + \beta_2X^k + \varepsilon$, де
β_j – параметри полінома, X^j – змінні, ε – похибка.

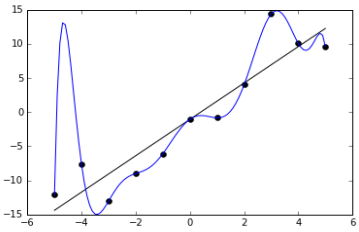


У логістичної регресії залежна змінна має двійковий характер (має дві категорії). Незалежні змінні можуть бути безперервними або двійковими. У многочленній логістичній регресії ви можете мати більше двох категорій в залежній змінній. Логістична модель має вигляд:
$p = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+..+b_kx_k)}}$, де

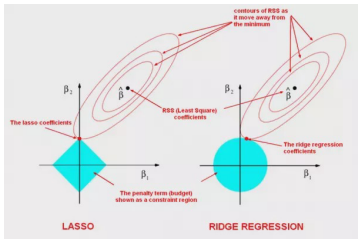
p – логістична функція, β^tX – вектори-стовпці значень незалежних змінних $X_1, \ldots, X_k$ і параметрів (коефіцієнтів регресії) – дійсні числа $\beta_1, \ldots, \beta_k$



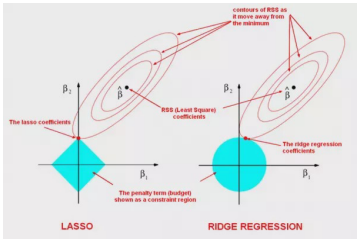
Квантільна регресія – регресія(прогноз), що спеціально вводить зміщення у результат. Замість пошуку середнього значення передбачуваної змінної, квантільна регресія прагне знайти медіану та будь-які інші кванти (які іноді називають «процентілями»). Квантілі особливо корисні для оптимізації товарних запасів. Поняття квантільної регресії представляє більш сучасну галузь статистики. Формула моделі регресії має вигляд:
$Q_\tau(y_i) = \beta_0(\tau) + \beta_1(\tau)x_{i1} + \cdots + \beta_p(\tau)x_{ip}, i = 1, \ldots, n,$
де
$\beta_p(\tau)$ – функція залежності від квантіля, x_{ip} – регресор



Гребнева регресія – один із методів для зменшення розмірності. Найчастіше використовується для боротьби із надмірністю даних, коли незалежні змінні корелюють одна з одною. Наприклад, для рівняння $Ax = b$ не існує єдиного розв'язку x, тоді ми можемо використати гребеневу регресію:
$\min||Ax - b||^2 + ||\Gamma x||^2$, де
Γ – матриця Тихонова, A – матриця, b – вектор.
Гребенева регресія запобігає перенавчанню та недонавчанню шляхом введення штрафної функції $||\Gamma x||^2$, де Γ – матриця Тихонова
Синя крива мінімізує похибку точок даних. Іноді це може привести до перенавчання. Введення матриці Тихонова відображається у чорну криву. Треба розуміти що вона не мінімізує помилки, але захищає від перенавчаннях



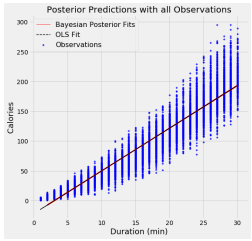
Суть регресії за методом «Ласо» є в тому, що вводиться додатковий додаток в функціонал оптимізації моделі, що часто дозволяє отримувати більш стійке рішення. Умова мінімізації квадратів помилки при оцінці параметрів $\hat{\beta}$ виражається наступною формулою:
$\hat{\beta} = argmin(\sum_{i=1}^{n}(y_j - \sum_{j=1}^{m}\beta_j x_{ij})^2 + \lambda|\beta|)$, де
λ – параметр регуляризації, що має сенс штрафу за складність.



