Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Магістерська кваліфікаційна робота на тему:

"ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЗАРОБІТНОЇ ПЛАТИ У СФЕРІ DATA SCIENCE У 2023 РОЦІ"

Виконав: студент 2 курсу, групи 2ІСТ-22м
спеціальності 126 – «Інформаційні системи та технології»
_____ Богдан ДОЛЕНКО
Керівник: к.т.н., доц. каф. САІТ
_____ Ілона ВАРЧУК
«08» _____12_____ 2023 р.
Рецензент: к.т.н., доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«12» _____12_____ 2023 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«08» _____12_____ 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«04» 09 2023 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Доленку Богдану Анатолійовичу

1. Тема роботи: “ Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році”,
керівник роботи: Ілона ВАРЧУК., к.т.н., доц. каф. САІТ,
затверджені наказом закладу вищої освіти від «18» 09 2023 року №247
2. Строк подання студентом роботи «30» 11 2023 року
3. Вихідні дані до роботи:
Датасет «Data Science Salaries 2023» з даними для передбачення заробітної плати у сфері Data Science
https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2023/
4. Зміст текстової частини:
   1) Актуальність проблеми;
   2) Аналіз бібліотек для вирішення задачі;
   3) Підготовка даних;
   4) Побудова моделей машинного навчання
   5) Економічна частина.
5. Перелік ілюстративного матеріалу:
   1) Графік розподілу виплат за роками;
   2) Графік вивчення середньої заробітної плати за рік;
   3) Графік розподілу заробітної плати за роками;
   4) Графік розподілу заробітної плати (у доларах) за роками;
   5) Графік середньої заробітної плати за рік;
   6) Діаграма розподілу на основі рівня досвіду;
   7) Графік розподілу рівнів досвіду працівників за роками стажу;
   8) Порівняння точності на тренувальних та тестових даних для різних моделей машинного навчання за допомогою графіку.
6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
4	Наталія БУРСЕННІКОВА, д. е. н., проф. каф. ЕПВМ	05.10.23	10.11.23

7. Дата видачі завдання «04» 09 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	04.09	20.09	
2	Аналіз методів машинного навчання для вирішення поставленої задачі	20.09	05.10	
3	Розробка моделі класифікації	05.10	25.10	
4	Економічна частина	25.10	10.11	
5	Оформлення матеріалів до захисту МКР	10.11	30.11	

Студент_____ Богдан ДОЛЕНКО

Керівник роботи_____ Ілона ВАРЧУК

АНОТАЦІЯ

УДК 004.09+336

Доленко Б. А. Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 97 с.

На укр. мові. Бібліогр.: 62 назв; рис.: 51; табл.: 11.

В магістерській кваліфікаційній роботі здійснено розробку інформаційної технології аналізу та передбачення заробітної плати у сфері Data Science у 2023 році. Здійснено збирання та обробку відповідних даних, які стосуються динаміки заробітної плати в даному сегменті протягом попередніх років. Використовуючи передові інструменти аналізу даних та технології обробки інформації, проведено детальний аналіз параметрів, що впливають на зміни в рівні оплати праці у галузі Data Science.

Проведено тестування розробленої інформаційної системи, включаючи аналіз інтерактивних графіків, які відображають динаміку змін у заробітній платі для спеціалістів Data Science, а також передбачення можливих тенденцій у майбутньому. Розроблена система надає можливість глибокого вивчення факторів, що визначають заробітну плату в цій сфері, що сприяє більш точному розумінню ринкових умов та прийняттю ефективних управлінських рішень.

Ілюстративна частина складається з 8 плакатів.

У розділі економічної частини розглянуто питання про доцільність розробки та впровадження інформаційної технології аналізу та передбачення заробітної плати у сфері Data Science у 2023 році.

Ключові слова: Інформаційна технологія, аналіз даних, передбачення, заробітна плата, Data Science.

ABSTRACT

Dolenko B. A. Information Technology for Analysis and Prediction of Salaries in the Field of Data Science in 2023. Master's Qualification Work in the specialty 126 – Information Systems and Technologies, educational-professional program – Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2023. 97 p.

In Ukrainian language. Bibliography: 62 titles; fig .: 51; tab .: 11.

In the master's qualification work involves the development of information technology for the analysis and prediction of salaries in the field of Data Science in 2023. Data collection and processing were carried out, focusing on the dynamics of salaries in this segment over the previous years. Utilizing advanced data analysis tools and information processing technologies, a detailed analysis of parameters influencing changes in the level of remuneration in the Data Science industry was conducted.

Testing of the developed information system was performed, including the analysis of interactive graphs depicting the dynamics of salary changes for Data Science professionals, as well as predicting possible future trends. The developed system enables in-depth exploration of factors determining salaries in this field, contributing to a more accurate understanding of market conditions and the adoption of effective management decisions.

The illustrative part consists of 8 posters.

The economic section discusses the feasibility of developing and implementing information technology for the analysis and prediction of salaries in the field of Data Science in 2023.

Keywords: Information Technology, Data Analysis, Prediction, Salary, Data Science.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному інформаційному суспільстві, визначеному стрімким темпом технологічного розвитку, однією з ключових сфер виявляється Data Science. В контексті цієї області виникає нагальна потреба в розробці і вдосконаленні методів аналізу та передбачення заробітної плати для спеціалістів у цьому сегменті ринку праці. Актуальність дослідження полягає в тому, що зростаюча конкуренція та сталий розвиток технологій вимагають від компаній та фахівців у галузі Data Science не тільки високого рівня кваліфікації, але й адаптації до динамічного ринкового середовища.

Мета і завдання роботи. Метою даної магістерської кваліфікаційної роботи є підвищення точності передбачення заробітної плати у сфері Data Science на ринку праці у 2023 році. Для досягнення поставленої мети необхідно розв’язати наступні завдання:

- провести огляд існуючих аналогів;
- підготувати дані для подальшої роботи ;
- провести розвідувальних аналіз даних;
- побудувати моделі та виконати прогнозування;
- оцінити результати роботи моделей.

Об’єктом дослідження магістерської кваліфікаційної роботи є процес розроблення інформаційної технології аналізу передбачення заробітної плати для спеціалістів у галузі Data Science у 2023 році.

Предметом дослідження магістерської кваліфікаційної роботи є інформаційна технологія аналізу та передбачення заробітної плати для спеціалістів у галузі Data Science у 2023 році.

Методи дослідження. Для досягнення поставлених завдань використано такі методи: аналіз літературних джерел, статистичний аналіз даних, математичне моделювання, а також застосування інструментів машинного навчання для розробки передбачувальних моделей.

Новизна одержаних результатів полягає в тому, що дістала подальший розвиток інформаційна технологія передбачення заробітної плати у сфері Data Science, за рахунок використання методів машинного навчання, що дозволяє підвищити точність цього передбачення.

Практичне значення. Практичне значення інформаційної технології полягає в можливості застосування отриманих результатів на практичних задачах, а також для оптимізації управління персоналом та кар'єрним розвитком фахівців у цьому сегменті ринку праці.

Апробація результатів магістерської кваліфікаційної роботи. Результати роботи доповідались на LIII Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (м. Вінниця, 2023-2024 рр.).

Публікації результатів магістерської кваліфікаційної роботи. Опубліковано тези на LIII Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (м. Вінниця, 2023-2024 рр.) [1].

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Огляд області досліджень

Data Science - це міждисциплінарна наука, яка об'єднує методи, процеси, алгоритми та системи для видобування знань та інсайтів з структурованих і неструктурованих даних. Головна мета Data Science - аналіз та використання даних для прийняття управлінських рішень, прогнозування подій та розробки нових продуктів та послуг. Ця галузь використовує методи машинного навчання, статистики, аналізу даних, обробки природних мов, великих даних та інших інструментів для досягнення своїх цілей [2].

Проблеми в роботі датасайнтистів включають:

- Очищення, обробка та інтеграція даних можуть бути складними завданнями, оскільки джерела даних можуть бути різнорідними та непередбачуваними;
- Деякі завдання можуть вимагати розробки нових методів аналізу або використання невеликої кількості даних;
- Робота з даними може стикатися з питаннями етики та конфіденційності, особливо в галузях, де обробка особистих даних є важливою;
- Поняття інтерпретації результатів та передачі їх іншими може бути викликом, особливо коли складні аналітичні моделі використовуються для прийняття рішень;
- Після розробки моделей їх потрібно постійно оновлювати та підтримувати, щоб вони залишалися актуальними та ефективними.

Роль датасайнтистів в сучасному світі полягає у вирішенні різноманітних завдань, від аналізу ринку та прогнозування тенденцій до розробки інтелектуальних систем, які здатні приймати автоматичні рішення. Вони грають важливу роль у бізнесі, наукових дослідженнях, урядових організаціях та багатьох інших сферах [3].

Розмір зарплати датасайнтистів може варіюватися відповідно до різних факторів, таких як рівень навчання, досвід роботи, географічне розташування та тип організації. Також великий вплив на зарплату має попит на датасайнтістів у конкретному регіоні та галузі, а також їхні навички та спеціалізація. Досвідчені датасайнтисти з високим рівнем навчання та спеціалізацією у важливих галузях (наприклад, медицина або фінанси) часто можуть очікувати вищі зарплати [4].

1.2 Аналіз методів та засобів для вирішення задачі

Exploratory Data Analysis (EDA) - це методологія, яка допомагає аналізувати та розуміти набір даних до того, як розпочинати більш глибокий аналіз чи моделювання. EDA допомагає виявити ключові властивості даних, зокрема розподіл, взаємозв'язки між змінними та можливі аномалії [5].

Це включає в себе використання графіків, гістограм, кореляційних матриць та інших візуалізаційних засобів для представлення даних у зрозумілій формі. Також проводяться статистичні обчислення, щоб зрозуміти основні характеристики даних, такі як середнє значення, медіана та розкид [6].

EDA важливий для ідентифікації патернів та трендів у даних, виявлення важливих залежностей і, в кінцевому підсумку, для прийняття рішення про те, яким чином аналізувати та моделювати дані для досягнення конкретних цілей. Власне, EDA служить "першим кроком" в розвідці та розумінні даних перед подальшим їх аналізом [7].

Random Forest Regressor - це потужна модель машинного навчання, яка використовується для завдань регресії. Вона базується на ідей ансамблю, де декілька дерев рішень об'єднуються для отримання більш точного та стійкого прогнозу.

У випадку Random Forest Regressor, модель створює кілька дерев рішень (випадкових лісів) і використовує їх для передбачення числових значень. Кожне дерево приймає випадковий піднабір даних для навчання та

використовує різні ознаки для прийняття прогнозів. Потім результати з різних дерев комбінуються шляхом середнього (у випадку регресії) для отримання остаточного прогнозу [8].

Random Forest Regressor володіє декількома перевагами, включаючи високу точність, здатність працювати з великими наборами даних та автоматичну обробку пропущених даних. Він також добре впорається з викидами та шумом у даних.

Ця модель знаходить застосування у багатьох областях, включаючи фінанси, медицину, екологію та багато інших, де потрібно прогнозувати числові значення на основі великих обсягів даних [9].

SVR, або Support Vector Regression, - це метод машинного навчання, який використовується для прогнозування неперервних числових значень, таких як ціни на нерухомість, акції на фондовому ринку або будь-які інші неперервні змінні. В основі цього методу лежить ідея побудови гіперплощини в просторі ознак, яка найкращим чином підходить до навчальних даних. Головною метою SVR є знаходження такої гіперплощини, яка б мінімізувала помилку прогнозування [10].

Під час навчання SVR спробує знайти оптимальну гіперплощину, яка максимізує відстань між нею і найближчими точками даних. Особливістю SVR є те, що він використовує функцію ядра для перетворення даних у вищорозмірний простір, де легше будувати гіперплощину. Цей метод може бути ефективним для розв'язання задач регресії в умовах, коли дані мають складну не лінійну залежність [11].

Додатково, SVR може бути використаний у великій кількості сфер, оскільки він дозволяє моделі працювати з нечіткими або нелінійними залежностями в даних. Ось кілька прикладів застосування SVR:

1. Прогнозування цін на нерухомість: SVR може бути використаний для прогнозування вартості нерухомості, враховуючи різноманітні фактори, такі як розташування, розмір і стан будинку, інфраструктура та інші чинники.

2. Фінансовий аналіз: SVR може допомогти в прогнозуванні цін на фінансові активи, такі як акції або валюта. Він може враховувати історичні ціни та інші економічні показники для створення прогнозів.

3. Медицинська діагностика: SVR може бути використаний для аналізу медичних даних, щоб прогнозувати ризики захворювань, результати тестів або виживання пацієнтів на основі клінічних та генетичних даних.

4. Прогнозування виробничих процесів: SVR може допомогти в прогнозуванні результатів виробничих процесів, оптимізації виробничої ланки, а також управлінні виробничими витратами.

5. Аналіз забруднення довкілля: SVR може використовуватися для аналізу забруднення повітря, води або грунту, допомагаючи прогнозувати рівні забруднення на основі різних факторів.

Узагальнюючи, SVR - це потужний інструмент машинного навчання для розв'язання задач регресії в різних галузях, де дані можуть мати складну структуру та залежність. Він дозволяє побудувати модель, яка може точно прогнозувати числові значення на основі навчальних даних [12].

Decision Tree Regressor - це модель машинного навчання, яка використовується для завдань регресії. Вона використовується для прогнозування числових значень (наприклад, цін на нерухомість, прибутковості акцій або температури) на основі вхідних даних. Основна ідея полягає в тому, щоб розділити набір даних на менші підгрупи, де значення цільової змінної максимально однорідні.

Decision Tree Regressor будує дерево рішень, де кожен вузол розбиває дані на дві частини, використовуючи певний критерій (найчастіше середньоквадратична помилка). Дерево росте до певної глибини або до досягнення певної точки зупинки, такої як мінімальна кількість прикладів у вузлі або максимальна глибина дерева [13].

Однією з переваг Decision Tree Regressor є їх інтерпретованість. Можливо вивести правила прийняття рішень, що легко розуміються. Однак це

може призвести до перенавчання, коли дерево занадто добре пристосовується до навчальних даних і втрачає здатність узагальнювати до нових даних.

Для подолання цього недоліку, можна використовувати техніки, такі як обрізка дерева (pruning) або ансамбльні методи, наприклад Random Forest, які поєднують кілька Decision Tree моделей для отримання більш точних результатів.

Decision Tree Regressor - це потужний інструмент для завдань регресії, і важливо добре розуміти його роботу та обмеження при роботі з ним у машинному навчанні [14].

1.3 Огляд аналогів для вирішення поставленої задачі

Однією з ключових складових успішного аналізу та передбачення заробітної плати у сфері Data Science є використання інноваційних інформаційних технологій та аналітичних методів. Для цього дослідження розглянуто існуючі аналоги та рішення, які були застосовані в галузі Data Science. Цей огляд допоможе нам зрозуміти, які інструменти та підходи вже використовувалися, а також знайти можливості для подальшого розвитку та вдосконалення нашої інформаційної технології для аналізу та передбачення заробітної плати у сфері Data Science [15].

«DS Salaries _ EDA & Salary predictions» - Kaggle Notebook зосереджений на аналізі набору даних, пов'язаних із зарплатами в галузі науки про дані у 2023 році. Ось розбивка основних розділів і завдань у ньому:

- Імпорт бібліотек: Цей розділ імпортує різні бібліотеки Python, включаючи NumPy, pandas, Matplotlib, Seaborn та інші.
- Завантаження набору даних: Набір даних завантажується з CSV-файлу з назвою "ds_salaries.csv", і перші три рядки відображаються для огляду даних.
- Короткий опис стовпців: Надає короткий опис кожного стовпчика в наборі даних, включаючи пояснення того, що вони представляють.

– Очищення даних та перші кроки з проведення аналізу даних: У цьому розділі виконується очищення даних і початковий дослідницький аналіз даних. Сюди входить заміна абревіатур у певних стовпчиках, перевірка відсутніх значень, відображення базової статистики та вивчення характеристик набору даних.

– Візуалізація даних: Для вивчення набору даних створюється кілька візуалізацій даних. Це гістограми, кругові діаграми, стовпчикові та секторні діаграми для візуалізації розподілу заробітної плати, співвідношення віддаленої та дистанційної роботи, місця проживання працівників, розташування компанії, назв посад та їхнього зв'язку із заробітною платою.

На рисунку 1.1 зображено візуалізацію даних за допомогою гістограми посади з найвищою заробітною платою.

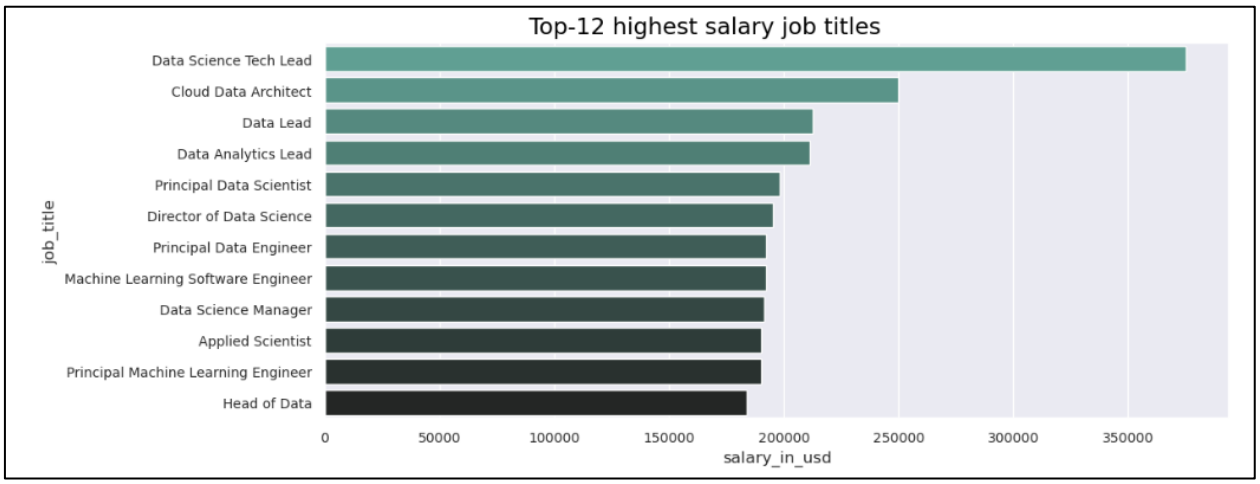


Рисунок 1.1 – Гістограма топ-12 посад з найвищою заробітною платою

На рисунку 1.2 зображено візуалізацію даних за допомогою гістограми посади з найнижчою заробітною платою.

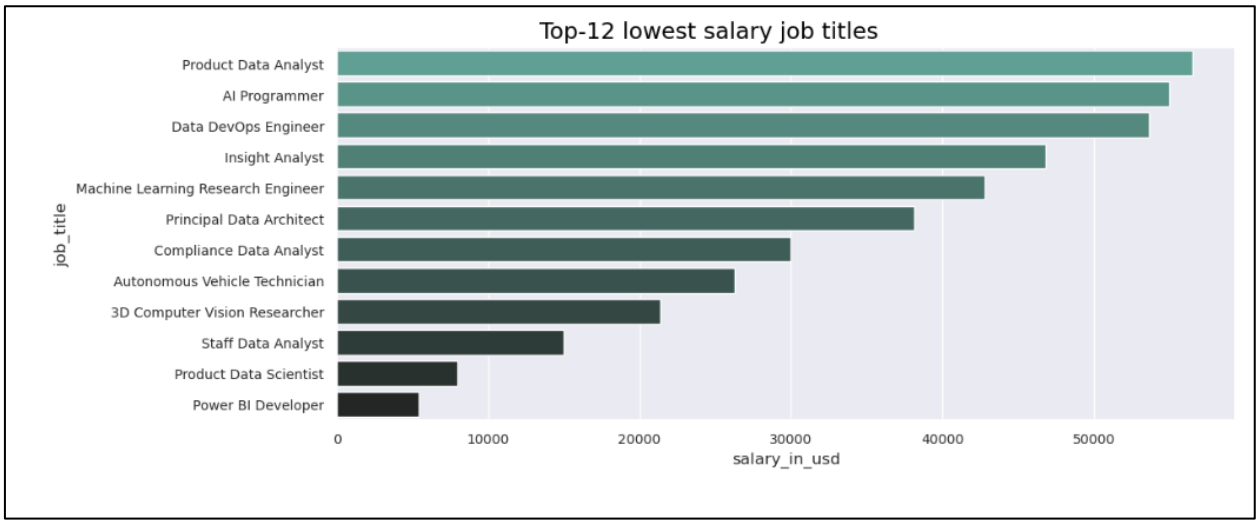


Рисунок 1.2 – Гістограма топ-12 посад з найнижчою заробітною платою

– Моделювання: У цьому розділі представлені моделі машинного навчання, які використовуються для прогнозування зарплат.

На рисунку 1.3 зображено графік розкиду фактичних і прогнозованих зарплат.

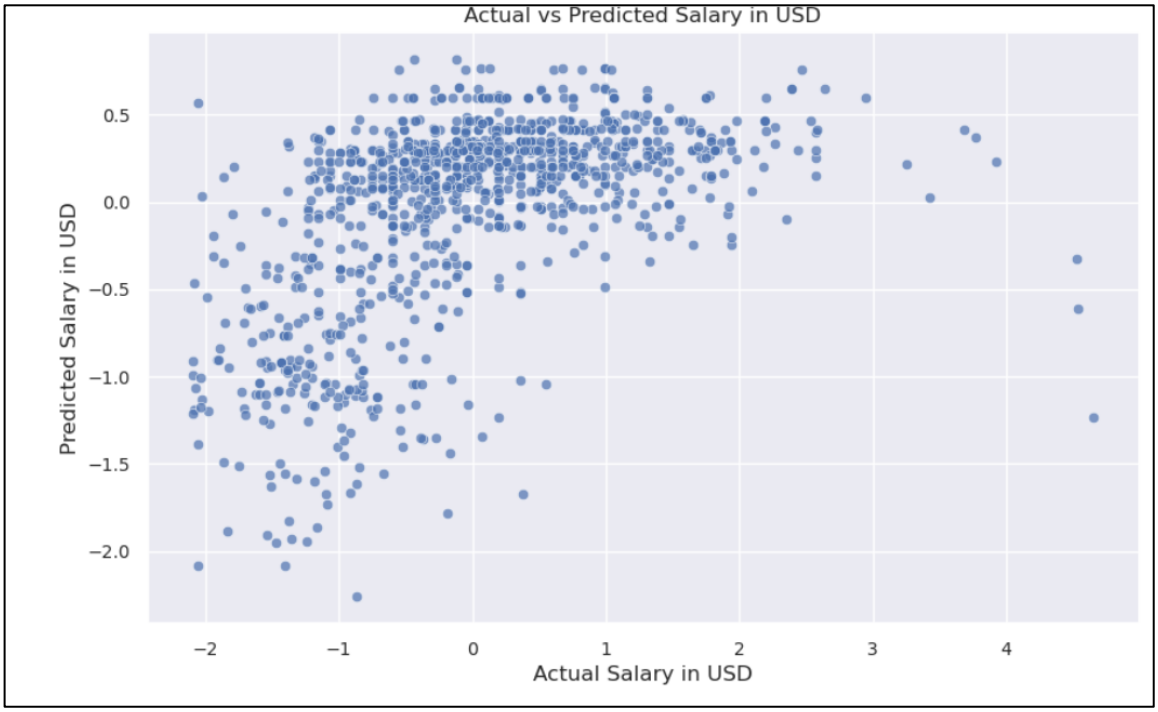


Рисунок 1.3 - Графік розкиду фактичних і прогнозованих зарплат

У згаданому ноутбуці детально розглядаються дані про заробітну плату в галузі науки про дані, включаючи очищення даних, візуалізацію та моделювання, з акцентом на розумінні факторів, які можуть впливати на заробітну плату в галузі науки про дані. Однак у ньому також згадується, що результати моделювання не є ідеальними і можуть потребувати подальшого налаштування або використання інших моделей для більшої точності.

«Data science salary EDA» - Kaggle Notebook для аналізу та візуалізації даних для набору даних, пов'язаних із зарплатами в галузі Data Science у 2023 році. Ось розбивка того, як він працює:

- Імпорт бібліотек: Блокнот починає з імпорту декількох бібліотек Python, зокрема pandas, numpy, matplotlib, seaborn та Plotly. Ці бібліотеки зазвичай використовуються для аналізу та візуалізації даних;
- Імпорт даних: Код встановлює бібліотеку "country-converter" за допомогою pip, а потім імпортує її як "coco". Ця бібліотека, ймовірно, використовується для перетворення назв країн у коди ISO3. Блокнот зчитує CSV-файл з назвою "ds_salaries.csv" за допомогою pandas і зберігає дані у фреймі даних з назвою "df";
- Дослідження та попередня обробка даних: Дослідження форми та основної інформації набору даних, використовуючи `df.shape` та `df.info()`. Він також перевіряє наявність пропущених значень за допомогою `df.isull().sum()`, і оскільки пропущених значень немає, він відзначає, що набір даних добре підготовлено.

На рисунку 1.4 зображено візуалізацію даних працівників за їх рівнем досвіду за допомогою кругової діаграми.

