Київський національний університет імені Тараса Шевченка

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

«КОРЕКЦІЯ ЗМІЩЕННЯ ПРОГНОЗОВАНОЇ ТЕМПЕРАТУРИ ПОВІТРЯ НА ОСНОВІ МЕТОДІВМАШИННОГО НАВЧАННЯ»

Кваліфікаційна робота бакалавра
зі спеціальності 123 «Комп'ютерна інженерія»
студента 4 курсу
Владислава ДОМНЮКА

_____

Науковий керівник:
к.ф.-м.н., асистент **Андрій КОНОВАЛОВ**

_____

Рецензент
доцент кафедри квантової радіофізики та наноелектроніки, д. ф.-м. н.
Андрій ГОРЯЧКО

_____

До захисту допускаю:

Завідувач кафедри, к.ф.-м.н.,
доцент Юрій БОЙКО

Ухвалено на засіданні кафедри "___"_____2022 р., протокол №___

КИЇВ - 2022

РЕФЕРАТ

Дипломна робота: 39 с., 10 рис., 5 табл., 1 додаток, 19 джерел.

Мета роботи – удосконалення методу корекції зміщення мінімальної і максимальної добових температур, що прогнозуються моделлю LDAPS.

Модель дослідження – регресія опорних векторів.

В першому розділі визначено мету та завдання для методів прогнозування мінімальної і максимальної добових температур; проведено аналіз попередніх досліджень; визначено інформацію про моделі машинного навчання, що застосовувались у процесі виконання.

У другому розділі визначено методики навчання використаної моделі дослідження, описані ознаки та метрики якості, які застосовувались та яким способом здійснювався підбір параметрів.

У третьому розділі проведено аналіз отриманих значень метрик якості, порівняння їх з наявними результатами опрацьованої літератури та навчально-виробничої практики.

Отримані точності прогнозування мінімальної та максимальної добових температур розроблених SVR моделей – вищі, ніж наведені у літературі для окремих типів моделей (LDAPS, RF, SVR, MLP) без використання ансамблевого підходу.

РЕГРЕСІЯ ОПОРНИХ ВЕКТОРІВ, МАКСИМАЛЬНА ТА МІНІМАЛЬНА ДОБОВІ ТЕМПЕРАТУРИ, МЕТРИКИ, СИСТЕМИ АСИМІЛЯЦІЇ І ПРОГНОЗУВАННЯ ЛОКАЛЬНИХ ДАНИХ, МАШИННЕ НАВЧАННЯ.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

kNN – Метод k-найближчих сусідів

LDAPS – Система асиміляції та прогнозування локальних даних

R^2 – Коефіцієнт детермінації

RMSE – Середньоквадратичне відхилення

RF – Випадковий ліс

ANN – Штучні нейронні мережі

SVR – Метод опорних векторів

MME – Ансамблевий метод

SVM – Машина опорних векторів

RBF – Радіально базисна функція

ВСТУП

Загальновідомо, що прогнози температур повітря відіграють важливу роль в повсякденному житті людей, оскільки вони дають можливість бути готовим як до стихійних природних лих, так і до буденних змін погоди. З розвитком технологій та інформатики метеорологія прийняла більш кількісний підхід, а моделі прогнозів стали доступнішими для дослідників, прогнозистів та інших зацікавлених сторін. Моделі кількісного передбачення погоди описують основні фізичні процеси в атмосфері та враховують їх вплив на змінні моделі, такі як температура, тиск, вітер, водяна пара, хмари та опади. Однією з таких моделей є система асиміляції та прогнозування локальних даних (Local Data Assimilation and Prediction System, LDAPS), яка розроблена для прогнозування мезомасштабних погодних явищ та поєднує широкий спектр спостережуваних метеорологічних даних в єдиний аналіз з певним часовим інтервалом.

Однак метеорологічні змінні корелюються у просторі та часі, ознаки, такі як опади, хмарність, тощо, можуть з'являтися і зникати в часових масштабах, набагато коротших за діапазон прогнозу, що, так чи інакше, впливає на модель LDAPS. Тому використання методів машинного навчання це перспектива покращення даної моделі.

Метою випускної кваліфікаційної роботи є удосконалення методу корекції зміщення мінімальної і максимальної добових температур, що прогнозуються моделлю LDAPS.