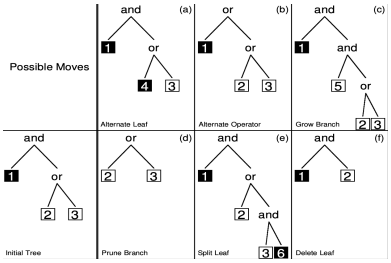
Еластична сітка - це гібрид методів регресії «Лассо» і «Гребнева». Він тренується з L1 і L2 регуляризаціями і бере найкраще. Еластична сітка корисна, коли є кілька взаємопов'язаних функцій. Метод «Лассо» ймовірно, обере один з них випадковим чином, в той час як еластична сітка, швидше за все, вибере обидва.
Виходячи з цього завдання оптимізації можна побачити на формулі.

$$\hat{\beta} = (\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta_1\|)$$

Практична перевага вибору компромісу між «Лассо» і «Гребневой» полягає в тому, що воно дозволяє «ElasticNet» успадковувати частину стабільності «Гребневой» регресії при обертанні



Байєсова регресія схожа на гребневу регресію, проте заснована на тому припущенні, що в даних шум (помилка) розподілений нормально - відповідно, передбачається, що загальне розуміння про структуру даних вже є, і це дає можливість отримувати більш точну модель (у порівнянні з лінійною регресією точно). Але у реальному житті, особливо коли ми працюємо з великим набором даних, початкові знання про дані не можуть похвалитися точністю, тому припущення будується на підставі пов'язаних величин, тобто воно штучно за своєю суттю. Формула для вирішення задачі має наступний вигляд:
$\tau = y(x, w) + \varepsilon$,
де τ – спостережувана змінна, а $\varepsilon \sim N(0, \sigma^2)$ – нормально розподілена помилка.



Логічна регресія – вид регресії, що використовується у випадку, коли змінні двійкові. Загалом, взаємодії між змінними враховуються не часто, ці взаємодії зазвичай залишаються простими (не більше двох-тристоронніх взаємодій). Але часто, особливо коли все змінні є двійковими, їх взаємодія є причиною відмінностей у відповідях.

Логічна регресія дозволяє дослідити, як зв'язок між змінними впливає на результат загалом. Приклад моделі логічної регресії:

$$g\big(E(y)\big) = \beta_0 + \sum_{j=1}^{t} \beta_j L_j$$

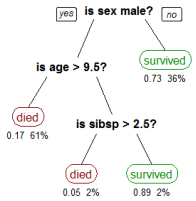
Де L_j — булева функція від змінних X_j.

Decision Trees

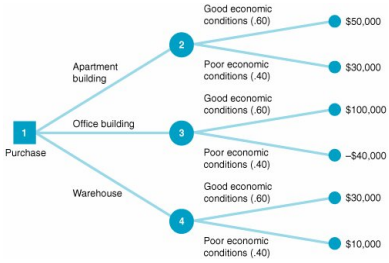
- Дерева рішень («Decision Trees») - це статистичний метод, що дозволяє передбачати приналежність спостережень або об'єктів до того чи іншого класу категоріальної залежною змінною або середнє значення кількісної змінної в залежності від відповідних значень однієї або декількох незалежних змінних.
- Мета методу дерев рішень – спрогнозувати значення цільової змінної в залежності від відповідних значень незалежних змінних (предикторів, атрибутів). В свою чергу, дерева рішень поділяються на дерева регресії і дерева класифікації.
- Якщо мова йде про дерева регресії, прогнозується значення цільової змінної, що залежить від відповідних значень предикторів.
- В деревах класифікації все виглядає інакше. Робиться прогноз приналежність об'єкта до тієї чи іншої категорії цільової змінної в залежності від відповідних значень предикторів.
- Наприклад, класифікуються добрі і погані учні в залежності від їх оцінок. Дерева класифікації працюють з Залежність значення цільової змінної від значень предикторів, представляється у вигляді ієрархічної структури - «дерева». Якщо залежна змінна є категоріальної, будується дерево класифікації. Якщо залежна змінна є кількісної, будується дерево регресії.