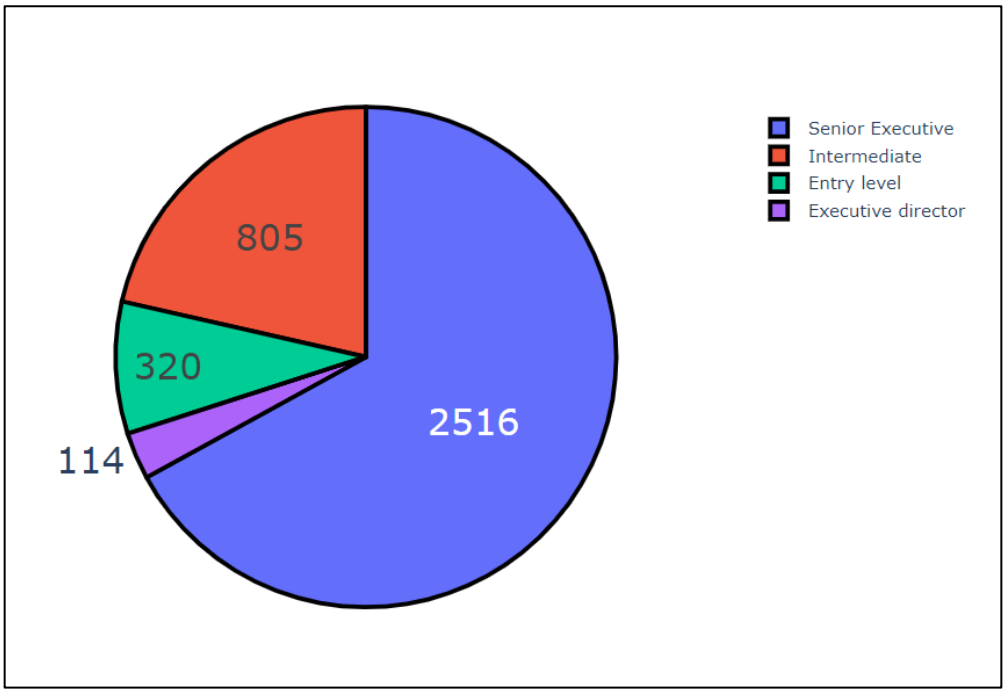


Рисунок 1.4 - Діаграма візуалізації даних працівників за їх рівнем досвіду

На рисунку 1.5 зображено топ-15 найпопулярніших посад у сфері Data Science використовуючи гістограму.

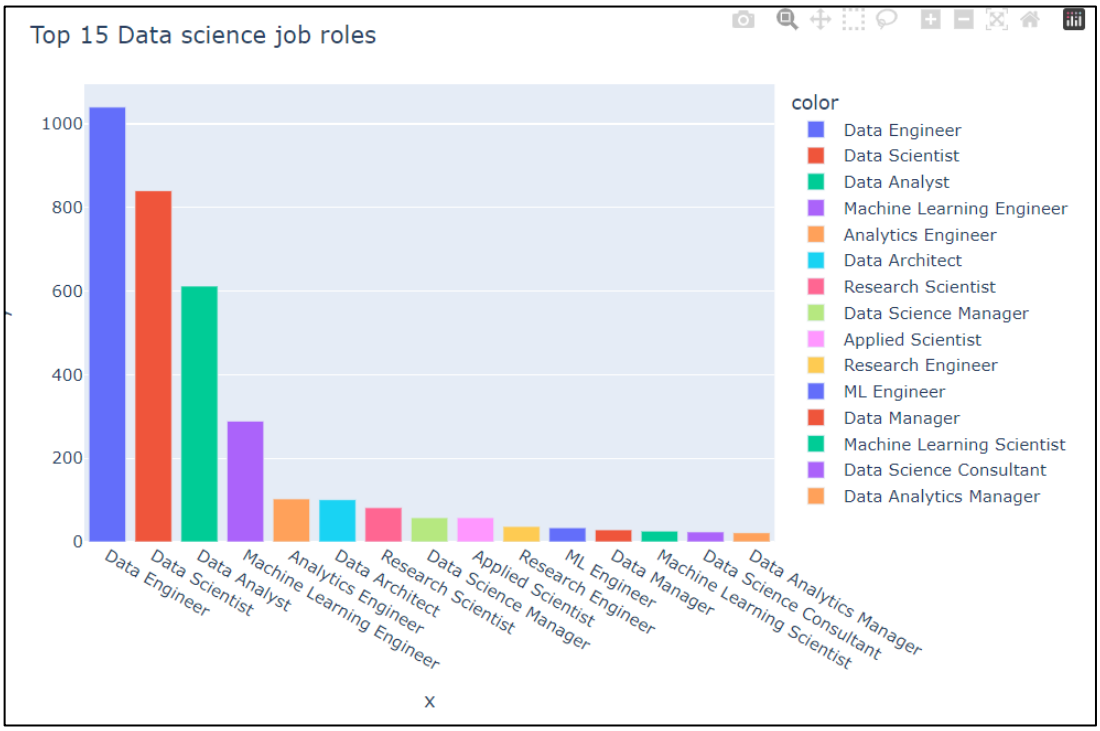


Рисунок 1.5 - Гістограма топ-15 найпопулярніших посад у сфері Data Science.

Даний ноутбук поєднує аналіз даних, попередню обробку даних і візуалізацію даних, щоб надати уявлення про різні аспекти набору даних, такі як посадові ролі, типи зайнятості, назви посад, місце проживання працівників, місцезнаходження компаній і розміри компаній. Ці візуалізації допомагають зрозуміти характеристики та тенденції в наборі даних.

1.4 Висновки

У даному розділі магістерської кваліфікаційної роботи виконано обґрунтовану характеристику об'єкта досліджень - інформаційних технологій аналізу та передбачення заробітної плати в галузі Data Science у 2023 році. Огляд області досліджень дозволив систематизувати ключові аспекти, які впливають на зміни у заробітній платі в даній сфері, зокрема врахувати важливість факторів, таких як динаміка ринку праці, специфіка вимог до кваліфікацій, та технологічний прогрес.

Аналіз методів та засобів для вирішення задачі надав можливість визначити ключові інструменти, що застосовуються в сучасних інформаційних технологіях для ефективного аналізу та передбачення заробітної плати. Відзначається, що застосування методів машинного навчання, статистичного аналізу та алгоритмів глибинного навчання є актуальними та перспективними для вирішення поставлених завдань.

Огляд аналогів показав, що в сучасній літературі та практиці вже існує певний обсяг робіт, присвячених аналізу заробітної плати в галузі Data Science. Однак, важливо відзначити, що багато з цих робіт можуть бути застарілими або не враховувати сучасні тенденції розвитку галузі.

Висновки з цього розділу вказують на актуальність теми та підтверджують необхідність подальших досліджень у напрямку оптимізації методів та інструментів для аналізу та передбачення заробітної плати у сфері Data Science у 2023 році. Також, вони створюють основу для наступних

розділів роботи, де будуть розглядатися конкретні методи, експерименти та отримані результати досліджень.

Такі кардинальні відмінності важко передбачити, оскільки вони дуже рідкісні у всьому спостереженні, тому існуючі моделі не можуть добре передбачити раптові зміни.

2 ОБҐРУНТУВАННЯ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ДОСЛІДЖЕННЯ

2.1 Обґрунтування вибору середовища розробки

Під час вибору інструментів для виконання поставленого завдання обрано платформу Kaggle, оскільки вона надає доступ до великої кількості готових наборів даних, які можна використовувати для аналізу та моделювання. Також вона дозволяє використовувати потужні обчислювальні ресурси, такі як GPU та TPU, для швидкого тренування та тестування моделей машинного навчання [16].

Kaggle — це найбільша у світі спільнота з машинного навчання та науки про дані, яка надає потужні інструменти та ресурси для досягнення цілей з аналізу даних. Одним з таких інструментів є Kaggle Notebooks, хмарне обчислювальне середовище, яке дозволяє виконувати та ділитися кодом, даними, візуалізаціями та іншими результатами аналізу. Kaggle Notebooks підтримує різні мови програмування, такі як Python, R, Julia, Scala та інші, а також різні фреймворки, такі як TensorFlow, PyTorch, Keras та інші. Kaggle Notebooks також надає доступ до великої кількості наборів даних, які можна використовувати для навчання моделей, а також до GPU та TPU, які можна використовувати для прискорення обчислень. Kaggle Notebooks є зручним та гнучким середовищем розробки для машинного навчання, яке сприяє відтворюваності та співпраці аналітиків даних.

На рисунку 2.1 зображено інтерфейс середовища Kaggle.

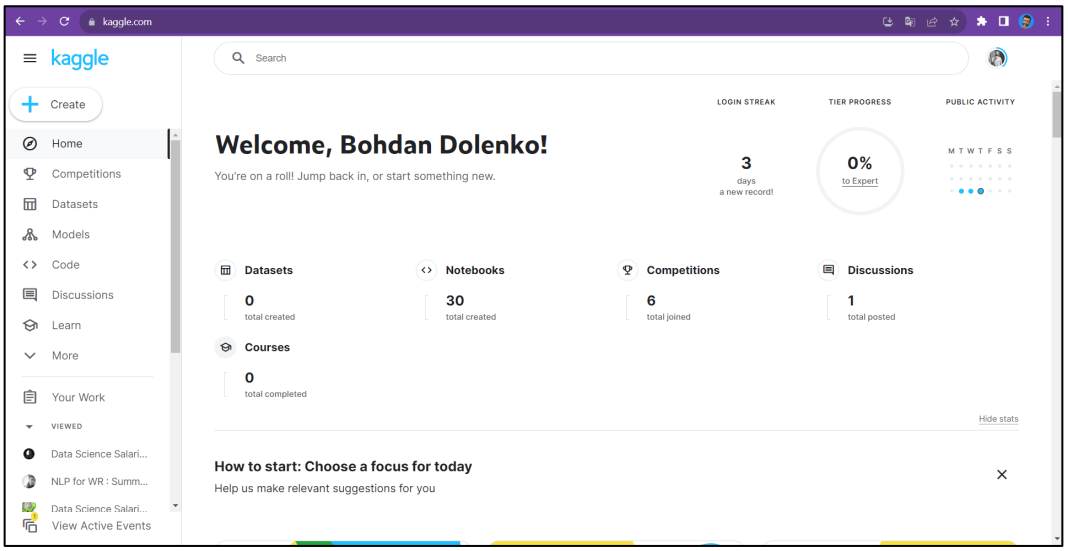


Рисунок 2.1 – Інтерфейс середовища Kaggle

Основний функціонал Kaggle включає участь у змаганнях та викликах для вирішення складних завдань, а також можливість обговорення та обміну ідеями на форумах. Використання графічних та тензорних процесорів дозволяє прискорити роботу з моделями машинного навчання [17].

Платформа забезпечує API для автоматизації завдань та можливість зберігання ноутбуків. Також Kaggle пропонує інтеграцію з іншими інструментами для полегшення роботи з даними та моделями. Можливість публікації та обміну роботами сприяє взаємодії та навчанню на прикладах інших користувачів.

Загалом, Kaggle створено для зручної спільної роботи в галузі науки про дані, забезпечуючи широкий набір інструментів для аналізу, моделювання та навчання [18].

2.2 Огляд моделі машинного навчання «Linear Regression»

Лінійна регресія - простий метод машинного навчання для прогнозування. Вона використовує лінійну залежність між вхідними факторами і вихідним значенням. Мета - знайти оптимальні коефіцієнти, які мінімізують середньоквадратичну помилку.

Оптимізація включає пошук коефіцієнтів, що мінімізують середньоквадратичну помилку. Градієнтний спуск - метод оптимізації.

Середньоквадратична помилка - міра відхилення прогнозованих значень від реальних.

Модель базується на припущеннях лінійності та незалежності. Ефективність оцінюється за допомогою метрик, таких як R-squared.

Лінійна регресія ефективна лише для лінійних взаємозв'язків. У складніших сценаріях можуть бути ефективніші інші моделі машинного навчання [19].

Лінійна регресія, як метод машинного навчання, використовується для аналізу та моделювання взаємозв'язків між змінними. Вона базується на припущенні, що ці взаємозв'язки можна виразити лінійним рівнянням. У випадку одновимірної лінійної регресії, коли є одна незалежна та одна залежна змінна, модель представлена рівнянням прямої лінії.

Застосовуючи цю модель, лінійна регресія знаходить такі значення коефіцієнтів, щоб утворити лінію, яка найкраще відображає взаємозв'язок між змінними. Головна ідея - зменшити різницю між фактичними та передбачуваними значеннями [20].

Лінійна регресія може бути розширена на випадок багатьох незалежних змінних, відомий як багатовимірна лінійна регресія. У цьому випадку рівняння буде мати вигляд гіперплощини.

Оцінка ефективності лінійної регресії зазвичай здійснюється за допомогою різних метрик, включаючи R-squared, яка визначає відсоток варіації вихідної змінної, яку пояснює модель.

Лінійна регресія також використовується в статистичних дослідженнях для визначення ступеня впливу однієї або кількох змінних на інші.

Хоча лінійна регресія є потужним інструментом, важливо пам'ятати, що її ефективність обмежена припущеннями про лінійність та незалежність помилок. У випадках складних нелінійних відносин та великої кількості

змінних, інші методи машинного навчання можуть бути більш відповідними [21].

На рисунку 2.2 зображено приклад лінійної регресії в машинному навчанні.

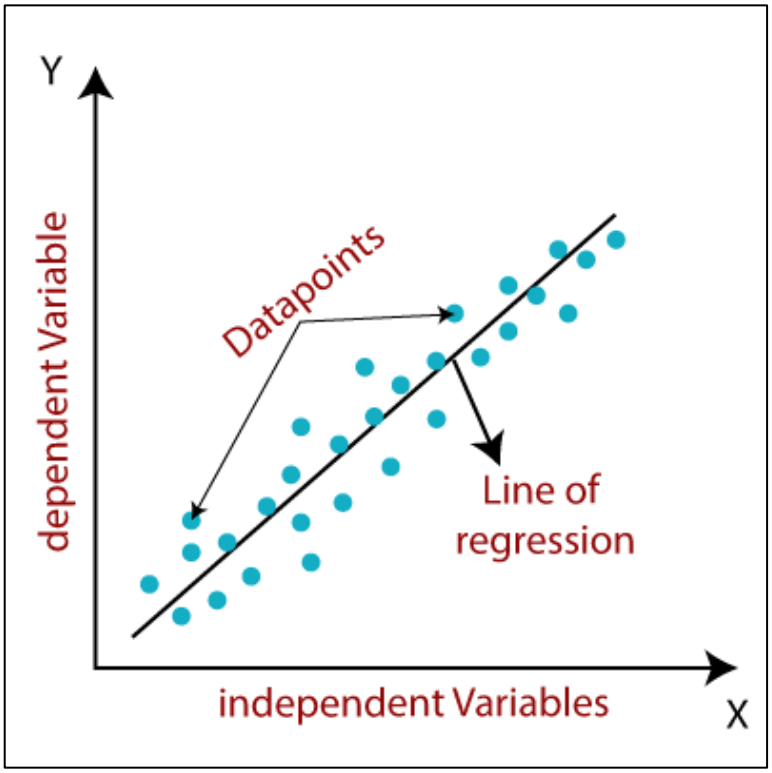


Рисунок 2.2 – Приклад лінійної регресії в машинному навчанні

Завдяки своїй простоті, лінійна регресія тренується та прогнозує швидко, що робить її привабливою для використання.

При використанні для початкового аналізу даних, лінійна регресія дозволяє швидко оцінити напрямок взаємозв'язку між змінними, навіть без використання більш складних моделей.

Важливо зазначити, що лінійна регресія не є універсальним рішенням і може бути неефективною для нелінійних взаємозв'язків. Перед використанням важливо враховувати особливості даних та природу взаємодії між змінними [22].

2.3 Огляд моделі машинного навчання «Support Vector Regression»

Support Vector Regression (SVR) — це метод машинного навчання, який використовує ідеї Support Vector Machines (SVM) для задачі регресії. Основна мета SVR — знайти гіперплощину, яка найкращим чином апроксимує функцію взаємозв'язку між залежними та незалежними змінними в просторі високої розмірності.

У випадку SVR, здійснено пошук гіперплощини, щоб регулюючи ширину коридору (англ. margin), який визначається відхиленням точок від гіперплощини, можна мінімізувати відхилення (помилки) точок від цієї гіперплощини [23].

У випадку SVR, точки, які лежать в межах коридору, вважаються прийнятними для моделі, тоді як точки, які виходять за його межі, враховуються як відхилення та впливають на формування моделі. Ширина коридору є гіперпараметром, який визначається під час налаштування моделі.

Задача оптимізації SVR включає мінімізацію величини, яка враховує відхилення точок від гіперплощини та ширина коридору. Додатково, у випадку, коли є точки, які виходять за межі коридору, але тепер розглядаються як прийняті (відомі як опорні вектори), вони впливають на формування гіперплощини.

SVR дозволяє враховувати нелінійні взаємозв'язки за допомогою ядерних функцій, які перетворюють вхідні дані в простори вищої розмірності. Це дозволяє моделі SVR ефективно працювати з наборами даних, де взаємозв'язки складніші, ніж просто лінійні.

Важливою характеристикою SVR є те, що вона робить модель стійкою до викидів та робить її ефективною в умовах, де дані містять шум чи невизначеності. Також варто зазначити, що SVR може бути витратною з точки зору обчислень, особливо при великих обсягах даних, через необхідність вирішення оптимізаційної задачі з великою кількістю параметрів [24].

2.4 Огляд моделі машинного навчання «Polynomial Regression»

Polynomial Regression є розширенням лінійної регресії, яке дозволяє моделювати нелінійні зв'язки між залежними та незалежними змінними. У відміну від лінійної регресії, де відносини виражаються лінійним рівнянням, поліноміальна регресія дозволяє використовувати поліноміальні функції вищих ступенів.

Для тренування моделі поліноміальної регресії використовуються ті ж методи оптимізації, що й для лінійної регресії, такі як градієнтний спуск. Оптимальні значення коефіцієнтів обираються так, щоб мінімізувати функцію втрат [25].

Важливо враховувати, що поліноміальна регресія може вести до перенавчання, особливо при використанні високих ступенів поліномів. Підбір оптимального ступеня полінома є важливим етапом в розробці моделі, оскільки недостатній чи занадто високий ступінь може призвести до невірного узагальнення.

Polynomial Regression дозволяє адаптувати модель до складних структур даних, де лінійні відносини недостатньо точно відображають реальний зв'язок між змінними. Використання поліноміальних функцій дозволяє більш гнучко моделювати вигляд графіка та адекватно апроксимувати нелінійні закономірності в даних.

Один з важливих аспектів використання Polynomial Regression - це визначення ступеня полінома. Недостатній ступінь може привести до недооцінки взаємозв'язків, тоді як занадто великий ступінь може призвести до перенавчання та неадекватного узагальнення на нових даних [26].

Важливим є також уникання перенавчання, оскільки Polynomial Regression може легко адаптуватися до шуму в даних та втрати узагальнювальної здатності. Для цього використовуються методи регуляризації, такі як LASSO або Ridge Regression.

Поліноміальна регресія широко використовується в різних областях, таких як фізика, економіка, біологія та інші, де залежність між змінними може бути складною та має нелінійний характер. Вона виступає ефективним інструментом для апроксимації та прогнозування в умовах складних зв'язків.

На рисунку 2.3 зображено приклад поліноміальної регресії.

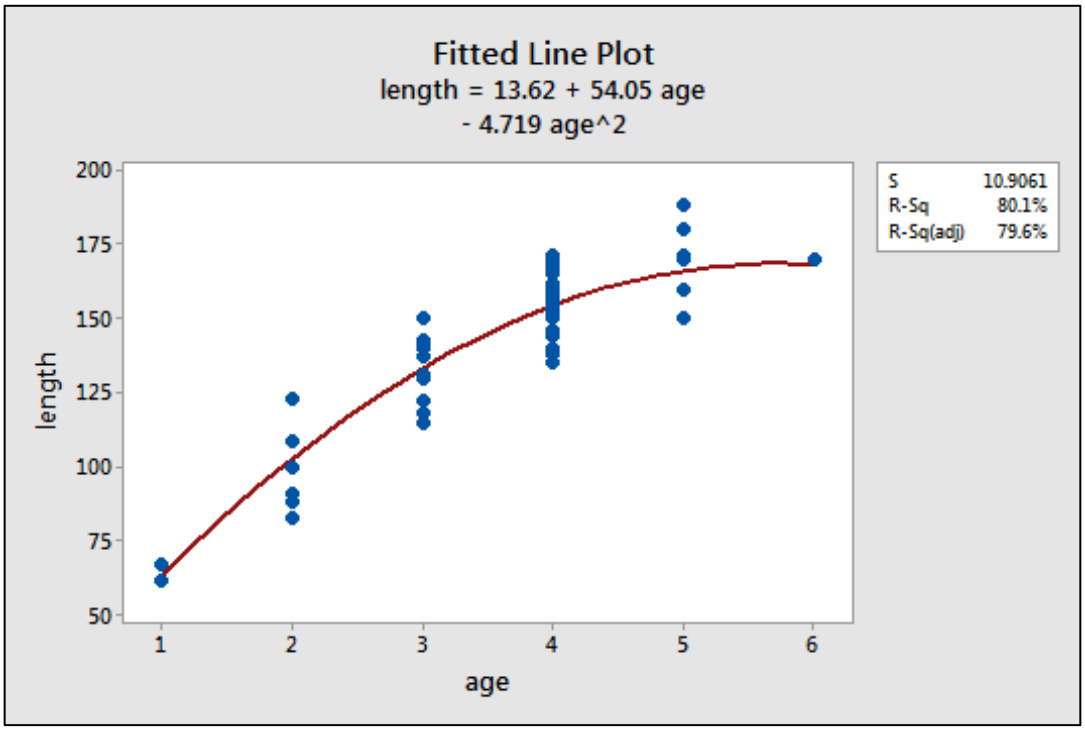


Рисунок 2.3 – Приклад поліноміальної регресії

Переваги поліноміальної регресії включають її гнучкість, здатність адаптуватися до нелінійних взаємозв'язків, можливість апроксимації складних функцій, збереження простоти лінійної регресії при низьких ступенях полінома та застосування в різних галузях. Поліноміальна регресія може ефективно розглядати взаємодії між змінними, зменшуючи вплив шуму в даних та знаходячи застосування там, де лінійні методи можуть бути неефективними. Однак важливо уникати перенавчання та визначати оптимальний ступінь полінома для кращої адаптації до конкретних даних [27].

2.5 Огляд моделі машинного навчання «Decision Tree Regressor»

Decision Tree Regressor є методом машинного навчання, який використовується для задач регресії. Основна ідея полягає в тому, щоб побудувати дерево рішень, яке допомагає прогнозувати неперервні числові значення на основі вхідних ознак.

При побудові дерева рішень, алгоритм рекурсивно розбиває набір даних на більш однорідні групи за допомогою різних порогів на вхідних ознаках. Кожен вузол у дереві представляє рішення для певного піднабору даних. Дерево росте до досягнення критерію зупинки, такого як максимальна глибина, мінімальна кількість прикладів у листку або інший визначений поріг.

У Decision Tree Regressor кожен листовий вузол містить прогнозоване числове значення для відповідного піднабору даних. Прогноз для нового прикладу обчислюється шляхом проходження дерева від кореня до листя, де значення листя стає прогнозом.

Цей метод володіє властивістю автоматичного вивчення нелінійних та взаємодіючих залежностей в даних. Однак Decision Tree Regressor може схильний до перенавчання, особливо при великій глибині дерева, що може призводити до неадекватного узагальнення на нових даних. Це може бути вирішено за допомогою технік обмеження глибини дерева або використання ансамблевих методів, таких як Random Forests або Gradient Boosted Trees.

Decision Tree Regressor може взаємодіяти з нелінійними та взаємодіючими залежностями в даних, і його гнучкість дозволяє враховувати різноманітні структури даних. Важливо враховувати його чутливість до перенавчання та те, що модель може вирішувати проблеми за рахунок деталізації, що може призвести до переоснащення.

Використання дерев рішень у вигляді Decision Tree Regressor дозволяє отримувати прогнози для числових значень та враховує вплив окремих ознак на результат. При цьому модель може бути особливо корисною, коли важливо розуміти та пояснити прогнози.

Рекомендується використовувати обмеження глибини дерева та враховувати гіперпараметри для забезпечення адекватної регуляризації та уникнення перенавчання. Decision Tree Regressor може бути важливим елементом у моделях ансамблю, які комбінують декілька дерев рішень для покращення загальної ефективності та стійкості моделі.

Decision Tree Regressor не вимагає нормалізації, легко інтерпретований, дозволяє важливість ознак, ефективно працює з категоріальними ознаками, використовується в ансамблях, таких як Random Forests, щоб поліпшити стійкість та уникнути перенавчання. Також важливо налаштовувати гіперпараметри для оптимальної ефективності та уникнення перенавчання.

2.6 Огляд моделі машинного навчання «Random Forest Regressor»

Random Forest Regressor - це метод машинного навчання, який базується на ансамблі дерев рішень. Він працює, комбінуючи прогнози багатьох дерев рішень, щоб отримати більш точний та стійкий прогноз. Кожне дерево будується на випадковому підмножині даних та випадковому підмножині ознак.

Random Forest Regressor вирішує проблему перенавчання, яка може виникнути при використанні одного дерева рішень, за рахунок рандомізації. Модель ефективно обробляє велику кількість ознак та може працювати з різноманітними типами даних.

Додатково, Random Forest Regressor надає важливість ознак, що дозволяє оцінити вплив кожної ознаки на прогноз. Це робить модель корисною для визначення важливих факторів у завданнях регресії.

Модель є потужним інструментом, який може автоматично адаптуватися до різноманітних завдань та виділяти важливі залежності в даних, зменшуючи при цьому ризик перенавчання.

Random Forest Regressor, будуючи кожне дерево на випадковому підмножині даних, дозволяє моделі більш ефективно уникати перенавчання та

робити більш стійкі прогнози. Це досягається завдяки рандомізації вибору прикладів та ознак для побудови кожного дерева, що забезпечує різноманітність усієї моделі.

Окрім того, важливою характеристикою є здатність Random Forest Regressor ефективно працювати з великою кількістю ознак, включаючи категоріальні та числові, і враховувати їх важливість у формуванні прогнозів.

Модель може автоматично обробляти відсутні дані, вона є стійкою до викидів і шуму в даних, що дозволяє використовувати її в різних сценаріях та природах даних.