РОЗДІЛ 1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1. Чисельний прогноз погоди

Прогноз погоди відноситься до наукового процесу прогнозування стану атмосфери на основі конкретних часових рамок і місць [1]. Кількісне прогнозування погоди використовує комп'ютерні алгоритми для надання прогнозу на основі поточних погодних умов шляхом розв'язування великої системи нелінійних математичних рівнянь, які базуються на конкретних математичних моделях. Більш конкретно, ці моделі визначають систему координат, яка розділяє Землю на тривимірну сітку. Однак, через грубу роздільну здатність сітки та недосконаліть фізичних параметрів, моделі кількісного передбачення погоди зазвичай спрощували детальні характеристики суші, атмосфери та океанських систем. Дуже важливою характеристикою модельної сітки є відстань сітки, тобто горизонтальна відстань сусідніх точок сітки. Чим менша відстань сітки, тим більш детальні структури атмосфери можна визначити за допомогою моделі кількісного передбачення погоди.

Концепція кількісного передбачення погоди була запропонована Льюїсом Фраєм Річардсоном у 1922 році, а її практичне використання почалося в 1955 році після розробки програмованих комп'ютерів [1]. Протягом останніх десятиліть здатність моделей кількісного передбачення погоди передбачати майбутній стан атмосфери постійно покращувалася та досягли надзвичайної точності в прогнозуванні відповідних метеорологічних величин. Сучасні прогнози погоди в значній мірі покладаються на масивні системи чисельного моделювання, якими регулярно керують національні метеорологічні агентства по всьому світу [2].

LDAPS

Модель LDAPS розроблена для прогнозування мезомасштабних (середній розмір метеорологічного явища, приблизно від 10 до 1000 кілометрів по горизонталі) погодних явищ у регіонах, включаючи моря. LDAPS поєднує широкий спектр спостережуваних метеорологічних даних в єдиний аналіз

атмосфери з певним часовим інтервалом. LDAPS має сітку широти-довготи з роздільною здатністю 1,5 км по горизонталі і висоту по вертикалі до 40 км поділену на 70 шарів.

Корекція LDAPS

Було проведено чимало досліджень [3 – 5] щодо удосконалення методу корекції зміщення максимальної та мінімальної добових температур на основі машинного навчання для кількісного передбачення погоди. Автори цих статей використовували різні моделі машинного навчання для порівняння ефективності результатів. За основу цієї роботи взято статтю [3], метою якої було покращення системи асиміляції та прогнозування локальних даних (LDAPS) на основі методів машинного навчання.

В статті [3] описано роботу таких моделей:

1) Випадковий ліс (RF);
2) Штучні нейронні мережі (ANN);
3) Регресія опорних векторів (SVR);
4) Ансамблевий метод (MME).

Результати моделі LDAPS показали, що для прогнозування максимальної добової температури коефіцієнт детермінації R^2 дорівнює 0.69, квадратний корінь із середньоквадратичного відхилення RMSE (Root Mean Square Error) = 2.08°C, а для прогнозування мінімальної добової температури – 0.77 та 1.43 °C відповідно. Щоб покращити ці метрики якості, описані вище методи машинного навчання використовують прогнозовані цією моделлю значення.

В цій літературі [3] після удосконалення моделі LDAPS методами SVR, RF, ANN, MME показники R^2 та RMSE покращились, найбільший результат досягнуто за допомогою MME. Це значення для максимальної температури відповідає R^2 0.78 та RMSE 1.55 °C, а для мінімальної температури – 0.87 та 0.98 °C.

1.2. Методи машинного навчання

Машинне навчання – це наукова галузь, яка перебуває на перетині статистики, штучного інтелекту та комп'ютерних наук і також відома як прогнозна аналітика чи статистичне навчання [6]. Процес навчання автоматизований та покращується на основі досвіду машин протягом усього процесу. Принцип навчання полягає в тому, що вхідні дані разом із вихідними подаються в машину під час фази навчання, і вона визначає закономірності та приймає рішення з мінімальним втручанням людини. (рис. 1.1) [7]. Інформація надходять до машин, і різні алгоритми використовуються для побудови моделей задля навчання машин на цих даних. Вибір алгоритму залежить від типу даних і виду діяльності, яку необхідно автоматизувати.