Основні терміни

- об’єкт – приклад, шаблон, дослідження;
- атрибут – ознака, незалежна змінна, властивість;
- цільова змінна – залежна змінна, мітка класу;
- вузол – внутрішній вузол дерева, вузол перевірки;
- кореневий вузол – початковий вузол дерева рішень;
- лист – кінцевий вузол дерева, вузол рішення;
- вирішальне правило – умова у вузлі, перевірка;
- ентропія – міра хаотичності інформації



- У наведеній вище моделі використовуються 3 атрибута з набору даних, а саме стать, вік і кількість подружжя або дітей разом.
- Дерево рішень намальовано догори ногами з коренем вгорі. На рисунку жирний текст чорного кольору є умовою(внутрішнім вузлом), на основі якого дерево розділяється на гілки(ребра). Кінець гілки, яка більше не поділяється, є рішенням(листком), в даному випадку, чи загинув пасажир або вижив, (представлено червоним або зеленим кольором) відповідно.



- Дерева регресії представлені так само, просто вони прогнозують безперервні значення, такі як ціна на нерухомість

Алгоритм розбиття «CART» для регресії

Загалом, алгоритми дерева рішень називаються «CART» або деревами класифікації і регресії

Функція вартості «CART», яка мінімізується для задач регресії:

$$J(T, x_h) = \frac{|T_{left}|}{|T|} MSE(T_{left}) + \frac{|T_{right}|}{|T|} MSE(T_{right})$$

$$MSE(T_{left}) = \sum_{k \in T_{left}} \left(\hat{y}_{T_{left}} - y_k\right)^2$$

$$\hat{y}_{T_{left}} = \frac{1}{|T_{left}|} \sum_{k \in T_{left}} y_k$$

$X_h^* = argmin_h\, J(T, x_h)$ – корінь дерева(піддерева), MSE – середньоквадратична похибка. «CART» рекурсивно розбиває множину на підмножини з мінімально можливими значеннями MSE з ваговими коефіцієнтами – відносними розмірами цих підмножин

Коефіцієнт детермінації — статистичний показник, в моделі є мірою залежності варіацій залежної змінної від варіацій незалежних змінних. Відображає наскільки спостереження побудованної моделі підтверджують реальні.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Загальний підхід до побудови Дерева Рішень

Ідея:

- Побудова дерева зверху-вниз;
- рекурсивне розбиття навчальної вибірки на максимально більш «чисті» підмножини;
- розбиття має бути значущим – класифікувати найбільшу кількість елементів навчальної вибірки.

Якщо множина T містить елементи, які відносяться до різних класів, тоді:

$$T = \bigcup_{i=1}^{q_h} T_i$$
$$T_i \subseteq T\colon\ x_h = c_{hi}$$

Множини T_i більш «чисті» в порівнянні з T.

$$T \coloneqq T_i, \forall_i = 1, \dots, q_h$$

Де грає	Суперник	Температура	Дощ	Перемога
Вдома	Вище	Висока	Так	Ні
Вдома	Вище	Висока	Ні	Так
Вдома	Нижче	Норма	Ні	Так
Вдома	Нижче	Висока	Так	Ні
В гостях	Нижче	Норма	Так	Так
В гостях	Нижче	Висока	Так	Ні
В гостях	Нижче	Висока	Ні	Ні
В гостях	Вище	Норма	Ні	Так

Критерії вибору змінної розбиття:

- ентропійні;
- джині.

Нехай множина T складається з n об'єктів, k з яких мають властивість S. Тоді ентропія множини T по відношенню до властивості S:

$$H(T,S) = -\frac{k}{n}\log_2\frac{k}{n} - \frac{n-k}{n}\log_2\frac{n-k}{n}$$

$H(T,S)$=1, коли $k = \frac{n}{2}$.