Random Forest Regressor є популярним інструментом в машинному навчанні, зокрема в завданнях регресії, завдяки його здатності до впорядкування складних взаємозв'язків у даних та високій стабільності прогнозів.

На рисунку 2.4 зображено приклад використання моделі Random Forest Regressor.

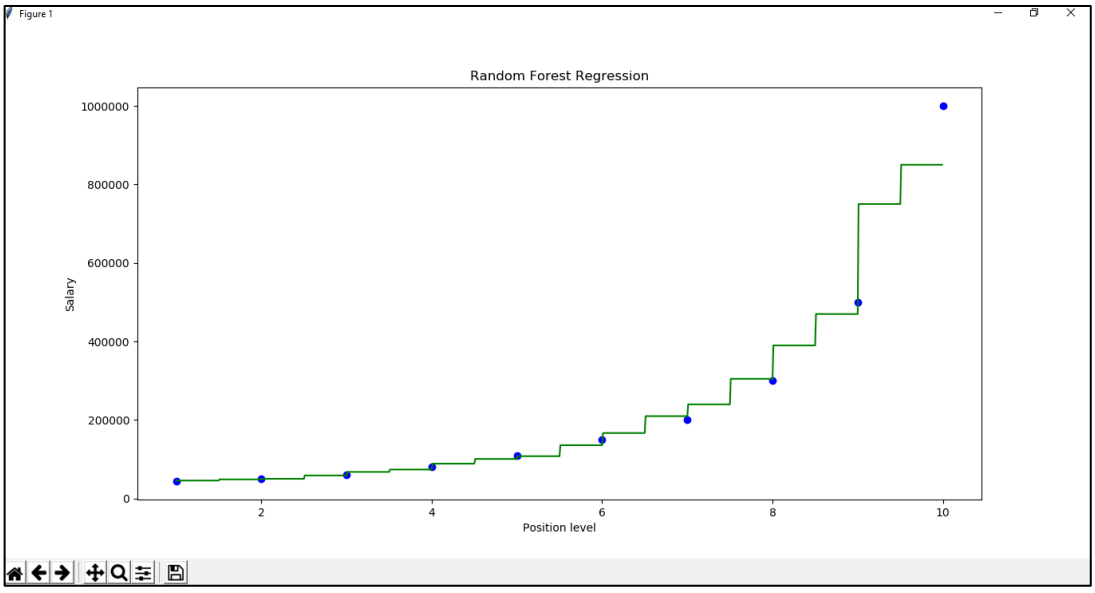


Рисунок 2.4 – Приклад використання моделі Random Forest Regressor

Random Forest Regressor володіє кількома перевагами. Це стійка до перенавчання завдяки рандомізації, яка враховується при побудові кожного дерева. Модель ефективно враховує багато ознак, включаючи категоріальні та

числові. Вона універсальна та застосовується до різноманітних завдань регресії. Random Forest Regressor може обробляти великі обсяги даних та взаємодіяти з відсутніми значеннями. Модель надає важливість кожній ознаці, полегшуючи інтерпретацію. Завдяки ансамблю дерев рішень, вона забезпечує точні та стабільні прогнози, розглядаючи різні аспекти даних. Висока точність та гнучкість роблять Random Forest Regressor ефективним інструментом для багатьох завдань регресії.

2.7 Висновки

У даному розділі обґрунтовано використання інформаційних технологій для рішення задачі дослідження. Вибір середовища розробки був обґрунтований з урахуванням потреб та специфіки дослідження.

Модель машинного навчання "Linear Regression" детально оглянуто, дозволяючи отримати уявлення про її принципи та можливості в контексті використання в дослідженні. Модель "Support Vector Regression" також пройшла огляд, розкриваючи основні аспекти та переваги цієї методології машинного навчання.

Особлива увага була приділена моделі машинного навчання "Random Forest Regressor". Її вивчення дозволило зрозуміти, як цей ансамбль дерев рішень може бути ефективним інструментом для регресійних завдань. Висвітлені переваги, такі як стійкість до перенавчання, важливість ознак та універсальність, роблять її цікавим вибором для використання у контексті дослідження.

Усе це створює технологічну базу для подальшого дослідження та розв'язання поставлених завдань у вибраному напрямку, використовуючи відомості та можливості, що надаються обраними моделями машинного навчання.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЗАРОБІТНОЇ ПЛАТИ

3.1 Огляд бібліотек

Для створення графіків за допомогою даних використано бібліотеки seaborn та plotly. Для нормалізації та кодування даних використано бібліотеки StandardScaler та OrdinalEncoder з бібліотеки scikit-learn. Також використано інші бібліотеки для побудови моделей машинного навчання, такі як xgboost, sklearn та tensorflow.keras.NumPy: Це фундаментальний пакет для наукових обчислень у Python. Він пропонує потужні N-вимірні масиви, зручні функції для роботи з ними, а також інструменти для проведення складних математичних обчислень.

3.1.1 Огляд бібліотеки Pandas

Pandas - це потужна бібліотека для маніпулювання та аналізу даних в середовищі Python. Вона надає структури даних високого рівня, такі як DataFrame і Series, які значно полегшують операції з даними.

Основні структури даних в pandas:

DataFrame: Це двовимірна таблична структура даних з іменованими рядками та стовпцями. Можна уявити її як аркуш електронної таблиці або SQL-таблицю.

На рисунку 3.1 зображено приклад коду використання DataFrame в Pandas.

```
import pandas as pd
data = {'Name': ['John', 'Jane', 'Bob'],
        'Age': [28, 24, 22],
        'City': ['New York', 'San Francisco', 'Los Angeles']}
df = pd.DataFrame(data)
```

Рисунок 3.1 – Код використання DataFrame в Pandas

Це приклад коду з використанням бібліотеки pandas для створення DataFrame з даними про імена, вік та місто проживання трьох осіб. Він перетворює словник з даними у табличну форму, яку можна аналізувати та візуалізувати. DataFrame - це двовимірна структура даних, яка дозволяє зберігати дані у вигляді таблиці з рядками та стовпцями.

Series – це одновимірний масив міткированих даних. Вони є основним елементом створення стовпців у DataFrame.

На рисунку 3.2 зображено приклад коду використання Series в Pandas.

```
ages = pd.Series([28, 24, 22], name='Age')
```

Рисунок 3.2 – Код використання Series в Pandas

Цей приклад коду створює серію даних в бібліотеці pandas з назвою 'Age', яка містить три вікові значення: 28, 24, та 22. Серія в pandas - це одновимірний масив, що може містити будь-який тип даних, і вона має мітки осі, які називаються індексами. Цей фрагмент коду може бути використаний для аналізу вікових даних у статистичних обчисленнях або як частина більшого набору даних.

Основні функції та можливості pandas:

На рисунку 3.3 зображено приклад коду зчитування та запису даних в Pandas.

```
df = pd.read_csv('file.csv')
   df.to_csv('new_file.csv', index=False)
```

Рисунок 3.3 – Код зчитування та запис даних в Pandas

Цей приклад коду використовує бібліотеку pandas для читання та збереження CSV файлів у Python. Він спочатку читає дані з 'file.csv' до DataFrame, а потім зберігає цей DataFrame у новий файл 'new_file.csv', не включаючи індекси рядків. Це корисно для обробки та обміну табличними даними без зайвої інформації.

На рисунку 3.4 зображено приклад коду індексації та вибірки даних в Pandas.

```
df['Name']            # Вибірка стовпця
df.loc[0]             # Вибірка рядка за міткою
df.iloc[0]            # Вибірка рядка за позначкою
df[['Name', 'Age']]   # Вибірка кількох стовпців
```

Рисунок 3.4 – Код індексації та вибірки даних в Pandas

Цей приклад коду демонструє, як вибирати дані з DataFrame у Python за допомогою бібліотеки pandas. Він показує вибірку стовпця, рядка за міткою або індексом, а також вибірку кількох стовпців. Це основні операції, які часто використовуються при аналізі даних.

На рисунку 3.5 зображено приклад коду операцій з даними в Pandas.

```
df.head(3)          # Перегляд перших 3 рядків
df.describe()       # Статистика даних
df['Age'].mean()    # Середнє значення стовпця
```

Рисунок 3.5 – Код операцій з даними в Pandas

Цей приклад коду використовує бібліотеку pandas для аналізу даних у DataFrame. Він включає методи для перегляду перших рядків, отримання статистичного опису та обчислення середнього значення в стовпці “Age”. Це базові функції pandas для початкового аналізу даних.

На рисунку 3.6 зображено приклад коду заповнення пропущених значень в Pandas.

```
df.fillna(value)      # Заповнення пропущених значень
```

Рисунок 3.6 – Код заповнення пропущених значень в Pandas

Цей приклад коду використовує метод `fillna()` бібліотеки pandas для заповнення пропущених значень у DataFrame `df`. Це стандартна операція при обробці даних, що дозволяє уникнути помилок при аналізі чи візуалізації даних, які містять пропуски. Метод `fillna()` може приймати різні параметри для визначення, якими значеннями заповнювати пропущені дані.

На рисунку 3.7 зображено приклад коду групування та агрегації в Pandas.

```
df.groupby('City')['Age'].mean()  # Середній вік за містом
```

Рисунок 3.7 – Код групування та агрегації в Pandas

У цьому прикладі коду використовується бібліотека Pandas для розрахунку середнього віку людей за містом. Це робиться за допомогою методу groupby(), який групує дані за містом, а потім методу mean(), який обчислює середнє значення для кожного міста.

На рисунку 3.8 зображено приклад коду зміни даних в Pandas.

```
df['Age'] += 1        # Збільшення віку на 1
```

Рисунок 3.8 – Код зміни даних в Pandas

У цьому прикладі коду використовується бібліотека Pandas для збільшення віку всіх людей у датафреймі на один рік. Це робиться за допомогою оператора +=, який додає значення 1 до стовпця "Age" у датафреймі.

На рисунку 3.9 зображено приклад коду сортування в Pandas.

```
df.sort_values(by='Age', ascending=False)  # Сортування за віком у зворотньому порядку
```

Рисунок 3.9 – Код сортування в Pandas

У цьому коді прикладі коду використовується бібліотека Pandas для сортування даних у датафреймі за віком у зворотньому порядку. Це робиться за допомогою методу sort_values(), який приймає два аргументи: стовпець, за яким сортувати дані, і параметр ascending, який визначає, чи сортувати дані у зростаючому чи спадаючому порядку. У цьому випадку параметр ascending встановлено на значення False, тому дані сортуються у спадаючому порядку.

Це лише декілька прикладів можливостей бібліотеки pandas. Вона є потужним інструментом для роботи з даними в Python та використовується широко в аналізі даних, машинному навчанні та інших областях. seaborn: Це бібліотека для створення статистичних графіків у Python. Вона базується на matplotlib та тісно інтегрована з pandas. seaborn дозволяє легко створювати привабливі та інформативні графіки.

3.1.2 Огляд бібліотеки Plotly

Plotly - це бібліотека для створення інтерактивних графіків та візуалізації даних в мові програмування Python. Вона надає можливість легко створювати високоякісні графіки, які можна взаємодіяти з ними в онлайн-середовищі. Основні характеристики бібліотеки Plotly:

Plotly підтримує широкий спектр типів графіків, таких як лінійні графіки, стовпчасті гістограми, точкові графіки, 3D-графіки, теплові карти, кругові графіки, графіки розподілу, барометри та багато інших.

Один з головних плюсів Plotly - це можливість створювати інтерактивні графіки, які можна взаємодіяти з ними, навіть відредагувати та анімувати. Користувачі можуть масштабувати, повертати та вибирати дані безпосередньо в графіці. [28]

Plotly дозволяє легко ділитися створеними графіками онлайн. Можна створити інтерактивний графік, опублікувати його на веб-сайті Plotly та поділитися посиланням.

Поміж іншими мовами, Plotly підтримує Python, R та JavaScript. Це означає, що ви можете створювати графіки в одній мові та взаємодіяти з ними в інших.

Можна легко виводити графіки у форматі HTML, які можна вбудовувати у веб-сторінки. Також можливе збереження графіків у різних форматах, таких як PNG, PDF, SVG тощо.

На рисунку 3.10 зображено основні етапи створення графіків у Plotly.

```python
import plotly.express as px
# Створення DataFrame або використання існуючого
df = px.data.iris()
# Створення графіку за допомогою вищеспецифікованих функцій Plotly Express
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='species', size='petal_length', title='Scatter Plot')
# Виведення графіку
fig.show()
```

Рисунок 3.10 – Код для створення графіку в Plotly

У цьому прикладі коду використовується бібліотека Plotly Express для створення діаграми розсіювання. Діаграма показує зв'язок між шириною і довжиною пелюсток квітки ірису.

Загалом, Plotly є потужним інструментом для візуалізації даних у Python і широко використовується в аналізі даних, наукових дослідженнях, а також для створення красивих та інтерактивних візуалізацій для веб-додатків.

3.1.3 Огляд інструменту StandardScaler в бібліотеці scikit-learn

StandardScaler є інструментом для стандартизації даних в машинному навчанні. Він належить до бібліотеки `scikit-learn` в Python і використовується для масштабування ознак. Стандартизація є процесом перетворення даних так, щоб вони мали середнє значення 0 і стандартне відхилення 1. Це може поліпшити результати багатьох алгоритмів машинного навчання, особливо тих, які використовують відстані між точками (наприклад, метод k-найближчих сусідів або метод опорних векторів).

Основні кроки, які виконує StandardScaler:

– Визначається середнє значення кожного стовпця даних;

– Визначається стандартне відхилення для кожного стовпця даних;

– Для кожного значення в стовпці віднімається середнє значення і результат ділиться на стандартне відхилення;

– Результати стандартизації мають тепер середнє значення 0 і стандартне відхилення 1.

На рисунку 3.11 зображено приклад використання StandardScaler в scikit-learn.

```python
from sklearn.preprocessing import StandardScaler
import numpy as np
# Приклад даних
data = np.array([[1.0, 2.0, 3.0],
                 [4.0, 5.0, 6.0],
                 [7.0, 8.0, 9.0]])
# Створення об'єкту StandardScaler
scaler = StandardScaler()
# Виклик fit_transform для стандартизації даних
scaled_data = scaler.fit_transform(data)
print("Оригінальні дані:\n", data)
print("\nСтандартизовані дані:\n", scaled_data)
```

Рисунок 3.11 – Код для використання StandardScaler в бібліотеці scikit-learn

fit_transform виконує обчислення середнього та стандартного відхилення і застосовує стандартизацію до даних. Після цього `scaled_data` містить стандартизовані значення. OrdinalEncoder: Це інструмент з scikit-learn для кодування категорійних ознак як масив цілих чисел. Він перетворює категорійні дані в порядкові цілі числа, що може бути корисним для певних типів моделей машинного навчання.

3.1.4 Огляд бібліотеки Keras

Keras - це високорівневий інтерфейс для побудови та тренування моделей глибокого навчання в мові програмування Python. Початково розроблений як незалежна бібліотека, Keras став стандартною обгорткою для нейронних мереж на платформі TensorFlow.

Основні риси та особливості бібліотеки Keras:

- Keras розроблений так, щоб бути простим та легким для використання. Його API є інтуїтивно зрозумілим, що робить його ідеальним вибором для початківців в глибокому навчанні.
- Keras є модульною бібліотекою, що дозволяє легко компонувати різні компоненти для побудови власних моделей. Модулі можна комбінувати, щоб створювати різні архітектури нейронних мереж.
- Керас підтримує різні обчислювальні міста (backend), такі як TensorFlow, Theano та Microsoft Cognitive Toolkit (CNTK). В основному, він використовується як обгортка для TensorFlow, але користувачі можуть змінити backend за необхідності.
- Keras підтримує останні інновації у глибокому навчанні, такі як автоматичне диференціювання (важливе для градієнтного спуску), механізми регуляризації та оптимізації.
- В Keras ви можете легко визначити моделі з одним або декількома входами та виходами. Це особливо корисно для задач, де потрібно об'єднати різні типи вхідних даних або реалізувати моделі з декількома виходами.

– Keras надає широкий вибір вбудованих функцій втрат і метрик, які можна використовувати під час тренування моделі.

– В Keras є можливість створювати графічне представлення структури моделі та візуалізувати процес тренування.

– В Keras є можливість контролювати процес тренування за допомогою зворотного визову (callbacks), які дозволяють вам викликати певні функції під час різних етапів тренування.

На рисунку 3.12 зображено простий приклад створення та тренування моделі в Keras:

```
from keras.models import Sequential
from keras.layers import Dense
# Створення моделі
model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
# Компіляція моделі
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
# Тренування моделі
model.fit(x_train, y_train, epochs=10, batch_size=32)
```

Рисунок 3.12 – Код створення та тренування моделі в Keras

Цей код створює та навчає просту нейронну мережу з двома шарами. Перший шар має 64 вузли з активацією ReLU, а другий шар має 10 вузлів з активацією Softmax. Метрика точності використовується для оцінки ефективності моделі.

3.2 Імпорт бібліотек та огляд датасету

На рисунку 3.13 зображено фрагмент коду на Python який включає імпорт бібліотек для різних завдань у сфері обробки даних та машинного навчання.

```python
# Data Loading and Managing
import numpy as np
import pandas as pd
from tensorflow import keras

# Data Visualization
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Data Preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split

# Model
import xgboost as xgb
from sklearn.svm import SVR
from tensorflow.keras import layers
from sklearn.tree import  DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

# Model Performance
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

Рисунок 3.13 – Імпорт бібліотек

Код завантажує дані з файлу, розбиває їх на навчальний і тестовий набори, масштабує і кодує ознаки, а потім тренує кілька моделей регресії. Моделі оцінюються за допомогою коефіцієнта детермінації R2, середньоквадратичної помилки MSE та середньої абсолютної помилки MAE.

У цьому розділі наша головна мета — заглибитися в процес завантаження даних, що дозволить нам отримати цінну інформацію та провести невізуальний аналіз. За допомогою цього аналізу досягнуто повне розуміння складної поведінки наших наборів даних, а також розкрити базові закономірності розподілу, властиві даним. Заглиблюючись у ці фундаментальні аспекти, можна озброїтися знаннями, необхідними для прийняття обґрунтованих рішень і отримувати значущу інформацію з наших наборів даних (рис. 3.14).

[52]:
```
# Load data
df = pd.read_csv(filepath_or_buffer=file_path)

# Quick look
df.head()
```

[52]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES	L
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US	S
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US	S
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA	M
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA	M

Рисунок 3.14 – Імпорт та огляд даних в датасеті

Маючи повне розуміння стовпців, включаючи їхні назви та відповідне значення, тепер можна вивчити конкретні типи даних, інкапсульованих у кожному стовпці (рис. 3.15).

[53]:
```
# Information about the data columns/features.=
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   work_year           3755 non-null   int64
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
```

+ Code + Markdown

Рисунок 3.15 – Інформація про стовпці даних та функцій

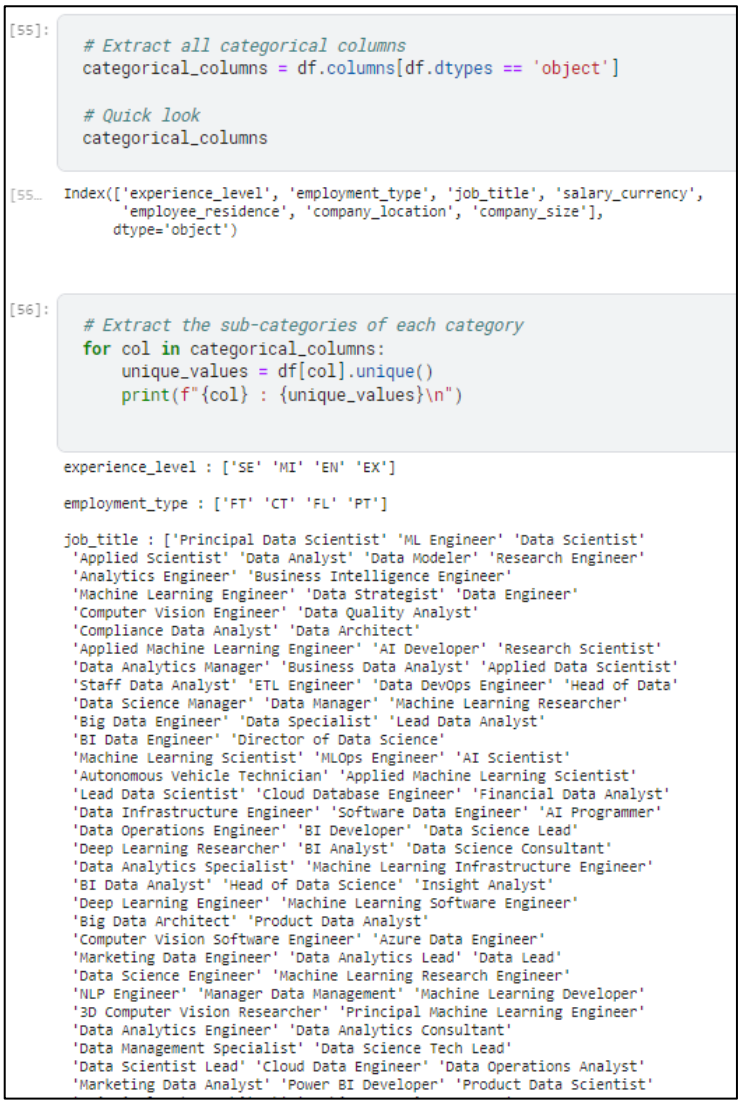
Ми отримали позитивні новини про те, що наш набір даних повний, тобто він не містить нульових або відсутніх значень. Це вигідна ситуація, оскільки вона звільняє нас від завдання заповнення відсутніх даних, дозволяючи нам зосередити увагу та ресурси на інших критичних завданнях. Крім того, отримано глибоку інформацію щодо складу нашого набору даних. Зокрема, визначено чотири стовпці, які містять числові дані, а саме:

- work_year;
- salary:
- salary_in_usd;
- remote_ratio.

Крім того, визначено сім стовпців, які містять категоричні дані, що представляють інформацію в окремих категоріях або групах:

- experience_level;
- employment_type;
- job_title;
- salary_currency;
- employee_residence;
- company_location;
- company_size.

Ця категоризація забезпечує більш чітке розуміння структури та складу нашого набору даних, відкриваючи шлях для ефективного аналізу та дослідження даних (рис. 3.16).



```
[55]:
# Extract all categorical columns
categorical_columns = df.columns[df.dtypes == 'object']

# Quick look
categorical_columns

[55…
Index(['experience_level', 'employment_type', 'job_title', 'salary_currency',
       'employee_residence', 'company_location', 'company_size'],
      dtype='object')

[56]:
# Extract the sub-categories of each category
for col in categorical_columns:
    unique_values = df[col].unique()
    print(f"{col} : {unique_values}\n")

experience_level : ['SE' 'MI' 'EN' 'EX']

employment_type : ['FT' 'CT' 'FL' 'PT']

job_title : ['Principal Data Scientist' 'ML Engineer' 'Data Scientist'
 'Applied Scientist' 'Data Analyst' 'Data Modeler' 'Research Engineer'
 'Analytics Engineer' 'Business Intelligence Engineer'
 'Machine Learning Engineer' 'Data Strategist' 'Data Engineer'
 'Computer Vision Engineer' 'Data Quality Analyst'
 'Compliance Data Analyst' 'Data Architect'
 'Applied Machine Learning Engineer' 'AI Developer' 'Research Scientist'
 'Data Analytics Manager' 'Business Data Analyst' 'Applied Data Scientist'
 'Staff Data Analyst' 'ETL Engineer' 'Data DevOps Engineer' 'Head of Data'
 'Data Science Manager' 'Data Manager' 'Machine Learning Researcher'
 'Big Data Engineer' 'Data Specialist' 'Lead Data Analyst'
 'BI Data Engineer' 'Director of Data Science'
 'Machine Learning Scientist' 'MLOps Engineer' 'AI Scientist'
 'Autonomous Vehicle Technician' 'Applied Machine Learning Scientist'
 'Lead Data Scientist' 'Cloud Database Engineer' 'Financial Data Analyst'
 'Data Infrastructure Engineer' 'Software Data Engineer' 'AI Programmer'
 'Data Operations Engineer' 'BI Developer' 'Data Science Lead'
 'Deep Learning Researcher' 'BI Analyst' 'Data Science Consultant'
 'Data Analytics Specialist' 'Machine Learning Infrastructure Engineer'
 'BI Data Analyst' 'Head of Data Science' 'Insight Analyst'
 'Deep Learning Engineer' 'Machine Learning Software Engineer'
 'Big Data Architect' 'Product Data Analyst'
 'Computer Vision Software Engineer' 'Azure Data Engineer'
 'Marketing Data Engineer' 'Data Analytics Lead' 'Data Lead'
 'Data Science Engineer' 'Machine Learning Research Engineer'
 'NLP Engineer' 'Manager Data Management' 'Machine Learning Developer'
 '3D Computer Vision Researcher' 'Principal Machine Learning Engineer'
 'Data Analytics Engineer' 'Data Analytics Consultant'
 'Data Management Specialist' 'Data Science Tech Lead'
 'Data Scientist Lead' 'Cloud Data Engineer' 'Data Operations Analyst'
 'Marketing Data Analyst' 'Power BI Developer' 'Product Data Scientist'
```

Рисунок 3.16 – Категоризація даних

На цьому етапі проведено детальний огляд категорійних змінних нашого набору даних, визначаючи унікальні значення, які вони приймають. Ця процедура допоможе нам розібратися в сутності та значимості кожної підгрупи в цих змінних [29].

Рівень досвіду:

- Унікальні значення: ['SE', 'MI', 'EN', 'EX']
- Значення: цей стовпець представляє різні рівні професійного досвіду. Категорії «SE» означають «старший рівень», «MI» — «середній рівень», «EN» — «початковий рівень» і «EX» — «виконавчий рівень». Ці значення вказують на рівень досвіду, пов’язаний з кожним введенням даних.

Тип зайнятості:

- Унікальні значення: ['FT', 'CT', 'FL', 'PT']

Значення: цей стовпець означає тип зайнятості для кожного запису даних. Категорії «FT» представляють «Full-time», «CT» — «Contract», «FL» — «Freelance» і «PT» — «Part-time». Ці цінності дають зрозуміти природу домовленостей про працевлаштування.

Посада:

- Унікальні значення: [Список різних посад]
- Значення: Стовпець посади вказує на конкретне призначення або роль, пов’язану з кожним записом даних. Він надає інформацію про посади, які обіймають окремі особи, наприклад «Інженер-програміст», «Аналітик даних» або «Керівник проекту».

Валюта зарплати:

- Унікальні значення: ['EUR', 'USD', 'INR', 'HKD', 'CHF', 'GBP', 'AUD', 'SGD', 'CAD', 'ILS', 'BRL', 'THB' ', 'PLN', 'HUF', 'CZK', 'DKK', 'JPY', 'MXN', 'TRY', 'CLP']
- Значення: у цьому стовпці вказується валюта, у якій позначається зарплата для кожного запису даних. Значення представляють різні валюти, такі як "EUR" (євро), "USD" (долар США), "GBP" (британський фунт), "JPY" (японська єна) тощо. Ці значення пропонують чіткість щодо валюти, яка використовується для зарплати представництво.

Місце проживання працівника:

- Унікальні значення: [Список різних місць проживання працівників]
- Значення. Стовпець місця проживання працівника представляє місцезнаходження або країну проживання для кожного запису даних. Він надає інформацію про місцеперебування співробітників, наприклад «Іспанія» (ES), «Сполучені Штати» (США), «Канада» (CA), «Німеччина» (DE) тощо.

Розташування компанії:

- Унікальні значення: [Список різних місцезнаходження компанії]

– Значення: цей стовпець позначає місцезнаходження або країну, де розташована компанія-наймач для кожного запису даних. Він пропонує інформацію про географічний розподіл компаній із такими значеннями, як «Іспанія» (ES), «Сполучені Штати» (США), «Канада» (CA), «Німеччина» (DE) тощо.

Розмір компанії:

– Унікальні значення: ['L', 'S', 'M']

– Значення: Стовпець розміру компанії вказує величину або масштаб компанії-роботодавця для кожного запису даних. Категорії «L» означають «Великий», «S» — «Малий» і «M» — «Середній». Ці значення дають змогу зрозуміти діапазон розмірів компаній у наборі даних [30].

Описова статистика: описова статистика надає вичерпний підсумок даних шляхом обчислення ключових показників, таких як середнє значення, медіана, мода, стандартне відхилення, діапазон і процентилі. Ці статистичні дані пропонують цінне розуміння центральної тенденції, мінливості та розподілу змінних, що дозволяє краще зрозуміти набір даних (рис. 3.17).

