**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

**Пояснювальна записка**

до магістерської роботи

на ступіть вищої освіти магістр

на тему: «**РОЗРОБКА АЛГОРИТМУ ПРОГНОЗУВАННЯ ЗАПОВНЕННЯ ЛІЦЕНЗІЙНОГО ОБСЯГУ УНІВЕРСИТЕТУ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ**»

Виконав: студент 6 курсу, групи ПДМ–61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Лакович М.С.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент .
(прізвище та ініціали)

Київ – 2020

**ДЕРЖАВНИЙ УНІВАЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Магістр»
Спеціальність - 121 «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
**_____Негоденко О.В.**
"___" _____ 20__ року

**З А В Д А Н Н Я**
**НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА**

Лаковича Михайла Сергійовича

1. Тема роботи: «Розробка алгоритму прогнозування заповнення ліцензійного обсягу університету за допомогою машинного навчання»

Керівник роботи Жебка Вікторія Вікторівна доцент кафедри, кандидат технічних наук, доцент

затверджені наказом вищого навчального закладу від 13.10.2020 року № 230

2. Строк подання студентом роботи 24.12.2020.
3. Вихідні данні до роботи:
   - 3.1. Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо машинного навчання та роботи з великими даними.
   - 3.2. Методи машинного навчання.
   - 3.3. Дані по ліцензійному обсягу та динаміка вступу абітурієнтів.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   - 4.1. Дослідити сучасні популярні алгоритми прогнозування.
   - 4.2. Проаналізувати вхідну вибірку.
   - 4.3. Розробити алгоритм прогнозування.
   - 4.4. Висновки.
5. Перелік графічного матеріалу:
   - 5.1. Тема та мета магістерської роботи.
   - 5.2. Сучасні алгоритми прогнозування.
   - 5.3. Аналіз вхідних даних.
   - 5.4. Формула згладжування.
   - 5.5. Результати кросс-валідації та прогнозування на тестових даних.
   - 5.6. Порівняння з іншим алгоритмом.
   - 5.7. Результати прогнозування вступної компанії.

6. Дата видачі завдання 02.11.2020
7. Календарний план-графік

| № пор | Завдання | Термін виконання | Відмітка про виконання |
|---|---|---|---|
| 1. | Ознайомлення з постановкою задачі та вивчення літератури<br>Написання 1 розділу, представлення керівнику | 02.11.2020 - 05.11.2020 | |
| 2. | Попередній друк 1 розділу та допоміжних сторінок (черновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел. | 06.11.2020 – 09.11.2020 | |
| 3. | Написання 2 розділу, представлення керівнику | 10.11.2020 – 14.11.2020 | |
| 4. | Написання 3 розділу, представлення керівнику | 15.11.2020 – 25.11.2020 | |
| 6. | Загальне редагування та друк пояснювальної записки, графічного матеріалу | 26.11.2020 – 04.12.2020 | |
| 7. | Проходження нормо-контролю, перепліт пояснювальної записки. | 06.12.2020 – 11.12.2020 | |
| 8. | Розробка тексту доповіді. Оформлення графічного матеріалу для презентації | 14.12.2020 – 18.12.2020 | |
| 9. | Отримання відгуку керівника, рецензії. | 23.12.2020 | |
| 10. | Захист дипломного проекту | 25.12.2020 | |

Студент _____ Лакович М.С.
(підпис) (прізвище та ініціали)
Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

# РЕФЕРАТ

Текстова частина магістерської роботи: 71 с., 2 табл., 30 рис., 2 дод., 21 джерело.

MACHINE LEARNING, ШТУЧНИЙ ІНТЕЛЕКТ, АЛГОРИТМИ, ЧАСОВІ РЯДИ, ПРОГНОЗУВАННЯ НА ОСНОВІ ДАНИХ

*Об'єкт дослідження* – прогнозування заповнення ліцензійного обсягу університету за допомогою машинного навчання .

*Предмет дослідження* – алгоритм прогнозування, за допомогою якого спрогнозовано статистику абітурієнтів для вступної компанії 2021 року.

*Мета роботи* – покращення процесу прогнозування заповнення ліцензійного обсягу університету за допомогою методів машинного навчання.

*Методи дослідження* – методи експоненціального згладжування, для згладжування вхідний даних, метод побудови моделі SARIMA та ARIMA, методи бустингу, методи лінійної регресії.

Досліджено вхідні дані, інформація, про абітурієнтів, які надані вищим навчальним закладом. На основі даних від вищого навчального закладу проведено дослідження та розроблений алгоритм для прогнозування статистики абітурієнтів у вступній компанії наступного року. Розглянуто різні ознаки об'єктів (абітурієнтів). Досліджено ступінь впливу у об'єкта ознак на його місце в рейтингу абітурієнтів.

Методом дослідження обрано та адаптовано кращий метод згладжування. За допомогою якого можна більш точно, на основі вхідних даних, спрогнозувати відсоток абітурієнтів для вступної компанії наступного року.

Отримані результати дають інформацію вищому навчальному закладу статистику абітурієнтів, які до них поступлять, на основі який роблять прогнозування на наступний рік. Також отримані результати дадуть інформацію Міністерству освіти та науки України про кошти та місця, які потрібно виділити на державні та контрактні місця вищому навчальному закладу.

*Галузь використання* – будь-який вищий навчальний заклад, який має дані для побудови моделі на основі вступу абітурієнтів до цього вищого навчального закладу за попередні 10-20 років.

ЗМІСТ

# ВСТУП

Суть роботи полягає в створенні алгоритму для вирішення проблеми, проблеми невідомої кількості абітурієнтів під час вступної компанії. Ґрунтуючись на даних вступної компанії за останні роки можна спрогнозувати кількість заявок від абітурієнтів в наступному році.

В роботі буде розглянуто сучасні методи та алгоритми прогнозування даних, описано принципи, алгоритми та математичні формули, які вони використовують.

*Об'єкт дослідження* – прогнозування заповнення ліцензійного обсягу університету за допомогою машинного навчання .

*Предмет дослідження* – алгоритм прогнозування, за допомогою якого спрогнозовано статистику абітурієнтів для вступної компанії 2021 року.

*Мета роботи* – покращення процесу прогнозування заповнення ліцензійного обсягу університету за допомогою методів машинного навчання.

*Методи дослідження* – методи експоненціального згладжування, для згладжування вхідний даних, метод побудови моделі SARIMA та ARIMA, методи бустингу, методи лінійної регресії.

Кожен рік абітурієнти вступають до вищих навчальних закладів для того щоб здобути нові знання, отримати нові можливості та реалізувати свій безмежний потенціал.

Під час процесу підготовки вищих навчальних закладів до щорічної вступної компанії витрачаються кошти на створення навчальної програми, кваліфікування, набір спеціалістів, виділення кабінетів, улагодження розкладів кабінетів та інше.

У вищих навчальних закладів є щорічна проблема. Вони не можуть знати свою актуальність на ринку, на районі, в місті та в іншому. Вони не можуть надати всім абітурієнтам можливість навчатися в них, оскільки вони не знають скільки абітурієнтів прийдуть та залишать свої заявки.

Частина вищих навчальних закладів не знають або не розуміють свою актуальність, тому не набирають більше абітурієнтів для сформування груп. Відмовляють багатьом, не зумів надати їм права на навчання у вищі. Хоча ця частина вищих навчальних закладів має можливість навчати не одну групу, а дві, три або більше.

Проаналізовано надані вхідні дані вступної компанії вищого навчального закладу за останні 10 років. Розібрано аномальні значення, які отримано на вхідній вибірці.

Описано алгоритм роботи прогнозування створеного алгоритму на кількості заявок абітурієнтів для наступного року. Протестовано алгоритм на інших вхідних даних для аналізу стабільності алгоритму. Порівняно результати навчальної моделі з результатами популярної лінійної регресії, проаналізовано дані які отримали.

# 1 ДОСЛІДЖЕННЯ ПОПУЛЯРНИХ АЛГОРИТМІВ ПРОГНОЗУВАННЯ

Для створення алгоритму машинного навчання, на основі моделі на деякій вибірці вхідних та вихідних даних, для прогнозування майбутніх значень потрібно дослідити вже існуючі алгоритми та методи прогнозування машинного навчання. Серед яких:

- сімейство регресійних методів;
- сімейство бустингу, серед який градієнтний бустинг та xgboost (розроблений відносно недавно, вихідний код якого доступний для вивчення та розробки будь-ким бажаючим, знаходиться на github);
- метод проміжного представлення для прогнозування часових рядів;
- локальні методи прогнозування;
- метод гнучких найменших квадратів.

Прогнозування – це передбачення майбутнього на підставі накопиченого досвіду і поточних припущень щодо нього.

Прогнозування – це складний процес, під час аналізу якого потрібно вирішувати велику кількість різних запитань. Для його знаходження потрібно використовувати декілька поєднань різних методів прогнозування, їх на даний час існує величезна кількість, на практиці використовуються не багато, лише 15 - 20.

## 1.1 Сімейство регресійних методів прогнозування

### 1.1.1 Лінійна регресія

Лінійна регресія (англ. linear regression) – метод відновлення залежності однієї (що пояснюється, залежною) змінної $y$ від іншої або кількох інших змінних (факторів, регресорів, незалежних змінних) $x$ з лінійною функцією залежності. Даний метод дозволяє прогнозувати значення залежної змінної $y$ про значень незалежної змінної $x$.

На вході отримаємо:

- $f_1(x), \ldots, f_n(x)$ – числові ознаки;

- модель багатовимірної лінійної регресії:

$$f(x,a)=\sum_{j=1}^{n} a_j f_j(x), \tag{1.1}$$

де $a \in R^n$;

- навчальна вибірка – деяка множина з пар $a(x_i, y_i)_{i=1\ldots n}$;
- $x_i$ – об’єкти з множини $X \in R^n$;
- $y_i$ – об’єкти з множини $X \in R$.

Матричне представлення:

$$\underset{l*n}{\mathrm{F}}=\begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_l) & \cdots & f_n(x_l) \end{pmatrix}, \underset{l*1}{\mathrm{y}}=\begin{pmatrix} y_1 \\ \cdots \\ y_l \end{pmatrix}, \underset{n*1}{\mathrm{a}}=\begin{pmatrix} a_1 \\ \cdots \\ a_n \end{pmatrix}, \tag{1.2}$$

де F – матриця об’єктів-ознак, де рядки відповідають об’єктам, а стовпці – ознакам, y – вектор відповідей або цільовий вектор, a – вектор коефіцієнтів.

У цих трьох векторно-матричних позначеннях дуже зручно розписати постановку задачі найменших квадратів:

$$Q(a,X^l)=\sum_{i=1}^{n}(f(x_i,a)-y_i)^2=\|Fa-y\|^2 \to \min_a. \tag{1.3}$$

Необхідно знайти вектор $a$ при відомій матриці F та відомому вектору стовпцю $y$.

Напишемо необхідні умови мінімуму в матричному вигляді:

$$\frac{\partial Q}{\partial a}(a)=2F^T(Fa-y)=0. \tag{1.4}$$

Звідси виходить нормальна система задачі методу найменших квадратів:

$$F^T Fa = F^T y, \tag{1.5}$$

де $F^T Fa$ - $n * n$.

Отримали систему рівнянь, звідки зможемо отримати вектор $a$, який і шукаємо:

$$a^* = (F^T F)^{-1} F^T y = F^+ y, \tag{1.6}$$

де $F^+$ – псевдо-обернена матриця.

Значення функціоналу: $Q(a^*) = \|P_F y - y\|^2$, де $P_F = FF^+ = F(F^T F)^{-1} F^T$ – проекція матриці.

У разі мультиколінеарності (стовпці матриці $F$ лінейно-залежні) нам не вдасться знайти зворотну матрицю до $F^T F$ (вона буде вироджена).

Якщо ж стовпці матриці $F$ майже лінійно-залежні, то у нас виникне маса обчислювальних проблем зі зверненням цієї матриці.

### 1.1.2 Множинна лінійна регресія

Множинною називають лінійну регресію, в моделі якої число незалежних змінних дві або більше.

Рівняння має вигляд:

$$Y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n. \tag{1.7}$$

Як і в простої лінійної регресії, параметри моделі $b_n$ обчислюються за допомогою методу найменших квадратів.

Відмінність між простою та множинної лінійною регресією полягає в тому, що замість лінії регресії в ній використовується гіперплощині.

Перевага множинної лінійної регресії в порівнянні з простою полягає в тому, що використання в моделі декількох вхідних змінних дозволяє збільшити частку пояснених дисперсій вихідної змінної, і таким чином поліпшити відповідність моделі даних. Тобто при додаванні в модель кожної нової змінної коефіцієнт детермінації зростає.

Однак в множинної лінійної регресії виникають і проблеми, нехарактерні для простої моделі:

- можлива поява мультіколінеарності;
- необхідно вибирати кращу модель, в якій мінімальний набір незалежних змінних зможе пояснити найбільшу частку дисперсії залежної.

### 1.1.3 Логістична регресія

Логістична регресія (з англ. - logistic regression) – це метод побудови лінійного класифікатора, що дозволяє оцінювати апостеріорні ймовірності приналежності об'єктів класів, див. рис. 1.1.

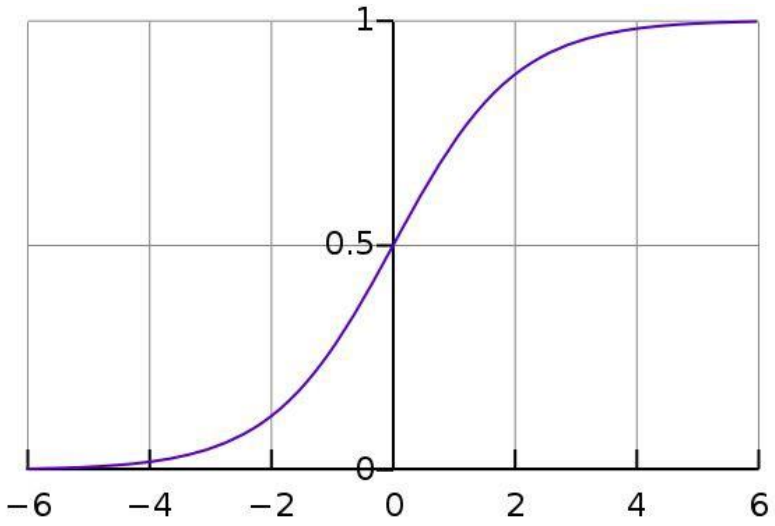



Рисунок 1.1 – Приклад логістичної регресії

Визначення:

Нехай об'єкти описуються n числовими ознаками $f_j: X \to \mathbb{R},\ j = 1, \ldots, n.$ Тоді простір описів об'єктів є $X = \mathbb{R}^n$.

Нехай $Y$ — кінцева безліч номерів (імен, міток) класів.

Нехай задана навчальна вибірка пару «об'єкт, відповідь»:

$$X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}. \quad (1.8)$$

Випадок двох класів

Випадок класифікації, коли на виході повинні отримати $Y = \{-1; +1\}$. У логістичної регресії будується лінійний алгоритм класифікації $a: X \to Y$ представлений у вигляді:

$$a(x, \omega) = sign\left(\sum_{j=1}^{n} \omega_j f_j(x) - \omega_0\right) = sign\langle x, \omega\rangle, \quad (1.9)$$

де $\omega_j$ – вага $j$-го признаку, $\omega_0$ — поріг прийняття рішення, $\omega = (\omega_0, \omega_1, \ldots, \omega_n)$ — вектор ваг, $\langle x, \omega\rangle$ — скалярний добуток опису об'єкта на вектор ваг.

Передбачається, що штучно введений «константних» нульовою ознака: $f_0(x) = -1$.

Задача навчання лінійного класифікатора полягає в тому, щоб за вибіркою $X^m$ налаштувати вектор ваг $\omega$. У логістичної регресії для цього вирішується завдання мінімізації емпіричного ризику з функцією втрат спеціального виду:

$$\varrho(\omega) = \sum_{i=1}^{m} \ln\big(1 + exp(-y_i\langle x_i, \omega\rangle)\big) \to \min_{\omega}. \quad (1.10)$$

Після того, як рішення $\omega$ знайдено, стає можливим не тільки обчислювати класифікацію $a(x) = sign\langle x, \omega\rangle$ для довільного об'єкта $x$ але і оцінювати апостеріорні ймовірності його приналежності класам:

$$\mathbb{P}\{y|x\} = \sigma(y\langle x, \omega\rangle), \qquad y \in Y, \quad (1.11)$$

де $\sigma(Z) = \frac{1}{1+e^{-Z}}$ — сігмоідна функція. У багатьох додатках апостеріорні ймовірності необхідні для оцінювання ризиків, пов'язаних з можливими помилками класифікації.

## 1.2 Сімейство бустингу

Бустинг (англ. boosting – покращення) – це процедура алгоритму машинного навчання, яка послідовно будує композицію взявши до уваги результати попередньої, прямуючи поліпшити результати попереднього алгоритму композиції компенсуванням недоліків всіх попередніх. Бустинг є жадібний алгоритм побудови композиції алгоритмів. Спочатку поняття бустингу виникло в роботах по ймовірно майже коректному навчанні у зв'язку з питанням: чи можливо, маючи безліч поганих (незначно відрізняються від випадкових) алгоритмів навчання, отримати хороший.

За останні 20 років бустинг залишається найбільш популярним методом машинного навчання, конкуруючи з нейронними мережами та алгоритмом опорних векторів. Основні плюси підходу - універсальність, простота, гнучкість (можливість побудови різних модифікацій), і, головне, висока точність даних на виході.

Бустинг з методом вирішальних дерев завжди вважався одним з найбільш ефективних методів якості класифікації. Проводилось безліч експериментів на яких спостерігається практично необмежене зменшення частоти помилок на незалежній вибірці даних для тестування по мірі нарощування композицій. Та якість на тестових даних зазвичай продовжували покращуватись навіть після досягнення майже ідеального розпізнавання всієї навчальної вибірки. Методи та алгоритми які існували довгий час змінило уявлення про те, що для підвищення точності та здатності необхідно обмежувати складність алгоритмів. На прикладі бустингу зрозуміло, що гарною якістю можуть володіти будь-які складні композиції, якщо їх правильно налаштовувати та обробляти.

Згодом феномен бустингу отримав теоретичне обґрунтування. Виявилося, що зважене голосування не збільшує ефективну складність алгоритму, а лише згладжує відповіді базових алгоритмів. Кількісні оцінки узагальнюючої здатності бустингу формулюються в термінах відступу. Ефективність бустингу пояснюється тим, що в міру додавання базових алгоритмів збільшуються відступи навчальних об'єктів. Причому бустинг продовжує розсовувати класи навіть після досягнення безпомилкової класифікації навчальної вибірки.

### 1.2.1 Метод найшвидшого бустингу

Метод найшвидшого бустингу – це техніка машинного навчання для задач класифікації і регресії, яка будує модель передбачення на формі дерев рішень, інакше кажучи модель ансамблю для слабких прогнозуючих моделей. Навчання ансамблю проводиться послідовно. На кожній ітерації розраховується відхилення від бачень прогнозів вже навченого ансамблю на навчальних даних. Наступна композиція буде враховувати ці відхилення та спробує передбачити їх та додатись до ансамблю. Додавши нове прогнозуюче дерево для передбачування до ансамблю то ми зможемо ще зменшити середнє значення відхилень моделі, яке і було метою оптимізаційної задачі. Нові дерева додаються в ансамбль до тих пір, поки помилка зменшується, або поки не виконується одна з правил "ранньої зупинки".

Знаходимо кінцевий алгоритм класифікації у вигляді композиції:

$$F_m(x) = \sum_{m=1}^{M} b_m h(x;\ a_m), b_m \in R, a_m \in A. \qquad (1.12)$$

Однак підбір оптимального набору параметрів $\{a_m, b_m\}_{m=1}^{M}$ – дуже трудомістке завдання. Тому потрібно будувати таку композицію шляхом жодного нарощування, кожен раз додаючи в суму доданок, що є найбільш оптимальним алгоритмом з можливих.

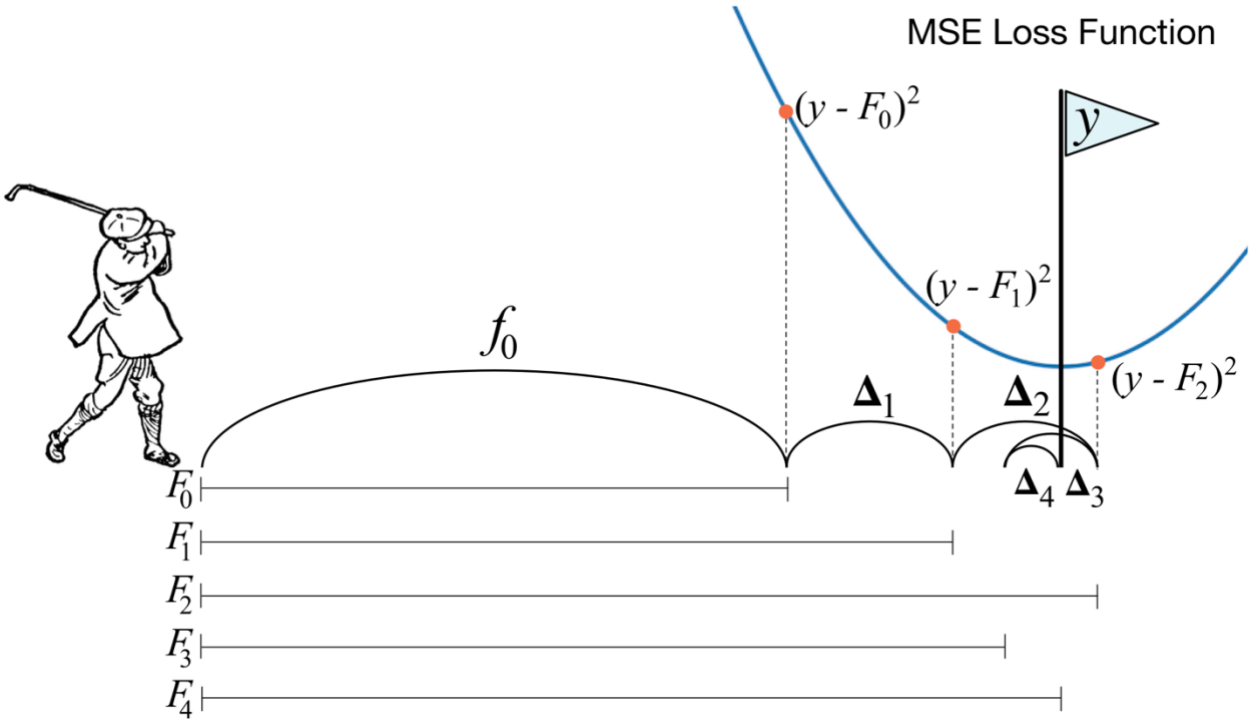



Рисунок 1.2 – Візуальний приклад бустингу

Будемо вважати, що ми вже побудували класифікатор $F_{m-1}$ довжини $m-1$. Таким чином задача зводиться до пошуку пари найбільш оптимальних параметрів $\{a_m, b_m\}$ для класифікатора довжини $m$:

$$\mathcal{Q} = \sum_{i=1}^{N} L\big(y_i, F_m(x_i)\big) \to min. \tag{1.13}$$

Функціонал помилки $\mathcal{Q}$ – речова функція, що залежить від точок $\{F_m(x_i)\}_{i=1}^N$ в $N$-вимірному просторі, і необхідно вирішити задачу мінімізації цього функціонала. Зробимо це, реалізуючи один крок методу градієнтного спуску. В якості точки, для якої ми шукаємо оптимальне збільшення, розглядаємо $F_{m-1}$. Знайдений градієнт функціоналу помилки:

$$\nabla\mathcal{Q} = \left[\frac{\partial \mathcal{Q}}{\partial F_{m-1}}\right]_{i=1}^{N} = \left[\frac{\partial\left(\sum_{i=1}^{N} L(y_i, F_{m-1})\right)}{\partial F_{m-1}}(x_i)\right]_{i=1}^{N} = \left[\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}\right]_{i=1}^{N}. \tag{1.14}$$

Таким чином, в силу методу градієнтного спуску, найбільш вигідно додати новий доданок в класифікатор наступним чином:

$$F_m = F_{m-1} - b_m \nabla Q, b_m \in \mathrm{R}, \tag{1.15}$$

де $b_m$ підбирається лінійним пошуком по речовим числах R:

$$b_m = \underset{b \in \mathrm{R}}{arg\min} \sum_{i=1}^{N} L(F_{m-1}(x_i) - b_m \nabla Q_i). \tag{1.16}$$

Однак $\nabla Q$ вдає із себе лише вектор оптимальних значень для кожного об'єкта $x_i$ а не базовий алгоритм з сімейства $H$ , певний $\forall x \in X$. Тому необхідно знайти $h(x, a_m) \in H$ схожій на $\nabla Q$. Знову мінімізуючи функціонал помилки, заснований на принципі явною максимізації відступів:

$$a_m = \underset{a \in \mathrm{A}}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(\nabla Q_i, h(x_i, a)\big) \equiv \text{навчити}\big(\{x_i\}_{i=1}^{N}, \{\nabla Q_i\}_{i=1}^{N}\big), \tag{1.17}$$

що просто відповідає базовому алгоритму навчання.

Коефіцієнт $b_m$ знаходится використовуючи лінійний пошук:

$$b_m = \underset{b \in \mathrm{R}}{arg\min} \sum_{i=1}^{N} L\big(F_{m-1}(x_i) - bh(x_i, a_m)\big). \tag{1.18}$$

Алгоритм::

Вхід: $\{x_i\}_{i=1}^{N}, \{y_i\}_{i=1}^{N}, M$

Вихід: $F_m(x)$

1. $F_0(x) = learn\big(\{x_i\}_{i=1}^{N}, \{y_i\}_{i=1}^{N}\big)$.
2. для $m = 1, M$.

3. $\nabla Q = \left[\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}(x_i)\right]_{i=1}^{N}$.

4. $a_m = \text{навчити}\left(\{x_i\}_{i=1}^{N}, \{\nabla Q_i\}_{i=1}^{N}\right)$.

5. $b_m = \underset{b \in R}{argmin} \sum_{i=1}^{N} L\left(F_{m-1}(x_i) - bh(x_i, a_m)\right)$.

6. $F_m(x) = F_{m-1}(x) - b_m h(x, a_m)$.