Якщо S може приймати s різних значень, кожне з яких – в k_i, тоді

$$H(T,S) = -\sum_{i=1}^{s}\frac{k_i}{n}\log_2\frac{k_i}{n}$$

Міра забрудненості Джині формула :

$$G(T_i,V) = 1 - \sum_{k=1}^{S}(p_{i,k})^2$$

Де $p_{i,k}$ – доля прикладів класу k серед навчальних прикладів в i – му вузлі.$G(T_i,V) = 0$, якщо T_i містить приклади одного класу

Де грає	Суперник	Температура	Дощ	Перемога
Вдома	Вище	Висока	Так	Ні
Вдома	Нижче	Норма	Ні	Так
В гостях	Нижче	Норма	Так	Так
В гостях	Нижче	Висока	Так	Ні
Вдома	Вище	Висока	Ні	Так
Вдома	Нижче	Висока	Так	Ні
В гостях	Нижче	Висока	Ні	Ні
В гостях	Вище	Норма	Ні	Так

$$H(T,Victory) = -\frac{4}{8}\log_2\frac{4}{8} - \frac{4}{8}\log_2\frac{4}{8} = 1$$

Перенавчання

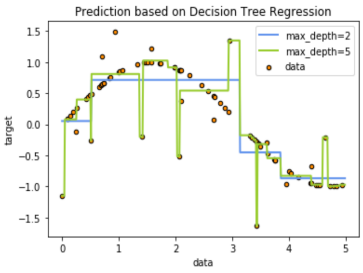
Якщо дотримуватися суто теоретичного погляду, то алгоритм зупиниться в той момент, коли усі параметри будуть розподілені по підмножинах T_i. На більш менш великому наборі даних відбудеться проблема перенавчання. Перенавчання – ознака моделі при котрій дерево рішень гарно справляється з навчальною вибіркою, але абсолютно неспроможне працювати на нових даних. Для уникнення цієї проблеми були розроблені наступні підходи:

- рання зупинка;
- зупинка в разі, якщо всі об'єкти в вершині відносяться до одного класу;
- обмеження максимальної глибини дерева;
- завдання мінімально допустимого числа прикладів у вузлі;

Бібліотека « Scikit-Learn»

В Scikit-Learn клас Decision TreeRegression має різні гіперпараметри.

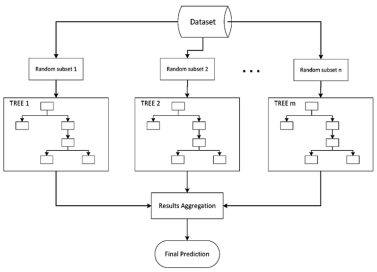
- Обмеження максимальної глибини дерева(max_depth) – зупинка алгоритму в момент досягнення заданої кількості розбиття гілок дерева.
- Завдання мінімально допустимого числа прикладів у вузлі(min_samples_split) – обмеження на мінімальну кількість прикладів у вузлі перш ній його можна буде розщепити. Це використовується для уникнення простих розбиттів в яких правило є невагомими.
- Min_samples_leaf – мінімальна кількість прикладів у листковому вузлі.
- Max_leaf_nodes – максимальна кількість листкових вузлів.
- Max_features – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла[28].
- Параметри max_... слід зменшувати, а параметри min_... слід збільшувати для зменшення ризику перенавчання дерева.



Ансамбль дерев рішень

Давайте уявимо, що ви проводите соціальне опитування серед великої кількості населення, а потім агрегуєте їх відповіді. З великою ймовірністю виявиться, що ця агрегована відповідь буде краща за відповідь експерта у галузі. Це називається колективним розумом. Аналогічно якщо ви агрегуєте прогнози групи прогнозаторів (таких як класифікатори або регресори), то в більшості випадків отримаєте кращі прогнози, ніж прогноз від найкращого індивідуального прогнозатора. Група прогнозаторів називається ансамблем. Відповідно, прийом носить назву ансамблеве навчання, а алгоритм – ансамблевий метод.