```
# Descriptive analysis
df.describe()
```

	work_year	salary	salary_in_usd	remote_ratio
count	3755.000000	3.755000e+03	3755.000000	3755.000000
mean	2022.373635	1.906956e+05	137570.389880	46.271638
std	0.691448	6.716765e+05	63055.625278	48.589050
min	2020.000000	6.000000e+03	5132.000000	0.000000
25%	2022.000000	1.000000e+05	95000.000000	0.000000
50%	2022.000000	1.380000e+05	135000.000000	0.000000
75%	2023.000000	1.800000e+05	175000.000000	100.000000
max	2023.000000	3.040000e+07	450000.000000	100.000000

Рисунок 3.17 – Описова статистика

З описової статистики змінних work_year, salary, salary_in_usd і remote_ratio можна отримати такі цінні відомості:

work_year:

- Набір даних містить загалом 3755 записів для змінної work_year.
- Середній (середній) робочий рік становить приблизно 2022,37, що свідчить про те, що більшість осіб у наборі даних нещодавно працювали.
- Стандартне відхилення 0,69 вказує на те, що точки даних відносно близькі до середнього значення, вказуючи на вузький розкид у розподілі років роботи.
- Мінімальне значення 2020 передбачає, що набір даних включає осіб із не менше двох років досвіду роботи.
- Квартилі (25%, 50%, 75%) надають інформацію про розподіл років роботи, причому більшість осіб потрапляє в діапазон 2022-2023 років.

salary та salary_in_usd:

- Змінні salary та salary_in_usd мають однакову статистику.
- Підрахунок 3755 означає, що для цих змінних немає пропущених значень.
- Середня зарплата становить приблизно 190 695,6 доларів США, що свідчить про відносно високий рівень середньої зарплати в наборі даних.
- Стандартне відхилення в 671 676,5 доларів США вказує на широкі варіації рівнів зарплати.
- Мінімальна зарплата в $6000 передбачає наявність у наборі даних низькооплачуваних посад.
- Квартилі надають інформацію про розподіл зарплат, причому медіана (50-й процентиль) зарплати становить приблизно 138 000 доларів США.
- Максимальна зарплата в 30 400 000 доларів США вказує на наявність потенційних викидів або надзвичайно високі зарплати.

remote_ratio:

- Змінна remote_ratio представляє частку віддаленої роботи.

- Кількість 3755 означає, що для цієї змінної немає пропущених значень.
- Середній віддалений коефіцієнт становить приблизно 46,27%, що вказує на те, що в середньому люди в наборі даних мають помірний обсяг віддаленої роботи.
- Стандартне відхилення 48,59 свідчить про широкий діапазон коефіцієнтів віддаленої роботи, де деякі люди не працюють віддалено, а інші мають 100% віддалену роботу.
- Квартилі дають уявлення про розподіл коефіцієнтів віддаленої роботи, причому більшість осіб не мають віддаленої роботи (коефіцієнт 0%) на основі значення 25-го процентиля.
- Максимальне значення 100 % означає, що деякі особи в наборі даних працюють виключно віддалено.

3.3 Проведення розвідувального аналізу

EDA або аналіз даних на основі відкритих джерел (Exploratory Data Analysis) - це метод дослідження даних, що дозволяє аналізувати, розуміти і виявляти патерни в наборі даних перед подальшим моделюванням або використанням їх для прийняття рішень. EDA використовується в багатьох галузях, включаючи статистику, машинне навчання, аналіз даних та науку про дані [8].

Основні етапи EDA включають такі процеси:

- Збір даних - це перший етап, де ви отримуєте доступ до даних і збираєте їх в один набір;
- Загальна інформація про дані, таку як кількість спостережень, ознаки (колонки), їх типи, розподіл значень і так далі;

- Виявлення і обробка відсутніх даних, аномалій, дублікатів і інших проблем в даних. Це може включати усунення викидів, нормалізацію і стандартизацію;
- Використовуйте графіки, діаграми та інші візуальні засоби для відображення розподілу даних, кореляцій між ознаками і іншої інформації. Це допомагає візуалізувати патерни та взаємозв'язки в даних;
- Розглядання статистичних характеристик даних, таких як середнє значення, медіана, дисперсія, квантилі та інші;
- Дослідження кореляцій між ознаками, що допомагає встановити, як вони впливають одна на одну;
- Пошук відомих або нових патернів у даних, що можуть бути важливими для подальшого аналізу;
- На основі виявлених патернів і взаємозв'язків створюються гіпотези для подальшого дослідження;
- Проведення статистичних тестів та інших методів, щоб підтвердити або відхилити гіпотези;
- Після завершення аналізу можна піти на наступний етап дослідження або приймати рішення на основі здобутої інформації.

EDA є важливим етапом в аналізі даних, оскільки дозволяє краще зрозуміти властивості даних, виявити можливі проблеми і патерни, а також сформулювати гіпотези, які можуть бути перевірені на подальших етапах аналізу.

На рисунку 3.18 зображено код для побудови графіку розподілу виплат за роками.

```python
# Extracting the values from the year column
data_values = df.ano_pago.value_counts()

# Creating the bar chart
fig = go.Figure(data=go.Bar(
    x=data_values.index,
    y=data_values.values,
    marker=dict(
        color=px.colors.sequential.Greens,
        line=dict(
            color='honeydew',
            width=2
        )
    ),
))

# Updating Layout
fig.update_traces(text=data_values.values, textposition='auto')
fig.update_layout(
    title='Distribution of payments made over the years',
    xaxis_title='Years',
    yaxis_title='Payments',
    autosize=False,
    width=500,
    height=500,
    showlegend=False
)

# Show the pie chart
fig.show()
```

Рисунок 3.18 - Код для побудови графіку розподілу виплат за роками

Цей фрагмент коду на Python створює стовпчасту діаграму за допомогою бібліотеки Plotly. Він встановлює значення даних з датафрейму df, створює фігуру та додає стовпець з даними значень та індексів, встановлює колір та ширину лінії стовпця, оновлює макет фігури та встановлює заголовок, мітки осей та розмір діаграми, і показує діаграму.

Розглянемо саму графік розподілу виплат за роками (рис. 3.19)

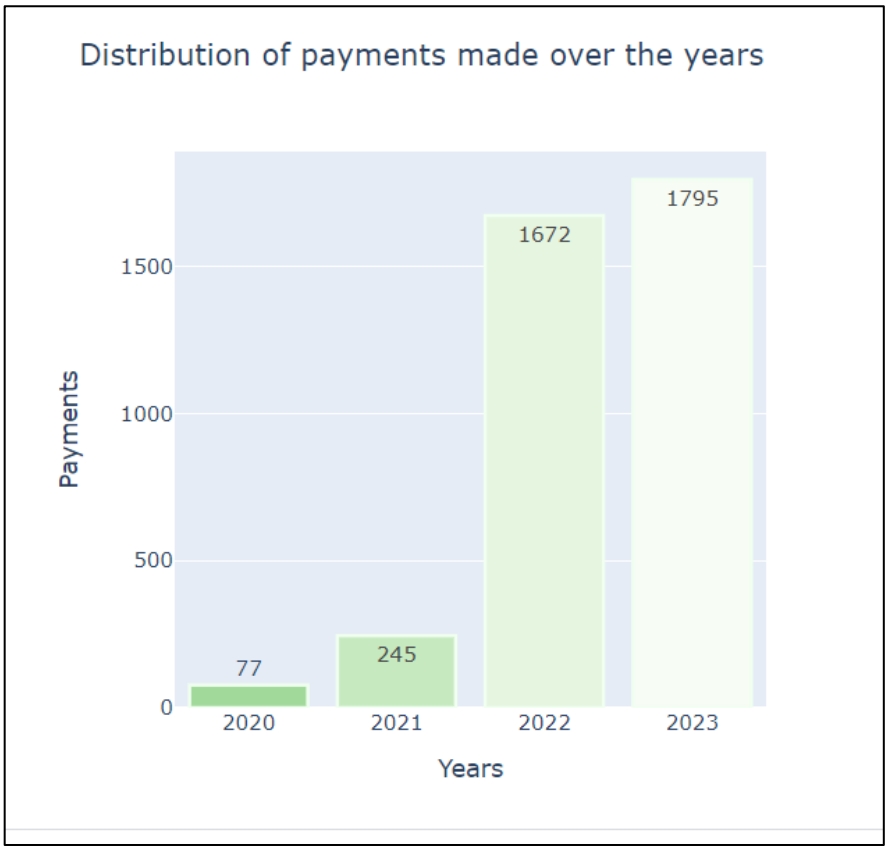


Рисунок 3.19 - Графік розподілу виплат за роками

Можна візуалізувати та інтерпретувати, що концентрація даних, яка уможливлює якісний аналіз, припадає на 2022 та 2023 роки. Це можна порівняти з 2021 і 2020 роками, які мають обмежену кількість даних за обидва роки, що може вплинути на загальний аналіз і висновки, зроблені на основі даних за кожний рік. [10]

З отриманих цифр видно, що вивчення та аналіз набору даних може бути упередженим через незбалансований розподіл даних у різні роки. Цей дисбаланс у розподілі даних викликає занепокоєння щодо потенційної упередженості результатів дослідження та аналізу. [14]

Обмеженість даних за 2020 та 2021 роки означає, що будь-які висновки, зроблені на основі аналізу, можуть не дати повного розуміння динаміки та тенденцій ринку в ці конкретні роки. Натомість, на отримані висновки може більше вплинути нещодавня динаміка ринку, представлена великою кількістю даних за 2022 та 2023 роки. [15]

На рисунку 3.20 зображено код для побудови графіку вивчення середньої заробітної плати за рік.

```
mean_salary = df.groupby('ano_pago')['salario_dolar'].mean()
median_salary = df.groupby('ano_pago')['salario_dolar'].median()


fig = make_subplots()


line_trace_median = go.Scatter(
    x=median_salary.index,
    y=median_salary.values,
    name="Median Salary",
    mode="lines+markers",
    line=dict(color='blue'),
    marker=dict(symbol='circle-open', size=15)
)
fig.add_trace(line_trace_median)


line_trace = go.Scatter(
    x=mean_salary.index,
    y=mean_salary.values,
    name="Average salary",
    mode="lines+markers",
    line=dict(color='darkorange'),
    marker=dict(symbol='circle-open', size=15)
)
fig.add_trace(line_trace)


fig.update_layout(
    title="Examining the average salary per year",
    xaxis_title="Year Worked",
    yaxis_title="Average salary",
    height=600
)


fig.show()
```

Рисунок 3.20 - Код для побудови графіку вивчення середньої заробітної плати за рік

Цей фрагмент коду на Python створює розсіювання за допомогою бібліотеки Plotly. Він використовує датафрейм та групує його за “ano_pago” та бере медіану “salario_dolar”. Він створює два сліди, один для медіанної

зарплати та один для середньої зарплати. Він встановлює назву осі X на “Year Worked” та назву осі Y на “Average Salary per Year”.

Розглянемо графік вивчення середньої заробітної плати за рік (рис. 3.21)

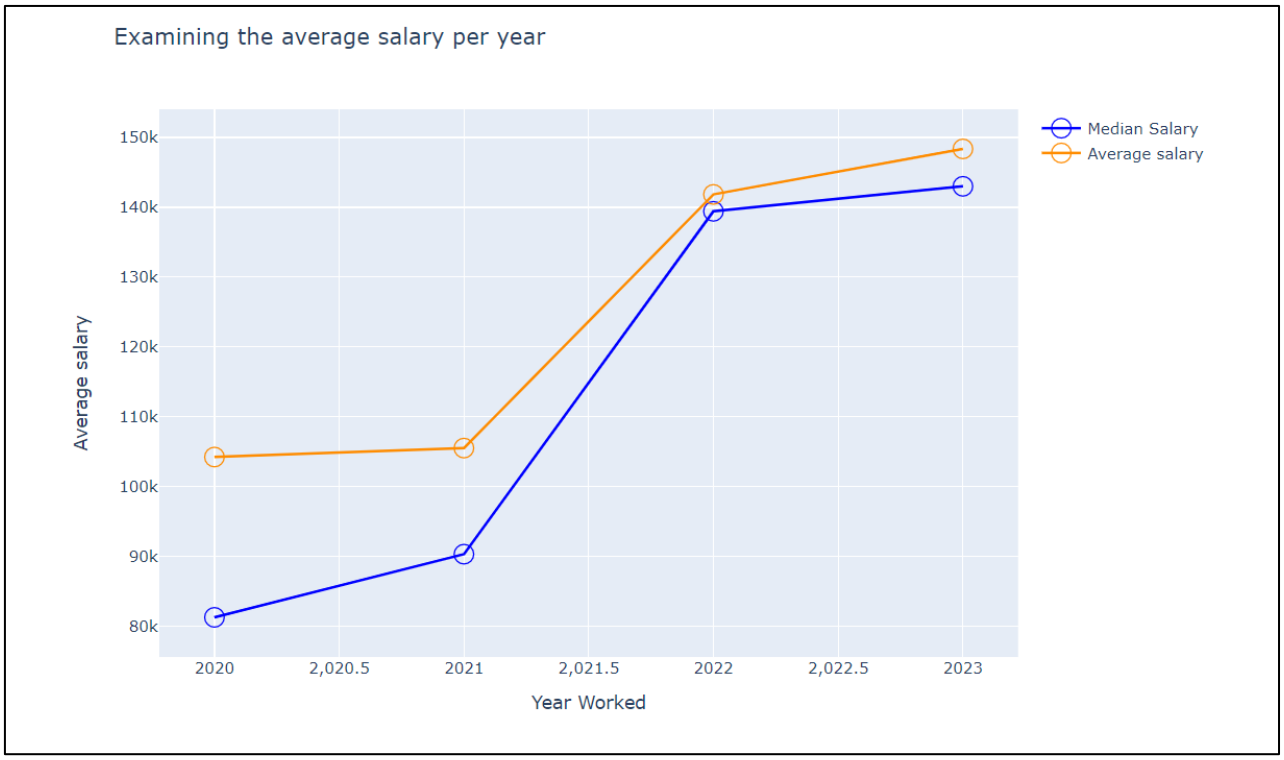


Рисунок 3.21 - Графік вивчення середньої заробітної плати за рік

На основі середньої заробітної плати (в доларах) за відпрацьований рік можна зробити наступні висновки:

Зростання заробітної плати: можна побачити зростання як середнього, так і медіанного показника заробітної плати в доларах за відповідні роки. Досягнувши у 2023 році таких значень:

– Середня: 148 589,4 USD;

– Медіана: 143 050 USD.

Також можна відзначити, що в даних є можливі викиди (які можна побачити на графіку нижче), які впливають на збільшення середнього показника в бік збільшення, особливо в 2020 і 2021 роках, коли різниця між двома показниками становила понад $10 000. [16]

З роками спостерігається стійка позитивна тенденція до зростання заробітної плати. Це свідчить про потенційний кар'єрний ріст і більші можливості для заробітку з більшим професійним досвідом. [17]

Розглянемо діаграми розподілу заробітної плати за роками (рис. 3.22), розподілу заробітної плати (у доларах) за роками (рис. 3.23), середньої заробітної плати за рік (рис. 3.24).

Розглянемо детальніше діаграму розподілу заробітної плати за роками (рис. 3.22):

- 2020 (червоний): Медіана та квартилі показують нижчі значення зарплати;
- 2021 (помаранчевий): Зростання медіани та квартилів, вищі за 2020;
- 2022 (зелений): Ще більше зростання, з медіаною вище 2021;
- 2023 (синій): Найвищі значення зарплати, з великим розмахом.

Кожен рік представлений боксплотом, що вказує на зростання середньої зарплати та її розподілу з часом.

На рисунку 3.22 зображено графік розподілу заробітної плати за роками.

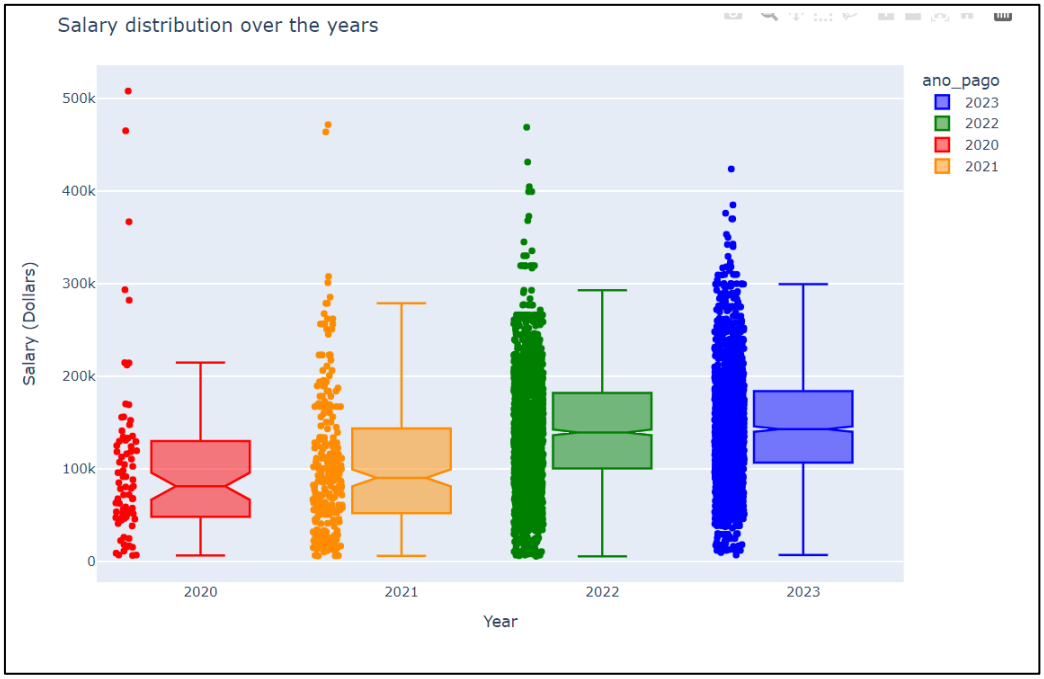


Рисунок 3.22 – Графік розподілу заробітної плати за роками

На рисунку 3.23 зображено діаграму розподілу заробітної плати (у доларах) за роками.

Згідно діаграми розподілу заробітної плати (у доларах) за роками (рис. 2.4) спостерігається:

– Лінія, що представляє розподіл заробітної плати у 2020 році, є найнижчою;

– Лінія 2021 року вища за лінію 2020 року, що свідчить про збільшення заробітної плати;

– Лінія 2022 року є найвищою, що вказує на подальше зростання заробітної плати;

– Усі лінії досягають піку у діапазоні заробітної плати 200k-300k доларів.

Ці дані можуть бути корисними для аналізу тенденцій заробітної плати протягом останніх років.

На рисунку 3.23 зображено графік заробітної плати за роками (у доларах).

Рисунок 3.23 – Графік розподілу заробітної плати (у доларах) за роками

Графік відображає позитивну тенденцію зростання середньої заробітної плати протягом цих років.

На рисунку 3.24 зображено графік середньої заробітної плати за рік.

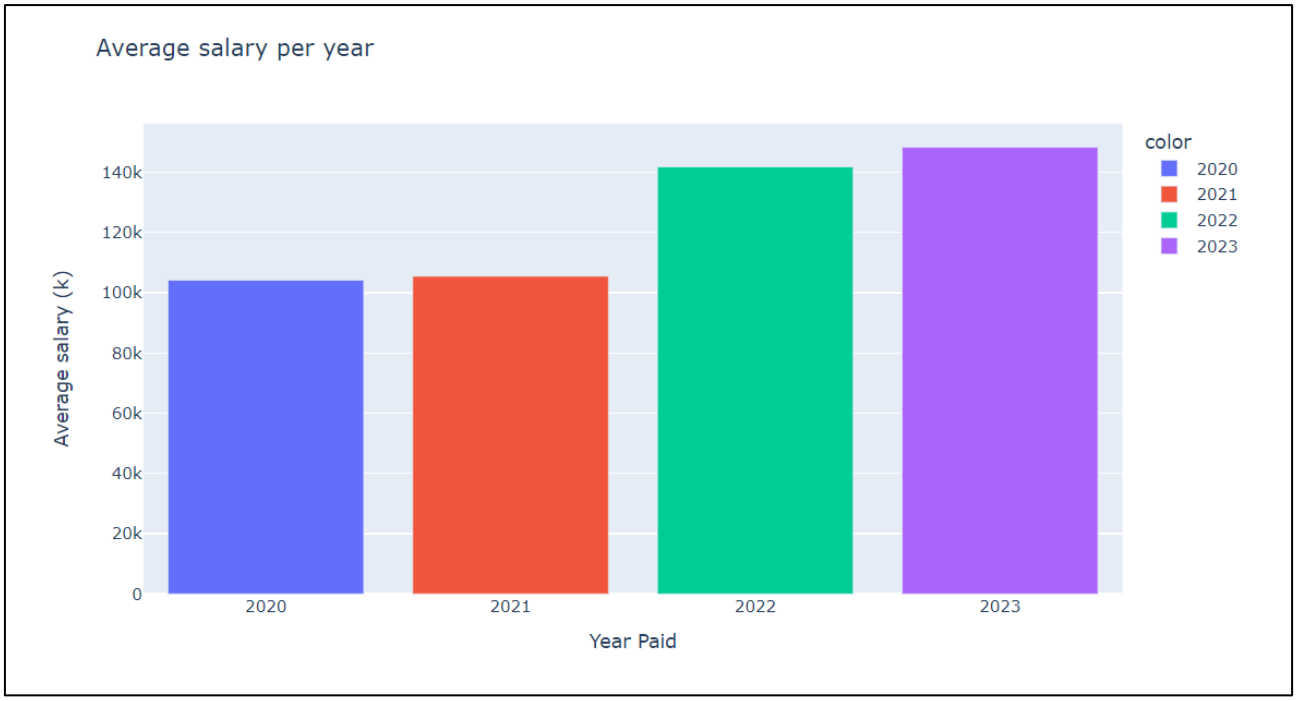


Рисунок 3.24 – Графік середньої заробітної плати за рік

Дивлячись на графіки вище, можна зробити наступні висновки:

По-перше, як вже показано на попередньому графіку (середнє та медіана). Очевидно, що кожен рік має значну кількість викидів. Ці викиди можна віднести до осіб, які на перший погляд працюють винятково добре, і, відповідно, отримують вищі зарплати, ніж інші.

По-друге, існує помітна тенденція в загальному розподілі заробітної плати з роками. Розподіл має тенденцію до розширення або поглиблення з плином часу. Зокрема, у 2023 році більше осіб отримують надзвичайно високі зарплати, тоді як інші отримують надзвичайно низькі зарплати. Таке розширення розподілу вказує на більший рівень варіації заробітних плат, оскільки значна кількість осіб перебуває на обох кінцях спектра, перевищуючи середнє значення.

Цікаво відзначити, що, незважаючи на загальну тенденцію до зростання заробітних плат у 2020-2023 роках, максимальна заробітна плата була зафіксована у 2020 році і становила 450 000 доларів США. Цей висновок

свідчить про те, що, хоча середня заробітна плата, можливо, поступово зростала протягом багатьох років, є випадки, коли окремі особи отримували винятково високу заробітну плату в попередні роки.[24]

Розглянемо на рисунку 3.25 діаграми розподілу на основі рівня досвіду:

– Ліва діаграма показує розподіл кількості значень за рівнем досвіду. Вона має три секції: S (3.94%), L (12.1%), і M (84%);

– Права діаграма відображає розподіл середньої зарплати за рівнем досвіду. Тут також три секції: L (34.8%), M (42.1%), і S (23%).

Кольори діаграм - червоний, помаранчевий, і фіолетовий, що відповідають різним розмірам компаній.

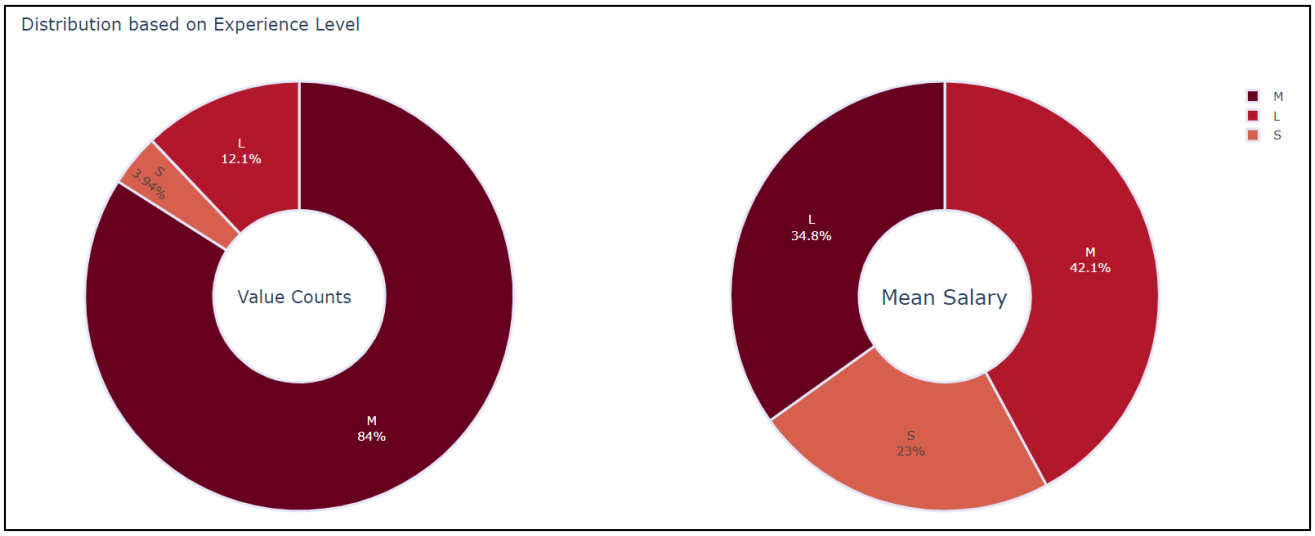


Рисунок 3.25 - Графік розподілу на основі рівня досвіду

Розглянемо на рисунку 3.26 графік розподілу середньої зарплати за рівнями досвіду.

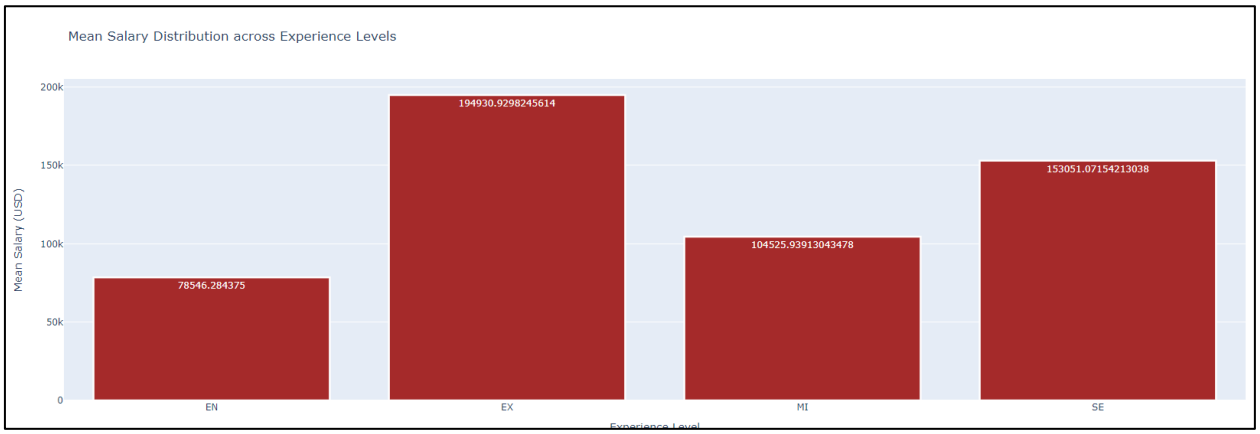


Рисунок 3.26 - Графік розподілу середньої зарплати за рівнями досвіду.

На основі наведеної вище візуалізації:

- Початковий рівень (EN): середня зарплата для працівників початкового рівня становить 78 546,28 доларів США. Це вказує на те, що люди на початку кар'єри або з обмеженим досвідом, як правило, мають нижчу зарплату порівняно з іншими рівнями досвіду. Позиції початкового рівня зазвичай мають нижчу винагороду, оскільки вони служать відправною точкою для формування навичок і отримання професійного досвіду.
- Керівний рівень (EX): середня зарплата для працівників на керівному рівні значно вища і становить 194 930,93 доларів США. Це свідчить про те, що особи, які займають керівні посади, такі як вище керівництво або посади рівня C, отримують вищу винагороду завдяки своєму великому досвіду, керівним обов'язкам і стратегічному впливу, який вони мають на організацію. Позиції керівного рівня часто вимагають поєднання досвіду, знань галузі та сильних здібностей до прийняття рішень.
- Середній рівень (MI): Середня зарплата працівників середнього рівня становить 104 525,94 доларів США. Це означає, що особи з середнім досвідом і навичками отримують помірний рівень винагороди. Посади середнього рівня зазвичай вимагають певного рівня знань і досвіду в цій галузі, і професіонали на цих посадах відповідають за виконання завдань, проектів і підтримку цілей організації.

– Старший рівень (SE): середня зарплата для працівників старшого рівня становить 153 051,07 доларів США. Це свідчить про те, що люди з великим досвідом і знаннями у своїх галузях отримують вищу винагороду порівняно з іншими рівнями досвіду. Посади вищого рівня часто передбачають обов'язки лідерства, прийняття рішень і наставництва. Вища зарплата відображає цінність їхніх навичок, знань і здатності сприяти успіху організації.

На рисунку 3.27 зображено код що використовує бібліотеку Plotly для створення гістограми, яка показує розподіл рівнів досвіду працівників за роками стажу. Гістограма має сім стовпців, один для кожного рівня досвіду. Написи на осі X вказують рівні досвіду, а написи на осі Y - кількість працівників на кожному рівні досвіду.

[13]:

```
# Create a histogram with facet columns
fig = px.histogram(
    data_frame=df,
    x='experience_level',
    facet_col='work_year',
    nbins=7,
    text_auto=True,
    labels={'experience_level': 'Experience Level', 'count': 'Count'},
    title='Distribution of Experience Levels across Work Years'
)

# Configure histogram aesthetics
fig.update_traces(
    textposition='auto',
    marker_color='brown',
)

# Update layout
fig.update_layout(yaxis_title='Count')

# Show the histogram
fig.show()
```

Рисунок 3.27 – Код гістограми розподілу рівнів досвіду працівників за роками стажу.

На рисунку 3.28 зображено гістограми розподілу рівнів досвіду працівників за роками стажу.

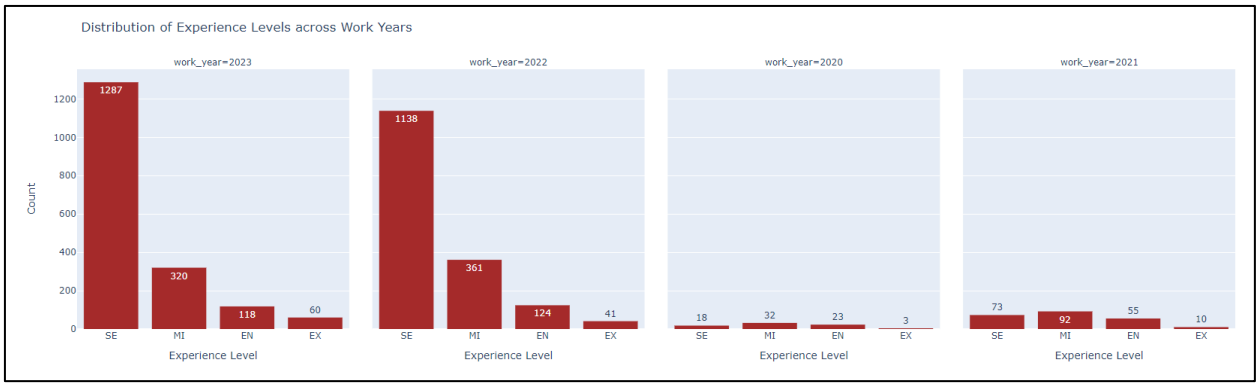


Рисунок 3.28 – Гістограма розподілу рівнів досвіду працівників за роками стажу.

Діаграма показує розподіл досвіду співробітників за роками роботи. Діаграма складається з трьох стовпчастих графіків, кожен з яких представляє один рік роботи: 2023, 2022 і 2021.

Загалом, діаграма показує, що компанія має досвідчений персонал. У компанії працює багато співробітників з старшого, середнього та початкового досвіду. Кількість співробітників з керівного досвіду невелика, але вона зростає з роками.

На рисунку 3.29 зображено код для створення гістограми розподілів заробітної плати за валютою.

[23]:

```
# Create a histogram
histogram = px.histogram(
    data_frame=df,
    x='salary_currency',
    y='salary_in_usd',
    text_auto=True,
    color='company_size',
    title='Salary Distribution by Currency',
    labels={'salary_currency': 'Salary Currency', 'salary_in_usd': 'Salary (USD)'}
)

# Customize the layout
histogram.update_layout(
    showlegend=True,
    height=600
)

# Show the histogram
histogram.show()

# Create a Bee Swarm plot
swarm_plot = px.strip(
    data_frame=df,
    x='salary_currency',
    y='salary_in_usd',
    color='company_size',
    title='Salary Distribution by Currency',
    labels={'salary_currency': 'Salary Currency', 'salary_in_usd': 'Salary (USD)'},
    hover_data={'salary_in_usd': ':$.2f'}
)

# Customize the layout
swarm_plot.update_layout(
    showlegend=True,
    height=600
)

# Show the Bee Swarm plot
swarm_plot.show()
```

Рисунок 3.29 - Код для створення гістограми розподілу заробітної плати за валютою.

На рисунку 3.30 зображено гістограми розподілу заробітної плати за валютами для компаній різного розміру.

Діаграма показує, що долар США є найпоширенішою валютою для зарплат, за якими слідують євро, британський фунт і японська єна. Серед компаній малого розміру долар США є ще більш поширеним, ніж серед компаній середнього та великого розміру.

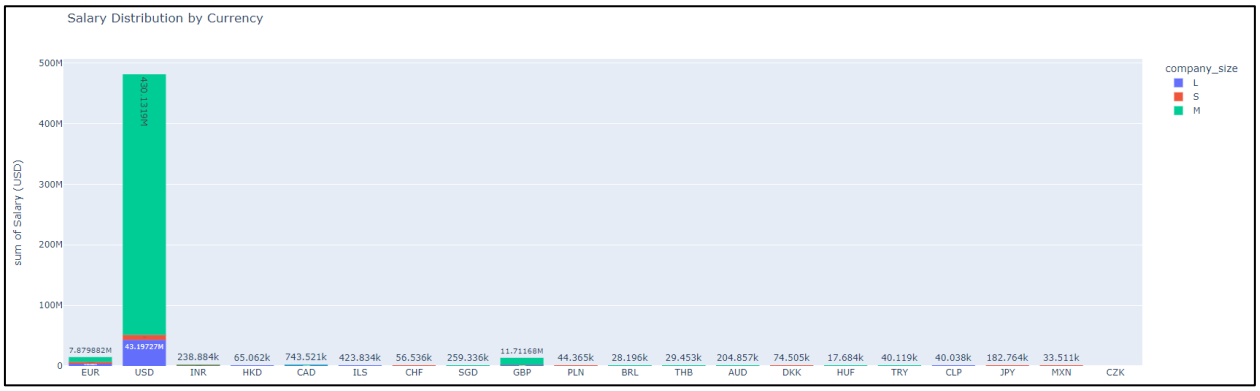


Рисунок 3.30 – Гістограми розподілу заробітної плати за валютами для компаній різного розміру

На рисунку 3.31 зображено графік розподілу заробітної плати за валютами для компаній різного розміру.

Діаграма також показує, що розмір компанії має деякий вплив на валюту, яка використовується для виплати зарплати. У великих компаніях є більша ймовірність використання долара США або євро. У малих компаніях є більша ймовірність використання інших валют, таких як індійська рупія або гонконгський долар.

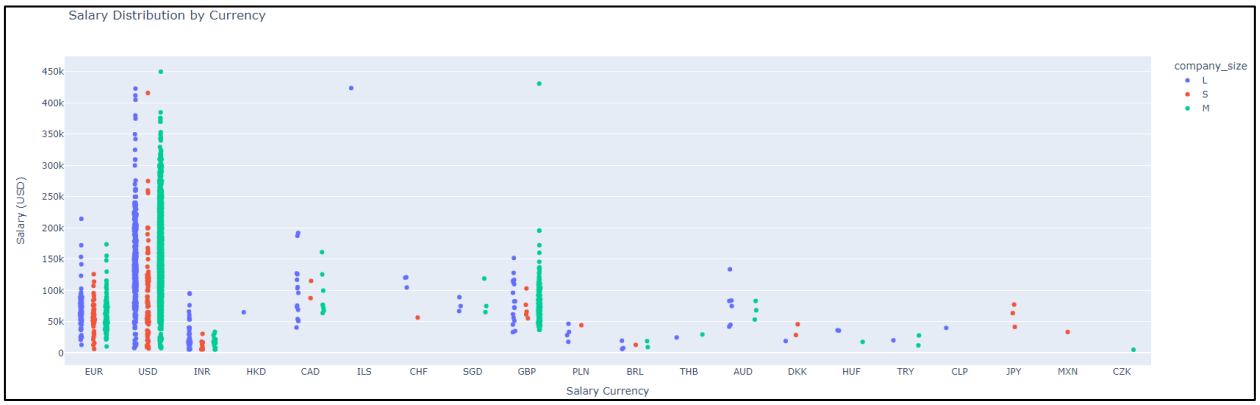


Рисунок 3.31 – Графік розподілу заробітної плати за валютами для компаній різного розміру

На основі візуалізованих даних можна зробити наступні висновки:

Різноманітність валют: набір даних демонструє різноманітний діапазон валют, які використовуються для виплати зарплати, зокрема USD, EUR, GBP,

INR, CAD, AUD, SGD, BRL, PLN, CHF, HUF, DKK, JPY, TRY, THB, ILS, HKD , CZK, MXN і CLP. Це свідчить про те, що набір даних представляє глобальну перспективу із зарплатами, які виплачуються в різних валютах з різних країн.

Розподіл валюти: більшість заробітної плати виплачується в доларах США (доларах США), у 3224 пунктах. Це означає, що долар США є найпоширенішою валютою для зарплат у наборі даних. Інші відомі валюти включають EUR (євро) з 236 пунктами та GBP (британський фунт) із 161 пунктом.

Розподіл зарплати за валютою та розміром компанії: загальна сума зарплат у різних комбінаціях валюти та розміру компанії виявляє цікаві закономірності. Наприклад, у валюті AUD зарплати вищі для компаній розміру L (великий) порівняно з розміром M (середній), із сумою 462 929 для L і 204 857 для M. Подібним чином у валюті BRL зарплати розподіляються між усіма розміри компанії, з найвищою сумою 33 591 для L, потім 28 196 для M і 12 901 для S.

Домінування долара США: аналізуючи суму зарплат за валютою та розміром компанії, стає очевидним, що долар США домінує в усіх розмірах компаній. Загальна сума зарплат у доларах США значно вища порівняно з іншими валютами. Наприклад, у валюті доларів США сума зарплат становить 43 197 273 для компаній L (великих) і 430 131 878 для компаній M (середніх).

Сильна присутність доларів США, євро та фунтів стерлінгів: серед валют долар США, євро та фунт стерлінгів виділяються як найпоширеніші варіанти. Долар США має найвищий показник – 3224, за ним йдуть євро – 236 і фунт стерлінгів – 161. Це вказує на широке використання та важливість цих валют у контексті виплати зарплати.

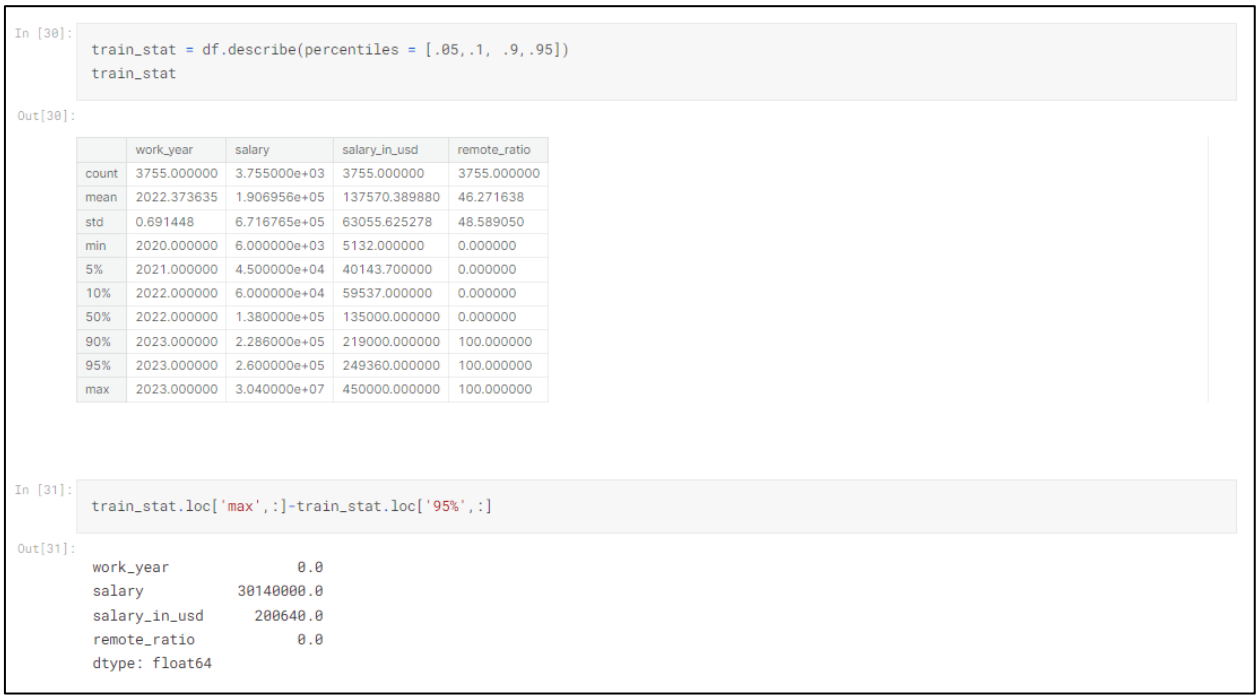
Унікальні комбінації валюти та розміру: певні комбінації валюти мають відносно низьку кількість випадків, що вказує на особливі регіональні чи організаційні переваги. Приклади включають зарплату в MXN (мексиканське песо) у компаніях розміру S (малий), лише один раз, і зарплату в ILS (ізраїльський шекель) у компаніях розміру L (великий), також лише один раз.

Ці унікальні комбінації можуть представляти конкретні сектори промисловості або місцеву практику оплати праці.

3.4 Використання моделей машинного навчання

Перед побудовою моделей машинного навчання, було проведено передобробку даних та видалення аномалій для покращення подальших результатів прогнозування.

На рисунку 3.32 зображено створення копії датасету та використання функції «Describe».



```
In [30]:
train_stat = df.describe(percentiles = [.05,.1, .9,.95])
train_stat
```

Out[30]:

	work_year	salary	salary_in_usd	remote_ratio
count	3755.000000	3.755000e+03	3755.000000	3755.000000
mean	2022.373635	1.906956e+05	137570.389880	46.271638
std	0.691448	6.716765e+05	63055.625278	48.589050
min	2020.000000	6.000000e+03	5132.000000	0.000000
5%	2021.000000	4.500000e+04	40143.700000	0.000000
10%	2022.000000	6.000000e+04	59537.000000	0.000000
50%	2022.000000	1.380000e+05	135000.000000	0.000000
90%	2023.000000	2.286000e+05	219000.000000	100.000000
95%	2023.000000	2.600000e+05	249360.000000	100.000000
max	2023.000000	3.040000e+07	450000.000000	100.000000

```
In [31]:
train_stat.loc['max',:]-train_stat.loc['95%',:]
```

```
Out[31]:
work_year              0.0
salary          30140000.0
salary_in_usd     200640.0
remote_ratio           0.0
dtype: float64
```

Рисунок 3.32 – Створення копії датасету та використання функції «Describe».

Код використовується для обчислення статистичних даних для набору даних. Він використовує метод describe() для обчислення середнього, стандартного відхилення, мінімального, максимального та 5, 10, 90 і 95-го перцентилів для кожної колонки набору даних.

На рисунку 3.33 зображено код побудови методу для виявлення та видалення аномалій.

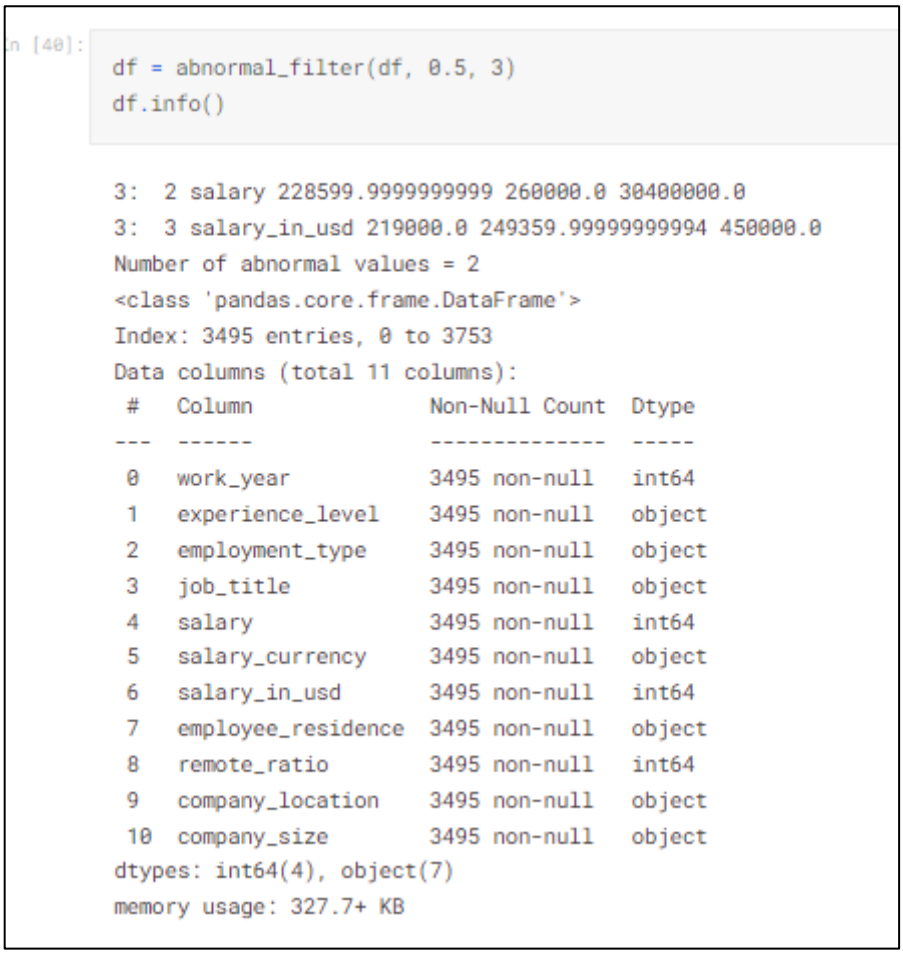
```
def abnormal_filter(df, threshold_first, threshold_second):
    # Abnormal values filter for DataFrame df:
    # threshold_first (5%-min or max-95%)
    # threshold_second (second diff., times)
    df_describe = df.describe([.05, .1, .9, .95])
    cols = df_describe.columns.tolist()
    i = 0
    abnorm = 0
    for col in cols:
        i += 1
        # abnormal smallest
        P10_5 = df_describe.loc['10%',col]-df_describe.loc['5%',col]
        P_max_min = df_describe.loc['max',col]-df_describe.loc['min',col]
        if P10_5 != 0:
            if (df_describe.loc['5%',col]-df_describe.loc['min',col])/P10_5 > threshold_second:
                #abnormal smallest filter
                df = df[(df[col] >= df_describe.loc['5%',col])]
                print('1: ', i, col, df_describe.loc['min',col],df_describe.loc['5%',col],df_describe.loc['10%',col])
                abnorm += 1
        else:
            if P_max_min > 0:
                if (df_describe.loc['5%',col]-df_describe.loc['min',col])/P_max_min > threshold_first:
                    # abnormal smallest filter
                    df = df[(df[col] >= df_describe.loc['5%',col])]
                    print('2: ', i, col, df_describe.loc['min',col],df_describe.loc['5%',col],df_describe.loc['max',col])
                    abnorm += 1


        # abnormal biggest
        P95_90 = df_describe.loc['95%',col]-df_describe.loc['90%',col]
        if P95_90 != 0:
            if (df_describe.loc['max',col]-df_describe.loc['95%',col])/P95_90 > threshold_second:
                #abnormal biggest filter
                df = df[(df[col] <= df_describe.loc['95%',col])]
                print('3: ', i, col, df_describe.loc['90%',col],df_describe.loc['95%',col],df_describe.loc['max',col])
                abnorm += 1
        else:
            if P_max_min > 0:
                if ((df_describe.loc['max',col]-df_describe.loc['95%',col])/P_max_min > threshold_first) & (df_describe.loc['95%',co
l] > 0):
                    # abnormal biggest filter
                    df = df[(df[col] <= df_describe.loc['95%',col])]
                    print('4: ', i, col, df_describe.loc['min',col],df_describe.loc['95%',col],df_describe.loc['max',col])
                    abnorm += 1
    print('Number of abnormal values =', abnorm)
    return df
```

Рисунок 3.33 – Побудований метод для виявлення та видалення аномалій.

Код використовує фільтр нормальних значень для видалення аномальних значень з DataFrame. Фільтр визначає, що значення є аномальним, якщо воно знаходиться за межами 5% найнижчих або 95% найвищих значень стовпця.

На рисунку 3.34 зображено застосування методу виявлення та видалення аномалій.