### 1.2.2 XGBoost

XGBoost – алгоритм машинного навчання, в основі якого знаходиться дерево пошуку рішень, він використовує алгоритм градієнтного бустингу. У завданнях передбачення, які використовують неструктуровані дані (наприклад, зображення або текст), штучна нейронна мережа вигідно відрізняється від інших алгоритми або фреймворки. Та коли задача доходить до структурованих або табличних даних невеликих за розміром, перші за коефіцієнтом ефективності знаходяться алгоритми засновані на дереві пошуку рішень [1].

XGBoost – один із ансамблів методів пошукових дерев, які використовують принцип бустингу навчання на слабких композиціях (найчастіше, алгоритм побудови бінарного дерева рішень) за допомогою архітектури градієнтного спуску [2]. Про ефективність та масштабування XGBoost було досліджено в роботі Тенці Чженом та Карлосом Ґестріном [3].

Він підтримує всі можливості таких бібліотек як scikit-learn з можливістю додатково додати регуляризацію. Підтримує три головні форми найшвидшого бустингу:

- стандартний градієнтний бустинг з можливістю зміни швидкості навчання;
- стохастичний градієнтний бустинг з можливістю семплірування по рядках і колонках вхідних даних;
- регуляризація градієнтного бустинг з L1 і L2 регуляризації.

Основна системна функція бібліотеки надається для використання в різних обчислювальних середовищах:

- паралельних процесів побудови дерева з використанням всіх доступних ядер та потоків процесора в момент навчання;
- розподілені обчислення для навчання великих моделей з використанням кластера машин;
- обчислення для великих наборів даних, які не вписуються в пам'ять;
- кеш оптимізація структури даних і алгоритму для найкращого використання апаратного забезпечення.

Основна особливість реалізація алгоритму це розроблення для ефективності обчислювальних ресурсів часу та пам'яті. Мета проекту полягала в тому, щоб найкращим чином використовувати наявні ресурси для навчання моделі. Деякі ключові функції реалізації алгоритму включають:

- різні стратегії обробки пропущених даних;
- блокова структура для підтримки навчання дерев в різних потоках;
- динамічне та просте навчання на нових даних в додатку до старих.

## 1.3 Метод проміжного представлення для прогнозування часових рядів

У процесі перетворення коду компілятори часто використовують проміжне представлення вихідної моделі, призначене насамперед для зручності генерації та роботи з моделлю, іноді на цьому етапі проводять різні оптимізації. Сама форма проміжного представлення залежить від цілей для яких її будуть використовувати в системі.

Найчастіше проміжне представлення використовують в формах де є орієнтований граф (наприклад, абстрактне синтаксичне дерево, в тому числі атрибутувати), трьох адресний код (у вигляді трійок або четвірок), префіксний та постфіксний запис.

Спостерігається система функцій дискретного аргументу $\left(f_i^{(k)}\right)_{i=1}^{N}$, де $k = 1, \ldots, s$. $s -$ кількість часових рядів, $k -$ номер ряду, $N -$ довжина часового ряду,

$i-$ номер ітерації. Потрібно розкласти ряд в суму компонент (використовуючи метод головних компонент), інтерпретувати кожну компоненту та побудувати продовження ряду $\left(f_i^{(k)}\right)_{i=1}^{N+M}$ по вибраній кількості компонент.

Для початку розглянемо одновимірний часовий ряд $(f_i)_{i=1}^{N}$. Візьмемо $n$ таке, щоб $0 < n \leq N-1$ – - час життя багатовимірної гусениці. Нехай $\sigma = N - n + 1$ – довжина гусениці. Розглянувши послідовність з $n$ векторів $R^{\sigma}$ наступного виду:

$$Y^{(i)} \in R^{\sigma}, \tag{1.19}$$

$$Y^{(i)} = (f_{i+l-1})_{i=1}^{\sigma}, \tag{1.20}$$

Визначимо:

$$Z = \left(Y^{(1)}, \ldots, Y^{(n)}\right): \tag{1.21}$$

Будемо називати $Z$ нецентрованою матрицею спостережень, породженої гусеницею з часом життя $n$. У разі багатовимірного часового ряду матрицею спостереження називається стовпець з матриць спостережень, відповідних кожної з компонент. Проведений аналіз головних компонент може проводитися як по центровій, так і по нецентрованій вибірці. Для спрощення викладок будемо розглядати найпростіший нецентрованого варіант.

Розглянемо ко варіаційну матрицю отриманої вибірки:

$$C = \frac{1}{n} Z Z^T. \tag{1.22}$$

Виконаємо для неї svd-розкладення:

$$C = V \Lambda V^T, \tag{1.23}$$

де $\Lambda = diag(\lambda_1, \ldots, \lambda_T) -$ діагональна матриця своїх чисел, $V = \left(v^{(1)}, \ldots, v^{(\tau)}\right)$, $\left(v^{(i)}\right)^T v^{(j)} = \delta_{ij} -$ ортогональна матриця власних векторів.

Після чого розглянемо систему компонент:

$$U = V^T Z, U = \left(U^{(1)}, \ldots, U^{(\tau)}\right)^T. \quad (1.24)$$

Після проведення аналізу головних компонент передбачається проведення операції відновлення вихідної матриці спостережень на деякому наборі головних компонент, для $V' = \left(v^{(i_1)}, \ldots, v^{(i_\tau)}\right)$ та $U' = V'^T Z$ розраховується матриця $Z' = V'U'$. Далі відновлюються вихідні послідовності. В одновимірну випадку і-у компоненту відновленого ряду, що є середнім значенням по і-ій діагоналі відновленої матриці спостережень $Z'$.

У багатовимірному випадку усереднення проводиться з урахуванням того, що матриця спостережень складається з підматриць, відповідних кожної компоненті ряду:

$$f_m'^{(k)} = \begin{cases} \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} x_i^{(m-i+1,k)} & 1 \le m \le \sigma, \\ \dfrac{1}{\sigma}\displaystyle\sum_{i=1}^{\sigma} x_i^{(m-i+1,k)} & \sigma \le m \le n, \\ \dfrac{1}{N-m+1}\displaystyle\sum_{i=1}^{N-m+1} x_{i+m-n}^{(m-i+1,k)} & n \le m \le N. \end{cases} \quad (1.25)$$

Часовий ряд $(f_i)_{i=1}^{N+1}$ називають продовженням ряду $(f_i)_{i=1}^{N}$, якщо породжувана ним під час гусеничної обробці вибірка лежить в тій же гіперплощини, що і вихідного ряду. Нехай у нас є певний набір обраних головних компонент $i_1, i_2, \ldots, i_\tau$. Визначимо:

$$w = \begin{pmatrix} v_\sigma^{(i_1)} & \cdots & v_\sigma^{(i_\tau)} \\ \vdots & \ddots & \vdots \\ v_\tau^{(i_1)} & \cdots & v_\tau^{(i_\tau)} \end{pmatrix}, \quad (1.26)$$

$$V_* = \begin{pmatrix} v_1^{(i_1)} & v_1^{(i_2)} & \cdots & v_1^{(i_\tau)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{\sigma-1}^{(i_1)} & v_{\sigma-1}^{(i_2)} & \cdots & v_{\sigma-1}^{(i_\tau)} \\ v_{\sigma+1}^{(i_1)} & v_{\sigma+1}^{(i_2)} & \cdots & v_{\sigma+1}^{(i_\tau)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{2\sigma-1}^{(i_1)} & v_{2\sigma-1}^{(i_2)} & \cdots & v_{2\sigma-1}^{(i_\tau)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{\tau-1}^{(i_1)} & v_{\tau-1}^{(i_2)} & \cdots & v_{\tau-1}^{(i_\tau)} \end{pmatrix}. \quad (1.27)$$

Також покладемо:

$$Q = \left(f_{N-\sigma+2}^{(1)}, \ldots, f_N^{(1)}, f_{N-\sigma+2}^{(2)}, \ldots, f_N^{(2)}, \ldots f_{N-\sigma+2}^{(s)}, \ldots, f_N^{(s)}\right)^T. \quad (1.28)$$

Тоді прогнозовані значення системи в точці $N+1$ обчислюється за формулою:

$$f_{N+1} = w(V_*^T V_*)^{-1} V_*^T Q. \quad (1.29)$$

## 1.4 Локальні методи прогнозування часових рядів

В підрозділі досліджується метод прогнозування часових рядів. Всі методи побудови прогнозу часових рядів можна розділити на локальні і глобальні. Глобальні методи використовують всю відому історію для побудови прогнозу. Локальні використовують тільки частина історії. Ми досліджуємо алгоритм, заснований на методі $k$ найближчих сусідів. Ми вирішуємо завдання налаштування параметрів алгоритму і параметрів метрики.

До параметрів алгоритму відносяться:

- число найближчих сусідів;
- метрика;
- довжина розглянутого вектора передісторії.

Задано кілька часових рядів, один з яких потрібно спрогнозувати на $h$ кроків вперед. Передбачається, що прогнозований відрізок лежить відразу ж за закінченням відомих нам значень рядів, тобто за кінцем історії. Будемо вважати, що в історії немає пропусків. Тоді об'єкт розглядатимемо як послідовності точок довжини $l$.

Потрібно знайти $k$ найбільш схожих об'єктів на закінчення історії в сенсі обраної метрики. Прогноз будується як лінійна комбінація прогнозів сусідів.

Переходимо до опису алгоритму. Основну складність становить визначення метрики, в сенсі якої визначається близькість об'єктів. Схема алгоритму:

1. Виділяємо об’єкт довжини $l$.

2. Нормалізуємо виділений об'єкт. Віднімаємо мінімум і ділимо на різницю максимального і мінімального значень за всіма даними рядах.

3. Далі, знаходимо відстань між виділеним і еталонним об'єктами:

$$dist(x, y) = \min_{A,b} \rho(x, Ay + b). \tag{1.30}$$

$$\rho(x, Ay + b) = \sum_{k=1}^{1} \omega^{2k} \big(x(k) - y(k)\big)^2 \tag{1.31}$$

або

$$\rho(x, Ay + b) = \sum_{k=1}^{1} \omega^{2k} |x(k) - y(k)| \tag{1.32}$$

У разі, коли ряд - багатовимірний, обчислюється метрика для кожного ряду зі своїм параметром $\omega$, та потім значення складаються. Параметр $\omega$ налаштовується в ході навчання моделі.

4. Якщо об'єкт опинився серед $k$ найближчих на даний момент, запам'ятовуємо отримані параметри і відстань. Викидаємо зі списку найдальший від еталонного об'єкт.

5. За результатами роботи виділяємо $k$ кращих об'єктів і виділяємо за ними ряд довжиною достатньою, розв’язати задачу пошуку історії.

6. З отриманих рядів будуємо опуклу лінійну комбінацію з рівними вагами.

7. Вибираємо ті елементи прогнозованого ряду, за якими необхідний прогноз.

Та розглянемо ще один цікавий метод - прогнозування подій. Це метод прогнозування подій на підставі інформації про наявність подій у тимчасових рядах, даної експертом. Спосіб породження і відбору ознак, що описують часовий ряд. Алгоритм заснований на розмітці інтервалів зростання і падіння тимчасового ряду і застосуванні логістичної регресії для налаштування параметрів лінійної моделі і оцінки її якості.

Дана вибірка часових рядів, розмічених експертом:

$$\{x^i, y^i\}, x^i = \left(x_1^i, \dots, x_T^i\right) \in R^t, i = 1 \dots l. \tag{1.33}$$

Тут цільова змінна:

$$y^i = \begin{cases} 1, & \text{в даному ряді є подія;} \\ 0, & \text{в даному ряді немає події.} \end{cases} \tag{1.34}$$

Необхідно ввести ознаковий опис тимчасового ряду:

$$g(x^i) = \left(g_1(x^i), \dots, g_n(x^i)\right). \tag{1.35}$$

На підставі опису (1.35) вирішуємо задачу класифікації - будуємо модель $f: R^w * R^T \to \{0, 1\}$, де $R^w$- простір параметрів моделі.

Розбиваємо задачу на три етапи.

1. Породження безлічі числових ознак $\left(g_1(x^i), \dots, g_n(x^i)\right)$, що описують часовий ряд;

2. Пропозиція критерію якості моделі;

3. Вибір найкращої моделі.

Зафіксуємо безліч міток $M = \{'U', 'D'\}$, де $'U'$ - підвищення значення ряду, $'D'$ - зниження значення ряду. Розмічаємо тимчасової ряд $(m_1, \dots m_T)$:

$$m_k = \begin{cases} 'U', & \text{якщо } x_k - x_{k-1} > 0, \\ 'D', & \text{якщо } x_k - x_{k-1} \leq 0. \end{cases} \quad (1.36)$$

Таким чином, для кожного часового ряду $x_i$ отримуємо рядок з $(T-1)$ символів $'U'$ або $'D'$: $m^i = m_1^i \dots m_{T-1}^i$ - слово довжини $(T-1)$ в алфавіті $M$. Таких слів $N = (2^w - 1)$. Нумеруємо їх: $R = (r_1, \dots r_N)$. Слово $r_k = r_k^1 \dots r_k^{n_k}$ та поставимо у відповідність 2 числових ознаки.

1. Дійсна ознака:

$$g_{2k}(x_i) = \begin{cases} x_{j+n^k}^i - x_j^i, & \text{є підстрока } r_k\text{: } (m_{j+1}^i, \dots m_{j+n_k}^i) = (r_k^1 \dots r_k^{n_k}), \\ 0, & \text{немає підстроки } r_k. \end{cases} \quad (1.37)$$

2. Бінарна ознака:

$$g_{2k-1}(x_i) = \begin{cases} 1, & \text{якщо в строці } m^i \text{ є підстрока } r_k, \\ 0, & \text{якщо в строці } m^i \text{ немає підстроки } r_k. \end{cases} \quad (1.38)$$

Якщо комбінація зустрічається в ряді кілька разів, то ми розглядаємо тільки її останнім за часом входження (і записуємо в дійсний ознака сумарне зміна ціни на ньому). Таким чином, ми отримуємо $2N = 2(2^w - 1)$ числових ознак і матрицю об'єкт-ознака розміру $l * 2N$:

$$G = \begin{bmatrix} g_1(x^1) & \cdots & g_{2N}(x^1) \\ \vdots & \ddots & \vdots \\ g_1(x^l) & \cdots & g_{2N}(x^l) \end{bmatrix}. \quad (1.39)$$

Будувати класифікатор і оцінювати якість моделі будемо за допомогою логістичної регресії. За оцінку якості моделі приймемо площа під ROC-кривої даного класифікатора. Відбір найкращої моделі будемо проводити за допомогою генетичного алгоритму. Фіксуємо парне число - довжину моделі n (кількість ознак в ній). Раніше ми побудували безліч слів $R = (r_1, \dots r_N)$ довжини не більше $w$ в алфавіті $M$ і кожному слову $r_k$ підставили відповідну ознаку $g_{2k-1}$ та $g_{2k}$. Активне безліч ознак $A$ - ознаки, на підставі яких будується поточна модель, $|A| = n$. Поставимо йому у відповідність активну безліч індексів $I = \left\{i_1, \dots, i_{\frac{n}{2}}\right\}, i_k \in \{1, \dots N\}, |I| = \frac{n}{2}$, з цього випливає:

$$I = \left\{i_1, \dots, i_{\frac{n}{2}}\right\} \Leftrightarrow A = \left\{g_{2i_1-1}, g_{2i_1}, \dots, g_{2i_{\frac{n}{2}}-1}, g_{2i_{\frac{n}{2}}}\right\}. \quad (1.40)$$

Таким чином, ознаки $g_{2k-1}$ та $g_{2k}$, завжди використовуються або не використовуються в моделі разом. Модель повністю описується активним безліччю індексів, тому далі генетичний алгоритм описуємо в термінах цих множин.

Генетичний алгоритм містить наступні параметри для відбору моделей: $F$-число кращих моделей в популяції, $F_1$ – число моделей для схрещування, $P_2$ – ймовірність вибрати модель для мутації.

Початкове безліч моделей (популяцію) вибираємо випадковим чином. Ділимо все безліч індексів $\{1, \dots N\}$ випадковим чином на підмножини розміру $\frac{n}{2}$ (для простоти вважаємо, що $2N$ ділиться на $n$):

$$\{1, \dots N\} = \left\{\left(k_1, \dots, k_{\frac{n}{2}}\right), \left(k_{\frac{n}{2}}, \dots, k_n\right), \dots\right\} = \left(I_1, \dots, I_p\right), \quad (1.41)$$

де $P = \frac{2n}{2}$ – розмір популяції.

Ітеративно виконуються наступні операції:

1. Відбір. Моделі упорядковано відповідно до убування якості (площа під ROC-кривої), і в наступне покоління потрапляють $F$ кращих.

2. Випадковим чином вибираються $F_1$ моделей для схрещування і мутації.

3. Схрещування. Вибрані моделі випадковим чином діляться на пари. У кожній парі активні безлічі індексів $I_p = \left(k_1^p, \ldots, k_{\frac{n}{2}}^p\right)$ та $I_q = \left(k_1^q, \ldots, k_{\frac{n}{2}}^q\right)$ розбиваються точкою кросинговера $m$, обраній випадково з безлічі $\left\{1, \ldots \frac{n}{2}\right\}$, на 2 частини. Відбувається обмін елементів множин $I_p$ та $I_q$ так, щоб в нових величезних кількостях не було однакових індексів:

$$\begin{cases} \left(k_1^p, \ldots, k_m^p, k_{m+1}^p, \ldots, k_{\frac{n}{2}}^p\right) \\ \left(k_1^q, \ldots, k_m^q, k_{m+1}^q, \ldots, k_{\frac{n}{2}}^q\right) \end{cases} \longmapsto \begin{cases} \left(k_1^q, \ldots, k_m^q, k_{m+1}^q, \ldots, k_{\frac{n}{2}}^q\right) \\ \left(k_1^p, \ldots, k_m^p, k_{m+1}^p, \ldots, k_{\frac{n}{2}}^p\right) \end{cases}. \quad (1.42)$$

4. Мутація. Кожна модель $I = \left(k_1, \ldots k_{\frac{n}{2}}\right)$ з отриманої безлічі з ймовірністю $P_2$ піддається мутації: випадковим чином з безлічі $\left\{1, \ldots \frac{n}{2}\right\}$ вибирається число $j$, та значення $k_j$ змінюється на випадково обраний індекс з безлічі $\{1, \ldots N\}$. При цьому контролюється, щоб в новій безлічі індексів не було однакових індексів.

5. Оцінка отриманих моделей.

### 1.5 Метод гнучких найменших квадратів

Для початку опишемо метод найменших квадратів.

Метод найменших квадратів (МНК) - математичний метод, застосовуваний для вирішення різних завдань, заснований на мінімізації суми квадратів відхилень деяких функцій від шуканих змінних. Він може використовуватися для «вирішення» перевизначених систем рівнянь (коли кількість рівнянь перевищує

кількість невідомих), для пошуку рішення в разі звичайних (НЕ перевизначених) нелінійних систем рівнянь, для апроксимації точкових значень деякої функції. МНК є одним з базових методів регресійного аналізу для оцінки невідомих параметрів регресійних моделей за вибірковими даними, для початку визначається лінія тенденції, див. рис. 1.3.

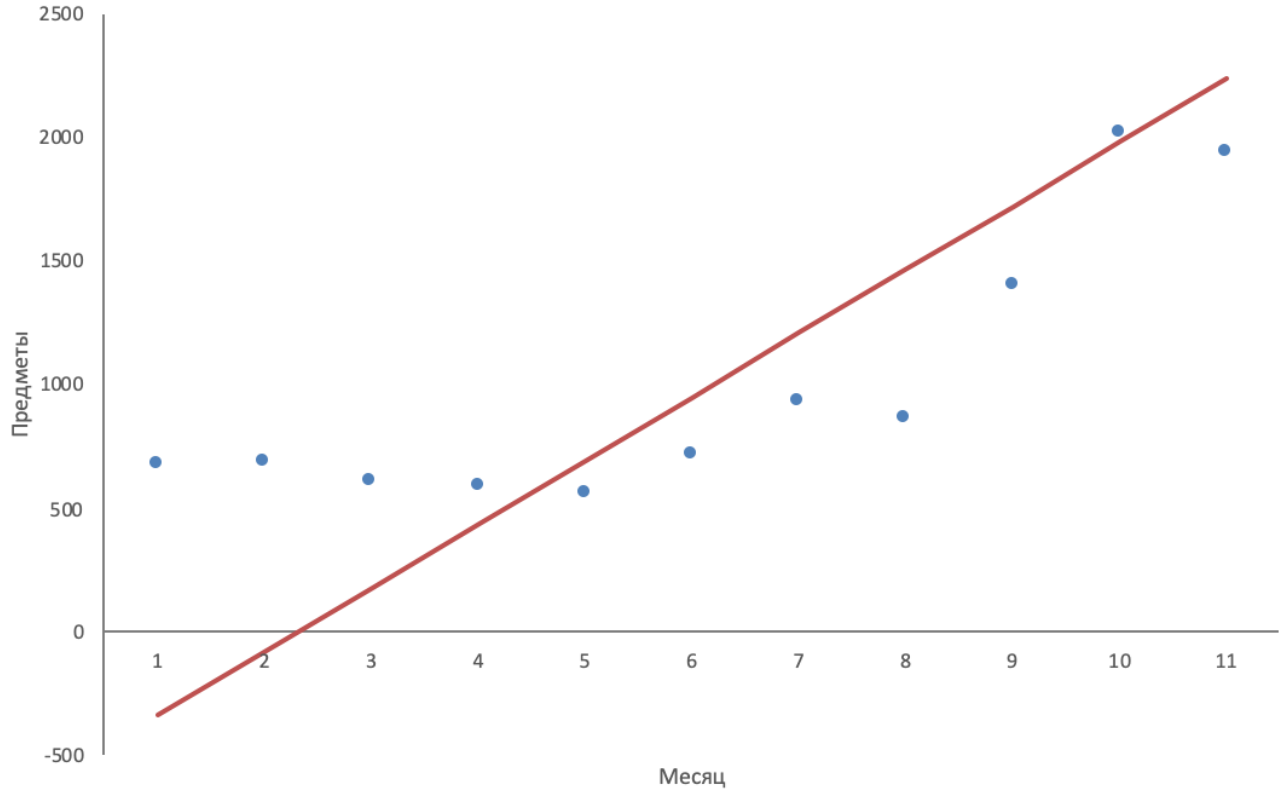



Рисунок 1.3 – Приклад лінії тенденції на графіку

Нехай $x$ – набір $n$ невідомих змінних (параметрів), $f_i(x)$, $i = 1,\ldots,m, m > n$ — сукупність функцій від цього набору змінних. Завдання полягає в підборі таких значень $x$, щоб значення цих функцій були максимально близькі до деяких значень $y_i$. По суті мова йде про «вирішенні» перевизначення системи рівнянь $f_i(x) = y_i, i = 1,\ldots,m$ в зазначеному сенсі максимальній близькості лівої і правої частин системи. Суть МНК полягає у виборі в якості «заходів близькості» - суми квадратів відхилень лівих і правих частин $|f_i(x) - y_i|$. Таким чином, сутність МНК може бути виражена таким чином:

$$\sum_i e_i^2 = \sum_i \big(y_i - f_i(x)\big)^2 \to \min_x. \tag{1.43}$$

У разі, якщо система рівнянь має розв'язок, то найменше значення суми квадратів дорівнюватиме нулю, і можуть бути знайдені точні рішення системи рівнянь аналітично або, наприклад, різними чисельними методами оптимізації. Якщо система перевизначена, тобто, кажучи не жорстко, кількість незалежних рівнянь більше кількості шуканих змінних, то система не має точного рішення і метод найменших квадратів дозволяє знайти певний «оптимальний» вектор $x$ в сенсі максимальній близькості векторів $y$ та $f(x)$ або максимальній близькості вектору відхилень $e$ до нуля (близькість розуміється в сенсі евклидова відстані).

В підрозділі розглянуто загальну постановку задачі та алгоритм розв'язання. За допомогою методу вирішується завдання відновлення нестаціонарної регресії. Нехай задана лінійна нормальна модель сигналу з прихованої компонентою:

$$\begin{aligned} x_t &= V_t x_{t-1} + \xi_t, \quad t = 2. \\ y_t &= c_t^T x_t + \eta_t, \quad t = 1. \end{aligned} \tag{1.44}$$

де $x_t \in \mathbb{R}^n$ - прихована компонента (векторів коефіцієнтів регресії), $y_t \in \mathbb{R}$ - спостерігається компонента (прогнозована змінна), $\xi_t \in \mathbb{R}^n$ – внутрішній шум, $\eta_t \in \mathbb{R}$ - зовнішній шум, $cov(\xi_t) = \frac{1}{\lambda} U_t^{-1}$, $\lambda$ – відношення інтенсивності внутрішнього та зовнішнього шуму (гіперпараметр моделі). Параметри $V_t \in \mathbb{R}^{n*n}, U_t \in \mathbb{R}^{n*n}$ – задаються апріорі, а $\mathrm{c}_t \in \mathbb{R}^n$ – значення багатовимірного часового ряду, на якому виконується прогнозування.

Алгоритм отримує на вхід вибірку $\{y_t, \mathrm{c}_t\}, t = 1 \dots T$, на виході отримаємо набір значень коефіцієнтів регресії $\{x_t\}$, $t = 1 \dots T$. Для оцінки коефіцієнтів використовується квадратична функція втрат:

$$J(x_1, \dots, x_T) = \sum_{t=1}^{n} \psi(x_t | y_t) + \lambda \sum_{t=2}^{n} \lambda_t(x_{t-1}, x_t). \tag{1.45}$$

де $\psi(x_t|y_t) = (y_t - c_t^T x_t)^2 = (x_t - x_t^0)^T Q_t^0 (x_t - x_t^0)$ – відхилення від моделі сигналу $y_t$,

$x_t^0 = \frac{y_t}{c_t^T c_t} c_t, Q_t^0 = c_t^T c_t$ – точка мінімуму та матриця квадратичної форми $\psi(x_t|y_t)$,

$\gamma_e(x_{t-1}, x_t) = (x_t - V_t x_{t-1})^T U_t (x_t - V_t x_{t-1})$ – відхилення від моделі прихованих змінних.

Ця функція втрат об'єднує 2 суперечливих вимоги до коефіцієнтів регресії $x_t$: точність відновлення сигналу $y_t$ (перший параметр) і гладкість зміни самих коефіцієнтів регресії (другий доданок). Компроміс між цими вимогами досягається вибором гіперпараметра $\lambda$.