Алгоритми машинного навчання можна поділити на:

- Класифікація – це процес пошуку або виявлення моделі або функції, яка допомагає розділити дані на дискретні значення. Тут дані розподіляються за різними мітками відповідно до деяких параметрів, наведених у вхідних даних, а потім для даних прогнозуються мітки [8].
- Регресія – це процес пошуку моделі або функції для розрізнення даних на безперервні реальні значення замість використання класів або дискретних значень. Він також може визначити рух розподілу залежно від історичних даних [8].

Рис 1.1. Принцип навчання машин (взято з [7]).

1.2.1. Метод k-найближчих сусідів

Принцип методу k-найближчих сусідів (kNN) полягає в тому, щоб знайти попередньо визначену кількість навчальних прикладів, найближчих за відстанню до нової точки, і передбачити мітку з них. Кількість прикладів може бути визначеною користувачем константою (вивчення k-найближчого сусіда) або змінюватися на основі локальної щільності точок (навчання сусідів на основі фіксованого радіусу) (рис. 1.2). Цей алгоритм відносно простий, але досить потужний, хоча рідко витрачається час на розуміння його обчислювальної складності та практичних питань. Його можна використовувати як для класифікації, так і для регресії з однаковою складністю.

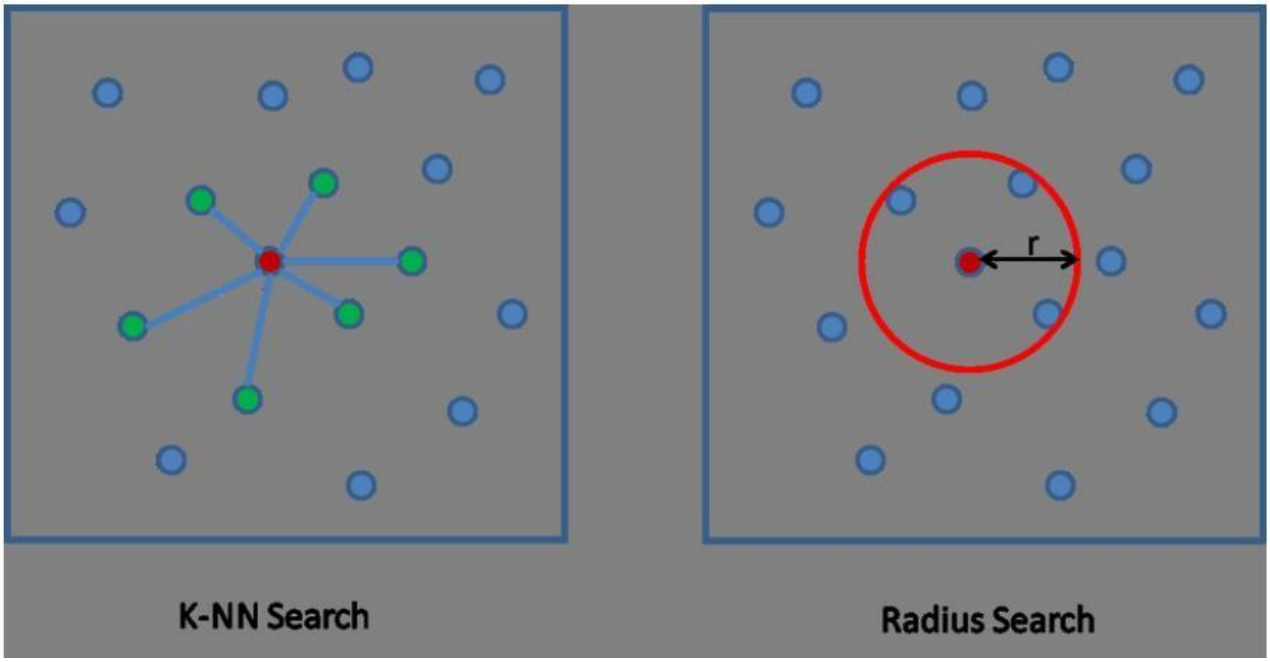

Рис. 1.2. Типи пошуку сусідів в kNN.

Головним параметром в методі найближчих сусідів є кількість сусідів або радіус, за яким відбувається пошук сусідів. Існує залежність між кількістю сусідів (значення k) та кінцевим результатом, так як при низькому значені k модель є занадто конкретною, а при великому – дуже узагальненою.