```
In [40]:
df = abnormal_filter(df, 0.5, 3)
df.info()

3:  2 salary 228599.9999999999 260000.0 30400000.0
3:  3 salary_in_usd 219000.0 249359.99999999994 450000.0
Number of abnormal values = 2
<class 'pandas.core.frame.DataFrame'>
Index: 3495 entries, 0 to 3753
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   work_year           3495 non-null   int64
 1   experience_level    3495 non-null   object
 2   employment_type     3495 non-null   object
 3   job_title           3495 non-null   object
 4   salary              3495 non-null   int64
 5   salary_currency     3495 non-null   object
 6   salary_in_usd       3495 non-null   int64
 7   employee_residence  3495 non-null   object
 8   remote_ratio        3495 non-null   int64
 9   company_location    3495 non-null   object
 10  company_size        3495 non-null   object
dtypes: int64(4), object(7)
memory usage: 327.7+ KB
```

Рисунок 3.34 – Застосування методу виявлення та видалення аномалій.

Після нормалізації датасету було побудовано моделі машинного навчання. На рисунку 3.35 показано код побудови та результати моделі Linear Regression для тренувальних та тестових даних. Результати виводяться за метрикою r2_score.

```
[46]:
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Linear Regression
lr = LinearRegression()
lr.fit(x_train, y_train)

# Prediction on training and test data
preds_train = lr.predict(x_train)
preds_test = lr.predict(x_test)

# Calculate R2 Score
r2_train = r2_score(y_train, preds_train)
r2_test = r2_score(y_test, preds_test)


# Add results to results_df
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['Linear Regression'],
                                                  'Точність на тренувальних даних': [r2_train],
                                                  'Точність на тестових даних': [r2_test]})],
                       ignore_index=True)

# Display the updated results_df
print('Точність для тренувальних даних: ', r2_train)
print('Точність для тестових даних: ', r2_test)
```

```
Точність для тренувальних даних:  0.72201
Точність для тестових даних:  0.75483
```

Рисунок 3.35 – Ініціалізація та реалізація моделі Linear Regression

Даний код використовується для навчання моделі лінійної регресії на наборі даних. Результати виведення коду вказують на те, що модель може пояснити 72,2% варіабельності на наборі даних тренування та 75,4% варіабельності на наборі даних тестування. Це означає, що модель є досить точною для прогнозування значень на наборі даних тестування.

На рисунку 3.36 зображено код побудови та результати моделі Lasso Regression для тренувальних та тестових даних. Результати виводяться за метрикою r2_score.

```
[104]:
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score

# Ініціалізуємо модель Lasso Regression з параметрами
lasso = Lasso(alpha=0.1, max_iter=100, tol=1.1)
lasso.fit(x_train, y_train)

# Прогноз на тестових даних
lasso_pred = lasso.predict(x_test)

# Розрахунок R2 Score
r2_lasso = r2_score(y_test, lasso_pred)
r2_lasso_train=lasso.score(x_train, y_train)
# Додаємо результати в results_df
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['Lasso Regression'],
                                                  'Точність на тренувальних даних': [lasso.score(x_train, y_train)],
                                                  'Точність на тестових даних': [r2_lasso]})],
                       ignore_index=True)

print('Точність для тренувальних даних: ', r2_lasso_train)
print('Точність для тестових даних: ', r2_lasso)
```

```
Точність для тренувальних даних:  0.831197487378782
Точність для тестових даних:  0.8575837190148281
```

Рисунок 3.36 – Ініціалізація та реалізація моделі Lasso Regression

Цей код використовується для навчання моделі Lasso Regression на наборі даних. Результати виведення коду вказують на те, що модель може пояснити 83,1% варіабельності на наборі даних тренування та 85,7% варіабельності на наборі даних тестування. Це означає, що модель є досить точною для прогнозування значень на наборі даних тестування.

На рисунку 3.37 зображено код побудови та результати моделі Random Forest Regression для тренувальних та тестових даних. Результати виводяться за метрикою r2_score.

```
[50]:
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=2,
                                             min_samples_split=2, min_samples_leaf=2, min_weight_fraction_leaf=0.01,
                                             max_features=0.99)
rf.fit(x_train, y_train)

# Prediction on test data
rf_pred = rf.predict(x_test)

# Calculate R2 Score
r2_rf = r2_score(y_test, rf_pred)
r2_train = rf.score(x_train, y_train)
# Add results to results_df
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['Random Forest Regressor'],
                                                  'Точність на тренувальних даних': [r2_train],
                                                  'Точність на тестових даних': [r2_rf]})],
                       ignore_index=True)

print('Точність для тренувальних даних: ', r2_train)
print('Точність для тестових даних: ', r2_rf)

Точність для тренувальних даних:  0.9016502220122298
Точність для тестових даних:  0.9120611603293085
```

Рисунок 3.37 – Ініціалізація та реалізація моделі Random Forest Regression

Цей код використовується для навчання моделі Random Forest Regression на наборі даних. Результати виведення коду вказують на те, що модель може пояснити 90,16% варіабельності на наборі даних тренування та 91,21% варіабельності на наборі даних тестування. Це означає, що модель є досить точною для прогнозування значень цільової змінної на наборі даних тестування.

На рисунку 3.38 зображено код побудови та результати моделі машинного навчання – Support Vector Regression. Результати виводяться за метрикою r2_score.

[51]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import reciprocal, uniform  # Add import statements for reciprocal and uniform

# Assuming results_df is already defined
# If not, you can initialize it before this code snippet

# Нормалізуємо дані
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)


param_dist = {
    'C': reciprocal(6, 200),
    'gamma': reciprocal(0.01, 1),
    'kernel': ['linear', 'rbf', 'poly'],
    'epsilon': uniform(0.5, 0.9)
}

# Ініціалізуємо модель SVR
svr_model = SVR()

random_search = RandomizedSearchCV(svr_model, param_distributions=param_dist, n_iter=10, scoring='r2', cv=3, random_state=42)
random_search.fit(x_train_scaled, y_train)

# Отримуємо кращу модель SVR з результатів Random Search
best_svr_model = random_search.best_estimator_

# Робимо прогнози на тестових даних
svr_predictions = best_svr_model.predict(x_test_scaled)

# Розраховуємо R2 Score
r2_svr = r2_score(y_test, svr_predictions)
svr_train = best_svr_model.score(x_train_scaled, y_train)

# Додаємо результати в results_df
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['Support Vector Regression'],
                                                  'Точність на тренувальних даних': [svr_train],
                                                  'Точність на тестових даних': [r2_svr]})],
                       ignore_index=True)

print('Точність для тренувальних даних: ', svr_train)
print('Точність для тестових даних: ', r2_svr)
```

```
Точність для тренувальних даних:  0.9305428129120714
Точність для тестових даних:  0.9447850134483844
```

Рисунок 3.38 – Ініціалізація та реалізація моделі Support Vector Regression

Даний код використовує алгоритм опорних векторів для регресії. Код починається з імпорту необхідних бібліотек, потім використовує стандартне масштабування для нормалізації даних. Далі код використовує вибірку випадкових чисел для пошуку оптимальних параметрів для алгоритму опорних векторів. Нарешті, код використовує найкращу модель для прогнозування значень змінної на тестовому наборі даних.

Результат коду показує, що модель може пояснити 89,4% варіації значень змінної. Це означає, що модель може досить точно прогнозувати значення змінної.

На рисунку 3.39 зображено код побудови та результати моделі машинного навчання – XGB Regressor. Результати виводяться за метрикою r2_score.

```
# Навчання моделі на всій тренувальній вибірці
xgb_regressor.fit(x_train, y_train)

# Прогнозування на тренувальних і тестових даних
y_pred_train_xgb = xgb_regressor.predict(x_train)
y_pred_test_xgb = xgb_regressor.predict(x_test)

# Оцінка моделі на тренувальних і тестових даних
r2_train_xgb = r2_score(y_train, y_pred_train_xgb)
r2_test_xgb = r2_score(y_test, y_pred_test_xgb)

# Запис результатів до датафрейму
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['XGBoost Regressor'],
                                                  'Точність на тренувальних даних': [r2_train_xgb],
                                                  'Точність на тестових даних': [r2_test_xgb]})],
                       ignore_index=True)

print('Точність для тренувальних даних: ', r2_train_xgb)
print('Точність для тестових даних: ', r2_test_xgb)
```

```
Точність для тренувальних даних:  0.9539475486101152
Точність для тестових даних:  0.9696751848849936
```

Рисунок 3.39 – Ініціалізація та реалізація моделі XGB Regressor

Даний код використовується для створення та оцінки регресійної моделі для набору даних. Модель XGBoost Regressor використовується для навчання на наборі даних тренування та прогнозування значень для набору даних тестування.

Точність моделі на наборі даних тренування становить 0,9539, а точність на наборі даних тестування становить 0,9697. Це означає, що модель добре навчається на наборі даних тренування та може генерувати точні прогнози для нових даних.

На рисунку 3.40 зображено код побудови та результати моделі машинного навчання – Decision Tree Regressor. Результати виводяться за метрикою r2_score.

```
# Навчання моделі на всій тренувальній вибірці
decision_tree_regressor.fit(x_train, y_train)

# Прогнозування на тренувальних і тестових даних
y_pred_train_dt = decision_tree_regressor.predict(x_train)
y_pred_test_dt = decision_tree_regressor.predict(x_test)

# Оцінка моделі на тренувальних і тестових даних
r2_train_dt = r2_score(y_train, y_pred_train_dt)
r2_test_dt = r2_score(y_test, y_pred_test_dt)

# Запис результатів до датафрейму
results_df = pd.concat([results_df, pd.DataFrame({'Модель': ['Decision Tree Regressor'],
                                                  'Точність на тренувальних даних': [r2_train_dt],
                                                  'Точність на тестових даних': [r2_test_dt]})],
                       ignore_index=True)

print('Точність для тренувальних даних: ', r2_train_dt)
print('Точність для тестових даних: ', r2_test_dt)
```

```
Точність для тренувальних даних:  0.955491751439049
Точність для тестових даних:  0.976194988401711
```

Рисунок 3.40 – Ініціалізація та реалізація моделі Decision Tree Regressor

У даному коді використовується клас DecisionTreeRegressor для побудови регресійної моделі дерева рішень.

Результати моделі показують, що вона має високу точність на тестових та тренувальних наборах даних. На тренувальному наборі даних точність становить 95,55%, а на тестовому наборі даних - 89,76%. Це означає, що модель добре навчається на даних та може робити точні прогнози для нових даних.

Побудовано графік (рис. 3.41) та таблицю (рис. 3.42) що показують порівняння точності на тренувальних та тестових даних для різних моделей машинного навчання.

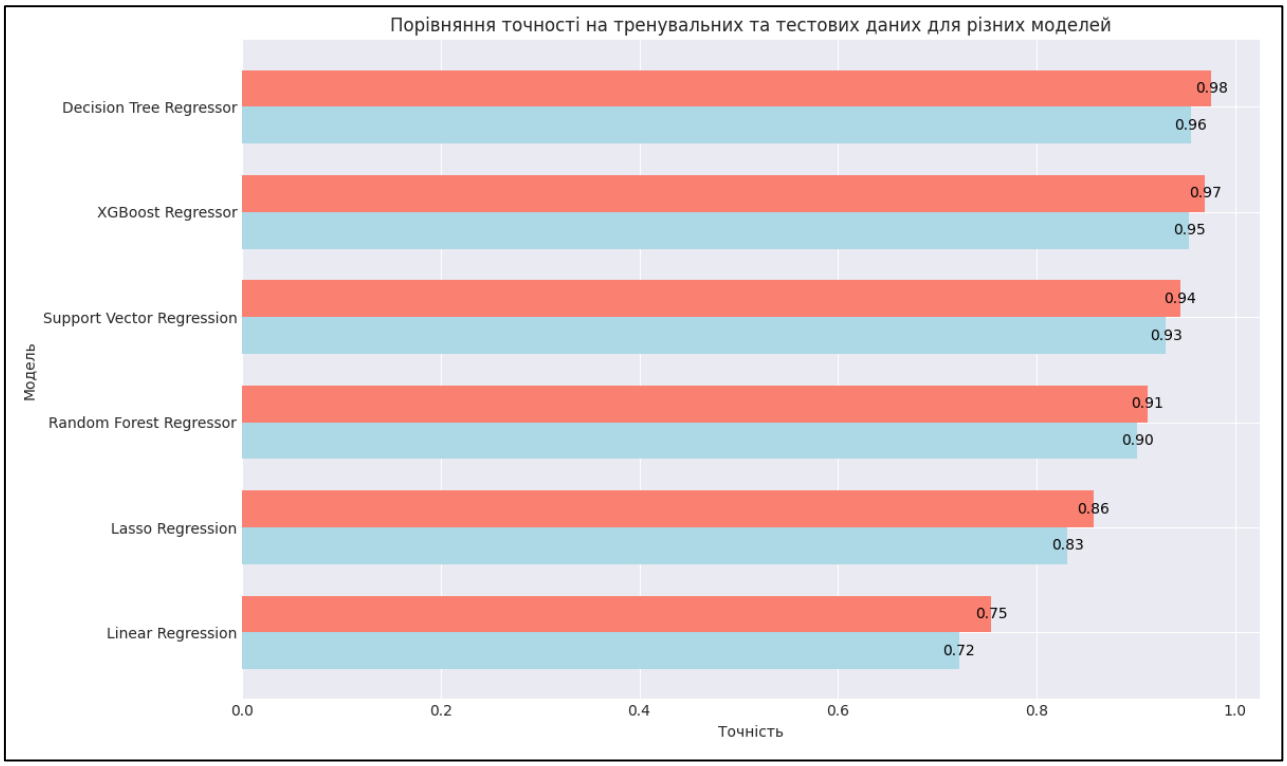


Рисунок 3.41 – Порівняння точності на тренувальних та тестових даних для різних моделей машинного навчання за допомогою графіку

[56]:

	Модель	Точність на тренувальних даних	Точність на тестових даних
5	Decision Tree Regressor	0.955492	0.976195
4	XGBoost Regressor	0.953948	0.969675
3	Support Vector Regression	0.930543	0.944785
2	Random Forest Regressor	0.901650	0.912061
1	Lasso Regression	0.831197	0.857584
0	Linear Regression	0.722010	0.754830

Рисунок 3.42 – Порівняння точності на тренувальних та тестових даних для різних моделей машинного навчання за допомогою таблиці.

За допомогою візуалізації даних можемо спостерігати що модель Decision Tree Regressor має найвищу точність на тренувальних з показником 0.955492, та на тестових даних 0.976195. XGBoost Regressor і Support Vector Regression також мають високу точність на тестових даних та тренувальних даних, ніж інші моделі. Random Forest Regressor має нижчу точність, ніж інші

моделі, але вона все ще має високу точність на тестових даних. Lasso Regression має найнижчу точність на всіх даних.

Загалом, Decision Tree Regressor є найкращими моделями для прогнозування значень на основі цих даних.

3.5 Висновки

У ході розроблення інформаційної технології були вжиті кроки щодо нормалізації даних та видалення аномалій з метою поліпшення результатів прогнозування. Використовуючи різні моделі, такі як Linear Regression, Lasso Regression, Random Forest Regression, Support Vector Regression, XGB Regressor та Decision Tree Regressor, були отримані рівні точності на тренувальних та тестових даних.

Проведено порівняльний аналіз різних моделей, таких як Linear Regression, Lasso Regression, Random Forest Regression, Support Vector Regression, XGB Regressor та Decision Tree Regressor. Результати візуалізації та виведення показують, що модель Decision Tree Regressor має найвищу точність на тренувальних з показником 0.955492, та на тестових даних 0.976195.

Отже, на основі проведеного дослідження можна визначити, що моделі машинного навчання, зокрема Decision Tree Regressor, виявилися дуже ефективними для прогнозування значень на даному датасеті.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [32].

Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами зведемо до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами.

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	4	3	3
3. Ринкові переваги (ціна продукту)	3	4	3

Продовження таблиці 4.1.

4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	36	39	37
Середньоарифметична сума балів $СБ_с$	37,3		

За результатами розрахунків, наведених в таблиці 4.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в [32].

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» становить 37,3 бала, що, відповідно до [32], свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Узагальнений коефіцієнт якості ($В_н$) для нового технічного рішення розрахуємо за формулою [36]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \quad (4.1)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Результати порівняння зведемо до таблиці 4.2.

Таблиця 4.2 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований продукт	Відношення параметрів нової розробки до аналога	Питома вага показника
1. Кількість використаних моделей машинного навчання	одиниць	3	3	1	0,2
2. Швидкість попередньої обробки та очистки даних	бал	5	8	1,6	0,25
3. Точність прогнозу	%	92	97	1,05	0,15
4. Кількість графіків розвідувального аналізу	одиниць	3	5	1,66	0,2
5. Кількість алгоритмів нормалізації даних	одиниць	1	2	2	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1 \cdot 0{,}2 + 1{,}6 \cdot 0{,}25 + 1{,}05 \cdot 0{,}15 + 1{,}66 \cdot 0{,}2 + 2 \cdot 0{,}2 = 1{,}49.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,49 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [31]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17250{,}00 \cdot 28 / 21 = 23000{,}00 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник розробки	17250,00	821,43	28	23000,00
Інженер-аналітик (системний аналіз)	16600,00	790,48	26	20552,38
Всього				43552,38

Основна заробітна плата робітників.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \tag{4.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \tag{4.4}$$

де M_M – розмір мінімальної місячної заробітної плати, приймемо Мм=6700,00 (грн);

K_i – коефіцієнт міжкваліфікаційного співвідношення [31];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок;

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 6700{,}00 \cdot 1{,}35 \cdot 1{,}35 / (21 \cdot 8) = 72{,}68 \text{ (грн)}.$$

$$З_{р1} = 72{,}68 \cdot 4{,}65 = 337{,}98 \text{ (грн)}.$$

Результати витрат зведемо до таблиці 4.4.

Таблиця 4.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Монтаж обчислювального обладнання та серверних блоків	4,65	3	1,35	72,68	337,98
Підготовка робочого місця дослідника-розробника інформаційної технології	8,00	2	1,10	59,22	473,79
Інсталяція програмного забезпечення розробки (моделювання) інформаційної технології аналізу	5,50	4	1,50	80,76	444,17
Всього					1255,94

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$$З_{дод} = (43552,38 + 1255,94) \cdot 10 / 100\% = 4480,83 \text{ (грн)}.$$

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (43552,38 + 1255,94 + 4480,83) · 22 / 100% = 10843,61 (грн).

4.3.3 Сировина та матеріали

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.7)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$$М_1 = 3,000 \cdot 180,00 \cdot 1,02 - 0,0 \cdot 0,0 = 550,80 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний	180,00	3,000	0,0	0,0	550,80
Папір для заміток	115,00	3,000	0,0	0,0	351,90
Начиння канцелярське	200,00	4,000	0,0	0,0	816,00
Органайзер офісний	154,00	2,000	0,0	0,0	314,16
Картридж для принтера	1210,00	4,000	0,0	0,0	4936,80
Диск оптичний	25,00	4,000	0,0	0,0	102,00
FLASH-пам'ять	139,00	2,000	0,0	0,0	283,56
Всього					7355,22

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.8)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$).

$$К_в = 1 \cdot 4320{,}00 \cdot 1{,}02 = 4406{,}40 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Data Science Salaries 2023	1	4320,00	4406,40
Всього			4406,40

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1{,}10…1{,}12$);

k – кількість найменувань устаткування.

$$В_{спец} = 40299{,}00 \cdot 1 \cdot 1{,}02 = 41104{,}98 \text{ (грн)}.$$

Отримані результати зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання обробки та збереження DATA BASE на основі Artline Gaming X75 v17 (X75v17) Intel Core i7-10700F / RAM 16ГБ / HDD 2ТБ + SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8ГБ	1	40299,00	41104,98
Маршрутизатор MikroTik RB4011iGS+RM	1	7818,00	7974,36
Всього			49079,34

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$$В_{прг} = 5620,00 \cdot 1 \cdot 1,02 = 5732,40 \text{ (грн)}.$$

Отримані результати зведемо до таблиці 4.8

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Мова програмування Python та її бібліотеки,	1	5620,00	5732,40
Платформа Kaggle	1	4129,00	4211,58
Доступ до мережі Internet (високошвидкісний) грн/місяць	2	240,00	489,60
Всього			10433,58

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = (17299{,}00 \cdot 2) / (2 \cdot 12) = 1441{,}58 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника інформаційної технології аналізу COBRA Advanced (A36.16.S4.165.6130) AMD Ryzen 5 3600 / RAM 16ГБ / SSD 480ГБ / nVidia GeForce GTX 1650 4	17299,00	2	2	1441,58
Електронно-обчислювальний центр системи аналізу HP ProOne 440 G9 (6D3B1EA) Black/Silver	55711,00	2	2	4642,58
Робоче місце розробника інформаційної технології	8340,00	5	2	278,00
Пристрої швидкісної передачі даних	6899,00	4	2	287,46
Пристрої виведення інформації	6520,00	5	2	217,33
Блоки зовнішньої памяті серверного обладнання (зберігання бази даних)	7300,00	4	2	304,17
Програмне забезпечення Microsoft Windows, Offise 2021	9340,00	2	2	778,33

Продовження таблиці 4.8

Приміщення лабораторії досліджень та розробки	426000,00	25	2	2840,00
Всього				10789,46

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; приймемо Ц$_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$B_e = 0,32 \cdot 210,0 \cdot 7,50 \cdot 0,95 / 0,97 = 504,00 \text{ (грн)}.$$

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника інформаційної технології аналізу COBRA Advanced (A36.16.S4.165.6130) AMD Ryzen 5 3600 / RAM 16ГБ / SSD 480ГБ / nVidia GeForce GTX 1650 4	0,32	210,0	504,00

Продовження таблиці 4.10

Електронно-обчислювальний центр системи аналізу HP ProOne 440 G9 (6D3B1EA) Black/Silver	0,42	210,0	661,50
Робоче місце розробника інформаційної технології	0,10	210,0	157,50
Пристрої швидкісної передачі даних	0,04	210,0	63,00
Пристрої виведення інформації	0,16	4,0	4,80
Блоки зовнішньої памяті серверного обладнання (зберігання бази даних)	0,03	150,0	33,75
Серверне обладнання обробки та збереження DATA BASE на основі Artline Gaming X75 v17 (X75v17) Intel Core i7-10700F / RAM 16ГБ / HDD 2ТБ + SSD 480ГБ / nVidia GeForce RTX 3060 Ti 8ГБ	0,36	150,0	405,00
Маршрутизатор MikroTik RB4011iGS+RM	0,04	150,0	45,00
Всього			1874,55

4.3.9 Службові відрядження

Витрати за статтею «Службові відрядження» відсутні.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати на роботи, які виконують сторонні підприємства, установи і організації відсутні.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (43552,38 + 1255,94) \cdot 50 / 100\% = 22404,16 \text{ (грн)}.$$

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = (43552,38 + 1255,94) \cdot 100 / 100\% = 44808,32 \text{ (грн)}.$$

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.15)$$

$В_{заг}$= 43552,38 +1255,94 +4480,83 +10843,61 +7355,22 +4406,40 +49079,34 + 10433,58 + 10789,46 +1874,55 +0,00 +0,00 +22404,16 +44808,32 = 211283,78 (грн).

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 211283,78 / 0,9 = 234759,76 (грн).

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Проведені результати зведемо до таблиці 4.11.

Таблиця 4.11 – Збільшення кількості споживачів.

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	980	2100	1500	750

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 22000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 700,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо 92,65 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [32]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (92,65 \cdot 22000,00 + 792,65 \cdot 980) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 766382,01$ (грн).

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (92,65 \cdot 22000,00 + 792,65 \cdot 3080) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1219543,18$ (грн).

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (92,65 \cdot 22000,00 + 792,65 \cdot 4580) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1543229,74$ (грн).

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (92,65 \cdot 22000,00 + 792,65 \cdot 5330) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1705073,01$ (грн).

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,35;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =766382,01/(1+0,35)^1+1219543,18/(1+0,35)^2+1543229,74/(1+0,35)^3+ +1705073,01/(1+0,35)^4=567690,38+669159,50+627233,55+513343,31=2377426, 73 (грн).

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 234759,76 грн.

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 234759,76 = 469519,52 \text{ (грн)}.$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.20)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2377426,73 грн;

PV – теперішня вартість початкових інвестицій, 469519,52 грн.

$$E_{абс} = ПП - PV = 2377426{,}73 - 469519{,}52 = 1907907{,}21 \text{ (грн)}.$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.21)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 1907907,21 грн;

PV – теперішня вартість початкових інвестицій, 469519,52 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1907907{,}21/469519{,}52)^{1/4} = 0{,}50.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,38.