Процедура оцінювання зводиться до застосування фільтра-інтерполятора Калмана-Бьюсі і виконується в 2 етапи:

1. Фільтрація:

1.1. $x_{1|1} = x_1^0, Q_{1|1} = Q_1^0, t = 1.$

1.2. $Q_{t|t} = Q_t^0 + \left( V_t Q_{t-1|t-1} V_t^{-1} + \frac{1}{\lambda} U_t^{-1} \right), t = 2 \dots T.$

1.3. $k_t = Q_{t|t}^{-1} c_t, t = 2 \dots T.$

1.4. $x_{t|t} = V_t x_{t-1|t-1} + k_t \left( y_t - c_t^T V_t x_{t-1|t-1} \right), t = 2 \dots T.$

2. Інтерполяція:

2.1. $H_t = \left( I + \frac{1}{\lambda} U_{t+1}^{-1} (V_{t+1}^{-1})^T Q_{t|t} \right)^{-1}, t = T - 1 \dots 1.$

2.2. $\hat{x}_t = x_{t|t} + H_t \left( \hat{x}_{t+1} - x_{t|t} \right), t = T - 1 \dots 1.$

Для оцінки гіперпараметра $\lambda$ використовується метод ковзаючого контролю $LOO$. У статті [4] отримані наступні співвідношення для значень параметрів $\hat{x}_t$ на підставі даних з пропущеним значенням $y_t$:

$$\hat{x}_t \left( Y^{(t)}, \lambda \right) = \left( Q_{t|T}^{(t)} \right)^{-1} \left( Q_{t|T} x_{t|T} - Q_t^0 x_t^0 \right), \tag{1.46}$$

де $Q_{t|T}^{(t)} = Q_{t|T} - Q_t^0, (t)$ означає викинуте вимір $y_t$;

$$Q_{t|T} = \left(H_t V_{t+1}^{-1} Q_{t+1|T}^{-1} (V_{t+1}^{-1})^T H_t^T + \left(Q_{t|t} + \lambda V_{t+1}^T U_{t+1} V_{t+1}\right)^{-1}\right)^{-1}. \quad (1.47)$$

Тоді зовнішній критерій $LOO$ обчислюється за визначенням:

$$LOO(\lambda) = J\left(\hat{x}_t\left(Y^{(1)}, \lambda\right), \ldots \hat{x}_t\left(Y^{(T)}, \lambda\right)\right). \quad (1.48)$$

При вирішенні задачі значення $\lambda$ вибирається з умови мінімізації зовнішнього критерію:

$$\hat{\lambda}_t = \underset{\lambda}{\operatorname{argmin}} LOO(\lambda). \quad (1.49)$$

Процедура має лінійної складністю по довжині $T$ часового ряду.

Для застосування алгоритму слід задати $U_t, t = 1 \ldots T$ (на практиці можна використовувати діагональні матриці), а також $V_t, t = 1 \ldots T$ , які визначають модель зміни прихованої компоненти. У вихідній роботі [5] використовуються одиничні матриці.

Застосування для прогнозування. Спочатку проводиться оцінка $\lambda$ за відомою історії часових рядів $hist$ . Далі виконується прогнозування в заданому часовому інтервалі $frc$.

1. Прогнозується $\mathrm{c_t}, \mathrm{t} \in \mathrm{frc}$ (за допомогою методу експоненціального згладжування).
2. Прогнозується попереднє значення $y_{\mathrm{t}|t-1}$ (за допомогою експоненціального згладжування).
3. FLS оцінює згладжений коефіцієнт регресії $x_{\mathrm{t}|t}$.
4. Обчислюється згладжені значення $y_{\mathrm{t}|t} = c_t^T x_{\mathrm{t}|t}$.
5. $\mathrm{t} \coloneqq t + 1$.

Реалізація вихідного коду [7].

# 2 РОБОТА НАД ВХІДНОЮ ВИБІРКОЮ

Мрія кожного розробника - очищені дані без помилок, пропущених значень і викидів, без пробілів або порожніх клітинок, необхідних для вирішення поставленого завдання. Однак, в реальності ми часто маємо справу з некоректною або часткової базою. А, оскільки ефективність побудованих моделей машинного навчання прямо пропорційно залежить від якості вихідного датасета, потрібно багато часу і зусиль на його підготовку до моделювання, саме в такому порядку: очищення, нормалізація і генерація змінних. При цьому в початковій вибірці можуть запросто бути відсутнім дані, потрібні для формування предикторов і цільових змінних. Таким чином, перед нами постане завдання заповнення датасета даними.

Перед тим, як застосовувати алгоритми машинного навчання, дані необхідно перетворити в табличне представлення для зручної роботи з ними.

Для машинного навчання потрібні вагові, якісні, правдиві, значущі значення, або дані, або ще їх називають змінними, які насправді впливають на підсумковий результат. Пошук, відбір і значимість даних один з головних моментів при роботі з машинним навчанням. Про нього далі поговоримо.

## 2.1 Фонові процеси для підготовки даних

Для того щоб продовжити говорити про дані, потрібно зрозуміти, а які види даних найчастіше трапляються в реальних додатках [9]:

- різнорідні (ознаки виміряні в різних шкалах);
- неповні (ознаки виміряні не всі, є пропуски);
- неточні (ознаки виміряні з похибками);
- суперечливі (об'єкти однакові, відповіді різні);
- надлишкові (надвеликі, не поміщаються в пам'ять);
- недостатні (об'єктів менше, ніж ознак);
- неструктуровані (немає знакових описів);

- «брудні» (помилкові, грубо не відповідають істині).

З усіма видами даних можна пробувати працювати, проводячи їх по процесам які описані в підрозділі нижче, але з «брудним» видом даних ми не можемо працювати, потрібно це розуміти.

Чим більше даних, тим краще, великий обсяг даних прийнято називати Big Data. Інтерес до Big Data ніколи не буде згасати, навпаки, зацікавленість людей все більше, особливо тим, як збираються, структуруються, аналізуються, ким обробляються і де застосовуватися. До бізнесу приходять величезні обсяги даних, які в наш час надходять фактично звідусіль, вони їм потрібні для маркетингу, ринку, контролювання ситуація та іншого. Це не дивно, адже за одну хвилину в Інтернеті з'являється тисячі нових сайтів, за день в світовому інформаційному просторі відбувається понад 5 млрд подій, які після алгоритмів обробки і стиснення все ця інформація займає близько 900 Гбайт, а кожен рік загальний обсяг одержуваних і збережених бізнес-даних потроюється.

Big Data - це величезний обсяг неструктурованих або структурованих даних, цінність яких залежить від того, де і для чого вони будуть застосуються [10]. У наш час все більше попит на великі дані для компаній, але все ж більшість завдань вимагає потребує малих обсягах даних. Великі дані – не завжди дають потрібний результат. Особливо важливою стає проблема наскільки дані якісні, так як разом з цінною інформацією генерується купа інформаційного сміття. Для хорошого аналізу і застосування цих дані в Big Data, потрібна хороша команда аналітиків, які зможуть отримати корисні знання з даних і правильно їх використовувати.

Пошук, відбір ознак або виділення - це процес обробки і відкидання не важливих змінних з датасета вибірки перед запуском алгоритму машинного навчання та інтелектуального аналізу даних (Data Mining). Є декілька причин при яким скорочення числа предикторів необхідно:

- швидкість обчислень - чим менше змінних, тим швидше будуть йти розрахунки;

- значимість ознак - зазвичай, вихідна вибірка завжди містить багато «сміттєвих даних»: шумів, викидів, аномалій, а на реальний результат впливають лише кілька предикторів;

- точність рішення - деякі моделі машинного навчання чутливі до величини вхідних параметрів. Наприклад, у нейромережах велике число вхідних даних може привести до перенавчання.

Процес перенавчання – основна фішка, вони можуть адаптуватися та перенавчатися на новоприбулих даних, для отримання надійних та репрезентативних результатів, і водночас проблема алгоритмів машинного навчання, за якої потрібно постійно стежити і контролювати. Адже вона: зменшує швидкість побудови моделі за даними, забирає ресурси системи, відбуватися зайва ітерація перенавчання алгоритму і не факт, що на виході отримаємо потрібну нам модель.

Data Mining не просто так виділяють підготовку даних як окрему фазу. Підготовка даних (Data Preparation) - вельми складний ітеративний процес, який займає приблизно 75% всіх ресурсів, а також часу в життєвому циклі Data Mining і проходить наступні завдання для обробки сирих даних [11]:

- вибірка даних - відбір об'єктів і ознак (features або предикторів) з урахуванням їх важливості для поставлених цілей, технічних обмежень (обсягу і типу) і якості;

- очищення даних - видалення помилок, неправильних значень (наприклад: рядок пошуку з набору числових параметрів та ін.), порожніх значень (позначаються як: Missing values або NA), видалення значень, які повторюються та різних описів одного і того ж об'єкта, відновлення унікальних об'єктів, цілісність та логічні зв'язки. Інакше кажучи на цьому етапі ми ставимо запитання для того щоб отримати відповіді: як потрібно підготувати дані, щоб використовувати їх максимально ефективно? Ми повинні позбутися помилкових даних, обробити відсутні записи, видалити дублікати і переконатися, що все відформатоване належним чином. Також на цьому етапі ми визначаємо набір ознак, на яких далі буде будуватися модель машинного навчання. Якщо

відбувається робота з текстом, потрібні будуть перевірки для додаткових кроків, скажімо перетворити неструктуровані дані в набір певних ознак, для використання в моделі. На практиці існує два способи очищення даних: автоматизована очищення даних за допомогою вбудованих засобів (СУБД для Big Data: Hive, Azure, SQL Server Data Tools та інші) або інтегрованих систем для статистичного аналізу(наприклад, IBM SPSS, SAS) та очищення даних власними силами, коли аналітик самостійно шукає готові або розробляє свої скрипти.

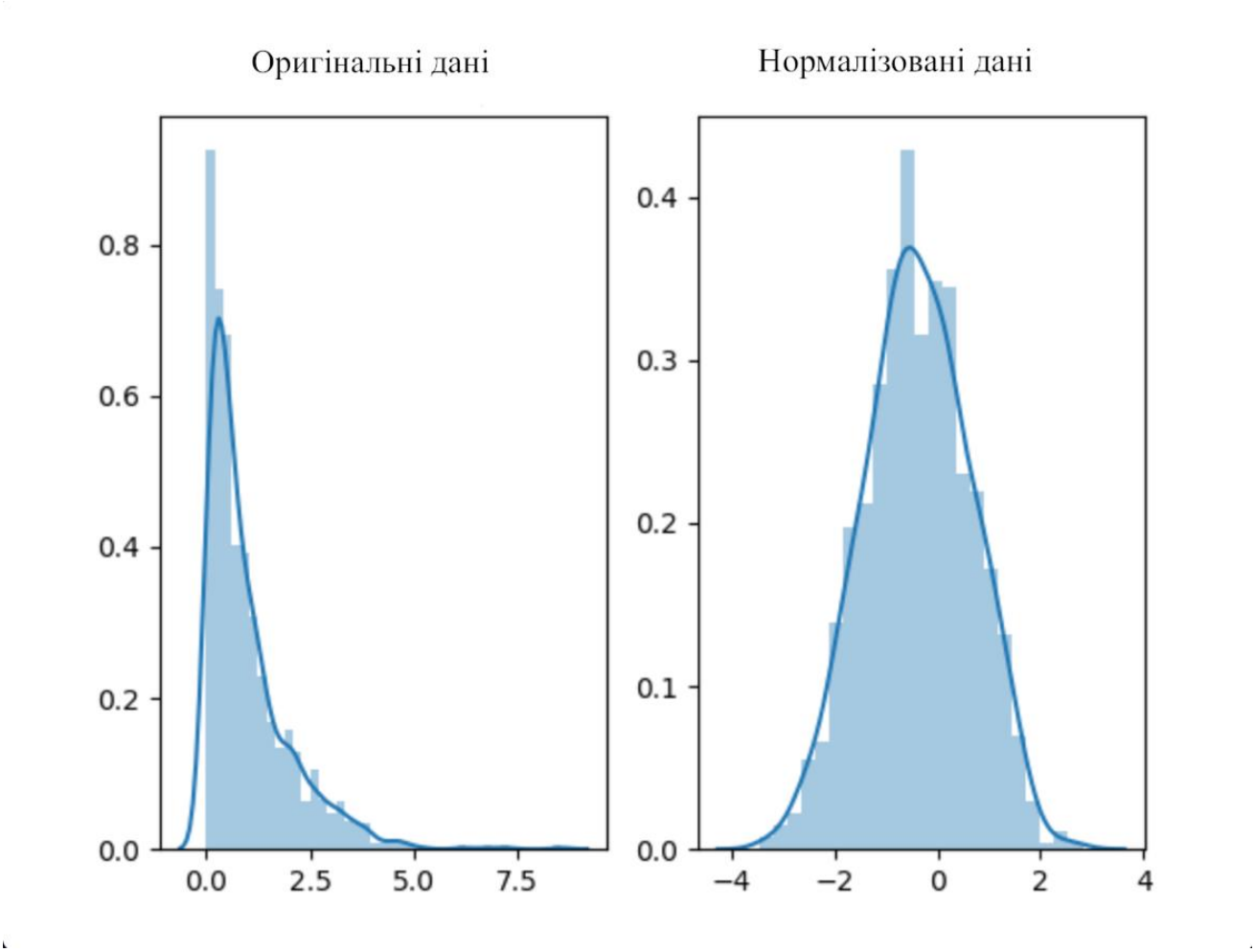



Рисунок 2.1 – Приклад нормалізованих даних [12]

Підготовка даних – це основний фундамент, на якому будуть будуватися наступні етапи [13]:

- генерація ознак – створення ознак та їх переформатування в вектори для моделі машинного навчання, а також трансформація для підвищеної точності алгоритмів машинного навчання;

- інтеграція – об'єднання даних з різних джерел (інформаційних систем, таблиць та інших), включаючи їх перетворення в нові дані, це процес, коли нові значення обчислюються шляхом підсумовування інформації з різних множин існуючих даних;

- форматування – зміна синтаксису, яка не змінює значення даних, а перетворює, вона потрібна для інструменту моделювання, наприклад, сортування тексту в певному порядку або видалення непотрібних знаків пунктуації в текстових полях, обрізка «довгих» слів, округлення дійсних чисел в більшу, меншу сторону або видавши ціле число, наприклад робота [14] і тому подібне.

Приклад стандартних проблем з даними, наприклад див. рис. 2.1, які навіть за ідеальних умов потрібно перевіряти і вирішувати:


Рисунок 2.2 – Графічний приклад неузгоджених даних

- неповнота – дані не містять атрибутів, або в них пропущені значення. Представимо приклад, якщо наші початкові данні мають пробіли в даних. Алгоритм не правильно побудує кінцеву модель машинного навчання. Передавши

данні компанії, хтось, наприклад бухгалтер або менеджер, зробила помилкові закупку із-за чого втратила мільйони, див. рис. 2.2, два значення на графіку 2013 року;

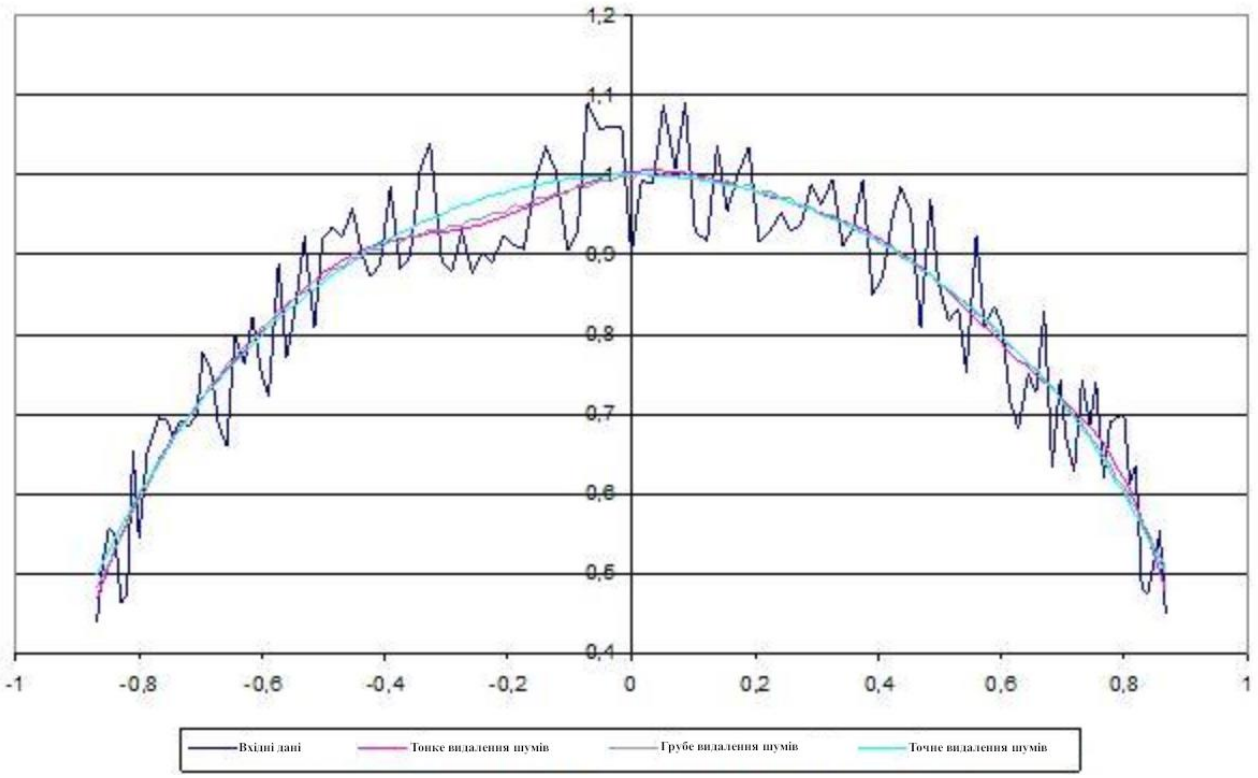



Рисунок 2.3 – Приклад шумів в даних та їх обробка

- шум – дані містять помилкові записи або викиди. Викиди можуть нам щось сказати про якість даних: наприклад, це може бути ознакою помилки: зміна розмірності, втрати знаку або кривої при кодуванні. Також вони можуть казати про те, наскільки сильно варіюються дані, також можуть сказати о граничних значеннях досліджуваних ознак, див. рис. 2.3;
- неузгодженість – дані містять конфліктуючі між собою записи або розбіжності. Наприклад: бухгалтер передав дані, на основі отриманих даних будуємо модель, віддаємо прогнозування по моделі замовнику, він каже що щось не так, по вашій моделі я в минулого року витрат n грошей, в вашій моделі 5*n. Розбіжність в даних можна вважати через помилково надані дані, див. рис. 2.4.


Рисунок 2.4 – Візуальний приклад неповних даних

Якісні дані – це обов’язкова умова для створення якісних моделей машинного прогнозування. Обов’язкова умова для уникання появи ситуації «сміття на вході, сміття на виході» що підвищить якість даних і, як наслідок, ефективність моделі. Необхідно проводити моніторинг працездатності даних, як можна раніше та виявити проблеми і вирішити їх, які дії по попередній обробці і очищенню даних необхідні.

Під моніторингом даних розуміємо сукупність методів і додатків, пов'язаних з алгоритмами обробки даних, вони не мають чітко зафіксованої відповіді на кожен вхідний об'єкт. Від конкретної реалізації класичного алгоритму залежить час його виконання і обсяг займаної пам'яті, але очікуваний результат його застосування строго зафіксований. На противагу цьому ми очікуємо від моделі, відповіді при , наприклад, вхідному зображенні, що зображає рукописну цифру, але не можемо вимагати саме ту цифру яка написана. Більш того, будь-яка (в розумному сенсі цього слова) модель буде іноді помилятися на тих чи інших варіантах коректних вхідних даних.

## 2.2 Аналіз та підготовка вхідних даних

Перш ніж почнемо аналізувати та підготовляти дані, розберемося, які уміння потрібно мати для аналізування даних.

Наука про дані – велика сфера, яка поєднує декілька сусідніх дисциплін. Це програмування, вища математика, статистика, бізнес-аналітика та машинне навчання.

Фахівці в цій сфері, аналітики даних, працюють з великими об'ємами масивів даних, аналізуючи їх та беручи з них тільки корисну інформацію. Результат дає відповіді на безліч питань: наприклад, чому один фахівець має більшу ефективність при роботі з клієнтами ніж інший, скільки одиниць товару потрібно закупити в наступному кварталі або які продукти потрібно додавати до рецепту для посилення смаку. Для вирішення деяких видів завдань фахівці розробляють алгоритми, які здатні генерувати результат без участі людини на протязі *n* років, рідкість але можливо, якщо правильно створити алгоритм, передавати в нього правильні коефіцієнти та отримувати ідеальні, чисті вхідні дані.

У нам потрібно буде розбиратися з деякими темами в:

- лінійній алгебрі – основи роботи з векторним і матричним представленням даних, операції над матрицями;
- математичний аналіз – для оптимізації моделі і алгоритму, розуміння, де можна оптимізувати модель, дописати формулу до неї, щоб вона працювала краще і швидше;
- теорія ймовірності і статистика – використовується майже у всіх алгоритмів машинного навчання, проведення аналізу даних і адекватного проведення a/b-тестів. Допоможе оцінити без сторонніх сервісів, наскільки взагалі можна покладатися на наявні дані, як працювати з шумами та аномаліями в даних, які можуть зіпсувати всю кінцеву модель;
- використання бібліотеки NumPy та інших бібліотек в мові програмування Python для обчислень і побудови моделей. Для математичних і

статистичних обчислень та перетворень, вже повсюдно використовують можливості Python, а для роботи з машинним навчанням знання цих інструментів необхідно.

Під аналітикою ми будемо називати широке використання баз даних, моделі з поясненнями, моделі побудовані на прогнозовані, статистичний аналіз та кількісний аналіз, а також менеджмент.

Залежно від мети і методів аналітику можна розділити на описову, передбачувану і нормативну.

- описова аналітика включає збір, систематизовану перевірку даних, подання даних у табличній формі, а потім знаходження потрібних характеристик. Даний вид аналітики орієнтується на інформуванні про характеристики даних. Він дуже корисний, але нічого не може сказати про причини поведінки, яка склалася або яка складеться в майбутньому;
- передбачувальна аналітика інакша, вона виходить за рамки звичайного опису даних і зв’язків між змінними (у вигляді коефіцієнтів, які можуть мати багато різних значень) і прогнозує динаміку значень на майбутнє на основі даних за минулі часи, дні, роки, не важливо. Спочатку визначаються зв'язку між змінними, можуть видаватися їм коефіцієнти ваги, а потім на основі цих даних аналізується та оцінюється ймовірність шуканої події: наприклад задача маркетингу, яка вірогідність, що користувач відреагує на банер на сайті і натисне клавішою миші на нього. Хоча зв'язку між змінними використовуються для прогнозування майбутнього, явна причинно-наслідковий зв'язок знаходяться не часто. Цей наслідок потрібен для отримання більш точного прогнозування;
- нормативна аналітика фокусується на ширший обсяг завдань і має такі методи: проведення експериментів і оптимізація. Подібно до того як програміст пише код алгоритму, нормативна аналітика пропонує напрямки дій. Експеримент потрібен для того щоб відповісти на питання про існування тих чи інших залежностей та даних. Для того щоб точно робити висновки про причинні зв'язки, фахівці змінюють одну чи кілька незалежних між собою змінних і спостерігають реакцію залежною змінною, водночас дивлячись на зовнішні чинники. Якщо

тестова група, що експериментуються, істотно показує кращі результати в порівнянні з групою для тестування, то відповідальний фахівець може прийняти рішення про імплементацію цієї умови в маси.

Залежно від наших цілей, цілей фахівця, можна застосовувати різні методи аналітики:

- прогнозування - оцінка потенціалу будь-якої змінної на момент в майбутньому на основі даних її динаміку в минулому;
- оптимізація - використання математичних методів для знаходження оптимальних рішень на основі заданих параметрів і обмежень;
- інтелектуальний аналіз даних (Data mining) - автоматизоване виявлення раніше пустих або глухих залежностей у великому об’ємі даних за допомогою спеціальних алгоритмів обчислювання або методів статистики;
- експеримент – формування групи даних для тесту та контролю методом випадкового взяття даних та знаходження причин і ступеня залежностей незалежних змінних на залежну змінну;
- інтелектуальний аналіз текстів (Text mining) - виявлення невідомих, сумнівних залежностей в даних або тенденцій в даних статистичними методами, подібними до інтелектуального аналізу даних;
- статистика це збір, оновлення, аналіз, інтерпретація і публікація даних. Давня проблема статистики полягає в тому, як найкраще використовувати вибірки, отримані з невідомих розподілів ймовірностей, щоб допомогти вирішити, з якого розподілу береться нова вибірка. Пов'язана проблема полягає в тому, як оцінити значення невідомої функції в новій точці, враховуючи значення цієї функції в наборі точок вибірки. Статистичними методами вирішення цих проблем можна вважати випадки машинного навчання, оскільки правила прийняття рішень та оцінки залежать від корпусу зразків, взятих із проблемного середовища [15].

В даному списку наведені приклади які широко відомі в аналітичних методах, деякі з них використовують однакові процедури та прийоми в аналітиці. Ми будемо використовувати прогнозування, алгоритм спрогнозує дані на наступний рік, інтелектуальний аналіз даних, в підрозділі ще проведемо

поверхневий аналіз даних, оптимізацію – основна проблема алгоритмів машинного навчання перенавчання та експеримент – перш ніж оприлюднити алгоритм, його потрібно протестувати на різних даних.

Важливий момент це відбір ознак - процес вибору ознак, що мають найбільш тісні взаємозв'язки з конкретною змінної.

Наявність в даних не потрібних значень призводить до зниження точності нашої моделі, особливо якщо вона лінійна, а саме модель лінійна або це логістична регресія.

Відбір ознак перед побудовою моделі дає нам наступні переваги [16]:

- зменшення перенавчання – менше порожніх, не потрібних даних, тим менше можливостей для моделі приймати рішення на основі незв’язних значень;
- підвищення точності. Менше спірних моментів - вище точність;
- скорочення часу навчання.

Існує 4 методи відбору ознак на мові Python. Нам не потрібно детально розбирати кожен метод відбору ознак, частина з них була вже описана в першому розділі про алгоритми, частина не складна. Але цими методами оперувався при відборі ознак для побудови моделі:

1. Одновимірний відбір ознак.

Ознаки, що мають найбільш значимий взаємозв'язок з цільової змінної, будуть відібрані за допомогою статистичних методів. Бібліотека scikit-learn містить клас SelectKBest, який реалізує одновимірний відбір ознак. Цей клас можна застосовувати спільно з різними статистичними критеріями для відбору заданої кількості ознак.

2. Рекурсивне виключення ознак.