Для формування прогнозів ви лише отримуєте прогнози всіх індивідуальних дерев і прогнозуєте клас, який став володарем більшості голосів дерев. Такий ансамбль дерев прийняття рішень називається випадковим лісом і, незважаючи на свою простоту, є одним з самих потужних алгоритмів МО, доступних на сьогоднішній день



- Bagging (Bootstrap Aggregation) використовується, коли наша мета - зменшити дисперсію дерева рішень. Тут ідея полягає в тому, щоб створити кілька підмножин даних з навчальної вибірки, обраної випадковим чином. Тепер кожна колекція даних підмножини використовується для навчання їх дерев рішень. В результаті ми отримуємо ансамбль різних моделей. Використовується середнє значення всіх прогнозів для різних дерев, яке є більш надійним, ніж одне дерево рішень.
- Boosting це ще один метод ансамблю для створення колекції предикторов. У цьому методі моделі формують ознаки і передають їх наступним моделям для покращення результату. Іншими словами, ми підбираємо послідовні дерева (випадкова вибірка), і на кожному кроці мета полягає в тому, щоб знайти чисту помилку з попереднього дерева

Extra Trees, Gradient Boosting

- Extremely Randomized Trees додають ще один крок рандомізації до алгоритму випадкового лісу. ExtraTrees виберe випадковий поділ для кожної ознаки в випадковій підмножині, а потім вибере кращу ознаку для поділу, і порівняє ці випадково вибрані ознаки для поділу. Надзвичайно рандомізовані дерева набагато ефективніше в обчислювальному відношенні. У деяких випадках вони можуть навіть працювати краще ніж Random Forest.
- Gradient Boosting - це розширення методу бустінг. Gradient Boosting = градієнтний спуск + бустінг. Він використовує алгоритм градієнтного спуску, який може оптимізувати будь-яку диференційовану функцію втрат. Ансамбль дерев будується один за іншим, а окремі дерева підсумовуються послідовно. Наступне дерево намагається відновити втрату (різницю між фактичними і прогнозними значеннями)[. Основна ідея полягає у тому, що увага приділяється випадкам, які трудно передбачити, і модель навчається на попередніх помилках. На кожному етапі виявляється слабкий учень і його прогноз порівнюється з правильною відповіддю, що ми очікуємо. Градієнт перш за все, часна похідна від нашої функції втрат, тому він описує крутизну функції помилок.

Random Forest

Метод Random Forest можна розглядати як BAGGing з невеликою зміною. Потрібно зробити ще один крок, де на додаток до випадкової підмножини даних, він також приймає випадковий вибір ознак, а не використовує всі ознаки для створення нових дерев. Коли в вас є багато дерев рішень це називається Random Forest.

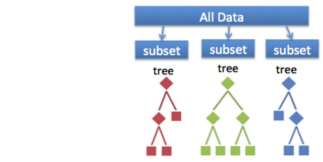
Припустимо, що в початковому наборі даних є N спостережень та M ознак. Алгоритм можна поділити на наступні етапи:

1.Спочатку вибірка з набору навчальних даних береться випадковим чином із повторенням.

2.Підмножини з M об'єктів вибираються випадковим чином, обирається найкраща ознака і використовується для ітеративного поділу вузла.

3.Процес побудови відбувається до вичерпання підвибірки.

4.Остаточний прогноз дається на основі агрегації прогнозів по n кількості дерев.



A random forest takes a random subset of features from the data, and creates n random trees from each subset. Trees are aggregated together at end.