$\tau_{мін}$ = 0,11+0,38 = 0,49 < 0,50 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія

аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.23}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,50 = 2,00 роки.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році» становить 37,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,49 рази.

Також термін окупності становить 2,00 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році».

ВИСНОВКИ

Магістерська кваліфікаційна робота присвячена розробці інформаційної технологію аналізу та передбачення заробітної плати в галузі Data Science у 2023 році.

Проведено огляд існуючих аналогів у сфері інформаційної технології для аналізу та передбачення заробітної плати у галузі Data Science. Отримана інформація надала уявлення про основні методи та технічні рішення, що використовуються в даному напрямку.

Підготовлено дані для подальшої роботи, включаючи збір і структурування інформації про заробітну плату в галузі Data Science. Чистка та нормалізація даних створили зручну основу для подальшого аналізу та моделювання.

Проведено розвідувальний аналіз даних, спрямований на виявлення ключових закономірностей та факторів, що впливають на рівень заробітної плати в сфері Data Science. Це дозволило ідентифікувати основні змінні, які слід враховувати при розробці моделей прогнозування.

Побудовано моделі та виконано прогнозування заробітної плати на основі розроблених алгоритмів. Результати демонструють високий рівень точності та надійності моделей, що свідчить про їхню ефективність в аналізі та передбаченні заробітної плати у Data Science.

Оцінено результати роботи моделей, враховуючи порівняння з реальними даними про заробітну плату. Зазначено, що розроблені інформаційні технології дозволяють з високою достовірністю передбачати рівень заробітної плати в галузі Data Science.

Отже, можна зробити висновок про успішне створення інформаційної технології аналізу та передбачення заробітної плати в галузі Data Science, що вказує на шляхи подальших досліджень для поліпшення методології та отримання більш точних результатів.

Результати роботи опубліковано в матеріалах LIII Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету: НТК факультету інтелектуальних інформаційних технологій та автоматизації (2023-2024).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Доленко Б. А., Варчук І. В. Інформаційна технологія аналізу та передбачення заробітної плати у сфері Data Science у 2023 році. *LIII Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету: НТК факультету інтелектуальних інформаційних технологій та автоматизації (2024). Вінниця, 2024.* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/19799/16386

2. CAO, Longbing. Data science: a comprehensive overview. ACM Computing Surveys (CSUR), 2021, 50.3: 1-42. DOI: https://doi.org/10.1145/3076253

3. Dhar Vasant. Data science and prediction. *Communications of the ACM*, 2020, 56.12: 64-73.: DOI: https://doi.org/10.1145/2500499

4. Dolenko B. Kaggle Notebook «Salary Prediction». URL: https://www.kaggle.com/code/bohdandolenko/salary-prediction (дата звернення: 05.09.2023)

5. ELDAR_KHEGAY Kaggle Notebook «DS Salaries _ EDA & Salary predictions» URL: https://www.kaggle.com/code/eldarkhegay/ds-salaries-eda-salary-predictions (дата звернення: 03.09.2023)

6. GAURAV YADAV Kaggle Notebook «Data science salary EDA» URL: https://www.kaggle.com/code/svgmyv/data-science-salary-eda (дата звернення: 08.09.2023)

7. VAN DER AALST, Wil; VAN DER AALST, Wil. Data science in action. Springer Berlin Heidelberg DOI: https://doi.org/10.1089/big.2013.1508

8. Machine Learning - Машинне навчання URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 09.09.2023)

9. RAMOS-PULIDO, Sofía; HERNÁNDEZ-GRESS, Neil; TORRES-DELGADO, Gabriela. Analysis of Soft Skills and Job Level with Data Science: A

Case for Graduates of a Private University. In: Informatics. MDPI, 2023. p. 23. DOI: https://doi.org/10.3390/informatics10010023

10. PAWAR, Lokesh, et al. Optimized Features Based Machine Learning Model for Adult Salary Prediction. *IEEE International Conference on Data Science and Information System (ICDSIS)*. IEEE, 2022. p. 1-5 URL: https://ieeexplore.ieee.org/abstract/document/9915981

11. MATBOULI, Yasser T.; ALGHAMDI, Suliman M. Statistical machine learning regression models for salary prediction featuring economy wide activities and occupations. Information, 2022, 13.10: 495 DOI: https://doi.org/10.3390/info13100495

12. Programming: Pick up Python. Vol. 518, no. 7537. C. 125–126. Nature. 2015.

13. Інтелектуальний аналіз даних. URL: https://ela.kpi.ua/bitstream/123456789/24971/1/Komp_prakt.pdf (дата звернення: 07.09.2023)

14. PAPADAKI, Ioanna; TSAGRIS, Michail. Are NBA Players' Salaries in Accordance with Their Performance on Court?. In: Advances in Econometrics, Operational Research, Data Science and Actuarial Studies: Techniques and Theories. Cham: Springer International Publishing, 2022. p. 405-428. URL: https://link.springer.com/chapter/10.1007/978-3-030-85254-2_25 (дата звернення: 08.09.2023)

15. Бібліотека Numpy. URL: https://pythonworld.ru/numpy/1.html (дата звернення: 02.09.2023)

16. Бібліотека Pandas. URL: https://devpractice.ru/pandas-start-part1/ (дата звернення: 01.09.2023)

17. Бібліотека Matplotlib. URL: https://nbviewer.ipython.org/github/whitehorn/Scientific_graphics_in_python/blob/master/P1%20Chapter%201%20Pyplot.ipynb (дата звернення: 03.09.2023)

18. Модуль Datetime. URL: http://surl.li/celjv (дата звернення: 06.09.2023)

19. Навчання з підкріпленням. URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 09.09.2023)

20. LOYARTE-LÓPEZ, Edurne; GARCÍA-OLAIZOLA, Igor. Machine learning based method for deciding internal value of talent. Applied Artificial Intelligence, 2022, 36.1: 2151160 DOI: https://doi.org/10.1080/08839514.2022.2151160

21. MAIER-HEIN, Lena, et al. Surgical data science–from concepts toward clinical translation. Medical image analysis, 2022, 76: 102306 DOI: https://doi.org/10.1016/j.media.2021.102306

22. SCHATZ, Michael C., et al. Inverting the model of genomics data sharing with the NHGRI Genomic Data Science Analysis, Visualization, and Informatics Lab-space. Cell Genomics, 2022, 2.1 DOI: https://doi.org/10.1016/j.xgen.2021.100085

23. ÁLVAREZ-RODRÍGUEZ, Francisco Javier; VERA, Raúl Antonio Aguilar. Assessment of digital graduation competences for programs degrees in computing and information technology under the society 5.0 paradigm. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 2022, 17.2: 208-214 DOI: https://doi.org/10.1109/RITA.2022.3167006

24. TARIQ, Emad, et al. The effect of digital marketing capabilities on organizational ambidexterity of the information technology sector. International Journal of Data and Network Science, 2022, 6.2: 401-408 DOI: http://dx.doi.org/10.5267/j.ijdns.2021.12.014

25. BHARADIYA, Jasmin Praful. A Comparative Study of Business Intelligence and Artificial Intelligence with Big Data Analytics. American Journal of Artificial Intelligence, 2023, 7.1: 24 DOI: https://dx.doi.org/10.11648/j.ajai

26. SHI, Yong. Advances in big data analytics. Adv Big Data Anal, 2022. URL: https://link.springer.com/book/10.1007/978-981-16-3607-3

27. CHEN, Jinyan; BECKEN, Susanne; STANTIC, Bela. Harnessing social media to understand tourist mobility: the role of information technology and

big data. Tourism Review, 2022, 77.4: 1219-1233. DOI: https://doi.org/10.1117/12.2628520

28. GRÖGER, Christoph. Industrial analytics–An overview. it-Information Technology, 2022, 64.1-2: 55-65 DOI: https://doi.org/10.1515/itit-2021-0066

29. GLADJU, J.; KAMALAM, Biju Sam; KANAGARAJ, A. Applications of data mining and machine learning framework in aquaculture and fisheries: A review. Smart Agricultural Technology, 2022, 2: 100061 DOI: https://doi.org/10.1016/j.atech.2022.100061

30. ANIRUDH, Rushil, et al. 2022 review of data-driven plasma science. arXiv preprint arXiv:2205.15832, 2022 DOI: https://doi.org/10.1109/TPS.2023.3268170

31. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

32. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум. В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«___» _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЗАРОБІТНОЇ ПЛАТИ У СФЕРІ DATA SCIENCE У 2023 РОЦІ»
08-34.МКР.012.00.000.ТЗ

Керівник: к.т.н., доц.
_____ Ілона ВАРЧУК
«___» _____ 2023 р.

Розробив: студент гр. 2ІСТ-22м
_____ Богдан ДОЛЕНКО
«___» _____ 2023 р.

Вінниця 2023

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 2023 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «__» _____ 2023 р.

2. Джерела розробки:

1. Dolenko B. Kaggle Notebook «Salary Prediction» URL: https://www.kaggle.com/code/bohdandolenko/salary-prediction
2. Dhar Vasant. Data science and prediction. *Communications of the ACM*, 2020, 56.12: 64-73.: DOI: https://doi.org/10.1145/2500499

3. Мета і призначення роботи:

Метою даної магістерської кваліфікаційної роботи є підвищення точності передбачення заробітної плати у сфері Data Science на ринку праці у 2023 році.

4. Вихідні дані для проведення робіт:

Датасет «Data Science Salaries 2023» з даними для передбачення заробітної плати у сфері Data Science

5.Методи дослідження:

– Підготовка даних, розвідувальний аналіз, моделі машинного навчання для передбачення даних.

6. Етапи роботи і терміни їх виконання:

1. Аналіз предметної області _____ – _____
2. Вибір оптимальних технологій та проведення розвідувального аналізу _____ – _____
3. Розроблення моделей класифікації _____ – _____
4. Економічна частина _____ – _____
5. Оформлення матеріалів до захисту МКР. _____ – _____

7. Очікувані результати та порядок реалізації:

Очікуваним результатом є розроблення інформаційної технології аналізу та передбачення заробітної плати у сфері Data Science у 2023 році та побудова моделей класифікації, які дадуть високу точність передбачення заробітної плати у сфері Data Science.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальності 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»).

9. Порядок приймання роботи

Публічний захист «____» _____ 2023 р.

Початок розробки «____» _____ 2023 р.

Граничні терміни виконання МКР «____» _____ 2023 р.

Розробив студент групи 2ІСТ-22м _____ Богдан ДОЛЕНКО

Додаток Б
Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технології аналізу та передбачення заробітної плати у сфері Data Science у 2023 році»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 96,22% Схожість 3.78%