Метод рекурсивного виключення ознак реалізується за допомогою наступного алгоритму: модель навчається на вхідних даних ознак і дає оцінку їхньої соціальної значимості, після чого з вибірки викидається одна або декілька найменш важливих ознак, модель далі навчається на залишившихся ознаках, і так поки, доки не залишиться задана кількість кращих ознак [17].

3. Метод головних компонент.

Суть роботи методу головних компонент описана в першому розділі.

4. Відбір на основі важливості ознак.

Ансамблеві алгоритми на основі дерев рішень, такі як випадковий ліс (описаний в першому розділі), дозволяють оцінити важливість ознак.

При аналізі теми роботи було виявлено за допомогою методів пошуку ознак основні ознаки, які потрібні для подальшого написання алгоритму та формування моделі. Звісно ознаки, які не мали вагомо впливу на цілісні дані були відкинуті. Ознаки які залишилися і будуть використані перелічені в таблиці 2.1 в головних стовпцях таблиці.

Як вже було описано вище, вхідні дані – сирі. Звичайно, є датасети з даними готовими для побудови на основі них моделі, є датасети з підготовленими даними, які були попередньо оброблені, але зазвичай ці явища велика рідкість, складно знайти датасет з потрібним тобі ідеальними даними, особливо якщо вони не великого об’єму, специфічні та не популярні.

Для мого випадку, потрібні дані специфічні, рідкісні та малі по об’єму. Під малими по об’єму маю на увазі за останні, хоча б, 10 рокі. Дані на основі який буде алгоритм буде прогнозувати наступні роки будуть за останні 10 років.

Які конкретні данні, за останні 10 років, потрібні для цієї роботи?

Побудова алгоритму буде проходити на основі даних від вступної компанії Державного Університету Телекомунікацій. Конкретно ці дані публічні. Можна знайти їх на просторах сайту університету, в них є електроні табличні подробиці вступної компанії за останні 2 роки, нажаль, не вони підходять, потрібно більше даних. Проблема пошуку даних вступної компанії університету гостріша з кожним датасет сайтом.

Таки знайшовся сайт з потрібними даними, який покриває мінімальний об’єм для нашої задачі, це «Сервіс пошуку абітурієнтів» (посилання на сайт – «abit-poisk.org.ua»). Всі дані для аналізу та подальша робота з ними будуть взяті з цього сайту, нажаль, є аномальні значення, про них буде розписано нижче, потрібно буде пройти етапи фонової обробки даних, які описані у підрозділі 2.1. Перенесемо дані в табличне представлення для зручної роботи з ними.

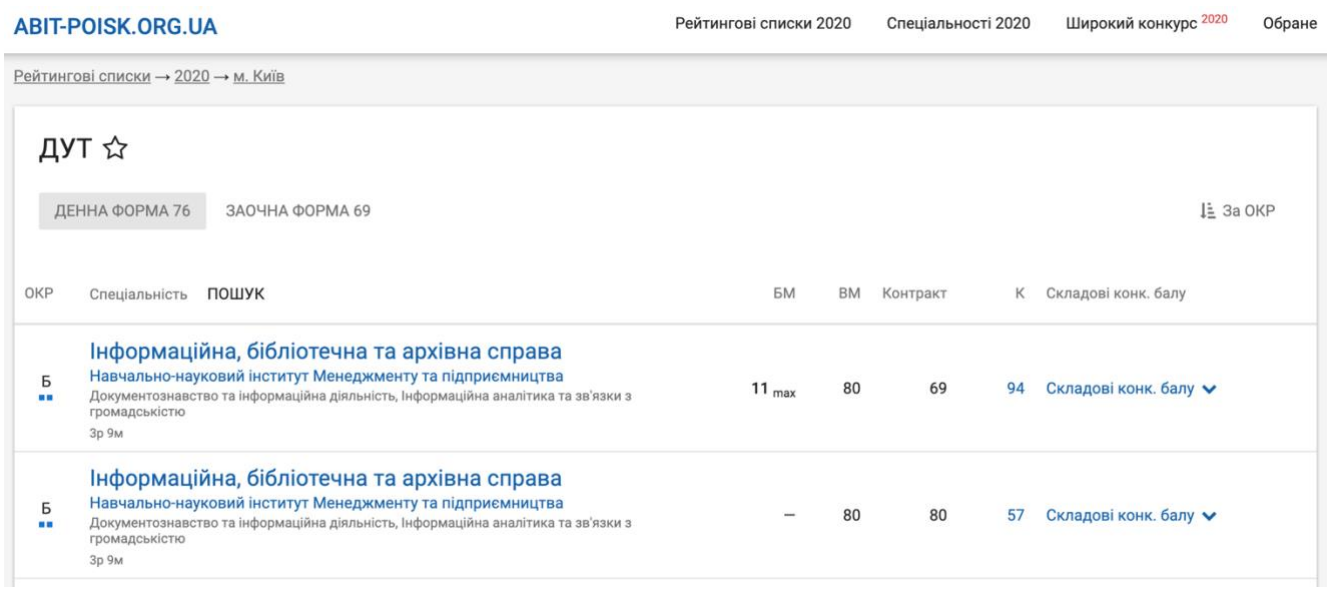



Рисунок 2.5 – Сайт на якому знаходяться потрібні дані

Дані представлені в вигляді, як представлено на рис. 2.5, ці дані потрібно перенести в табличний варіант, для початку в Excel, для зручного поверхневого аналізу та графічного представлення вхідних даних, після чого потрібно їх зберегти в файлі з форматом cvm та за допомогою бібліотеки завантажити в наш сформований датасет та наш алгоритм.

Дані в табличному вигляді представленні в таблиці А.1. На сайті представлено дані на бакалавра та магістратура, також до кожної дані по денній формі навчання та заочній. Вхідні дані на магістратуру сильно пошкодженні або не вносились, робити аналіз на такій кількість та якості даних, а особливо прогнозування по ним не має ніякого сенсу. Передивившись вхідні дані в таблиці ми бачимо багато значень виду «n\a» - це пусті данні, пробіли, биті дані, не має значення як ми їх будемо називати. Суті не змінює, потрібно прийняти рішення що з ними роботи, оскільки вони мають вплив на кінцеву побудову моделі та прогнозування.

Відразу відкидаємо значення заочної форми навчання, вхідних даних недостатньо для аналізу та побудови моделі. На цьому етапі ми визначили що нам цікаві лише дані від абітурієнтів при вступі на денну форму навчання бакалавру.

Переглянувши дані прийому абітурієнтів на другий та третій курс денної форми навчання можемо бачити що в даних або немає значень або їх замало,

нажаль дані за другий та третій курс приймальної комісії ми теж не можемо використовувати.

Як видно на таблиці А.1, в сформованих даних попадаються значення «n\a», які мають цільове значення для подальшої обробки. Приймаємо рішення, що дані текстові значення в комірках таблиці будемо замінювати на нулі, нулі нічого не важать і не вплинуть на кінцеву модель машинного навчання.

В публічному доступі на сайті можна зберегти не тільки дані представлені в таблиці А.1, також нам доступні особисті дані абітурієнтів та їх рейтингове положення при вступі на спеціальність до вищого навчального закладу. Ці дані не мають для нашої роботи ніякого цільового значення, тому не були використані при збережені.

Також бачимо дублі перших курсів, в даних інших років буде багато таких випадків, будемо вирішувати їх однаково для всіх, агрегуємо їх, для більш точного групування.

Розберемо, які дані ми отримаємо для аналізу на виході:

- 2010 рік - 2020 рік вступної компанії Державного Університету Телекомунікацій;
- абітурієнти, які прийшли до університету і хочуть вступити на перший курс денної форми навчання на бюджет або контракт одного із направлень університету;
- кількість абітурієнтів, які поступили на бюджет. В Державному Університеті Телекомунікацій динамічна зміна кількості студентів в направленні на бюджет;
- кількість контрактників – кількість абітурієнтів, які поступили до університету не перший курс та віддали всі необхідні дані для їх подальшого оформлення;
- кількість абітурієнтів, які поступили до університету, це значення дорівнює сумі кількості абітурієнтів та кількості контрактників;
- усього – запланована кількість абітурієнтів університетом;

- заяв – кількість заяв поданих до університету абітурієнтами з метою вступу та подальшого навчання.

Також для розрахунків нам потрібні середні статистичні дані кількості заяв, які було відхилено через не вірно оформлені заяви, не правильні передані документи, помилкові документи, тощо. В середнього така кількість заяв 10 на 1000 заяв на вступній компанії.

Таблиця 2.1 – Підсумкова таблиця отриманих даних про вступну компанію з 2010 року по 2020 рік

| | Бюджет | Контракт | Всього прийнято | Всього місць | Заяв |
|---|---|---|---|---|---|
| 2010 | 179 | 503 | 682 | 690 | 2851 |
| 2011 | 179 | 511 | 690 | 690 | 2301 |
| 2012 | 235 | 375 | 610 | 610 | 4447 |
| 2013 | 232 | 358 | 590 | 590 | 5269 |
| 2014 | 218 | 347 | 565 | 565 | 4822 |
| 2015 | 214 | 511 | 725 | 715 | 4728 |
| 2016 | 209 | 729 | 938 | 975 | 5846 |
| 2017 | 22 | 849 | 871 | 2362 | 5709 |
| 2018 | 340 | 1070 | 1410 | 1410 | 4276 |
| 2019 | 283 | 1737 | 2020 | 2050 | 5319 |
| 2020 | 281 | 1661 | 1942 | 1980 | 5588 |

Таблиці з кінцевими даними будуть прикріплені до додатку А. Зараз розглянемо підсумкову таблицю. В якій будуть дані:

- по рокам, від 2010 до 2020;
- кількість бюджетних місць по рокам;
- кількість контрактних місць, підсумкова кількість абітурієнтів які поступили до університету;
- кількість місць для бакалаврів в університеті;
- кількість заяв від абітурієнтів.

Вже з першого погляду на табличне представлення ми бачимо аномальне значення в колонці бюджетних місць. Це значення за 2017 рік, в 2017 році кількість абітурієнтів, які поступили до Державного Університету Телекомунікацій на бюджетне місце склала 22 людини. Якщо не нормалізувати дане число і віддати ці дані алгоритму машинного навчання побудована модель на виході може спрогнозувати не вірне значення.

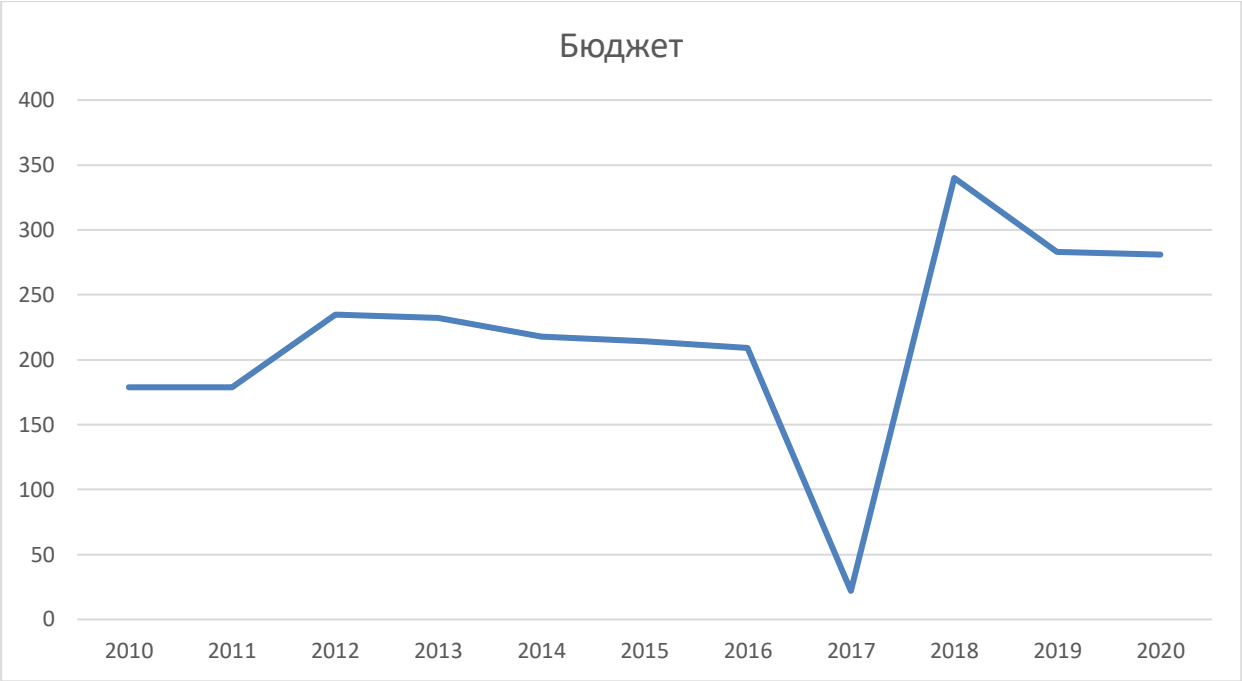



Рисунок 2.6 – Графік підсумкових даних по бюджетним місцям

Також бачимо скачок графіка у 2018 році, потрібно розбиратися з ними. Представимо такий варіант, у 2017 році було подано не вірне значення кількості абітурієнтів, які потрапили на бюджетні місця і 2018 році подали звіт в якому постаралися вирівняти загальну ситуацію інакше кажучи статистику. Людський фактор помилки ніхто не відміняв. Розраховуємо по формулі середнє значення бюджетних місць з 2010 року до 2020 року по формулі:

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.1}$$

де n – кількість років, в нашому випадку 11, $x_j$ - значення в даних.

Отримаємо значення $m \cong 217.46$. Отримане значення потрібно зрівняти з середньою сумою двох знайдених аномальних значень $\frac{x+y}{2m}$, де $x$ та $y$ – два аномальних значення за 2017 та 2018 роки. Отриманий результат – приблизно 0.83, що значно менше середнім значенням.

Будем вважати тільки значення за 2017 рік аномальним, а за 2018 рік реальним, схоже в 2018 році університету і правда виділили 340 місць.

Не можна викинути всі значення із таблиці за 2017 рік, кожен рік даних нам потрібен, хоча б для того що б побачити графік тенденції, тому потрібно нормалізувати значення за 2017 рік. Тому нормалізуємо графік.


Рисунок 2.7 – Нормалізований графік підсумкових даних по бюджетним місцям

Візьмемо формулу по знаходження середнього числа, яка була описана вище:

$$x_{2017} = \frac{1}{n-1}\sum_{i=1}^{n-1} x_i, \tag{2.2}$$

де n – кількість років, в цьому випадку 10, $x_j \in X, X \neq value$ 2017

$x_{2017} = 237$, отримане значення підставимо до таблиці та будуємо новий графік, див. рис. 2.7.

На новому графікі, який представлений на рис. 2.7 більше не залишилось порожніх даних, переходимо до наступного графіку.


Рисунок 2.8 – Графік кількості студентів, які поступили на контрактній основі

Аналізуючи дані, які пред'явлені на рис. 2.8 нам залишається тільки здогадуватись, що 2020 році щось вплинуло на результати, графік не може мати стабільну тенденцію росту і без якихось чинників перерости в тенденцію падіння. Скоріше за все це через COVID-19, який наклав певні рамки не тільки на світову економіку, а й на Державний Університет Телекомунікацій. Відхилення не є аномальним або шумовим, його не потрібно обробляти.

На рис. 2.9 побудований графік кількості абітурієнтів, які отримали можливість навчатися в Державному Університеті Телекомунікацій.

На графіку виділяються два роки. 2020 рік знову в порівняні з попереднім роком статистика нижче, не критично тому вважаємо значення нормальним. Та знову 2017 рік, графік «просить» щоб значення за 2017 рік було приблизно 1200, це не критичне відхилення можемо його поки ігнорувати, значення звичайних рамках, не виділяється сильно.


Рисунок 2.9 – Графік кількості прийнятих абітурієнтів

На рис. 2.10 явно бачимо аномальне значення, знову характерне для 2017 року та різницю розвитку тенденції графіка в порівнянні 2020 року та 2019 року, схоже є зовнішні або внутрішні фактори, які вплинули на статистику університету в 2020 році. Чи потрібно тут нормалізувати дані за 2017 рік, на перший погляд так, на другий ні. Графік проситься встановити значення 2017 року в межах 2016 та 2018, тобто від 975 до 1410 місць.

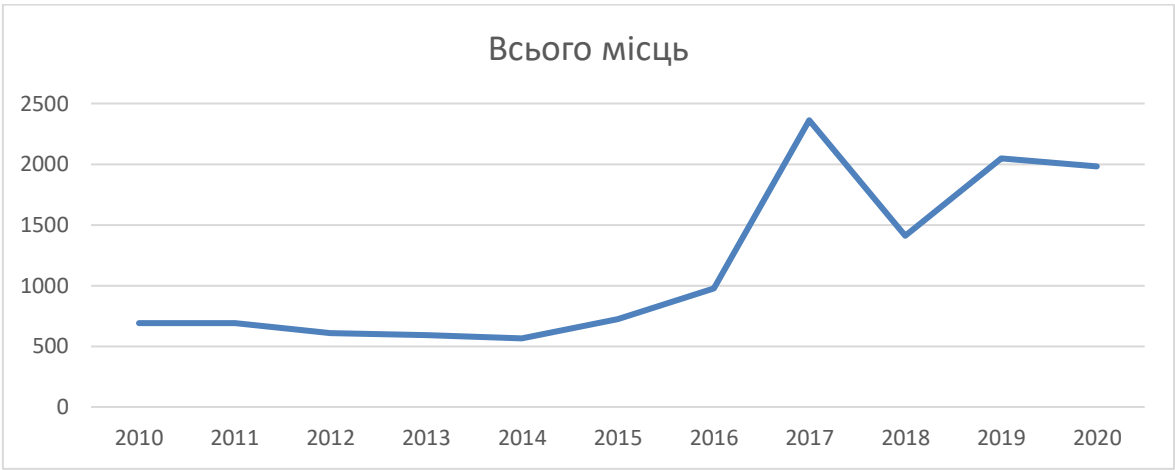



Рисунок 2.10 – Графік загальної кількості місць в університеті

Робимо припущення, що оскільки кількість заявок в 2016 році була максимально за останні 10 років, можна побачити на рис. 2.8, в 2017 році збільшили кількість контрактних місць до 2362, але це себе не виправдало, на рис.

3.7 та таблиці 1 бачимо реальне положення дій в 2017 році – взяли 871 абітурієнта. А це 1491 місце залишилося порожніми.

Саме в цьому і полягає суть даної роботи, розробити алгоритм на наступний рік, який дасть нам модель і по ній буде видно що кількість виділених контрактних місць не виправдана, а це викинуті державні гроші, які могла піти на користь, або в інших напрямках, або на потреби університетів.

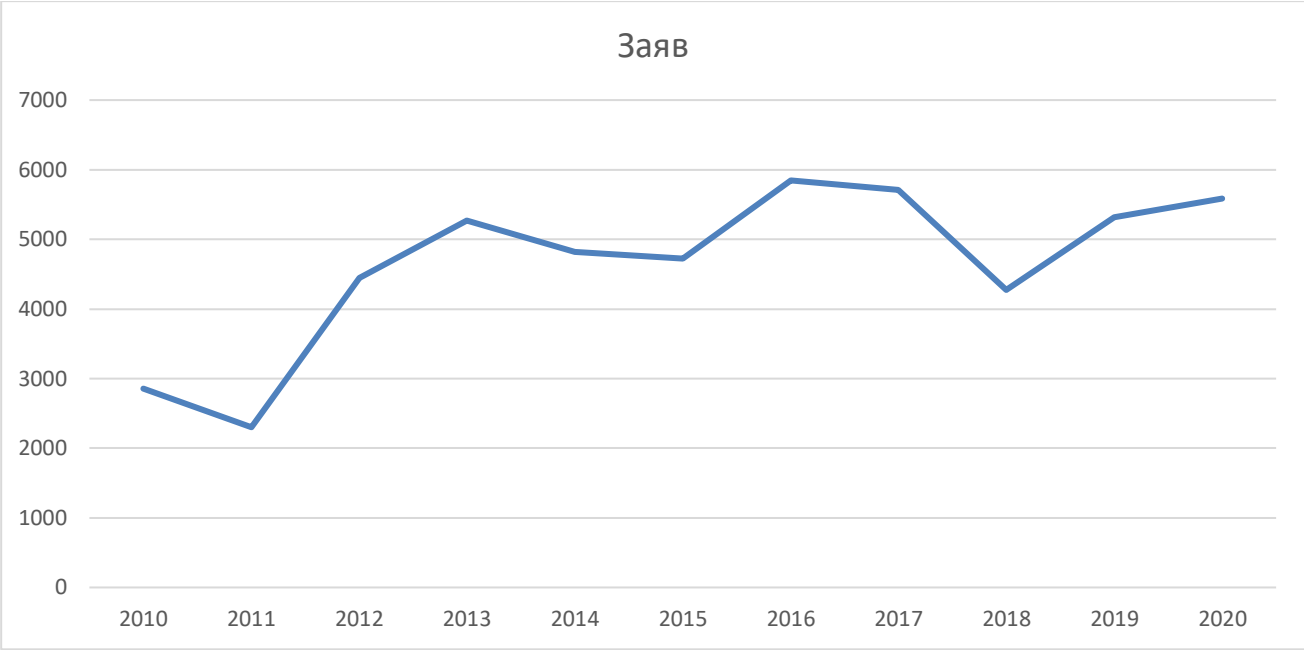



Рисунок 2.11 – Графік кількості поданих заявок до вступної компанії університету

Зауважень, аномалій або шумів в рис. 2.11 не бачимо. Можемо впевнено стверджувати що кількість місць на контракт в 2017 році була не випадкова. Тенденція подання студентами заявок до Державного Університету Телекомунікацій зменшується. Не в рамках даної роботи аналізувати факт зменшення Тенденція подання студентами заявок, на даний момент з цим нічого не можливо зробити, лише сповістити керівництво університету про результати аналізу вступної компанії.

Основні графіки по таблиці 2.1 були розглянуті. Для аналізу та подальшої побудови алгоритму, розглянемо ще декілька графіків в рамках додаткової інформації.

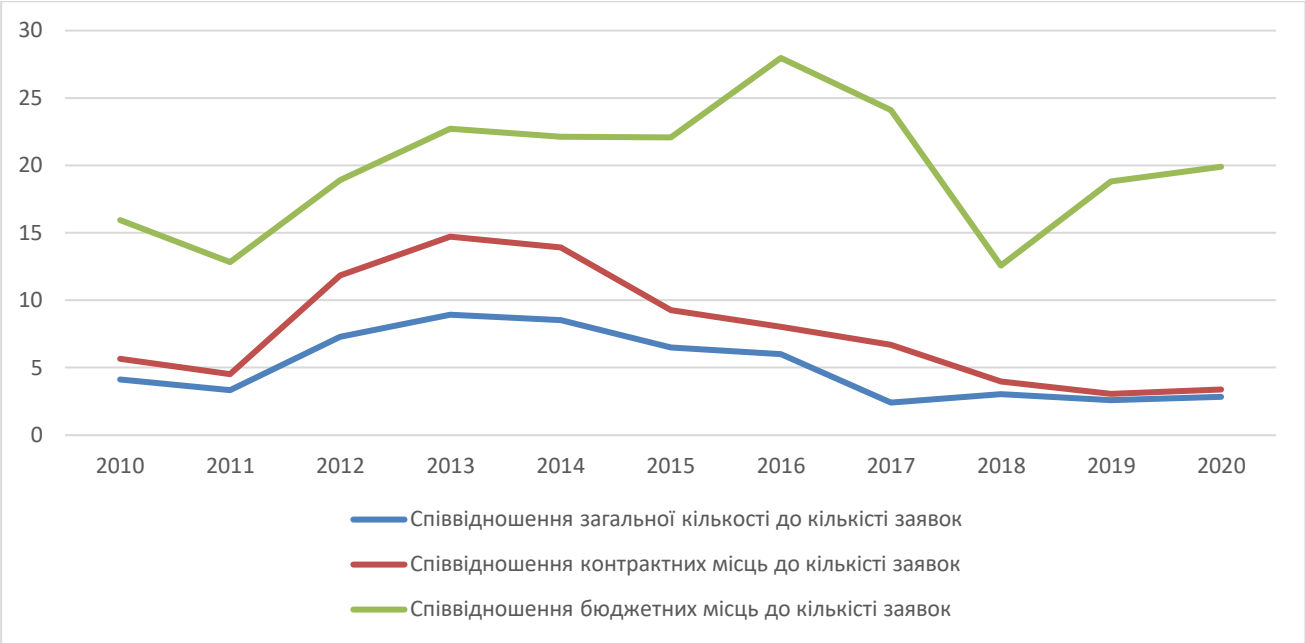



Рисунок 2.12 – Графік співвідношень заявок до інших значень

На рис. 2.12 зображено співвідношення заявок до різних параметрів:

- кількість бюджетних місць;
- кількість контрактних місць;
- загальна кількість виділених місць.

Спостерігається аномальне значення в 2018 році по співвідношенню кількості виділених місць на бюджетній основі до кількості заявок на вступ від абітурієнтів. Але пройшовшись очима по таблиці 1 бачимо максимальне значення виділених бюджетних місць за останні 10 років, дане значення вже розібрали вище, тому ігноруємо значення на рис. 3.12 за 2018 рік.

Одна тенденція зберігається на протязі останніх 4-х років, а саме коефіцієнт співвідношення загальної кількості до кількості заявок в районі 3.3. Останні 3 роки співвідношення контрактних місць до кількості заявок – 3.6 та знижається. Коефіцієнти до значення 4 вважаються поганими, гадаю адміністрація Державного Університету Телекомунікацій знає та пробує знайти рішення даної проблеми.

Проблеми низького співвідношення можуть бути різні. Наприклад:

- на скільки легко майбутнім студентам добиратися до свого майбутнього вищого навчального закладу? На прикладі Державного Університету

Телекомунікацій – складні та не зручно, відносно інфраструктури знаходиться далеко, поруч не має метро та вокзалу, під поруч розуміється в районі двох кілометрів;

- програма навчання;
- наявність воєнної кафедри;
- можливість безкоштовного навчання;
- матеріально-технічна база та інше.

В назві Державного Університету Телекомунікацій не просто так є слово «телекомунікацій». Давайте розглянемо таблицю 2.2, яка демонструє дані за 2013 рік. Проаналізуємо вплив назви на вибір абітурієнтами сучасних університетів для вступу.