Бібліотеки

Використані бібліотеки:

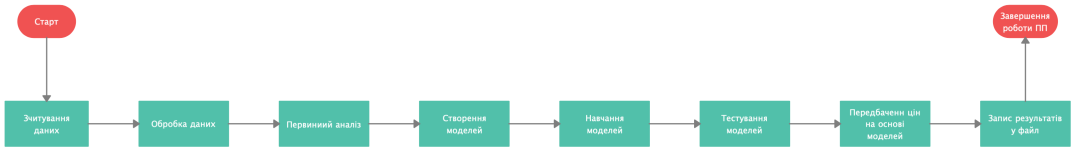
- pandas – використовується для роботи з даними
- numpy – використовується для роботи з багатовимірними масивами та матрицями
- sklearn.ensemble – бібліотека для робити з ансамблями дерев рішень
- sklearn.tree – бібліотека для роботи з деревами рішень
- sklearn.linear_model – бібліотека з лінійними моделями
- sklearn.model_slection – бібліотека для роботи з даними
- sklearn.metrics – бібліотека функцій оцінки, продуктивності
- matplotlib.pyplot – використовується для побудови графіків.

Метрики моделей

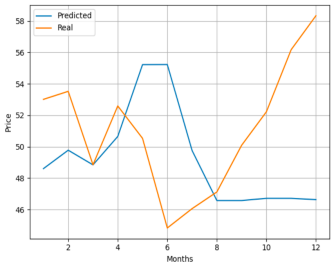
Назва моделі	R-square Value	Mean Squared Error	Explained Variance Score	Mean Absolute Error	Median Absolute Error
Random Forest	0.9600078	46.14370	0.960588	4.770672	2.866699
Extremely Trees	0.98745845	14.4706820	0.987468	2.583831	1.5214999
Gradient Boosting	0.9833593	19.200260	0.983360	3.01084194	2.1799603
Decision Tree Regression	0.8864948	130.964509	0.891784	7.203219	3.6531451
Linear Regression	0.7018832	343.97296	0.7412638	14.553832	12.377887
Hist Gradient Boosting	0.892661	123.84887	0.90026	7.302708	3.703953

Реальні ціни на нафту за 2017 рік

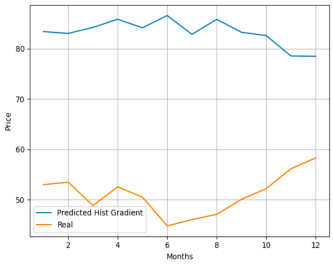
Принцип роботи програмного продукту



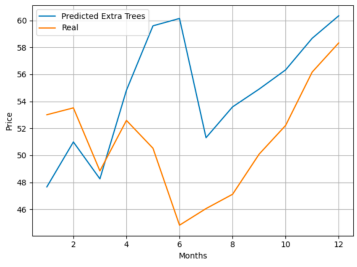
Далі наведено порівняння моделей з реальними значеннями



Random Forest

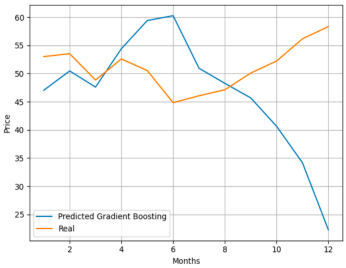


Hist Gradient

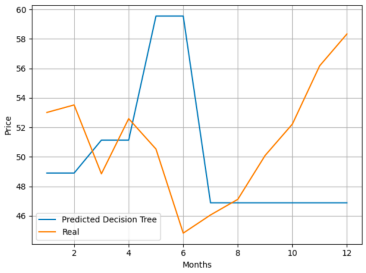


Extra Trees

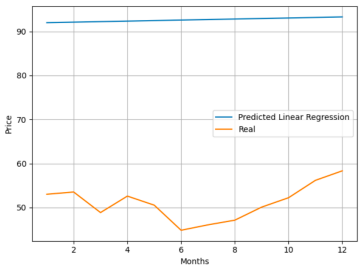
Порівняння результатів моделей з реальними значеннями