Класифікація kNN

Класифікація kNN – це тип навчання на основі екземплярів або навчання без узагальнення: вона не намагається побудувати загальну внутрішню модель, а просто зберігає екземпляри навчальних даних. Класифікація обчислюється простою більшістю голосів найближчих сусідів кожної точки, при цьому випадок відноситься до класу, найбільш поширеного серед його K-найближчих сусідів, виміряних функцією відстані [9].

Регресія kNN

Регресія kNN – це метод, який інтуїтивно наближає зв'язок між незалежними змінними та безперервним результатом шляхом усереднення спостережень. Регресія kNN використовує ті ж функції відстані, що й класифікація KNN (табл. 1.1).

identifier	class name	distance function
"euclidean"	EuclideanDistance	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	`sum(|x - y|)`
"minkowski"	MinkowskiDistance	`sum(|x - y|^p)^(1/p)`

Таблиця 1.1. Метричні функції відстані.

Регресія найближчих сусідів використовує рівномірні вагові функції: тобто кожна точка в локальному околиці рівномірно вносить свій внесок у класифікацію точки запиту. За деяких обставин може бути вигідно зважувати точки таким чином, щоб найближчі точки внесли більше в регресію, ніж віддалені точки (рис. 1.3) [9].

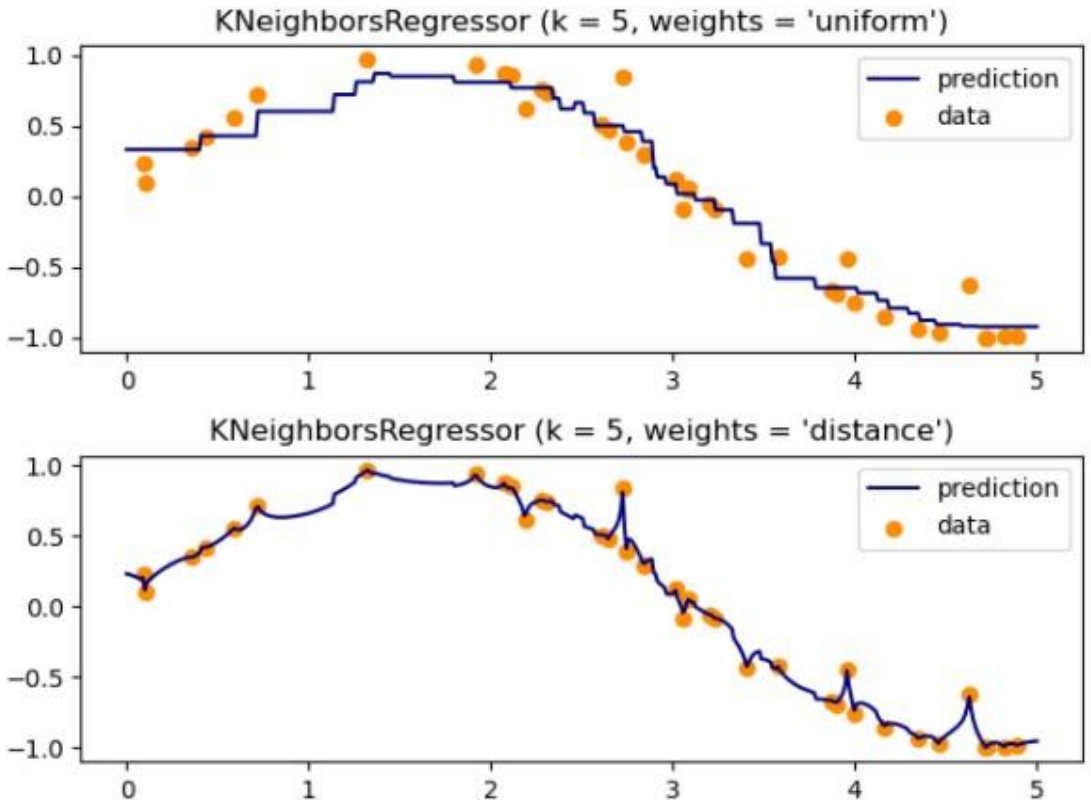

Рис. 1.3. kNN у випадку різних вагових функцій (взято в [9]).