Таблиця 2.2 – Дані за 2013 рік вступної компанії Державного Університету Телекомунікацій

| Інститут | Направлення | Денна форма навчання | | | | |
|---|---|---|---|---|---|---|
| | | Бюджет | Контракт | Було прийнято | Всього | Заяв |
| Телекомунікацій | Телекомунікації | 60 | 65 | 125 | 125 | 768 |
| | Радіотехніка | 25 | 75 | 100 | 100 | 355 |
| Менеджменту | Соціологія | 0 | 30 | 30 | 30 | 79 |
| | Менеджмент | 12 | 38 | 50 | 50 | 206 |
| Інформаційної безпеки | Системи технічного захисту інформації | 15 | 15 | 30 | 30 | 556 |
| | Безпека інформаційних і комунікаційних систем | 15 | 35 | 50 | 50 | 897 |
| | Управління інформаційною безпекою | 15 | 15 | 30 | 30 | 647 |
| Інформаційних технологій | Комп’ютерна інженерія | 30 | 20 | 50 | 50 | 1096 |
| | Телекомунікації | 60 | 65 | 125 | 125 | 665 |
| | Всього: | 232 | 358 | 590 | 590 | 5269 |

Якщо підрахувати кількість заявок на направлення «телекомунікацій» та «радіотехніка» отримаємо 1788 заявок, це 34% заявок. Дане направлення вже не користується популярністю. По статистичним даним за 2020 рік кількість поданих заявок на направлення телекомунікацій склала 371-у заявку, це 7% від загальної кількісті. Університет йде в «ногу» за часом але абітурієнти про це погано обізнані. Схоже реклама, сарафанне радіо або маркетинг не зовсім працюють на університет.

І знову 2017 рік, та що говорити про нього, вже вище описано причини, які стояли при формуванні кількості бюджетних та контрактних місць. Нас цікавить лише одне, що останні 2 роки університет має недобір, див рис. 2.13, не можуть правильно спрогнозувати значення на наступні роки.

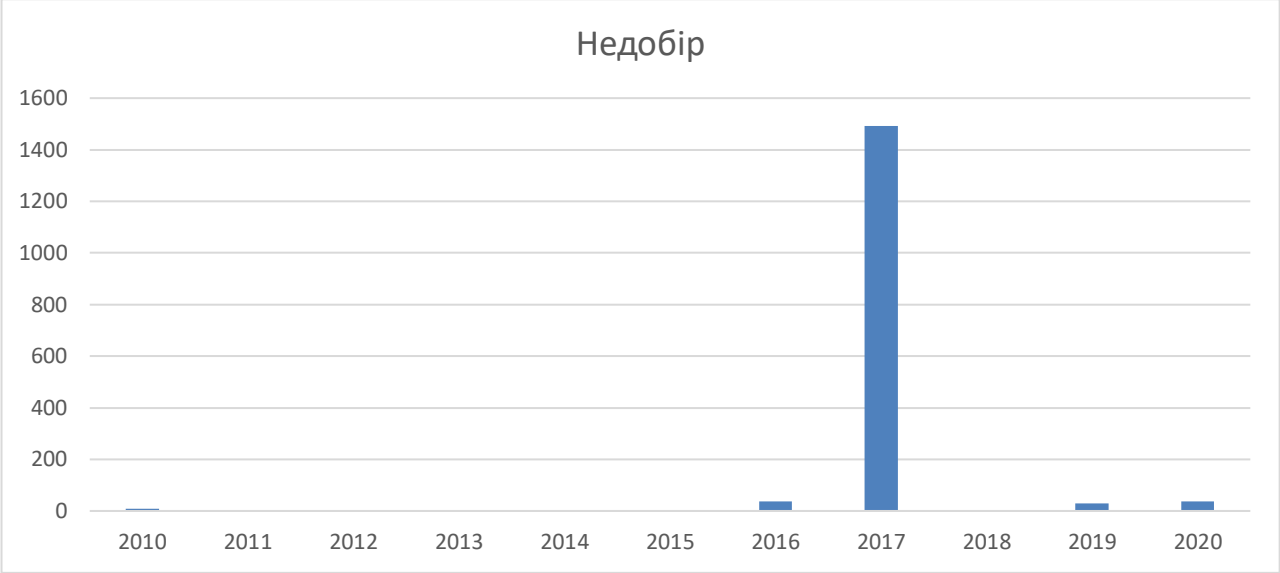



Рисунок 2.13 – Графік недобору студентів до університету

Проблеми недобору університетів можуть також бути через низький рівень знань абітурієнтів. Університет визначив мінімальні рейтинги для вступу на спеціальності по яким більшість абітурієнтів не проходить, звідси недобір. Перехід на 12 рокове навчання в школах ще не скоро покаже результати, його ввели декілька років назад, ще потрібно приблизно 8. Університетам потрібно якось всі ці роки «існувати» та зменшувати державну заборгованість, рекламувавши свої університети за власний кошт.

# 3 ЕТАПИ СТВОРЕННЯ АЛГОРИТМУ ПРОГНОЗУВАННЯ

## 3.1 Бібліотеки, на основі який побудований алгоритм

Python – це найпопулярніша мова програмування з динамічною семантикою. Вона досить проста для роботи, має велику підтримку серед розробників, легка в читанні і що найголовніше – легко взаємодіє з іншими мовами програмування. Її використання знижує вартість розробки та обслуговування програм, вважається однією з самих простих мов програмування. На даний момент в Python велика кількість бібліотек. Його обрали для машинного навчання через те що з ним легко збирати, систематизувати та аналізувати данні, після чого можемо просто написати алгоритм для штучного інтелекту.

Для початку роботи алгоритму потрібно імпортувати потрібні бібліотеки:

- sys – забезпечує доступ до змінних та функцій, які взаємодіють з інтерпретатором python;
- warnings – для видалення попереджувальних повідомлень, вони зазвичай видаються в ситуаціях, коли корисно попередити розробника про якусь проблему програми, коли ця проблема не вимагає винятку та припинення програми. Наприклад, можна зажадати попередження, коли програма використовує застарілу бібліотеку або залежність яка не використовується;
- tqdm – індикатор прогресу, візуально відображає процес виконання програми. Він потрібен для того щоб не сумніватися в тому, чи не зависла наша програма, дає інтуїтивне уявлення про швидкість виконання і підказує, скільки часу залишилося до завершення. В ході розробки допомогла, генерував модель десятки разів, в цей момент часу вентилятори мого ноутбуку працювали на 100% разом з процесором, через 7-30хв процес навчання закінчувався та програма продовжила працювати;
- pandas – програмна бібліотека написана на мові Python для обробки і аналізу даних;

- numpy – потрібна для зручної роботи з масивами, підтримка та робота багатовимірних масивів, високорівневих математичних функцій;
- scikit-learn – бібліотека для навчання та роботи з машинним навчанням на мові програмування Python;
- statsmodels – реалізовані численні методи статистичного моделювання, в тому числі для часових рядів, вони нам потрібні будуть далі;
- sciру - бібліотека призначена для виконання наукових і інженерних розрахунків;
- plotly – бібліотека для візуалізації даних з відкритим кодом на мові програмування Python, яка в свою чергу побудована на plotly.js, та в свою чергу, побудована на d3.js;
- jupyter notebook – це вкрай зручний інструмент для створення красивих аналітичних звітів, так як він дозволяє зберігати разом код, зображення, коментарі, формули і графіки [18].

### 3.2 Опис тенденції розвитку алгоритму згладжування

В цьому підрозділі ми поетапно розглянемо еволюцію написаного алгоритму згладжування, яку потім будемо використовувати для навчання нашої моделі.

На вході вступної компанії отримуємо 11 точок, за 11 років. Чого не вистачить для чіткої візуальної демонстрації ефективності роботи алгоритму. Для більш чіткої демонстрації роботи алгоритму потрібні дані великого об’єму, починаючи від 1000 точок часового ряду з явними ознаками сезонності. В якості такого прикладу для опису алгоритму в цьому розділі візьмемо публічні дані за 3 місяці 2006 року, 3000 значень, при більшому об’ємі графік стає не читабельним та важко аналізувати його, це статистика вартості погодинного продажу електроенергії в валюті євро, див. рис. 3.1.

Для відображення даних на графіку та підключення даних до нашого алгоритму використаємо наступний код:

```
dataset = panda.read_csv('./Electricity_Market_PZ_dayahead_price_volume.csv', index_col=['timestep'], parse_dates=['timestep'])
def drafGraph(series):
  fig, ax = pyplot.subplots(figsize=(15,5))
  pyplot.title("Вхідні дані")
  std =  series.rolling(window=smoothing).std()
  pyplot.plot(series, label="Вхідні дані")
  pyplot.legend(loc="upper left")
ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MONDAY))
  ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m'));
  pyplot.xticks(rotation=60)
drafGraph(dataset)
```

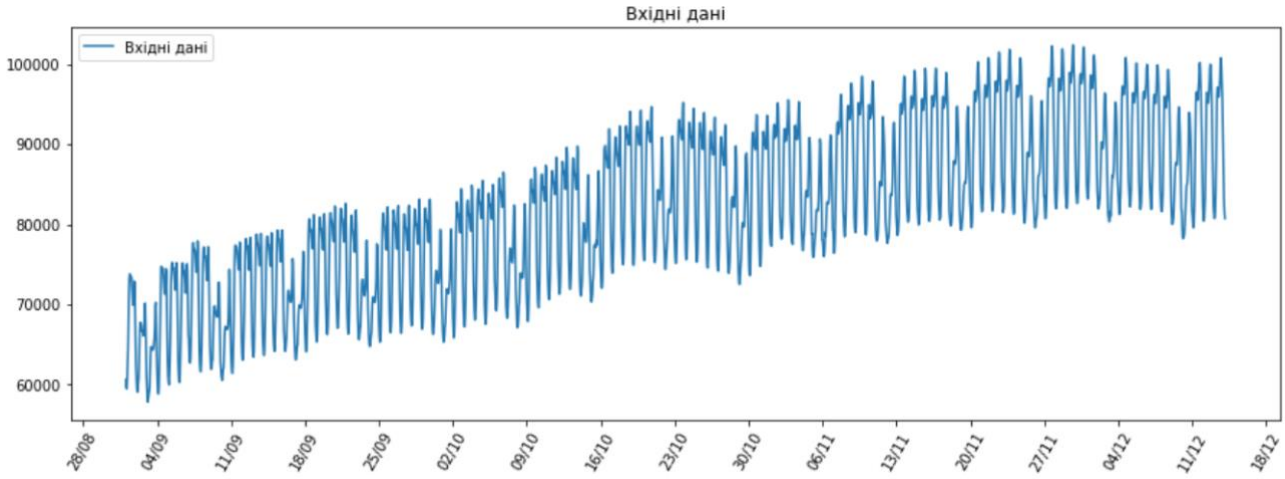



Рисунок 3.1 – Графічне представлення вхідних даних

Частина коду представленого вище відповідає за зчитування даних з файлу, ініційована та викликана функція візуалізації даних. Виставили в якості індексів в датасеті одну із колонок таблиці форматувавши його в формат дати для мови програмування PYTHON.

Дані підготовлені, починаємо почнемо описувати алгоритм програми. Почнемо з прогнозування. Можна зробити примітивне прогнозування часового ряду бравши середнє значення за попередню годину, за попередні 7 годин, за попередні 24 години та так далі.

$$\hat{y}_t = \frac{1}{k}\sum_{n=0}^{k-1} y_{t-n} \tag{3.1}$$

де $\hat{y}_t$ – прогноз, $k$ – кількість значень, $n$ – точка відліку

```
def moving_average(series, n):
    return nump.average(series[-n:])
moving_average(dataset.consumption_eur, 1)
```

На виході отримаємо значення - 80733.0.

Цього не достатньо для прогнозування, він не довготривалий, може бути ефективним, для нашого випадку, декілька годин але що робити далі. На практиці немає сенсу використовувати для цієї формули машинне навчання, з цим може справитись хто завгодно.

На допомогу приходить інше застосування - згладжування вихідного ряду за допомогою виявлення трендів. Скористаємося однією з підключених бібліотек у нашу програму, саме у бібліотеці pandas є готова реалізація для виявлення трендів - DataFrame.rolling(window).mean(). Чим більшу виставимо ширину інтервалу - тим більш плавним буде тренд.

Для визначення інтервалу потрібно оперувати інформацією о вхідній вибірці. Ми знаємо що дані представлені за кожну годину. Інтервал починається від години, дві, три і так далі, поки на знайдемо потрібний параметр згладжування.

У разі, якщо дані сильно аномальні, така поведінка особливо часто зустрічається, наприклад, у нашому випадку, фінансовому, така процедура може допомогти з визначенням загальних шаблонів.

Для наших даних тренди цілком очевидні, але якщо згладити за 24 години, стає краще видно динаміка приросту або упадку графіку, при тижневому

згладжуванні. Добре відображається тенденція, але нам вона не підходить, ці моменти описаний далі в розділі.

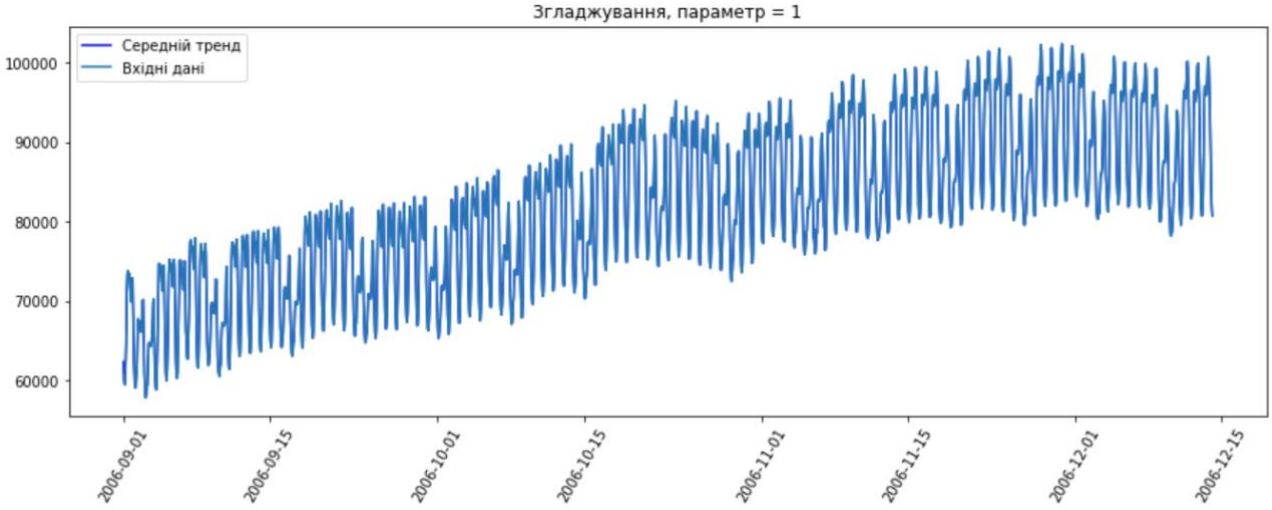



Рисунок 3.2 – Графік відображення тренду з годинним згладжуванням

На рис. 3.2 зображений графік погодинного згладжування, який повною мірою повторює графік вхідних даних. Це говорить нам про те що параметр згладжування занадто малий, оскільки на вході ми отримуємо дані за кожну годину операції.

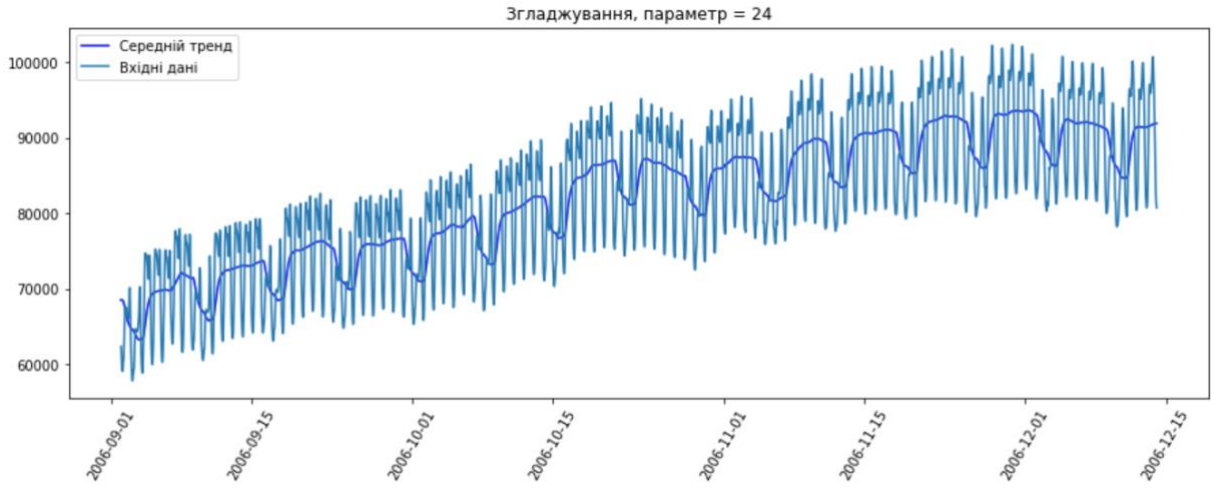



Рисунок 3.3 – Графік відображення тренду з 24 годинним згладжуванням

На рис. 3.3 зображений графік зі згладжуванням по дням, на якому видно що 5 із 7 днів в тижні ціна на електромережу була вища ніж ціна на неї ж в вихідні дні.

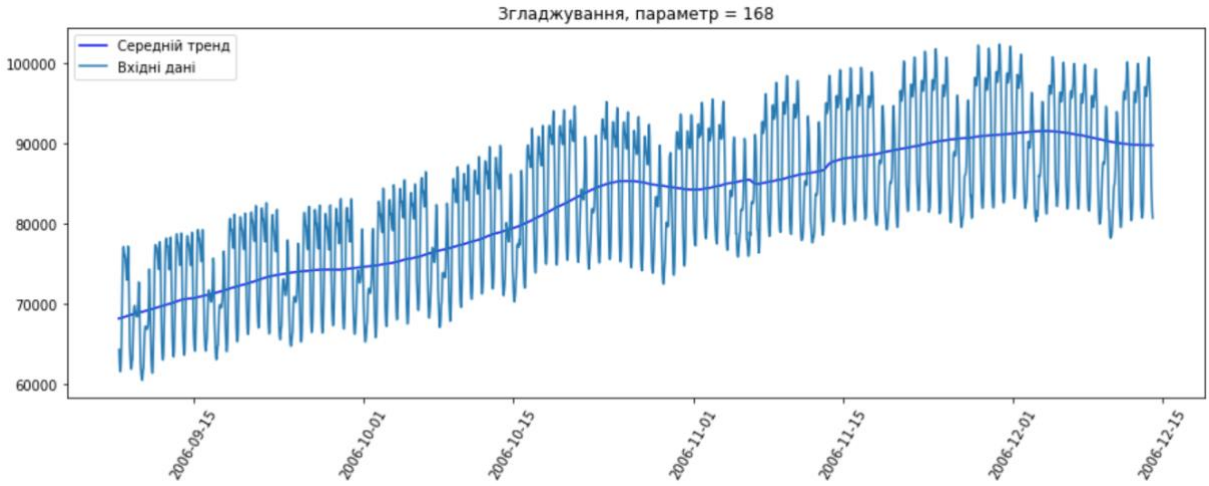



Рисунок 3.4 – Графік відображення тренду з тижневим згладжуванням

На рис. 3.4 зображений графік з тижневим згладжуванням, на якому очевидно видна лінія тенденції вартості електромережі.

```
def plotMovingAverage(series, smoothing):
    average = series.rolling(window=smoothing).mean()
    pyplot.figure(figsize=(15,5))
    pyplot.title("Згладжування, параметр = {}".format(smoothing))
    pyplot.plot(average, "b", label="Середній тренд")
    pyplot.plot(series[smoothing:], label="Вхідні дані")
    pyplot.legend(loc="upper left")
    pyplot.xticks(rotation=60)
plotMovingAverage(dataset, 1)
plotMovingAverage(dataset, 24)
plotMovingAverage(dataset, 24*7)
```

Також розрахуємо зважену середню, всередині якої за допомогою спостережень надаються ваги, які в сумі повинні давати одиницю, при цьому зазвичай останніми спостереженнями присвоюється більшу вагу [19].

$$\hat{y}_t = \sum_{n=1}^{k} w_n y_{t+1-n} \tag{3.2}$$

де $\hat{y}_t$ – прогноз, $k$ – кількість значень, $n$ – точка відліку, $w$ – вага.

```
def weightedAverage(series, weights):
    result = 0.0
    for n in range(len(weights)):
        result += series[-n-1] * weights[n]
    return result
weightedAverage(dataset.consumption_eur, [0.5, 0.26, 0.14, 0.08, 0.02])
```

На виході отримаємо значення 81687.36000000002, що більш точно відображає картину.

Давайте дослідимо що покаже графік, якщо замість зважування останніх значень в ряді ми дамо зважування всім доступнім спостереження, та експоненціально зменшуючи вагу через поглиблення в історичні дані. Формула виглядає наступним чином:

$$\hat{y}_t = \alpha * y_t + (1 - \alpha) * \hat{y}_{t-1} \tag{3.3}$$

де $\hat{y}_t$ – прогноз, $y_t$ – поточний елемент, $\alpha$ - згладжуваний фактор.

Експоненціальність скривається в рекурсивності даної функції – кожну ітерацію значення множимо на попереднє модельне значення, яке також містив у собі вагу.

```
def exponential_smoothing(series, a):
    result = [series[0]]
    for n in range(1, len(series)):
        result.append(a * series[n] + (1 - a) * result[n-1])
    return result
```

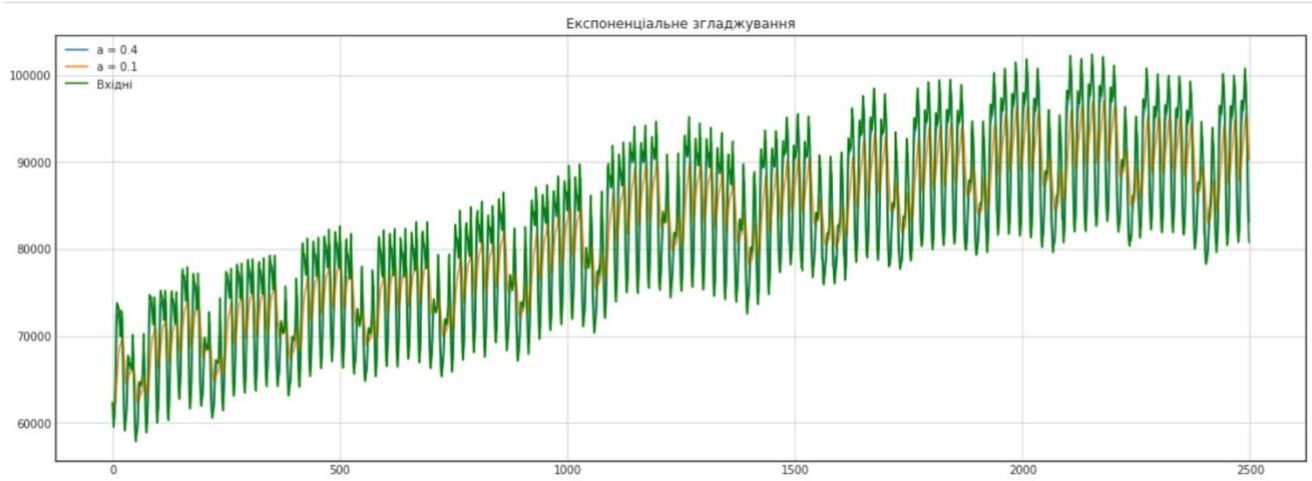



Рисунок 3.5 – Графік експоненційного згладжування

Але цього не достатньо, за допомогою даних методів можемо отримати лише одну точку прогнозу на майбутнє. Розширимо згладжування яке описано вище для опису декількох точок.

Для цього нам потрібно розбити ряд на дві складові – рівень та тренд. Рівень ми передбачили за допомогою метода описаного вище, тепер застосуємо для тренду таке згладжування та виведемо формулу:

$$\begin{gathered} l_x = \alpha y_x + (1-\alpha)(l_{x-1} + b_{x-1}) \\ b_x = \beta(l_x - l_{x-1}) + (1-\beta) b_{x-1} \\ \hat{y}_{x+1} = l_x + b_x \end{gathered} \tag{3.4}$$

де $\hat{y}_{x+1}$ – прогноз, $y_x$ – поточний елемент, $\alpha, \beta$ - згладжуваний фактор, $b$ – тренд, $l$ – рівень.

Отримали результат із набору функцій. Перша функція залежить від поточного значення та описує рівень, друга частина – це сума попередніх значень рівня та тренду. Друга функция – залежить від милого значення тренду та від зміни рівня на поточному кроці [20]. Переведемо формулу 3.4 в код програми python та виконаємо його:

```
def secondSmoothing(series, a, b):
    result = [series[0]]
    for i in range(1, len(series)+1):
```

```
        if i == 1:
            current, trend = series[0], series[1] - series[0]
        if i >= len(series):
            value = result[-1]
        else:
            value = series[i]
        next_level, current = current, a*value + (1-a)*(current+trend)
        trend = b*(current-next_level) + (1-b)*trend
        result.append(current+trend)
    return result
```

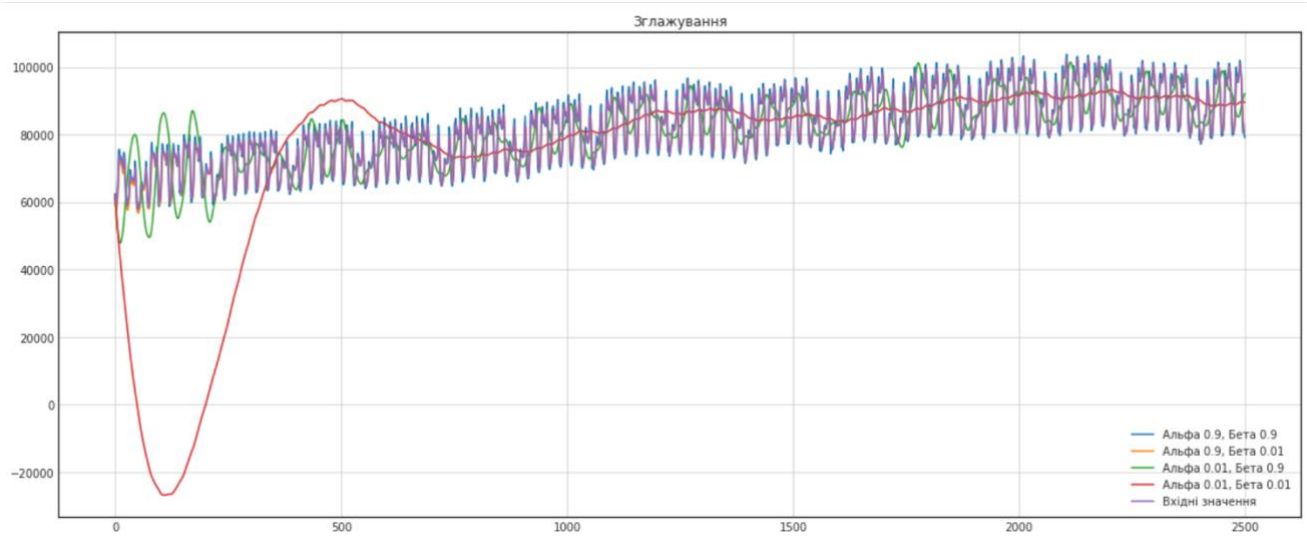



Рисунок 3.6 – Графік згладжування з прогнозом на декілька днів з підбором різних коефіцієнтів

При дослідженні рис. 3.6 бачимо що при підстановці різних пар найбільших та найменших коефіцієнтів $\alpha$ та $\beta$ отримаємо зовсім різні графіки згладжування. Так при найменших значеннях коефіцієнтів графік згладжування невірно відображається відносно вхідних даних, починаючи десь з 750-ого значення графік приходить до норми. При високих значення графік майже точно повторює вхідну вибірку. Вибір коефіцієнтів залежить від постановки задачі або точності якої нам потрібну буде добитися використовуючи цей метод.