Gradient Boosting

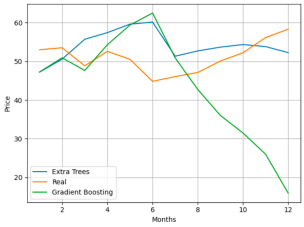


Decision Tree

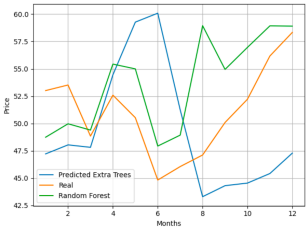


Linear Regression

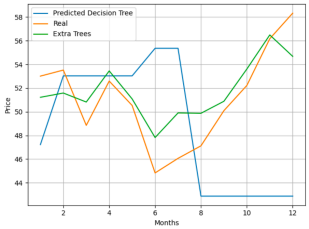
Порівняння результатів моделей



Extra Trees та Gradient Boosting

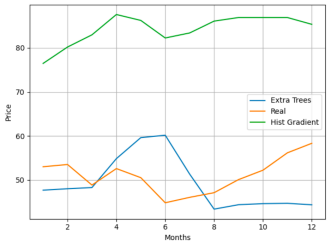


Extra Trees та Random Forest

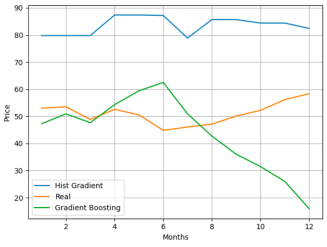


Decision Tree та Extra Trees

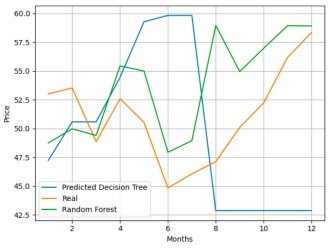
Порівняння результатів моделей



Extra Trees та Hist Gradient

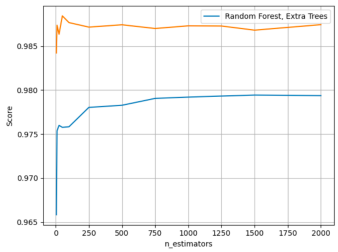


Hist Gradient та Gradient Boosting

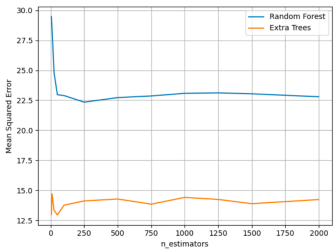


Decision Tree та Random Forest

Порівняння метрик

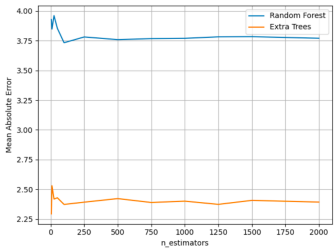


Порівняння R-square value

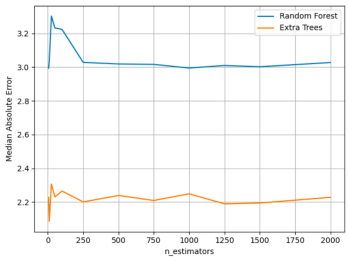


Порівняння Mean Squared Error

Порівняння метрик



Порівняння Mean Absolute Error



Порівняння Median Absolute Error

Висновки

- На мою думку, реалізовані алгоритми показали себе досить непогано. Результати досить різні і залежать від методу та його параметрів. Використовуючи GridSearch вдалося підібрати вдалі параметри, що позитивно вплинули на результат роботи програмного продукту. Краще всього показали себе такі методи як ExtraTrees та Random Forest, в окремих випадках різниця спрогнозованих результатів з реальними не перевищує 5%. На мою думку поставлена задача була виконана, програмний продукт є закінченим та максимально оптимізованим.
- Що з приводу вдосконалення та розвинення цього програмного продукту, на мою думку, було б дуже практично та умістно розробити модель для прогнозування цін на паливо. Спробувати встановити залежність між ціною на паливо та нафту, можливо додати якісь інші енергоресурси або відстежити залежність курсу USD/UAN. Розробка friendly – інтерфейсу зробила б продукт більш практичним та можливим для використання більш широкою аудиторією користувачів.