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Богдан ДОЛЕНКО

Керівник роботи _____ Ілона ВАРЧУК

Додаток В
Лістинг програми

```
# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.436364Z","iopub.status.idle":"2023-12-17T17:13:47.436761Z","shell.execute_reply.started":"2023-12-17T17:13:47.436567Z","shell.execute_reply":"2023-12-17T17:13:47.436585Z"}}
# Data Loading and Managing
import numpy as np
import pandas as pd
from tensorflow import keras

# Data Visualization
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Data Preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split

# Model
import xgboost as xgb
from sklearn.svm import SVR
from tensorflow.keras import layers
from sklearn.tree import  DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

# Model Performance
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# %% [code]


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.438635Z","iopub.status.idle":"2023-12-17T17:13:47.439029Z","shell.execute_reply.started":"2023-12-17T17:13:47.438837Z","shell.execute_reply":"2023-12-17T17:13:47.438856Z"}}
# Set a random seed
seed = 42
np.random.seed(42)

# Specify the path to the data file
file_path = '/kaggle/input/data-science-salaries-2023/ds_salaries.csv'

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.440846Z","iopub.status.idle":"2023-12-17T17:13:47.441244Z","shell.execute_reply.started":"2023-12-17T17:13:47.441046Z","shell.execute_reply":"2023-12-17T17:13:47.441065Z"}}
# Load data
df = pd.read_csv(filepath_or_buffer=file_path)

# Quick look
```

```
df.head()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.442526Z","iopub.status.idle":"2023-12-
17T17:13:47.442883Z","shell.execute_reply.started":"2023-12-
17T17:13:47.442704Z","shell.execute_reply":"2023-12-17T17:13:47.442722Z"}}
# Information about the data columns/features.=
df.info()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.444258Z","iopub.status.idle":"2023-12-
17T17:13:47.444655Z","shell.execute_reply.started":"2023-12-
17T17:13:47.444469Z","shell.execute_reply":"2023-12-17T17:13:47.444489Z"}}
# Cross check for null values
df.isnull().sum()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.445724Z","iopub.status.idle":"2023-12-
17T17:13:47.446073Z","shell.execute_reply.started":"2023-12-
17T17:13:47.445896Z","shell.execute_reply":"2023-12-17T17:13:47.445913Z"}}
# Extract all categorical columns
categorical_columns = df.columns[df.dtypes == 'object']

# Quick look
categorical_columns

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.447378Z","iopub.status.idle":"2023-12-
17T17:13:47.448148Z","shell.execute_reply.started":"2023-12-
17T17:13:47.447947Z","shell.execute_reply":"2023-12-17T17:13:47.447968Z"}}
# Extract the sub-categories of each category
for col in categorical_columns:
    unique_values = df[col].unique()
    print(f"{col} : {unique_values}\n")


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.449149Z","iopub.status.idle":"2023-12-
17T17:13:47.449957Z","shell.execute_reply.started":"2023-12-
17T17:13:47.449755Z","shell.execute_reply":"2023-12-17T17:13:47.449775Z"}}
# Descriptive analysis
df.describe()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.451831Z","iopub.status.idle":"2023-12-
17T17:13:47.452194Z","shell.execute_reply.started":"2023-12-
17T17:13:47.452011Z","shell.execute_reply":"2023-12-17T17:13:47.452028Z"}}
# Extract the value counts for years
data_values = df.work_year.value_counts()

# Create a pie chart
fig = go.Figure(data=go.Pie(
    labels=data_values.index,
    values=data_values.values,
    hole=0.4,
    textinfo='label+percent',
    insidetextorientation='radial',
    marker=dict(
        colors=px.colors.sequential.RdBu,
        line=dict(
            color='honeydew',
```

```
            width=2
        )
    ),
))

# Update layout
fig.update_layout(
    title="Distribution of Data Entries across Years",
    annotations=[dict(text="Year Overview", showarrow=False, font_size=20)],
    height=600
)

# Show the pie chart
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.455069Z","iopub.status.idle":"2023-12-
17T17:13:47.455625Z","shell.execute_reply.started":"2023-12-
17T17:13:47.455348Z","shell.execute_reply":"2023-12-17T17:13:47.455375Z"}}
# Understanding Salary Distribution
mean_salary = df.groupby('work_year')['salary_in_usd'].mean()

# Create subplots with shared x-axis
fig = make_subplots()

# Add bar trace
bar_trace = go.Bar(
    x=mean_salary.index,
    y=mean_salary.values,
    name="Mean Salary",
    marker=dict(color='steelblue')
)
fig.add_trace(bar_trace)

# Add line trace
line_trace = go.Scatter(
    x=mean_salary.index,
    y=mean_salary.values,
    name="Mean Salary",
    mode="lines+markers",
    line=dict(color='darkorange'),
    marker=dict(symbol='circle-open', size=15)
)
fig.add_trace(line_trace)

# Update layout
fig.update_layout(
    title="Examining Mean Salary by Year",
    xaxis_title="Working Year",
    yaxis_title="Mean Salary",
    height=600
)

# Show the combined plot
fig.show()
```

```
# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.456591Z","iopub.status.idle":"2023-12-17T17:13:47.457122Z","shell.execute_reply.started":"2023-12-17T17:13:47.456840Z","shell.execute_reply":"2023-12-17T17:13:47.456864Z"}}
# Create the box plot
fig = px.box(
    data_frame=df,
    x='work_year',
    y='salary_in_usd',
    color='work_year',
    points='all',
    height=600,
    notched=True
)

# Update layout
fig.update_layout(
    title="Salary Distribution across Years",
    xaxis_title="Year",
    yaxis_title="Salary (USD)"
)

# Show the box plot
fig.show()


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.459212Z","iopub.status.idle":"2023-12-17T17:13:47.460122Z","shell.execute_reply.started":"2023-12-17T17:13:47.459830Z","shell.execute_reply":"2023-12-17T17:13:47.459857Z"}}
# Create the box plot
fig = px.box(
    data_frame=df,
    x='work_year',
    y='salary_in_usd',
    color='work_year',
    points='all',
    height=600,
    notched=True
)

# Update layout
fig.update_layout(
    title="Salary Distribution across Years",
    xaxis_title="Year",
    yaxis_title="Salary (USD)"
)

# Show the box plot
fig.show()



# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.461355Z","iopub.status.idle":"2023-12-17T17:13:47.461880Z","shell.execute_reply.started":"2023-12-17T17:13:47.461600Z","shell.execute_reply":"2023-12-17T17:13:47.461624Z"}}
# Create a histogram with facet columns
```

```
fig = px.histogram(
   data_frame=df,
   x='experience_level',
   facet_col='work_year',
   nbins=7,
   text_auto=True,
   labels={'experience_level': 'Experience Level', 'count': 'Count'},
   title='Distribution of Experience Levels across Work Years'
)

# Configure histogram aesthetics
fig.update_traces(
   textposition='auto',
   marker_color='brown',
)

# Update layout
fig.update_layout(yaxis_title='Count')

# Show the histogram
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.463726Z","iopub.status.idle":"2023-12-
17T17:13:47.464271Z","shell.execute_reply.started":"2023-12-
17T17:13:47.463981Z","shell.execute_reply":"2023-12-17T17:13:47.464005Z"}}
# Determine the mean salary based on Experience Level
count_values = df.experience_level.value_counts()
salary_values = df.groupby('experience_level').mean(
   numeric_only=True)['salary_in_usd']

# Create subplots
fig = make_subplots(rows=1, cols=2, specs=[
               [{'type': 'domain'}, {'type': 'domain'}]])

# Count of Experience Level
count_trace = go.Pie(
   labels=count_values.index,
   values=count_values.values,
   pull=[0, 0, 0, 0.2],
   name="Value Counts"
)

# Salary based on Experience Level
salary_trace = go.Pie(
   labels=salary_values.index,
   values=salary_values.values,
   pull=[0, 0.2, 0, 0],
   name="Mean Salary"
)

# Add traces to the subplots
fig.add_trace(count_trace, row=1, col=1)
fig.add_trace(salary_trace, row=1, col=2)

# Update the traces
```

```
fig.update_traces(
   textinfo='label+percent',
   hole=0.4,
   marker=dict(
      colors=px.colors.sequential.RdBu,
      line_color='lavender',
      line_width=2.5
   )
)

# Update the figure layout
fig.update_layout(
   title_text="Distribution based on Experience Level",
   height=600,
   annotations=[
      dict(
         text="Value Counts",
         font_size=17,
         showarrow=False,
         x=0.17
      ),
      dict(
         text="Mean Salary",
         font_size=20,
         showarrow=False,
         x=0.83
      )
   ]
)

# Show the final figure
fig.show()

# Create a figure
fig = make_subplots()

# Add bar trace
bar_trace = go.Bar(
   x=salary_values.index,
   y=salary_values.values,
   name="Mean Salary (Bar)",
   text=salary_values.values,
   marker=dict(
      color='brown',
      line_color='white',
      line_width=2.5,
   )
)

# Add traces to the figure
fig.add_trace(bar_trace)

# Update Layout
fig.update_layout(
   title="Mean Salary Distribution across Experience Levels",
```

```
    xaxis_title="Experience Level",
    yaxis_title="Mean Salary (USD)",
    height=600
)

# Show final figure
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.465558Z","iopub.status.idle":"2023-12-
17T17:13:47.466081Z","shell.execute_reply.started":"2023-12-
17T17:13:47.465813Z","shell.execute_reply":"2023-12-17T17:13:47.465837Z"}}
# Create box plots with facets
fig = px.box(df, x='experience_level', y='salary_in_usd', color='experience_level', facet_col='work_year')

# Update the layout
fig.update_layout(
    title="Salary Distribution by Experience Level",
    yaxis_title="Salary (USD)",
    height=600
)

# Show the figure
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.467511Z","iopub.status.idle":"2023-12-
17T17:13:47.468039Z","shell.execute_reply.started":"2023-12-
17T17:13:47.467768Z","shell.execute_reply":"2023-12-17T17:13:47.467793Z"}}
# Create violin plots with facets
fig = px.violin(df, x='experience_level', y='salary_in_usd', color='experience_level', facet_col='work_year',
points='all')

# Update the layout
fig.update_layout(
    title="Salary Distribution by Experience Level",
    yaxis_title="Salary (USD)",
    height=600
)

# Show the figure
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.469816Z","iopub.status.idle":"2023-12-
17T17:13:47.470411Z","shell.execute_reply.started":"2023-12-
17T17:13:47.470102Z","shell.execute_reply":"2023-12-17T17:13:47.470137Z"}}
# Determine the mean salary based on Experience Level
count_values = df.employment_type.value_counts()
salary_values = df.groupby('employment_type').mean(
    numeric_only=True)['salary_in_usd']

# Create subplots
fig = make_subplots(rows=1, cols=2, specs=[
                [{'type': 'domain'}, {'type': 'domain'}]])

# Count of Experience Level
count_trace = go.Pie(
```

```
    labels=count_values.index,
    values=count_values.values,
    name="Value Counts"
)

# Salary based on Experience Level
salary_trace = go.Pie(
    labels=salary_values.index,
    values=salary_values.values,
    name="Mean Salary"
)

# Add traces to the subplots
fig.add_trace(count_trace, row=1, col=1)
fig.add_trace(salary_trace, row=1, col=2)

# Update the traces
fig.update_traces(
    textinfo='label+percent',
    hole=0.4,
    marker=dict(
        colors=px.colors.sequential.RdBu,
        line_color='lavender',
        line_width=2.5
    )
)

# Update the figure layout
fig.update_layout(
    title_text="Distribution based on Experience Level",
    height=600,
    annotations=[
        dict(
            text="Value Counts",
            font_size=17,
            showarrow=False,
            x=0.17
        ),
        dict(
            text="Mean Salary",
            font_size=20,
            showarrow=False,
            x=0.83
        )
    ]
)

# Show the final figure
fig.show()

# Create a figure
fig = make_subplots()

# Add bar trace
bar_trace = go.Bar(
```

```
    x=salary_values.index,
    y=salary_values.values,
    name="Mean Salary (Bar)",
    text=salary_values.values,
    marker=dict(
        color='brown',
        line_color='white',
        line_width=2.5,
    )
)

# Add traces to the figure
fig.add_trace(bar_trace)

# Update Layout
fig.update_layout(
    title="Mean Salary Distribution across Experience Levels",
    xaxis_title="Experience Level",
    yaxis_title="Mean Salary (USD)",
    height=600
)

# Show final figure
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.471713Z","iopub.status.idle":"2023-12-
17T17:13:47.472244Z","shell.execute_reply.started":"2023-12-
17T17:13:47.471968Z","shell.execute_reply":"2023-12-17T17:13:47.471992Z"}}
# Create a box plot
box_plot = px.box(
    data_frame=df,
    x='employment_type',
    y='salary_in_usd',
    points='outliers',
    title="Salary Distribution across Employment Type",
    color='employment_type',
    notched=True
)

# Customize the layout
box_plot.update_layout(
    xaxis_title="Employment Type",
    yaxis_title="Salary (USD)",
    height=600
)

# Show the box plot
box_plot.show()

# Create a facetted box plot
facet_box_plot = px.box(
    data_frame=df,
    x='employment_type',
    y='salary_in_usd',
    points='outliers',
```

```
    title="Salary Distribution across Employment Type",
    color='employment_type',
    facet_col='experience_level'
)

# Customize the layout
facet_box_plot.update_layout(
    xaxis_title="Employment Type",
    yaxis_title="Salary (USD)",
    height=600
)

# Show the facetted box plot
facet_box_plot.show()

# Create a violin plot
violin_plot = px.violin(
    data_frame=df,
    x='employment_type',
    y='salary_in_usd',
    points='all',
    title="Salary Distribution across Employment Type",
    color='employment_type'
)

# Customize the layout
violin_plot.update_layout(
    xaxis_title="Employment Type",
    yaxis_title="Salary (USD)",
    height=600
)

# Show the violin plot
violin_plot.show()


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.474025Z","iopub.status.idle":"2023-12-
17T17:13:47.474585Z","shell.execute_reply.started":"2023-12-
17T17:13:47.474309Z","shell.execute_reply":"2023-12-17T17:13:47.474335Z"}}
# Create a box plot
box_plot = px.box(
    data_frame=df,
    x='work_year',
    y='salary_in_usd',
    color='employment_type',
    title="Salary Distribution across Employment Types",
    labels={'work_year': 'Work Year', 'salary_in_usd': 'Salary (USD)', 'employment_type': 'Employment Type'}
)

# Customize the layout
box_plot.update_layout(
    legend_title="Employment Type",
    height=600
)
```

```
# Show the box plot
box_plot.show()


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.475986Z","iopub.status.idle":"2023-12-17T17:13:47.476552Z","shell.execute_reply.started":"2023-12-17T17:13:47.476263Z","shell.execute_reply":"2023-12-17T17:13:47.476304Z"}}
# Create a Sunburst chart
fig = px.sunburst(df, path=['experience_level', 'employment_type', 'work_year'], values='salary_in_usd')

# Update layout settings
fig.update_layout(
    title='Total Salary Distribution by Work Year and Experience Level and Employment Type',  # Set the title
    height=600  # Set the height of the figure
)

# Display the figure
fig.show()


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.478518Z","iopub.status.idle":"2023-12-17T17:13:47.478940Z","shell.execute_reply.started":"2023-12-17T17:13:47.478718Z","shell.execute_reply":"2023-12-17T17:13:47.478763Z"}}
# Determine the mean salary based on Experience Level
count_values = df.company_size.value_counts()
salary_values = df.groupby('company_size').mean(
    numeric_only=True)['salary_in_usd']

# Create subplots
fig = make_subplots(rows=1, cols=2, specs=[
                [{'type': 'domain'}, {'type': 'domain'}]])

# Count of Experience Level
count_trace = go.Pie(
    labels=count_values.index,
    values=count_values.values,
    name="Value Counts"
)

# Salary based on Experience Level
salary_trace = go.Pie(
    labels=salary_values.index,
    values=salary_values.values,
    name="Mean Salary"
)

# Add traces to the subplots
fig.add_trace(count_trace, row=1, col=1)
fig.add_trace(salary_trace, row=1, col=2)

# Update the traces
fig.update_traces(
    textinfo='label+percent',
    hole=0.4,
    marker=dict(
```

```
        colors=px.colors.sequential.RdBu,
        line_color='lavender',
        line_width=2.5
    )
)

# Update the figure layout
fig.update_layout(
    title_text="Distribution based on Experience Level",
    height=600,
    annotations=[
        dict(
            text="Value Counts",
            font_size=17,
            showarrow=False,
            x=0.17
        ),
        dict(
            text="Mean Salary",
            font_size=20,
            showarrow=False,
            x=0.83
        )
    ]
)

# Show the final figure
fig.show()

# Create a figure
fig = make_subplots()

# Add bar trace
bar_trace = go.Bar(
    x=salary_values.index,
    y=salary_values.values,
    name="Mean Salary (Bar)",
    text=salary_values.values,
    marker=dict(
        color='brown',
        line_color='white',
        line_width=2.5,
    )
)

# Add traces to the figure
fig.add_trace(bar_trace)

# Update Layout
fig.update_layout(
    title="Mean Salary Distribution across Experience Levels",
    xaxis_title="Experience Level",
    yaxis_title="Mean Salary (USD)",
    height=600
)
```

```
# Show final figure
fig.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.485497Z","iopub.status.idle":"2023-12-17T17:13:47.485994Z","shell.execute_reply.started":"2023-12-17T17:13:47.485772Z","shell.execute_reply":"2023-12-17T17:13:47.485793Z"}}
# Create a histogram
histogram = px.histogram(
    data_frame=df,
    x='salary_currency',
    y='salary_in_usd',
    text_auto=True,
    color='company_size',
    title='Salary Distribution by Currency',
    labels={'salary_currency': 'Salary Currency', 'salary_in_usd': 'Salary (USD)'}
)

# Customize the layout
histogram.update_layout(
    showlegend=True,
    height=600
)

# Show the histogram
histogram.show()

# Create a Bee Swarm plot
swarm_plot = px.strip(
    data_frame=df,
    x='salary_currency',
    y='salary_in_usd',
    color='company_size',
    title='Salary Distribution by Currency',
    labels={'salary_currency': 'Salary Currency', 'salary_in_usd': 'Salary (USD)'},
    hover_data={'salary_in_usd': ':$.2f'}
)

# Customize the layout
swarm_plot.update_layout(
    showlegend=True,
    height=600
)

# Show the Bee Swarm plot
swarm_plot.show()


# %% [code] {"execution":{"iopub.status.busy":"2023-12-17T17:13:47.487417Z","iopub.status.idle":"2023-12-17T17:13:47.487816Z","shell.execute_reply.started":"2023-12-17T17:13:47.487622Z","shell.execute_reply":"2023-12-17T17:13:47.487641Z"}}

# Calculate the count of matching and non-matching locations
location_counts = (df['company_location'] == df['employee_residence']).value_counts()
```

```
# Create a pie chart
pie_chart = px.pie(
    names=['Matching Locations', 'Non-Matching Locations'],
    values=location_counts.values,
    title='Matching vs Non-Matching Company and Employee Residence Locations',
    color_discrete_sequence=px.colors.sequential.RdBu,
    height=600,
    hole=0.4,
)

# Customize the layout
pie_chart.update_traces(
    insidetextorientation='radial',
    textinfo='label+percent',
    marker=dict(
        line=dict(
            color='honeydew',
            width=2
        )
    )
)

# Show the pie chart
pie_chart.show()
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЗАРОБІТНОЇ ПЛАТИ У СФЕРІ DATA SCIENCE У 2023 РОЦІ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2023 р.

Вінниця 2023

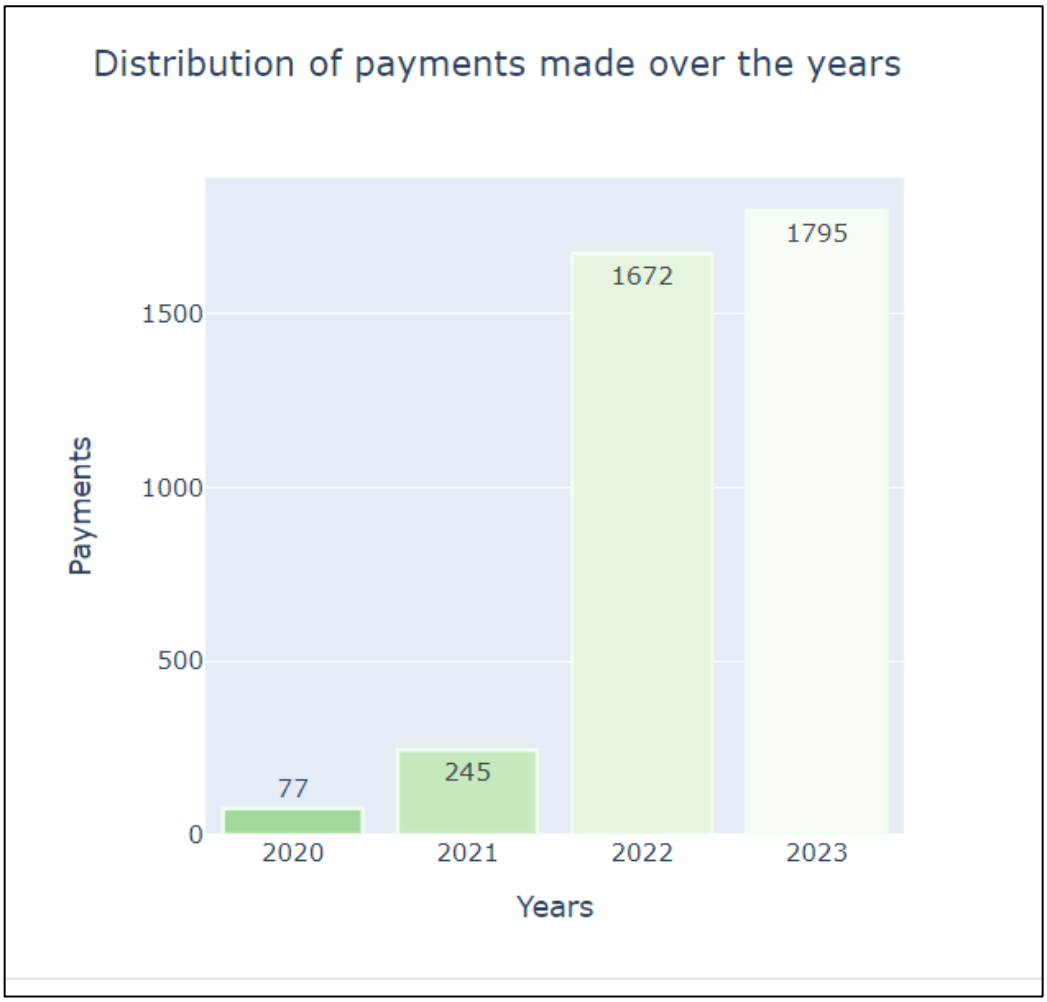


Рисунок Г.1 - Графік розподілу виплат за роками

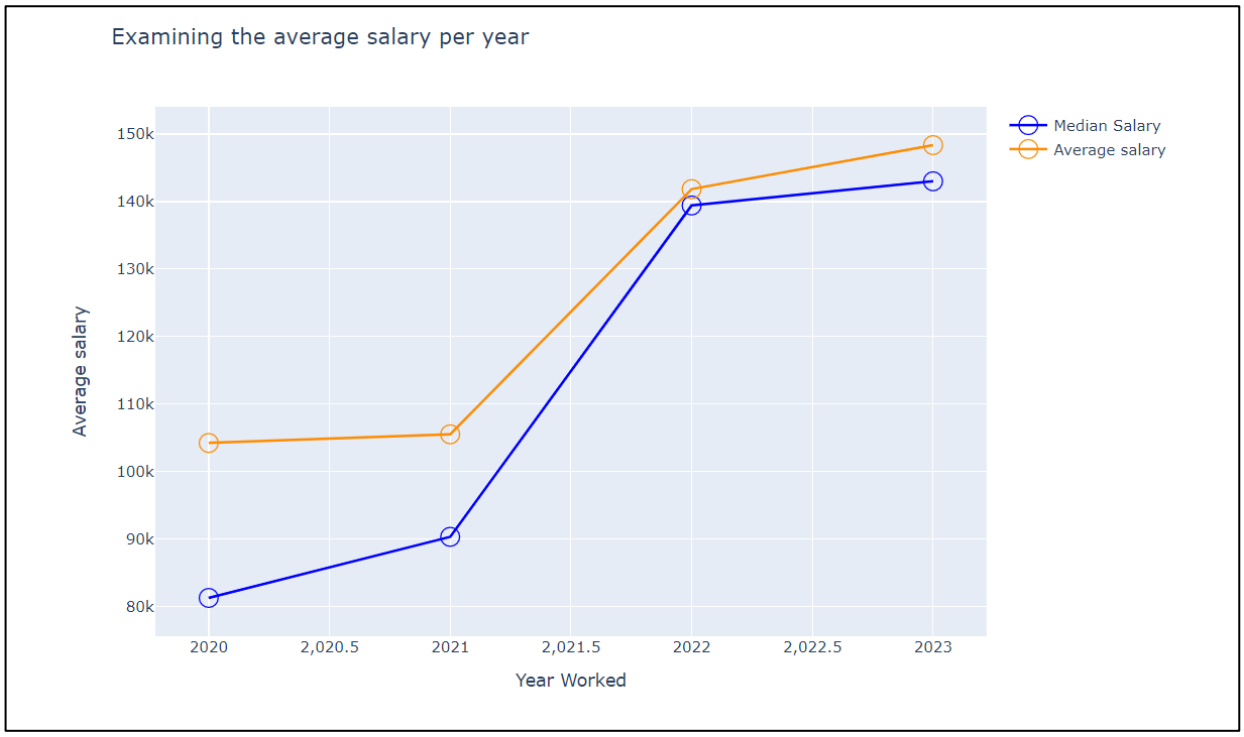


Рисунок Г.2 - Графік вивчення середньої заробітної плати за рік

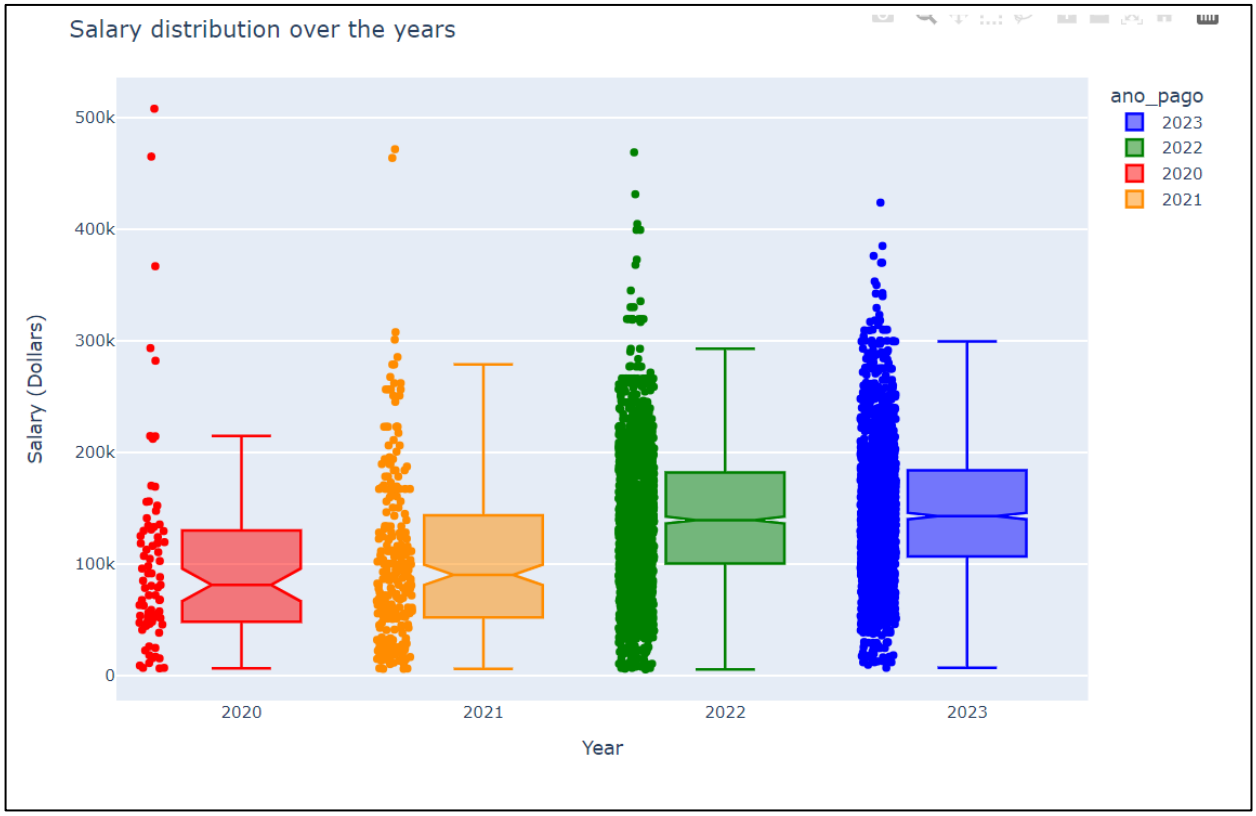


Рисунок Г.3 – Графік розподілу заробітної плати за роками

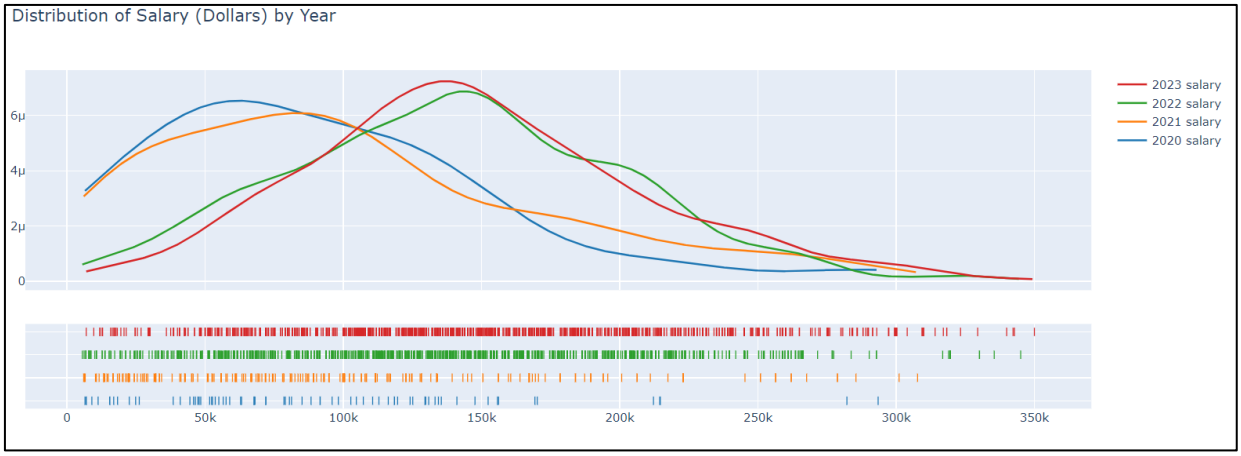


Рисунок Г.4 – Графік розподілу заробітної плати (у доларах) за роками

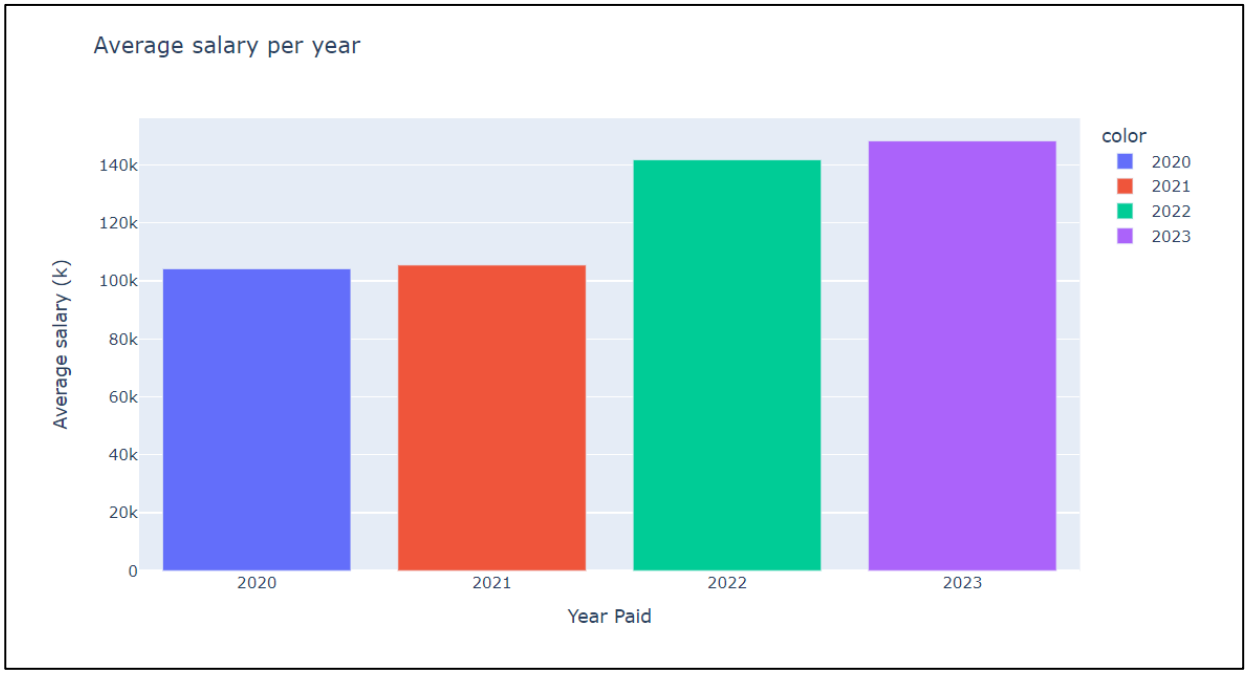


Рисунок Г.5 – Графік середньої заробітної плати за рік

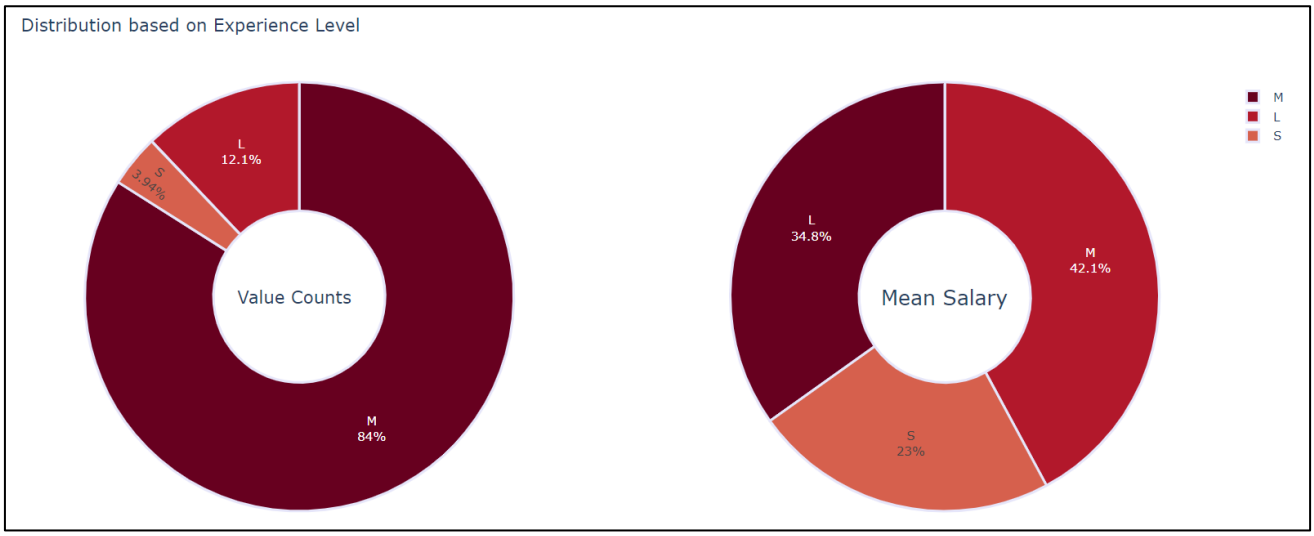


Рисунок Г.6 - Діаграми розподілу на основі рівня досвіду

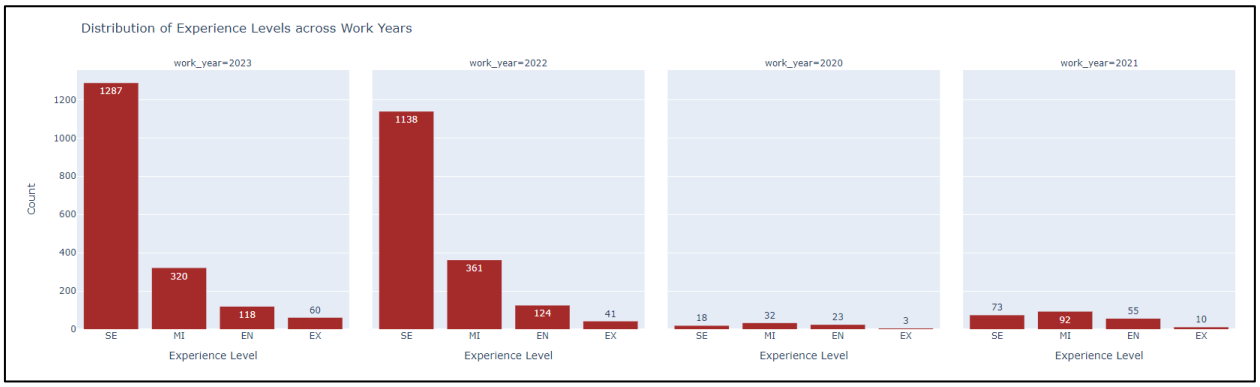


Рисунок Г.7 – Графік розподілу рівнів досвіду працівників за роками стажу.

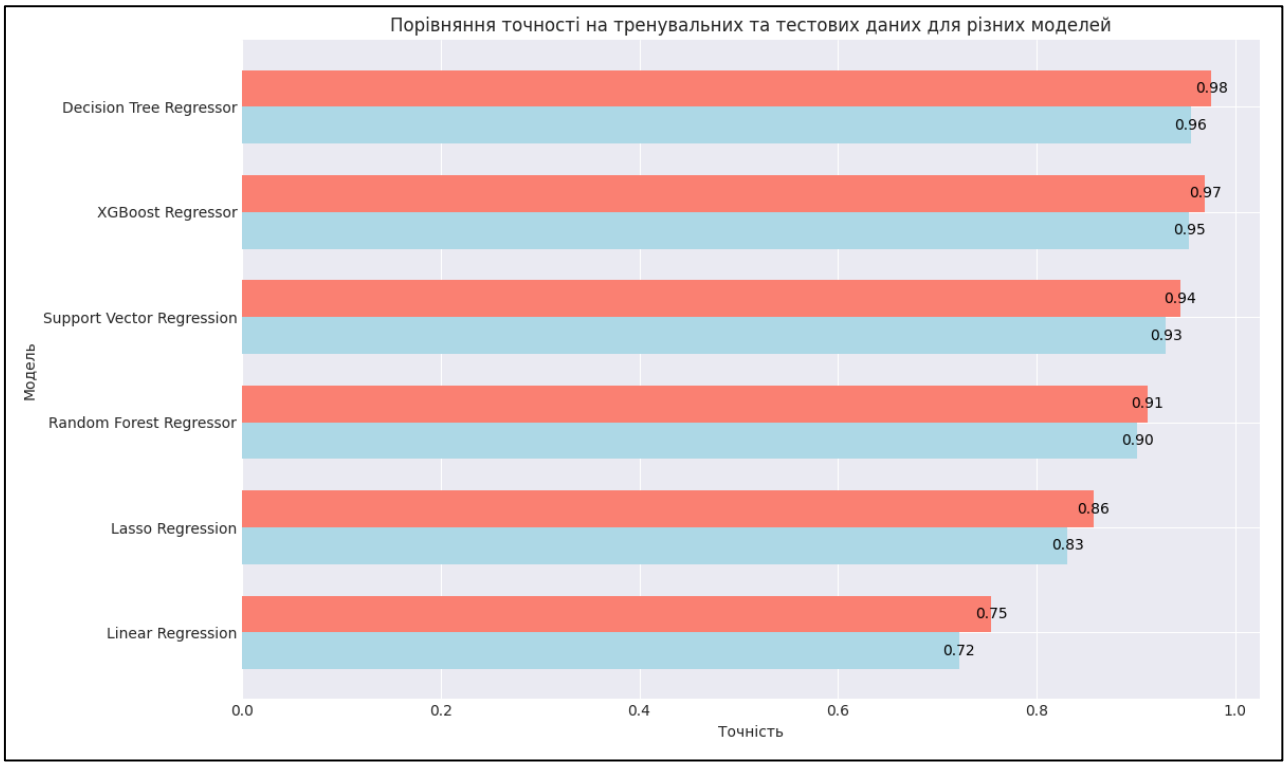


Рисунок Г.8 – Порівняння точності на тренувальних та тестових даних для різних моделей машинного навчання за допомогою графіку