Налаштовуємо вже два значення, як видно на малюнку вище, від цих значень залежить графік прогнозування, де $\alpha$ – згладжування навколо лінії тренду, а $\beta$ – згладжування самої лінії тренду.

На жаль, застосування окремих методів прогнозування не приводить до оптимального і досить точному результату, оскільки прогнози можуть враховувати не тільки фактори, що надає вплив на предмет прогнозування, а й різні складові прогнозу, такі як його основна тенденція розвитку, сезонна і циклічні складові, випадкова компонента.

Одним з методів, що враховує декілька складових прогнозу, є метод Хольта-Вінтерса. Метод Хольта-Вінтерса - це трипараметричної модель прогнозу, яка враховує:

- згладжений експонентний ряд;

- тренд;

- сезонність.

Покращимо варіант двійного згладжування додавши ще один коефіцієнт – сезонність. Для того щоб покращити результат прогнозування алгоритму – потрібні дані, якщо ми хочемо зробити прогноз на рік вперед, потрібно мінімум даних за 2-3 роки, краще за 3-5 повних років, без пробілів та аномальних значень.

Виведемо формулу на основі методу Хольта-Винтерса.

$$\begin{gathered} l_i = \alpha(y_i - s_{i-L}) + (1-\alpha)(l_{i-1} + t_{i-1}) \\ t_i = \beta(l_i - l_{i-1}) + (1-\beta)t_{i-1} \\ s_i = \gamma(y_i - l_t) + (1-\gamma)s_{i-L} \\ \hat{y}_{i+m} = l_i + mt_i + s_{i+1+(m-1)modL} \end{gathered} \tag{3.5}$$

де $l_i$ – згладжуваний ряд, $t_i$ – значення тренду в поточний період, $s_i$ – оцінка сезонності, $\hat{y}_{x+m}$ – значення лінії тенденції, $\alpha, \beta, \gamma$ – константи в діапазоні від 0 до 1, $i$ – індекс поточного спостереження.

Код програми наведений у додатку Б. Даний метод має ряд переваг.

Перевагою наявності коефіцієнту тренду з включеним експоненціальним згладжуванням за допомогою якого можна виявляти напрямок розвитку часового

ряду в динаміці та шліфувати дрібні неточності в даних для того щоб знайти спади та стрибки в ряді. Коефіцієнт сезонності дозволить будувати прогнози на майбутні періоди на основі даної сезонності, хочемо зробити прогноз на наступні 24 години, задаємо сезонність, за рахунок коефіцієнту метод будує прогнози на 24 години сезонності, на прикладах, які будуть розглядаються в даній роботі сезонність виставлена на різні параметри. Так для задачі прогнозування часового ряду для вступної компанії сезонність складає 1, 1рік, для задачі вартості електроенергії сезонність 24*7, іншими словами – тиждень, дані подані по стану на кожну години та судячи з графіку, див. рис. 3.2, 5 днів пік вартості, наступні 2 мала ціна, сезон дорівнює 7 днів.

За допомогою даного методу можна розраховувати прогнози на великий проміжок часу. Додавши сезонність та тренд метод став ще точніший. Але як і у кожної формули, тут теж є свої обмеження, які не можна залишати без уваги.

Для того щоб побудувати дуже точний прогноз, потрібні дані, бажана за великий проміжок часу. Для побудови точного прогнозу потрібно використовувати дані за 1-5 років, чим більше тим краще. В задачах, які були описані в роботі даних була різна кількість. Так для задачі з вартістю електромережі це 2500 точок, дані за 3 місяця, для задачі з прогнозуванням статистики вступної компанії це дані за 11 років, а це 11 точок. На рис. 3.12, рис. 3.13 та рис. 3.14 візуально можна побачити, що модель може не встигти навчитися на даній кількості даних.

Написаний код даного потрійного згладжування буде представлений в додатку разом із усім кодом алгоритму.

### 3.3 Опис та тестування алгоритму на тестових даних

Для початку розробки алгоритму визначимо який вид навчання ми будемо використовувати, їх два:

1. Навчання з вчителем - коли для навчальних даних у нас є X і Y, інакше кажучи, вхідні дані і відповідні їм цільові змінні (мітки). Для завдань

навчання з вчителем потрібно реалізувати дві головні функції для кожної побудованої моделі: навчання та прогнозування [21].

2. Навчання без вчителя – на вході отримали лише X та не отримали цільових змінних. Тому для задач без вчителя ми проводимо тільки оцінку щільності, кластеризації, в більш загальному сенсі - вивчаємо структуру даних. Для задач навчання без вчителя характерний розподіл на дві частини: задачі класифікації та регресії. Класифікація - це коли мітки дискретні. Наприклад, ми хочемо спрогнозувати, чи буде завтра сніг. Відповідь або «так» або «ні» - значить, це класифікація. Регресія намагається спрогнозувати значення уздовж отриманої осі, осі X. Наприклад, спроба спрогнозувати кількість хто завтра прийде до фінішу перший.

Якщо раніше при підстановці в формулу ми брали коефіцієнти на основі методу проб і помилок, для даного методу знайти точні коефіцієнти $\alpha, \beta, \gamma$.

Також при знаходженні коефіцієнтів потрібно мінімізувати функцію помилок втрат. Для цього будемо використовувати метод minimize з пакету scipy.optimize. Для його використання потрібно визвати його передавши в нього мінімум 3 параметра:

- функція для кросс-валідації на часовому ряді:
- визначити початкову точку $x_0$;
- метод за допомогою якого будуть мінімізуватися дані, цей метод вибирається методом найкращого результату із великого списку методів.

Результатом виконання методу minimize буде масив з трьома елементами, наші шукані константи дорівнюють: $\alpha = 0.015826655880716755$, $\beta = 0.01479205281421566$, $\gamma = 0.004297054322122084$.

При використанні функції мінімізації потрібно відкласти частину даних на тест, краще за все відкладати 30-40% даних для тесту, а на 60-70% навчати алгоритм машинного навчання. В нашому випадку ми відклали 500 значень для крос-валідації.

Великий нюанс виникає вже на етапі крос-валідації. Проблема в тому, що часовий ряд має не стабільну структуру, сьогодні тенденцій пішла вгору, завтра

вниз і це ніяк не структуровано, цей момент випадковий. Часові ряди можуть бути проблематичними для перехресної перевірки. Якщо якийсь зразок з'являється в 3-му році і залишається на 4–6 роки, то наша модель може його взяти за основу, але це може не буди частиною 1-го та 2-го років. Тому нам довелось використовувати більш хитріший спосіб при оптимізації параметрів.

Підхід, який іноді є більш принциповим для часових рядів, - це прямий ланцюжок, де наша процедура буде приблизно такою:

- fold 1: тренується на (1), тест (2);
- fold 2: тренується (1 2), тест (3];
- fold 3: тренується (1 2 3], тест (4);
- fold 4: тренується (1 2 3 4), тест (5);
- fold 5: тренується (1 2 3 4 5), тест (6);
- і так далі.

Така процедура точніше моделює ситуацію, яку ми побачимо під час прогнозування, де ми модулюємо минулі дані та прогнозуємо майбутні дані. Це також дає нам відчуття залежності нашого моделювання від обсягу даних.

Розберемо цю модель інакше, з точки зору програмування – починаємо навчати модель на невеликому відрізку пам'яті, десь 30-40%, часового ряду, від 0 індексу до якоїсь

Суть досить проста - починаємо навчати модель на невеликому відрізку тимчасового ряду, від початку до якоїсь умовної $t$, робимо прогнозування на $t + n$ кроків вперед та знову розраховуємо помилку. Після чого потрібно змістити початкову точку з 0 до $t + n$, зробити прогноз на відрізку з $t + n$ по $t + n * 2$, розрахувати помилку і так циклічно продовжувати тестовий відрізок доки не дійдемо до кінця часового ряду. В нашому випадку – для початку беремо відрізок часового ряду за перший місяць, нагадаю що в нас їх 3. Рухатися будемо по неділям, оскільки помітна по недільна тенденція, це логічно, оскільки заводи використовують велику кількість електроенергії в будень день, коли працюють.

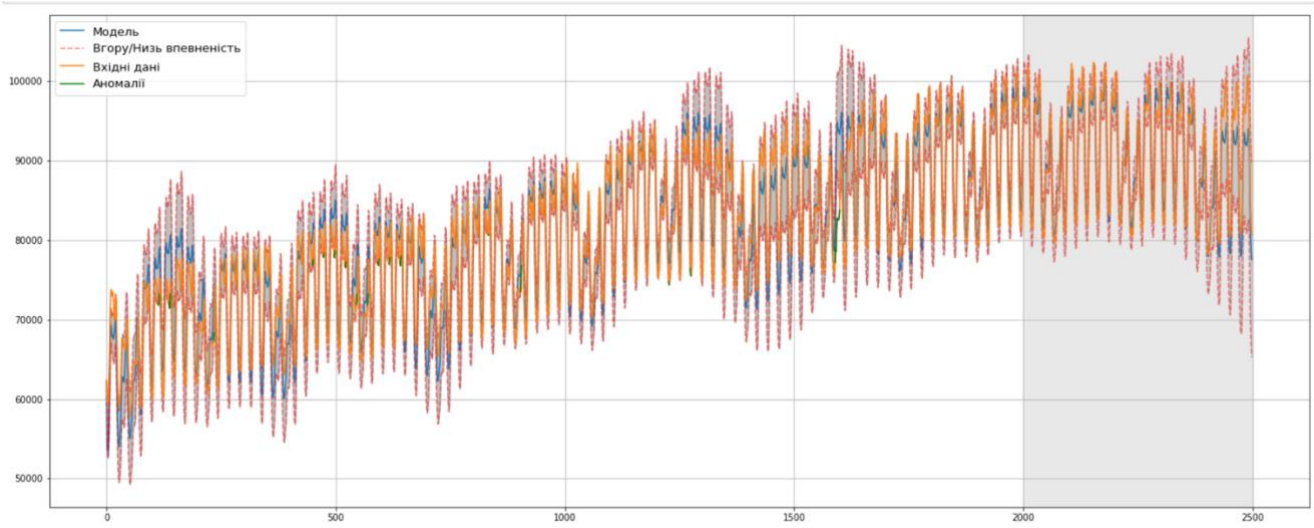



Рисунок 3.7 – Графік результатів крос-валідації

На графіку крос-валідації видно високу точність в вимірюваннях, де помаранчевою лінію намальований графік вхідних даних, побудована модель синім кольором, критерії це межі які виділені червоним пунктиром, добре видно на загальній картині графіку що в нас немає ні одної аномалії, вона була б відмічена зеленим кольором.

Наступний етап буде позбавлення від нестаціонарності за допомогою критерії Діккі-Фуллера і будування SARIMA моделі.

Критерій Дікі-Фуллера використовується для перевірки тимчасового ряду на стаціонарність.

Як нульової гіпотези розглядається наявність одиничного кореня, тобто нестаціонарності ряду. Критерій є одностороннім, в якості альтернативної для цієї гіпотези може розглядатися гіпотеза про стаціонарність ряду (всі корені конкретного полінома знаходяться поза радіусом одиничного кола), інший (рідкісний для часових рядів) варіант - наявність коренів всередині одиничного кола (вибухове зростання елементів ряду).

Потрібно враховувати якщо значення $p$ велике то ряд можна вважати нестаціонарним, якщо значення $p$ мале то з рівнем значущості 0.01 даний часовий ряд можна вважати стаціонарним.

Після декількох підгонок, отримали нульове значення критерію Діккі-Фуллера, це значить що тепер можемо визначати коефіцієнти для моделі SARIMA.

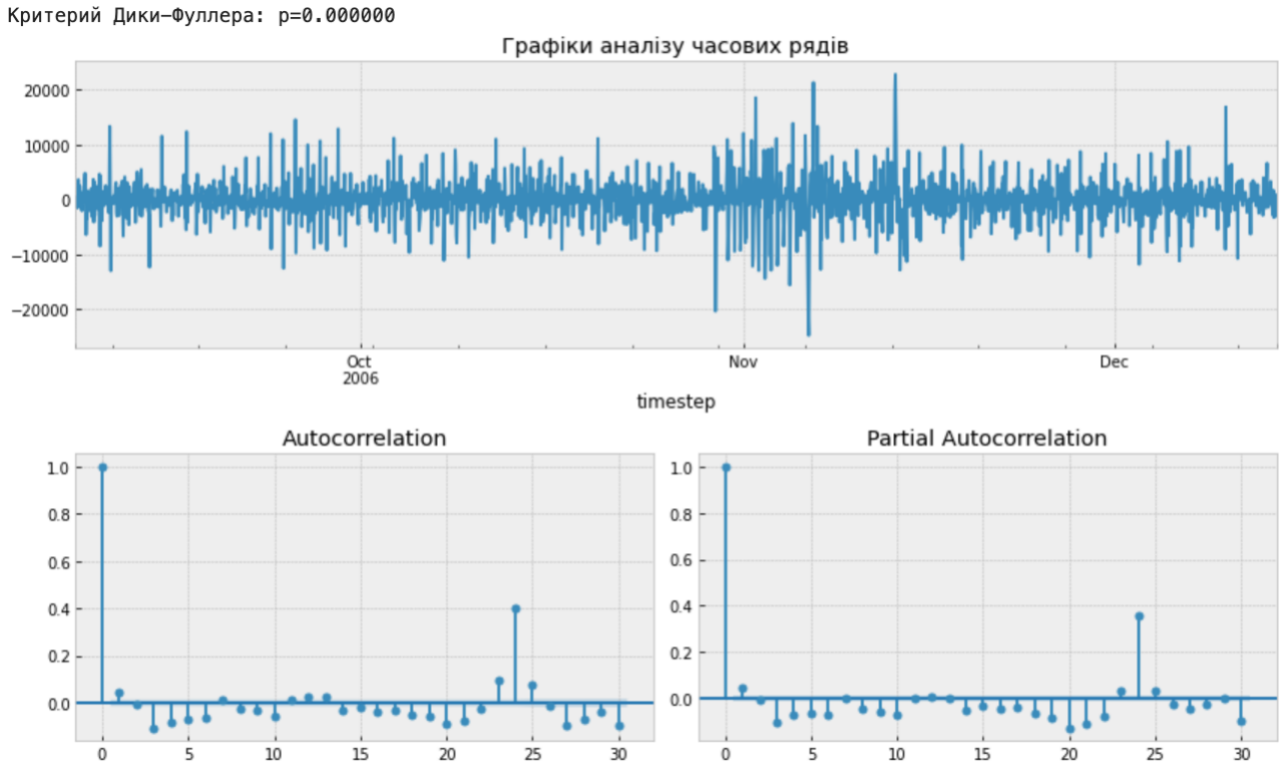



Рисунок 3.8 – Графік аналізу часових рядів з нульовим значенням критерію Діккі-Фуллера

Авторегресійна інтегрована ковзна середня, або ARIMA, є одним з найбільш широко використовуваних методів прогнозування для однофакторного прогнозування даних часових рядів. Метод може обробляти дані в яких є тренд, але він не може підтримувати тимчасові ряди з сезонним даними. SARIMA - розширення ARIMA, яке може працювати з моделюванням сезонного компонента.

Для елементі тренду потрібно три параметри, які потребують налаштування, вони точно такі ж ,як в моделі ARIMA:

- $p$ - порядок авторегресії тренда;
- $d$ - порядок зміни тренда;
- $q$ – тренд;
- середньої.

Також потрібні налаштування сезонні елементи, вони не є частиною ARIMA, їх 4:

- P - сезонний порядок авторегресії;
- $D$ - порядок сезонних різниць;
- $Q$ - сезонний порядок ковзних середніх;
- $M$ - кількість тимчасових кроків за один сезонний період.

SARIMAX Results

| | | | |
|---|---|---|---|
| Dep. Variable: | consumption_eur_container | No. Observations: | 2499 |
| Model: | SARIMAX(3, 1, 4)x(3, 1, [1], 24) | Log Likelihood | -23986.629 |
| Date: | Thu, 03 Dec 2020 | AIC | 47997.259 |
| Time: | 01:51:47 | BIC | 48067.022 |
| Sample: | 09-01-2006 | HQIC | 48022.599 |
| | - 12-14-2006 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| ar.L1 | -0.1047 | 0.308 | -0.340 | 0.734 | -0.708 | 0.499 |
| ar.L2 | -0.3144 | 0.236 | -1.334 | 0.182 | -0.776 | 0.148 |
| ar.L3 | -0.3699 | 0.154 | -2.406 | 0.016 | -0.671 | -0.069 |
| ma.L1 | 0.2696 | 0.309 | 0.873 | 0.382 | -0.335 | 0.875 |
| ma.L2 | 0.4409 | 0.208 | 2.122 | 0.034 | 0.034 | 0.848 |
| ma.L3 | 0.4561 | 0.148 | 3.078 | 0.002 | 0.166 | 0.747 |
| ma.L4 | 0.0794 | 0.018 | 4.382 | 0.000 | 0.044 | 0.115 |
| ar.S.L24 | 0.4782 | 0.008 | 59.723 | 0.000 | 0.463 | 0.494 |
| ar.S.L48 | -0.0835 | 0.008 | -10.965 | 0.000 | -0.098 | -0.069 |
| ar.S.L72 | -0.0337 | 0.003 | -13.357 | 0.000 | -0.039 | -0.029 |
| ma.S.L24 | -0.8204 | 0.009 | -87.071 | 0.000 | -0.839 | -0.802 |
| sigma2 | 1.572e+07 | 1.34e-08 | 1.17e+15 | 0.000 | 1.57e+07 | 1.57e+07 |

| | | | |
|---|---|---|---|
| Ljung-Box (L1) (Q): | 163.68 | Jarque-Bera (JB): | 18040.97 |
| Prob(Q): | 0.00 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 0.79 | Skew: | 0.01 |
| Prob(H) (two-sided): | 0.00 | Kurtosis: | 16.23 |

Рисунок 3.9 – Табличні значення SARIMA моделі

Для нашого випадку значення $d$, D дорівнює 1, інші потрібно знаходити на рис. 3.7. $p = 24, q = 1, P = 1, Q = 1$.

Підставимо ці значення в модель SARIMA і почнемо навчання. Цей та наступний етап найдовші, для їх виконання рекомендую проводити навчання на стаціонарній швидкій машині. Проводив навчання на своєму комп’ютері, який для цього не призначений, один цикл виконання програми займає від 30 хвилин, при цьому сильно навантажує систему, що є небезпечним моментом.

Після виконання коду отримаємо моделі та відразу запускаємо код для пошуку найкращого значення серед цих моделей, після чого виведемо ці дані на екран, як зображено на рис. 3.9.

В отриманих даним ми бачимо табличне представлення результатів моделі.

Авторегресія (AR) - це характеристика часових рядів, коли існує певна кореляція між розглянутими значеннями та значеннями, що передують їм і переходять до них. Для цього типу часових рядів ми використовували минулі дані для моделювання поведінки ряду.

Часові ряди ковзного середнього (MA) - це ті, які можна змоделювати за допомогою їх ковзного середнього. Наприклад, ковзну середню 12-місячну ціну за електроенергію можна обчислити, взявши середньомісячну з січня по грудень, а потім з лютого по січень і так далі. Він прогресує, скидаючи найдавніше значення та додаючи останнє значення.

Coef – коефіцієнти, std err – стд помилки, z – статистика, $P > |z|$ – ймовірність.

Тест Ljung Box - це спосіб перевірити відсутність серійної автокореляції до заданого відставання k. Тест визначає, чи є помилки білого шуму, чи є щось більше за ними; незалежно від того, чи є автокореляції помилок чи залишків ненульовими. По суті, це тест на відсутність придатності: якщо автокореляції залишків дуже малі, ми говоримо, що модель не демонструє „значної відсутності придатності”.

Авторегресійна умовна гетероскедастичних (Heteroskedasticity)- це метод, який явно моделює зміна дисперсії в часі в часі ряду.

Тест Ярка-Бера (англ. Jarque-Bera) різновид Лагранжа, - це тест на нормальність. Нормальність є одним із припущень для багатьох статистичних тестів, таких як тест t або тест F. Тест Ярка-Бери зазвичай проводять перед одним із попередніх тестів для підтвердження нормальності. Зазвичай його використовують для великих наборів даних.

У статистиці нерівність (англ. skewness) - це міра асиметрії розподілу ймовірності випадкової величини щодо її середнього значення. Іншими словами –

перекіс, говорить нам про величину та напрямок перекосу (відхід від горизонтальної симетрії). Значення перекосу може бути позитивним чи негативним, або навіть невизначеним. Якщо асиметрія дорівнює 0, дані є абсолютно симетричними, хоча для реальних даних це малоймовірно.

Куртоз (англ. kurtosis) - повідомляє вам висоту та різкість центрального піку щодо стандартної кривої дзвона.

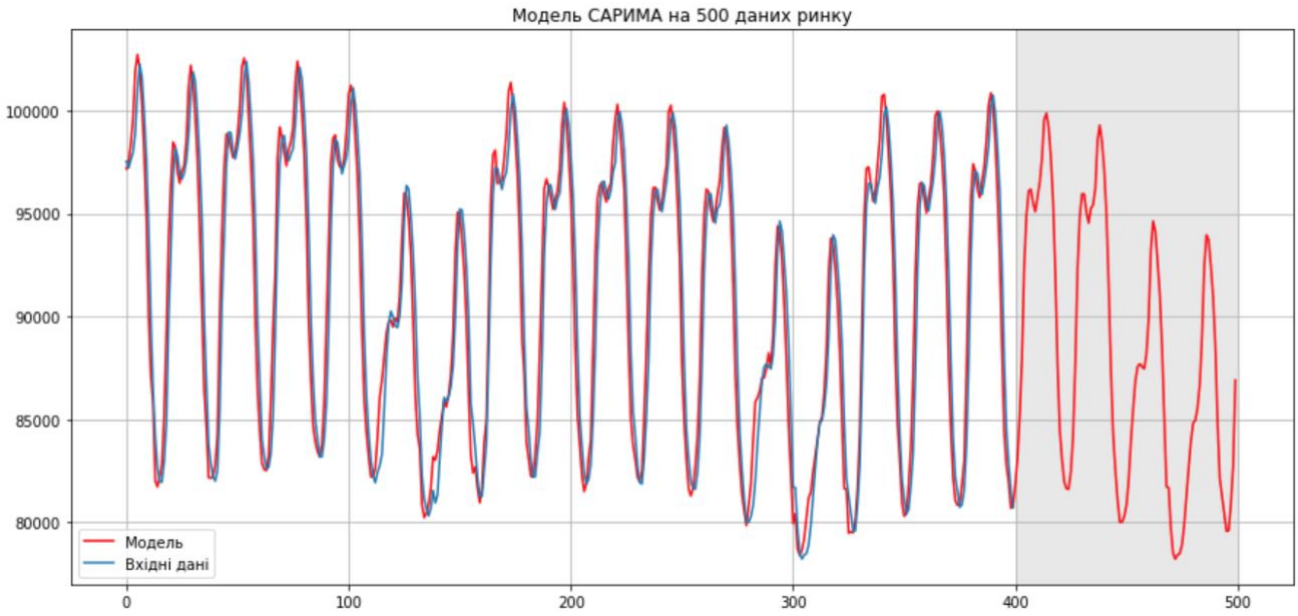



Рисунок 3.10 – Графік прогнозування моделі SARIMA

Судячи з рис. 3.10 прогнозовані дані відображають модель поведінки графіку, бачимо 5 днів коли тариф електроенергії найвищий, це будні дні, та 2 дні коли тариф менше, це вихідні дні. Залишилось лише дізнатися, які це цифри ми отримали на прогнозі на наступні 100 годин, для цього запустимо код та отримаємо табличні значення.

Зрівняємо результати з лінійним регресією, для цього будемо використовувати вже готову бібліотеку sklearn.linear_model та візьмемо функцію LinearRegression. Код програми:

*X_train, X_test, y_train, y_test = prepareData(dataset.consumption_eur, test_size=0.3, lag_start=12, lag_end=48)*

*lr = LinearRegression()*

```
lr.fit(X_train, y_train)
prediction = lr.predict(X_test)
pyplot.figure(figsize=(15, 7))
pyplot.plot(prediction, "r", label="Передбачення")
pyplot.plot(y_test.values, label="Вхідні дані")
pyplot.legend(loc="best")
pyplot.title("Лінійна регресія")
pyplot.grid(True);
```

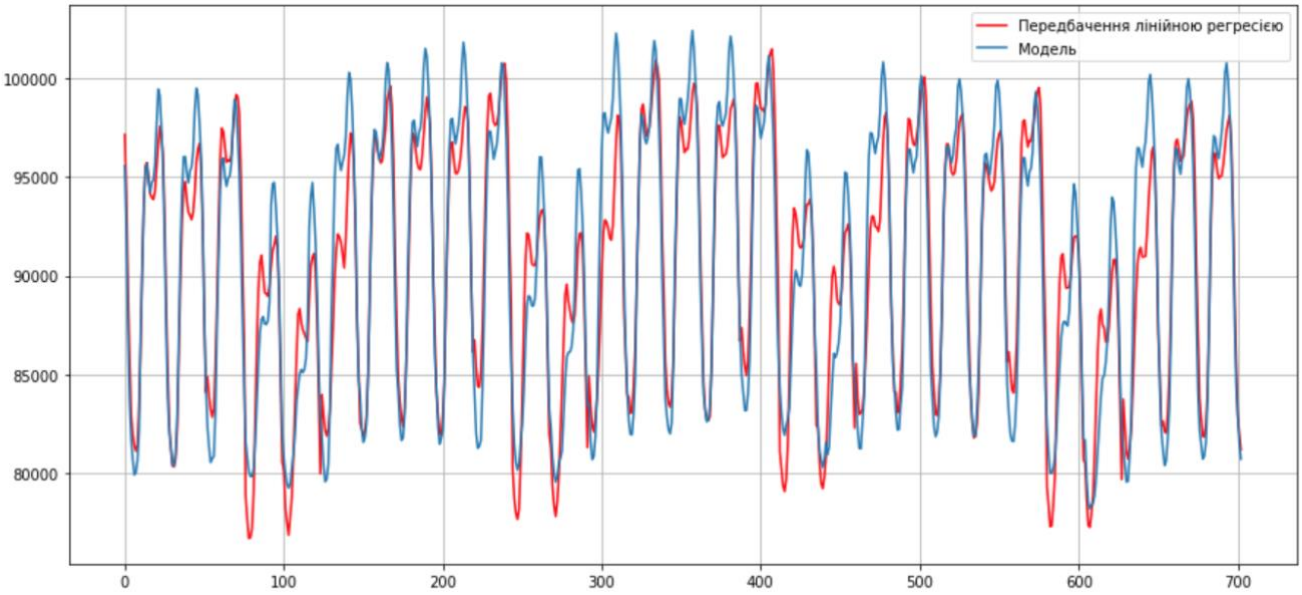



Рисунок 3.11 – Графік відображає наші вхідні дані та передбачення лінійною регресією

Як видно на графіку рис 3.11, лінія регресії не точно відображає реальне положення дій. Якщо розрахувати оцінку крос-валідації по тому ж принципу що був описаний, на виході отримаємо помилку в 2010.9964249140564, а це, переводячи на мову ринку, 2011 євро, велика сума помилки, за умовою того що наш алгоритм має краще показники крос-валідації та менші помилки можемо точно сказати, що розроблений алгоритм більш точно повторює поведінку вхідних даних, тому в нього більше шансів дати правильний прогноз на майбутнє.

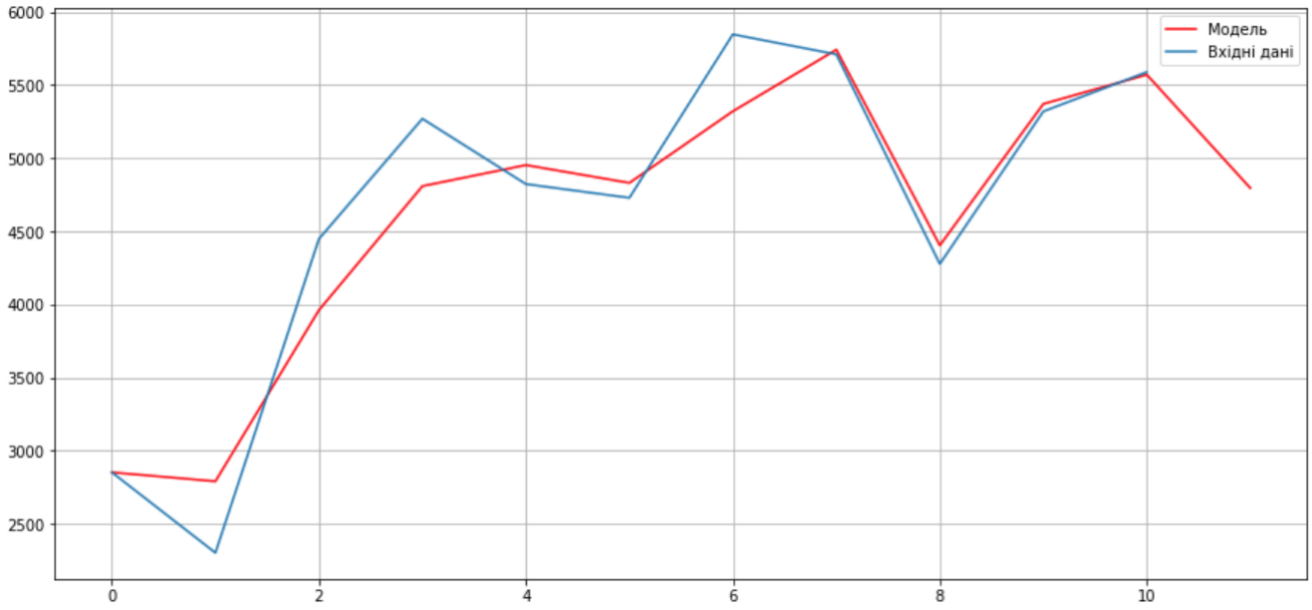



Рисунок 3.12 – Графік кількості заявок з моделлю прогнозування

Підігнавши наш алгоритм під вхідні дані ми отримали результати, які були зображені на рис. 3.12, рис. 3.13 та рис. 3.14. Навчити модель давати більш-менш правильні та логічні прогнози задача не з простих, особливо за такої кількості даних.

Зіткнулися з рядом проблем в алгоритмі, в основному вони були зв’язані через малу кількість даних. 11 точок потрібно розподілити на данні для навчання і на данні для тестування, припустимо що виділяємо останні 3 точки для тестування, на рис. 3.13 та рис. 3.14 видно що останні 3 роки на шаблонну поведінку, спрогнозувати такий момент складно, навіть людині.

На рис. 3.12 можна виразити сезонність, тому модель прогнозування зможе дати більш чіткі відповіді. На маю думку, данні прогнозування повністю відображають стан на 2021 рік.

В основному враховуючи проблему, яка тягнеться з 2019 року, а саме COVID, який вплинув на всіх нас, звичайно це ж стосується і державних фінансів, особистих фінансів, пристосованість людей до нових звичок, а саме дистанційного навчання та іншого, деяких психологічних нововведень – дистанційне спілкування та боязнь людних місць.

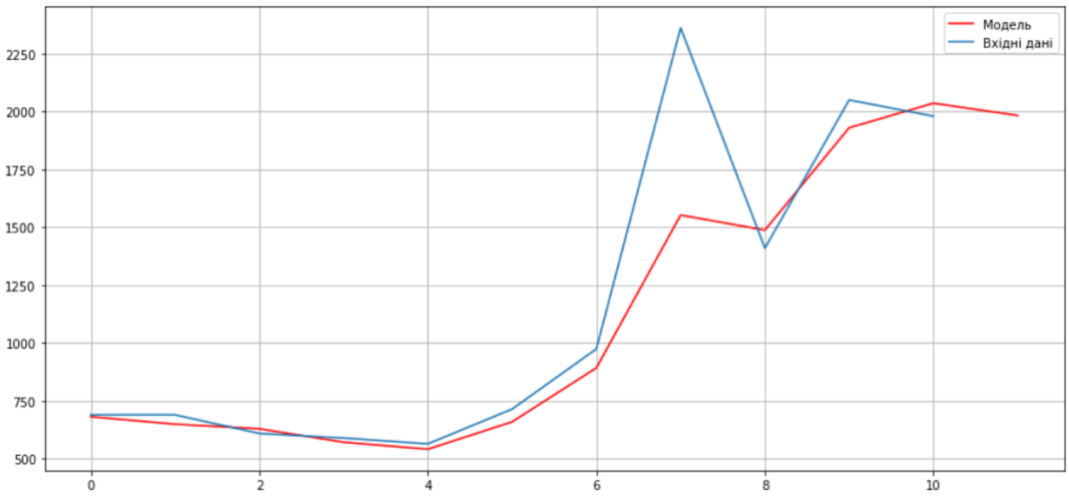



Рисунок 3.13 – Графік кількості вступних місць з моделлю прогнозування

Переглядаючи зображення прогнозування моделі на основі даних вступної компанії вищого навчального закладу можемо бачити тенденцію навчання моделі, чим більше даних модель змога отримати для того щоб навчитися тим легше їй спрогнозувати дані на майбутнє.

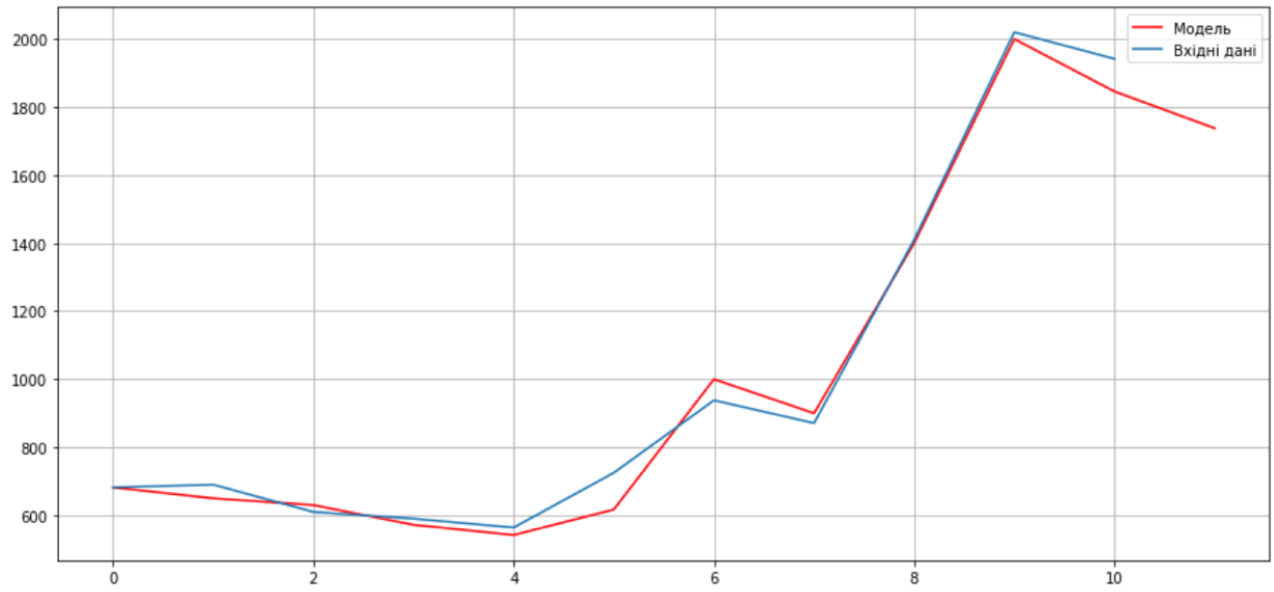



Рисунок 3.14 – Графік кількості прийнятих абітурієнтів з моделлю прогнозування

Проаналізувавши та нормалізувавши дані підставили їх до розробленого алгоритму отримавши приблизний прогноз на наступний 2021 рік.

# ВИСНОВОК

В ході виконання магістерської роботи було досліджено сучасні методи прогнозування майбутніх даних на основі навчальної вибірки, серед яких:

- сімейство регресійних методів прогнозування;
- сімейство бустингу;
- метод проміжного представлення;
- локальні методі прогнозування часових рядів;
- метод гнучких найменших квадратів.

Серед яких найбільш популярними, на час написання роботи, являються сімейство регресійних методів прогнозування та сімейство бустингу. Вони використовують різні алгоритмічні підходи та різну кількість даних, за рахунок цього в них різні результати на вихідній моделі. Лінійна регресія може швидко видати модель грунтуючись невиликим об’ємом даних, для сімейства бустингу цього недостатньо, а при великому об’ємі даних сімейство регресійних методів працює довше в помітні N раз, чим більше даних на вході там помітніше, в зрівняні з XGBoost із сімейства бустингу.

Проаналізовано вхідні дані вступної компанії за останні 11 років, з 2010 по 2020 рік. Побудовано графіки по кожному з основних параметрів статистики вступної компанії по рокам:

- кількість бюджетних місць;
- контрактних місць;
- кількість студентів, які поступили на перший курс;
- загальна кількість заявок.

Виявлені та проаналізовано лінії тенденцій для кожного з параметрів. Явно помітна сезонність подачі заявок студентів, 3 роки, серед яких два роки пік кількості поданих заявок, третій спад. Проаналізувавши вхідні дані, на графіку, вступної компанії знайдено декілька аномальних значень. Серед яких невідоме зменшення кількості бюджетних місць в 2017 році. Було прийнято рішення подібну аномалію оптимізувати через приблизне середнє значення кількості

бюджетних місць, оскільки дана аномалія впливає на модель навчання та знаходиться досить близько до тестових даних. Відповідно на виході отримали б не точно навчену модель. Також залишається невідома причина появи даного аномального значення, це може бути і як реальна статистика того року так і людський фактор помилки через який занесли в статистику помилкові дані.

Розроблено алгоритм, який працює точніше за сучасний методи лінійних регресійних методів прогнозування. Досліджено різницю між двома алгоритмами та продемонстровано на графіку, використовувавши однакові тестові дані, вихідні дані моделі навчання обох алгоритмів, див. рис. 3.11. Розроблений алгоритм працює краще як на етапі крос-валідації так і на етапі прогнозування кінцевих даних. Чим точніший етап навчання тим точніший прогноз.

Роботу можна удосконалити. За рахунок автоматизації процесу аналізування та нормалізації вхідних даних. На даний момент для цього потрібно залучати спеціаліста з області Data Science. Та за рахунок перенесення кодової бази до більш дружнього інтерфейсу користувача.

# ЛІТЕРАТУРА


1. Medium.com [Електронний ресурс]: [Інтернет-портал]. – Режим доступа: medium.com - вільний (дата звернення 30.12.2020). - Gradient Boosting and XGBoost, Gandhi, Rohith.

2. XGBoost - ML winning solutions (incomplete list). Процитовано 2016-08-01.

3. Chen, Tianqi; Guestrin, Carlos (2016). XGBoost: A Scalable Tree Boosting System. У Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. с. 785–794.

4. Vadim Mottl and Olga Krasotkina and Michael Markov and Ilya Muchnik Time-varying regression model with unknown time-volatility for nonstationary signal analysis, 2006.

5. Kalaba, R. and Tesfatsion, L. Time-varying linear regression via flexible least squares,1989.

6. J. McNames Innovations in local modeling for time series prediction, Stanford University, 1999.

7. Nikolay Savinov Flexible Least Squares, FlexibleLeastSquares, 2011.

8. U. B. Kjaerulff and A. L. Madsen. Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis. Springer, 2008.

9. Ин, Су - Теоретичний мінімум по Big Data. Все що потрібно знати про великі дані, 2016.

10. Винн Р., Холден К. Введення в прикладної економетричний аналіз. 1981. 294 с.

11. Грубер Й. Економетрика. Навчальний посібник для студентів економічних спеціальностей. 1996. 400 с.

12. Your-scorpion.ru [Електронний ресурс]: [Інтернет-портал]. – Режим доступа: your-scorpion.ru - вільний (дата звернення 30.12.2020). - Проверка результатов A/B теста, Цветков Максим.

13. D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical Correlation Analysis: An Overview with Application to Learning Methods. Neural Computation, 16(12):2639–2664, 2004.

14. Anderson, T. W., An Introduction to Multivariate Statistical Analysis, New York: John Wiley, 1958.

15. Маленбо Є. Статистичні методи економетрики, 1975. 423 с.

16. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical Correlation Analysis: An Overview with Application to Learning Methods. Neural Computation, 16(12):2639–2664, 2004.

17. D. Toomey - Jupyter for Data Science, 2017.

18. D. Barber. Expectation Correction for smoothing in Switching Linear Gaussian State Space models. Journal of Machine Learning Research, 7:2515–2540, 2006.

19. G. Kitagawa. The Two-Filter Formula for Smoothing and an implementation of the Gaussian-sum smoother. Annals of the Institute of Statistical Mathematics, 46(4):605–623, 1994.

20. Basu, S. and Christensen, J. (2013). Teaching classification boundaries to humans. InAAAI’2013.

21. Barlow, H. B. (1989). Unsupervised learning. Neural Computation, 1, 295–311.

Додаток А. Таблиця з даними про вступну компанію

Таблиця А.1 – Дані про вступну компанію на бакалавра за 2020 рік

| Інститут | Направлення | Курс | Денна форма навчання | | | | | Заочна форма навчання | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Бюджет | Контракт | Було прийнято | Всього | Заяв | Бюджет | Контракт | Було прийнято | Всього | Заяв |
| Навчально-науковий інститут Менеджменту та підприємництва | Інформаційна, бібліотечна та архівна справа | I | 11 | 69 | 80 | 80 | 94 | n/a | n/a | n/a | n/a | 8 |
| | | I | 0 | 80 | 80 | 80 | 57 | n/a | n/a | n/a | n/a | 5 |
| | | II | n/a | n/a | 0 | n/a | 2 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 15 | n/a | n/a | n/a | n/a | 9 |
| | Економіка | I | 5 | 30 | 35 | 40 | 117 | n/a | n/a | n/a | n/a | 5 |
| | | I | 0 | 40 | 40 | 40 | 70 | n/a | n/a | n/a | n/a | 9 |
| | | II | n/a | n/a | 0 | n/a | 0 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 1 | n/a | n/a | n/a | n/a | 0 |
| | Соціологія | I | 5 | 25 | 30 | 30 | 60 | n/a | n/a | n/a | n/a | 2 |
| | | I | 0 | 30 | 30 | 30 | 78 | n/a | n/a | n/a | n/a | 9 |
| | | II | n/a | n/a | 0 | n/a | 0 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 1 | n/a | n/a | n/a | n/a | 1 |
| | Менеджмент | I | 5 | 50 | 55 | 60 | 221 | n/a | n/a | n/a | n/a | 25 |
| | | I | 0 | 60 | 60 | 60 | 211 | n/a | n/a | n/a | n/a | 12 |
| | | II | n/a | n/a | 0 | n/a | 4 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 1 | n/a | n/a | n/a | n/a | 2 |
| | Маркетинг | I | 5 | 30 | 35 | 40 | 198 | n/a | n/a | n/a | n/a | 18 |
| | | I | 0 | 40 | 40 | 40 | 251 | n/a | n/a | n/a | n/a | 31 |
| | | II | n/a | n/a | 0 | n/a | 0 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 2 | n/a | n/a | n/a | n/a | 1 |

Продовження таблиці А.1 - дані про вступну компанію на бакалавра за 2020 рік

| Інститут | Направлення | Курс | Денна форма навчання | | | | | Заочна форма навчання | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Бюджет | Контракт | Було прийнято | Всього | Заяв | Бюджет | Контракт | Було прийнято | Всього | Заяв |
| Навчально-науковий інститут Менеджменту та підприємництва | Підприємництво, торгівля та біржова діяльність | I | 5 | 30 | 35 | 40 | 146 | n/a | n/a | n/a | n/a | 16 |
| | | I | 0 | 40 | 40 | 40 | 153 | n/a | n/a | n/a | n/a | 12 |
| | | II | n/a | n/a | 0 | n/a | 3 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 12 | n/a | n/a | n/a | n/a | 3 |
| Навчально-науковий інститут Інформаційних технологій | Інженерія програмного забезпечення | I | 11 | 89 | 100 | 100 | 729 | n/a | n/a | n/a | n/a | 8 |
| | | I | n/a | n/a | 0 | n/a | 105 | n/a | n/a | n/a | n/a | 19 |
| | | II | n/a | n/a | 0 | n/a | 28 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 0 | n/a | n/a | n/a | n/a | 2 |
| | Комп'ютерні науки | I | 11 | 69 | 80 | 80 | 538 | n/a | n/a | n/a | n/a | 21 |
| | | I | n/a | n/a | 0 | n/a | 103 | n/a | n/a | n/a | n/a | 11 |
| | | II | n/a | n/a | 0 | n/a | 11 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 22 | n/a | n/a | n/a | n/a | 2 |
| | Комп’ютерна інженерія | I | 24 | 63 | 87 | 90 | 406 | n/a | n/a | n/a | n/a | 21 |
| | | I | n/a | n/a | 0 | n/a | 122 | n/a | n/a | n/a | n/a | 10 |
| | | II | n/a | n/a | 0 | n/a | 8 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 45 | n/a | n/a | n/a | n/a | 13 |
| Навчально-науковий інститут Телекомунікацій | Системний аналіз | I | 6 | 30 | 36 | 40 | 160 | n/a | n/a | n/a | n/a | 6 |
| | | I | 0 | 40 | 40 | 40 | 39 | n/a | n/a | n/a | n/a | 7 |
| | | II | n/a | n/a | 0 | n/a | 0 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 9 | n/a | n/a | n/a | n/a | 1 |
| Навчально-науковий інститут Захисту інформації | Кібербезпека | I | 53 | 271 | 324 | 330 | 702 | n/a | n/a | n/a | n/a | 19 |
| | | I | 0 | 100 | 100 | 100 | 164 | n/a | n/a | n/a | n/a | 19 |
| | | II | n/a | n/a | 0 | n/a | 17 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 69 | n/a | n/a | n/a | n/a | 6 |

Продовження таблиці А.1 - дані про вступну компанію на бакалавра за 2020 рік

| Інститут | Направлення | Курс | Денна форма навчання | | | | | Заочна форма навчання | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Бюджет | Контракт | Було прийнято | Всього | Заяв | Бюджет | Контракт | Було прийнято | Всього | Заяв |
| Навчально-науковий інститут Інформаційних технологій | Інформаційні системи та технології | I | 11 | 39 | 50 | 50 | 274 | n/a | n/a | n/a | n/a | 13 |
| | | I | 0 | 50 | 50 | 50 | 70 | n/a | n/a | n/a | n/a | 7 |
| | | II | n/a | n/a | 0 | n/a | 1 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 20 | n/a | n/a | n/a | n/a | 12 |
| Навчально-науковий інститут Телекомунікацій | Телекомунікації та радіотехніка | I | 124 | 176 | 300 | 300 | 302 | n/a | n/a | n/a | n/a | 8 |
| | | I | 0 | 100 | 100 | 100 | 69 | n/a | n/a | n/a | n/a | 2 |
| | | II | n/a | n/a | 0 | n/a | 28 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 44 | n/a | n/a | n/a | n/a | 13 |
| Навчально-науковий інститут Захисту інформації | Публічне управління та адміністрування | I | 5 | 50 | 55 | 60 | 82 | n/a | n/a | n/a | n/a | 6 |
| | | I | 0 | 60 | 60 | 60 | 67 | n/a | n/a | n/a | n/a | 5 |
| | | II | n/a | n/a | 0 | n/a | 0 | - | - | - | - | - |
| | | III | n/a | n/a | 0 | n/a | 0 | n/a | n/a | n/a | n/a | 0 |
| | | Всього: | 281 | 1661 | 1942 | 1980 | 5931 | 0 | 0 | 0 | 0 | 399 |

# Додаток Б. Код програми

```
import sys
import warnings
warnings.filterwarnings('ignore')
from tqdm import tqdm
import pandas as panda
import numpy as nump
from sklearn.metrics import mean_absolute_error, mean_squared_error
import statsmodels.tsa.api as smta
import statsmodels.api as sma
import scipy.stats as scst
from scipy.optimize import minimize
import matplotlib.pyplot as pyplot
import matplotlib.dates as mdates
%matplotlib inline
from plotly.offline import init_notebook_mode, plot, iplot
init_notebook_mode(connected = True)
dataset = panda.read_csv('./Electricity_Market_PZ_dayahead_price_volume.csv', index_col=['timestep'], parse_dates=['timestep'])
def moving_average(series, n):
    return nump.average(series[-n:])
moving_average(dataset.consumption_eur, 1)
def plotMovingAverage(series, smoothing):
    average = series.rolling(window=smoothing).mean()
    pyplot.figure(figsize=(15,5))
    pyplot.title("Згладжування, параметр = {}".format(smoothing))
    pyplot.plot(average, "b", label="Середній тренд")
    pyplot.plot(series[smoothing:], label="Вхідні дані")
    pyplot.legend(loc="upper left")
    pyplot.xticks(rotation=60)
plotMovingAverage(dataset, 1)
plotMovingAverage(dataset, 24)
plotMovingAverage(dataset, 24*7)
def weightedAverage(series, weights):
    result = 0.0
    for n in range(len(weights)):
        result += series[-n-1] * weights[n]
    return resultweightedAverage(dataset.consumption_eur, [0.5, 0.26, 0.14, 0.08, 0.02])
def exponential_smoothing(series, a):
    result = [series[0]]
    for n in range(1, len(series)):
        result.append(a * series[n] + (1 - a) * result[n-1])
    return result
with pyplot.style.context('seaborn-white'):
    pyplot.figure(figsize=(20, 7))
    for a in [0.4, 0.1]:
        pyplot.plot(exponential_smoothing(dataset.consumption_eur, a), label="a = {}".format(a))
    pyplot.plot(dataset.consumption_eur.values, "g", label = "Вхідні")
    pyplot.legend(loc="best")
    pyplot.axis('tight')
    pyplot.title("Експоненціальне згладжування")
    pyplot.grid(True)
def secondSmoothing(series, a, b):
    result = [series[0]]
    for i in range(1, len(series)+1):
        if i == 1:
            current, trend = series[0], series[1] - series[0]
        if i >= len(series):
            value = result[-1]
        else:
            value = series[i]
        next_level, current = current, a*value + (1-a)*(current+trend)
        trend = b*(current-next_level) + (1-b)*trend
        result.append(current+trend)
    return result
with pyplot.style.context('seaborn-white'):
    pyplot.figure(figsize=(20, 8))
    for a in [0.9, 0.01]:
        for b in [0.9, 0.01]:
            pyplot.plot(secondSmoothing(dataset.consumption_eur, a, b), label="Альфа {}, Бета {}".format(a, b))
    pyplot.plot(dataset.consumption_eur.values, label = "Вхідні значення")
    pyplot.legend(loc="best")
    pyplot.axis('tight')
    pyplot.title("Зглажування")
    pyplot.grid(True)
class akaHoltWinters:
```

```
    def __init__(self, series, slen, a, b, g, p, scalingFactor):
        self.series = series
        self.slen = slen
        self.a = a
        self.b = b
        self.g = g
        self.p = p
        self.scalingFactor = scalingFactor
    def initial_trend(self):
        sum = 0.0
        for i in range(self.slen):
            sum += float(self.series[i+self.slen] - self.series[i]) / self.slen
        return sum / self.slen
    def initial_seasonal_components(self):
        seasonals = {}
        season_averages = []
        n_seasons = int(len(self.series)/self.slen)
        for j in range(n_seasons):
            season_averages.append(sum(self.series[self.slen*j:self.slen*j+self.slen])/float(self.slen))
        for i in range(self.slen):
            sum_of_vals_over_avg = 0.0
            for j in range(n_seasons):
                sum_of_vals_over_avg += self.series[self.slen*j+i]-season_averages[j]
            seasonals[i] = sum_of_vals_over_avg/n_seasons
        return seasonals
    def triple_exponential_smoothing(self):
        self.result = []
        self.Smooth = []
        self.Season = []
        self.Trend = []
        self.PredictedDeviation = []
        self.UpperBond = []
        self.LowerBond = []
        seasonals = self.initial_seasonal_components()
        for i in range(len(self.series)+self.p):
            if i == 0:
                smooth = self.series[0]
                trend = self.initial_trend()
                self.result.append(self.series[0])
                self.Smooth.append(smooth)
                self.Trend.append(trend)
                self.Season.append(seasonals[i%self.slen])
                self.PredictedDeviation.append(0)
                self.UpperBond.append(self.result[0] +
                                      self.scalingFactor *
                                      self.PredictedDeviation[0])
                self.LowerBond.append(self.result[0] -
                                      self.scalingFactor *
                                      self.PredictedDeviation[0])
                continue
            if i >= len(self.series):
                m = i - len(self.series) + 1
                self.result.append((smooth + m*trend) + seasonals[i%self.slen])
                self.PredictedDeviation.append(self.PredictedDeviation[-1]*1.01)
            else:
                val = self.series[i]
                last_smooth, smooth = smooth, self.a*(val-seasonals[i%self.slen]) + (1-self.a)*(smooth+trend)
                trend = self.b * (smooth-last_smooth) + (1-self.b)*trend
                seasonals[i%self.slen] = self.g*(val-smooth) + (1-self.g)*seasonals[i%self.slen]
                self.result.append(smooth+trend+seasonals[i%self.slen])
                self.PredictedDeviation.append(self.g * nump.abs(self.series[i] - self.result[i])  + (1-self.g)*self.PredictedDeviation[-1])
            self.UpperBond.append(self.result[-1] +
                                  self.scalingFactor *
                                  self.PredictedDeviation[-1])
            self.LowerBond.append(self.result[-1] -
                                  self.scalingFactor *
                                  self.PredictedDeviation[-1])
            self.Smooth.append(smooth)
            self.Trend.append(trend)
            self.Season.append(seasonals[i%self.slen])
from sklearn.model_selection import TimeSeriesSplit
def timeseries(x):
    errors = []
    values = data.values
    a, b, g = x
    tscv = TimeSeriesSplit(n_splits=10)
    for train, test in tscv.split(values):
        model = akaHoltWinters(series=values[train], slen = 1, a=a, b=b, g=g, p=len(test), scalingFactor=2.5)
        model.triple_exponential_smoothing()
```

```
        predictions = model.result[-len(test):]
        actual = values[test]
        error = mean_squared_error(predictions, actual)
        errors.append(error)
    return nump.mean(nump.array(errors))
%%time
data = dataset.consumption_eur[:-500]
x = [0, 0, 0]
opt = minimize(timeseries, x0=x, method="Nelder-Mead")
a, b, g = opt.x
print(a, b, g)
data = dataset.consumption_eur
model = akaHoltWinters(data[:-128], slen = 24*7, a = a, b = b, g = g, p = 128, scalingFactor = 2.2)
model.triple_exponential_smoothing()
def plotHoltWinters():
    Anomalies = nump.array([nump.NaN]*len(data))
    Anomalies[data.values<model.LowerBond] = data.values[data.values<model.LowerBond]
    pyplot.figure(figsize=(25, 10))
    pyplot.plot(model.result, label = "Модель")
    pyplot.plot(model.UpperBond, "r--", alpha=0.5, label = "Вгору/Низ впевненість")
    pyplot.plot(model.LowerBond, "r--", alpha=0.5)
    pyplot.fill_between(x=range(0,len(model.result)), y1=model.UpperBond, y2=model.LowerBond, alpha=0.5, color = "grey")
    pyplot.plot(data.values, label = "Вхідні дані")
    pyplot.plot(Anomalies, "g", markersize=10, label = "Аномалії")
    pyplot.axvspan(len(data)-500, len(data), alpha=0.5, color='lightgrey')
    pyplot.grid(True)
    pyplot.axis('tight')
    pyplot.legend(loc="best", fontsize=13);
plotHoltWinters()
def plotProcess(n_samples=500, r=0):
    x = w = nump.random.normal(size=n_samples)
    for t in range(n_samples):
        x[t] = r * x[t-1] + w[t]
    with pyplot.style.context('bmh'):
        pyplot.figure(figsize=(20, 8))
        pyplot.plot(x)
        pyplot.title("R параметр = {}\n Дікі-Фуллер".format(r))
for r in [0, 0.6, 0.8, 1]:
    plotProcess(r=r)
def tsplot(y, lags=None, figsize=(12, 7)):
    if not isinstance(y, panda.Series):
        y = panda.Series(y)
    with pyplot.style.context('bmh'):
        fig = pyplot.figure(figsize=figsize)
        layout = (2, 2)
        ax1 = pyplot.subplot2grid(layout, (0, 0), colspan=2)
        ax2 = pyplot.subplot2grid(layout, (1, 0))
        ax3 = pyplot.subplot2grid(layout, (1, 1))
        y.plot(ax=ax1)
        ax1.set_title('Графіки аналізу часових рядів')
        smta.graphics.plot_acf(y, lags=lags, ax=ax2, alpha=0.5)
        smta.graphics.plot_pacf(y, lags=lags, ax=ax3, alpha=0.5)
        print("Критерий Дики-Фуллера: p=%f" % sma.tsa.stattools.adfuller(y)[1])
        pyplot.tight_layout()
    return
tsplot(dataset.consumption_eur, lags=30)
def invboxcox(y,lamda):
    if lamda == 0:
        return(nump.exp(y))
    else:
        return(nump.exp(nump.log(lamda*y+1)/lamda))
data = dataset.copy()
data['consumption_eur_container'], lamda = scst.boxcox(data.consumption_eur+1)
tsplot(data.consumption_eur_container, lags=30)
print("Оптимальный параметр преобразования Бокса-Кокса: %f" % lamda)
data['consumption_eur_container_day'] = data.consumption_eur_container - data.consumption_eur_container.shift(24*7)
tsplot(data.consumption_eur_container_day[24*7:], lags=30)
data['consumption_eur_container_day_diff'] = data.consumption_eur_container_day - data.consumption_eur_container_day.shift(1)
tsplot(data.consumption_eur_container_day_diff[24*7+1:], lags=30)
p = range(0, 24)
d=1
q = range(0, 1)
P = range(0, 1)
D=1
Q = range(0, 1)
from itertools import product
parameters = product(p, q, P, Q)
parameters_list = list(parameters)
len(parameters_list)
```

```
%%time
results = []
best_aic = float("inf")
warnings.filterwarnings('ignore')
for param in parameters_list:
    try:
        model=sma.tsa.statespace.SARIMAX(data.consumption_eur_container, order=(param[0], d, param[1]),
                                         seasonal_order=(param[3], D, param[3], 24)).fit(disp=-1)
    except ValueError:
        continue
    aic = model.aic
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param, model.aic])
warnings.filterwarnings('default')
result_table = panda.DataFrame(results)
result_table.columns = ['parameters', 'aic']
%%time
best_model=sma.tsa.statespace.SARIMAX(data.consumption_eur_container, order=(1, d, 3),  seasonal_order=(3, D, 1, 24)).fit(disp=-1)
print(best_model.summary())
tsplot(best_model.resid[24:], lags=30)
data["arima_model"] = invboxcox(best_model.fittedvalues, lamda)
forecast = invboxcox(best_model.predict(start = data.shape[0], end = data.shape[0]+100), lamda)
forecast = data.arima_model.append(forecast).values[-500:]
actual = data.consumption_eur.values[-400:]
pyplot.figure(figsize=(15, 7))
pyplot.plot(forecast, color='r', label="Модель")
pyplot.title("Модель САРИМА на 500 даних ринку")
pyplot.plot(actual, label="Вхідні дані")
pyplot.legend()
pyplot.axvspan(len(actual), len(forecast), alpha=0.5, color='lightgrey')
pyplot.grid(True)
def code_mean(data, cat_feature, real_feature):
    return dict(data.groupby(cat_feature)[real_feature].mean())
data = pd.DataFrame(dataset)
# data.columns = ["y"]
data.head(n=100)
best_model.plot_diagnostics(figsize=(15,12));
```

**ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)**



**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# Розробка алгоритму прогнозування ліцензійного обсягу абітурієнтів на основі машинного навчання

Виконав студент 6 курсу
групи ПДМ-61
Лакович Михайло Сергійович
Керівник роботи
Жебка Вікторія Вікторівна

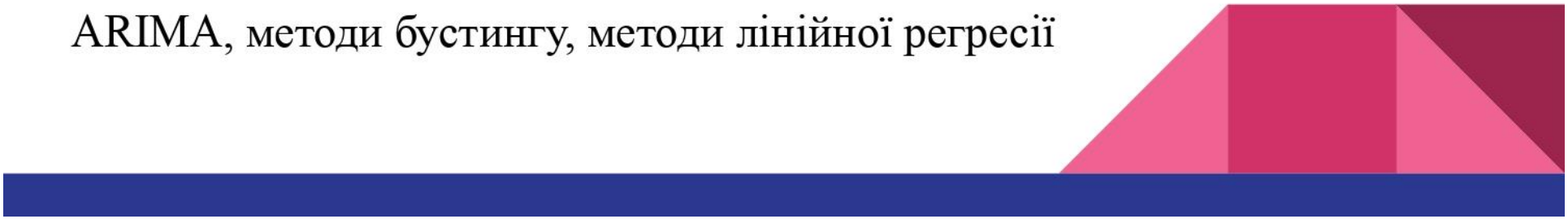
Київ – 2020

Об’єкт дослідження – прогнозування заповнення ліцензійного обсягу університету за допомогою машинного навчання .

Предмет дослідження – алгоритм прогнозування, за допомогою якого спрогнозовано статистику абітурієнтів для вступної кампанії 2021 року.

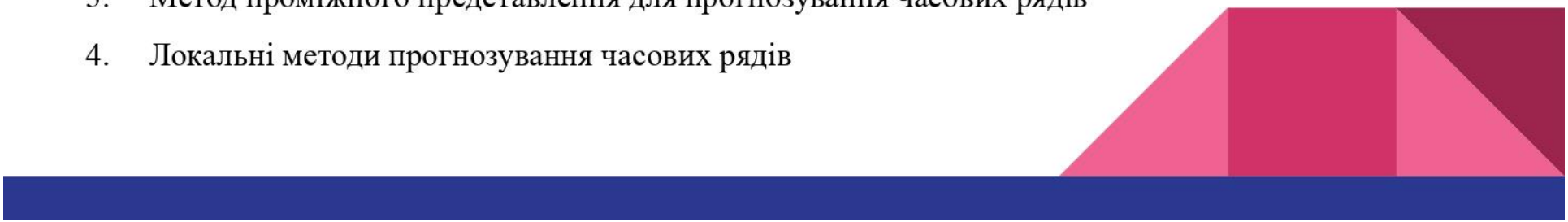
Мета роботи – покращення процесу прогнозування заповнення ліцензійного обсягу університету за допомогою методів машинного навчання.

Методи дослідження – методи експоненціального згладжування, для згладжування вхідний даних, метод побудови моделі SARIMA та ARIMA, методи бустингу, методи лінійної регресії



# Сучасні алгоритми прогнозування

1. Сімейство регресійних методів прогнозування
    1. Лінійна регресія
    2. Множинна лінійна регресія
    3. Логістична регресія
2. Сімейство бустингу
    1. Метод найшвидшого бустингу
    2. XGBoost
3. Метод проміжного представлення для прогнозування часових рядів
4. Локальні методи прогнозування часових рядів



# Аналіз вхідних даних

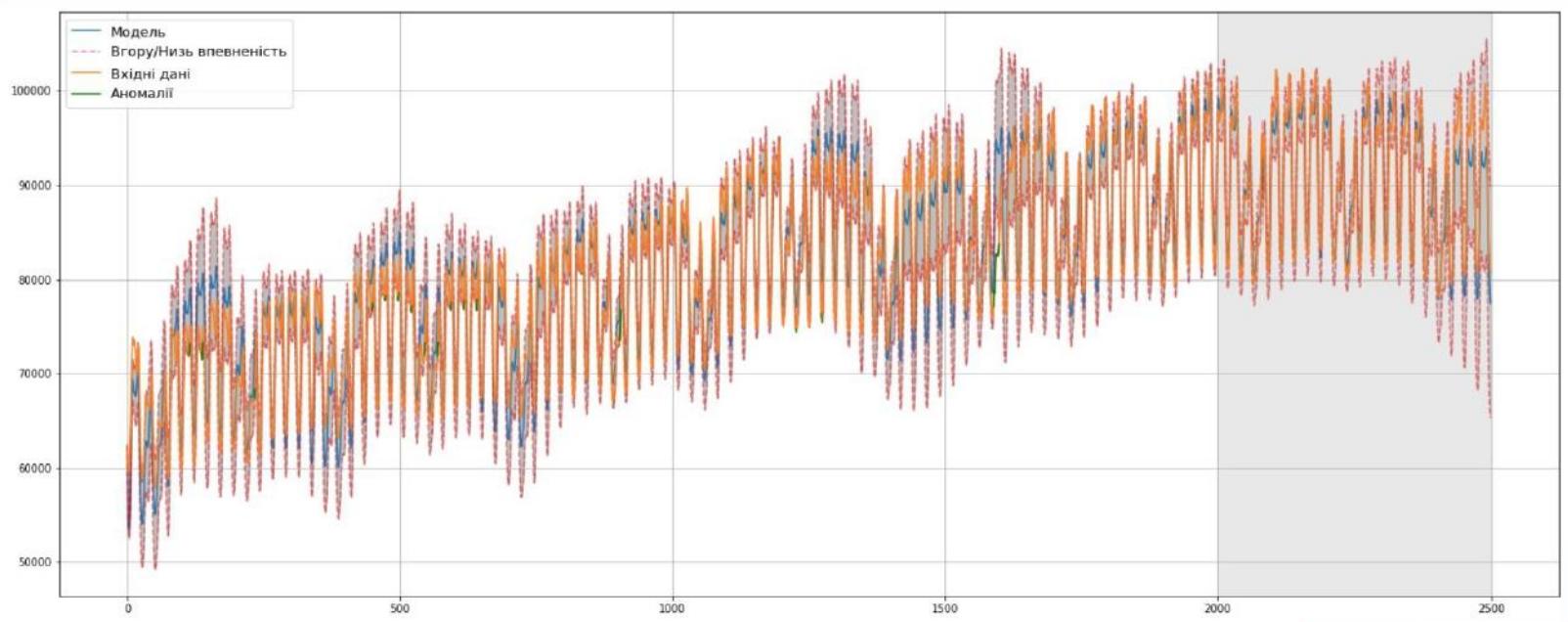
| Рік | Бюджет | Контракт | Всього прийнято | Всього місць | Заяв |
|---|---|---|---|---|---|
| 2010 | 179 | 503 | 682 | 690 | 2851 |
| 2011 | 179 | 511 | 690 | 690 | 2301 |
| 2012 | 235 | 375 | 610 | 610 | 4447 |
| 2013 | 232 | 358 | 590 | 590 | 5269 |
| 2014 | 218 | 347 | 565 | 565 | 4822 |
| 2015 | 214 | 511 | 725 | 715 | 4728 |
| 2016 | 209 | 729 | 938 | 975 | 5846 |
| 2017 | 22 | 849 | 871 | 2362 | 5709 |
| 2018 | 340 | 1070 | 1410 | 1410 | 4276 |
| 2019 | 283 | 1737 | 2020 | 2050 | 5319 |
| 2020 | 281 | 1661 | 1942 | 1980 | 5588 |

# Формула згладжування на основі сезонності

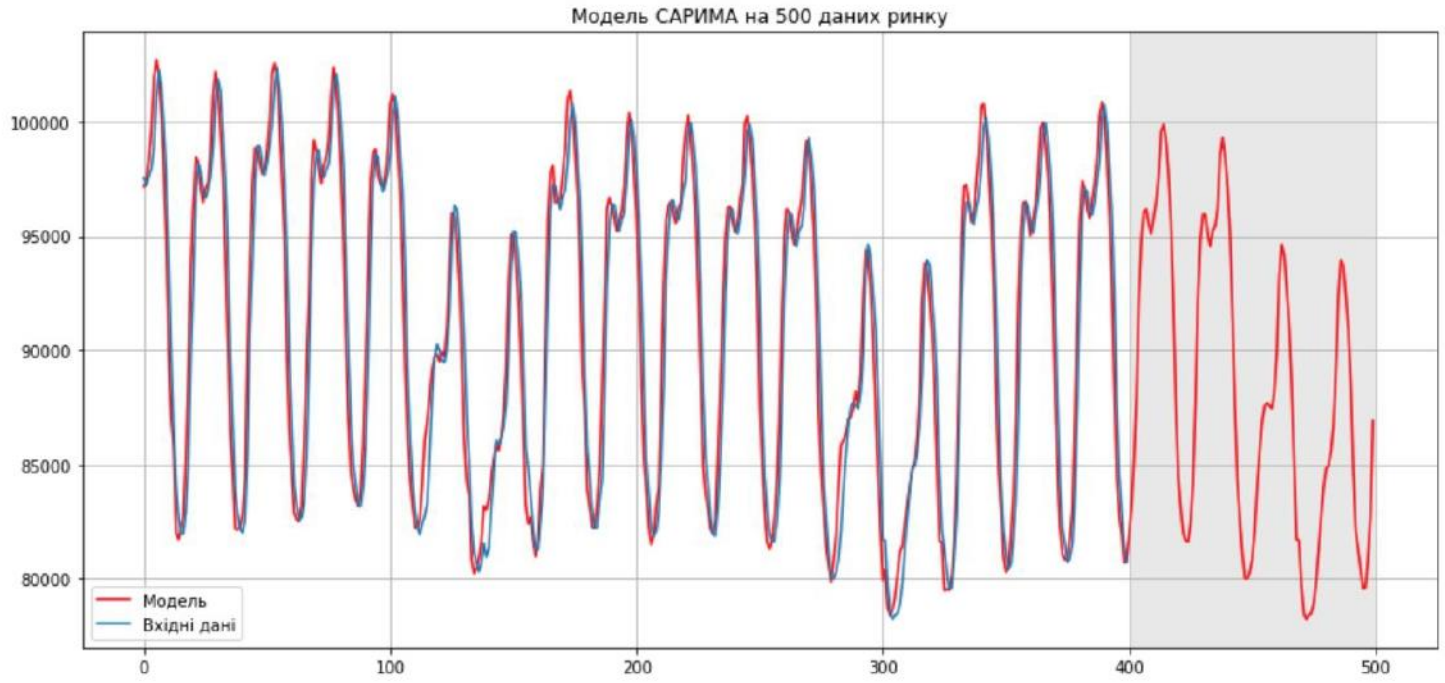
$$
\begin{aligned}
l_i &= \alpha(y_i - s_{i-L}) + (1-\alpha)(l_{i-1} + t_{i-1}) \\
t_i &= \beta(l_i - l_{i-1}) + (1-\beta)t_{i-1} \\
s_i &= \gamma(y_i - l_t) + (1-\gamma)s_{i-L} \\
\hat{y}_{i+m} &= l_i + mt_i + s_{i+1+(m-1)modL}
\end{aligned}
\qquad (3.5)
$$

де $l_i$ – згладжуваний ряд, $t_i$ – значення тренду в поточний період, $s_i$ – оцінка сезонності, $\hat{y}_{x+m}$ – значення лінії тенденції, $\alpha$ - згладжування чиннику рівня, $\beta$ - згладжування чиннику лінії тренду, $\gamma$ - чинник згладжування сезонності $\alpha, \beta, \gamma$ – константи в діапазоні від 0 до 1, $i$ – індекс поточного спостереження.

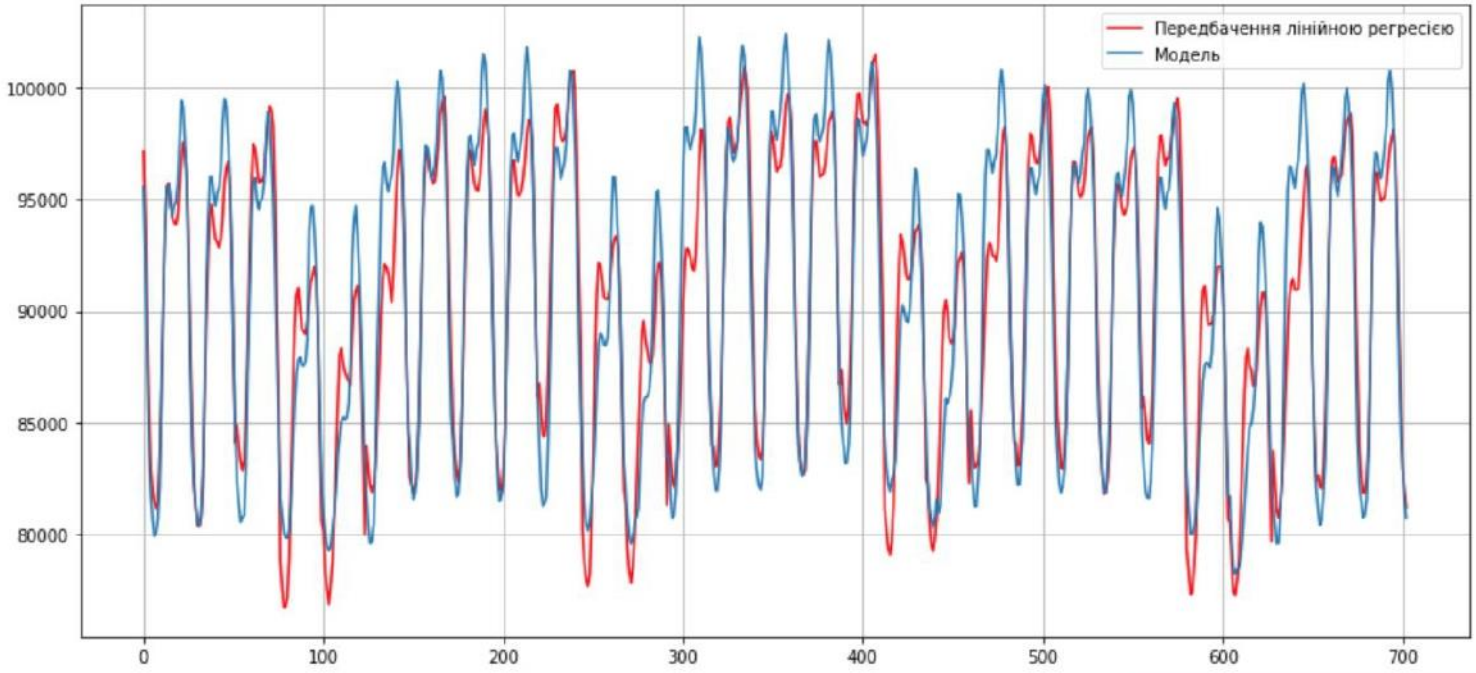
# Результати кросс-валідації



# Результати прогнозування алгоритму

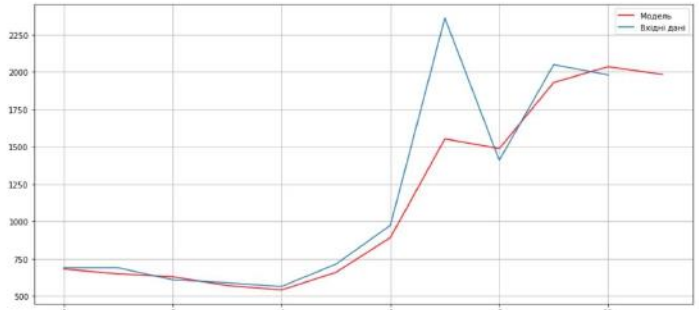


# Порівняння розробленого алгоритму з алгоритмом лінійної регресії

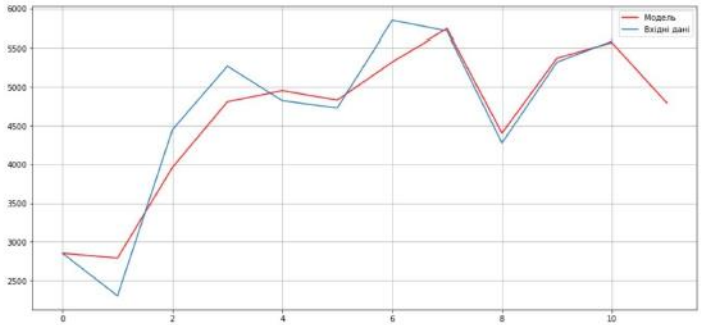


# Навчання та прогнозування алгоритму на наступний рік по:

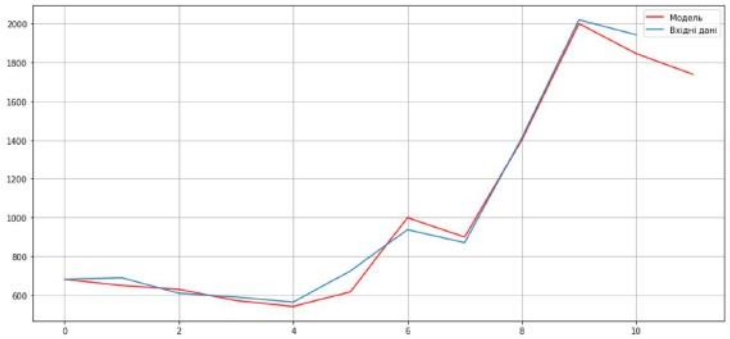
загальній кількості місць




заяв від абітурієнтів




студентів першого курсу



# Апробація

Теза:

- Лакович М.С. “Аналіз популярних алгоритмів машинного навчання” // Науково-практична конференція “Проблеми комп’ютерної інженерії”: 2 грудня 2020 року.

Стаття:

- М.С. Лакович, В.В. Жебка “Використання семантичних мереж онтології в задачах пошукових запитів”, Журнал "Зв'язок", №6, 2020, Київ.

# Висновки

1. Досліджено та проаналізовано роботу сучасних алгоритмів прогнозування.
2. Створена формула для прогнозування часових рядів з урахуванням сезонності.
3. Розроблено більш точний алгоритм прогнозування.
4. Роботу можна продовжувати розвивати, додавши додаток з удосконаленою автоматизацією процесів згладжування та виявлення параметрів.

# Дякую за увагу!