Алгоритми вибору найближчого сусіда:

- Грубої сили (brute force) – алгоритм, що використовується для швидкого обчислення найближчих сусідів, шляхом обчислення відстаней між усіма парами точок у наборі даних. Часова асимптотична складність алгоритму дорівнює $O(DN^2)$, де N – кількість зразків, а D – розмірність. Застосовується здебільшого для невеликих вибірок даних [9].

- KD дерево (k вимірне дерево) – це бінарна деревовидна структура, яка рекурсивно розбиває простір параметрів уздовж осей даних, поділяючи його на вкладені області, в які подаються точки даних. Розбиття виконується тільки вздовж осей даних, тому не потрібно обчислювати D у відстані. Тому часова складність алгоритму обчислення відстані між сусідами становить $O[log(N)]$. Підхід дерева KD швидкий для низькорозмірних (D<20) сусідів, але стає неефективним як D виростає дуже великим. Це один із проявів так званого «прокляття розмірності» [9].

- Кулькове дерево – алгоритм, що рекурсивно розділяє дані на вузли, визначені центроїдом (центральна точка многокутника) C і радіус r так, що кожна точка у вузлі лежить у межах гіперсфери, визначеної r і C. Через сферичну геометрію вузлів кулькового дерева він може перевершити *KD*-дерево у великих розмірах, хоча фактична продуктивність сильно залежить від структури навчальних даних [9].

Під час **науково-виробничої практики** дослідження показало, що використання методу k найближчих сусідів (kNN) не є ефективним, оскільки він не зміг перевершити жодну з використаних в статті [3] моделей. Тому **в випускній кваліфікаційній роботі** дослідження зосереджене на покращенні наведених в публікації [3] результатів удосконаленням моделі регресії опорних векторів (SVR).

1.2.2. Машина опорних векторів

Машина опорних векторів (SVM) є одним із найпопулярніших алгоритмів контрольованого навчання, який використовується для задач класифікації та регресії.

Метою цього алгоритму є створення найкращої лінії або межі рішення, яка може розділити n-вимірний простір на класи, щоб ми могли легко помістити нову точку даних у правильну категорію в майбутньому.

Щоб зрозуміти роботу SVM, потрібно знати такі терміни, як: гіперплощина – межа рішення для прогнозування безперервного результату; опорні вектори – точки (вектори) даних по обидва боки від гіперплощини, які є найближчими до неї (для побудови потрібної лінії, яка показує прогнозований результат алгоритму)[10].

Алгоритми SVM використовують набір математичних функцій, які визначаються як ядро. Зазвичай вони застосовуються для пошуку гіперплощини у просторі вищих вимірів. Найбільш широко використовуваними функціями є (табл. 1.2): лінійна, поліноміальна, сигмоїдна

та радіально базисна (Гаусова, RBF). Кожне з цих ядер обирається залежно від набору даних.

Name	Definition	Parameters
Gaussian kernel	$K(x_i, x) = \exp\left(-\gamma \|x_i - x\|^2\right)$	γ
Linear kernel	$K(x_i, x) = (x_i \cdot x)$	-
Sigmoid kernel	$K(x_i, x) = \tanh(\gamma(x_i \cdot x) + R)$	γ, R
Polynomial kernel	$K(x_i, x) = (\gamma(x_i \cdot x) + R)^d$	γ, R, d

Таблиця 1.2. Допустимі функції ядра (взято в [11]).

SVM є потужним інструментом, але його часова складність обчислень швидко зростає із збільшенням кількості навчальних векторів. Вона масштабується між $O(n_{features} \times n^2_{samples})$ і $O(n_{features} \times n^3_{samples})$, в залежності від набору даних.

Регресія опорних векторів

Регресія опорних векторів (SVR) використовує ті ж принципи, що й класифікація опорних векторів, але застосовується для прогнозування дискретних значень. Модель, створена за допомогою класифікації опорних векторів, залежить лише від підмножини навчальних даних, оскільки функція вартості для побудови моделі не піклується про точки навчання, які лежать за межами. Аналогічно, модель, створена за допомогою SVR, залежить лише від підмножини навчальних даних, оскільки функція вартості ігнорує приклади, прогноз яких близький до цільових точок [12]. Даний алгоритм також використовує ідентичні функції ядра (рис. 1.4)

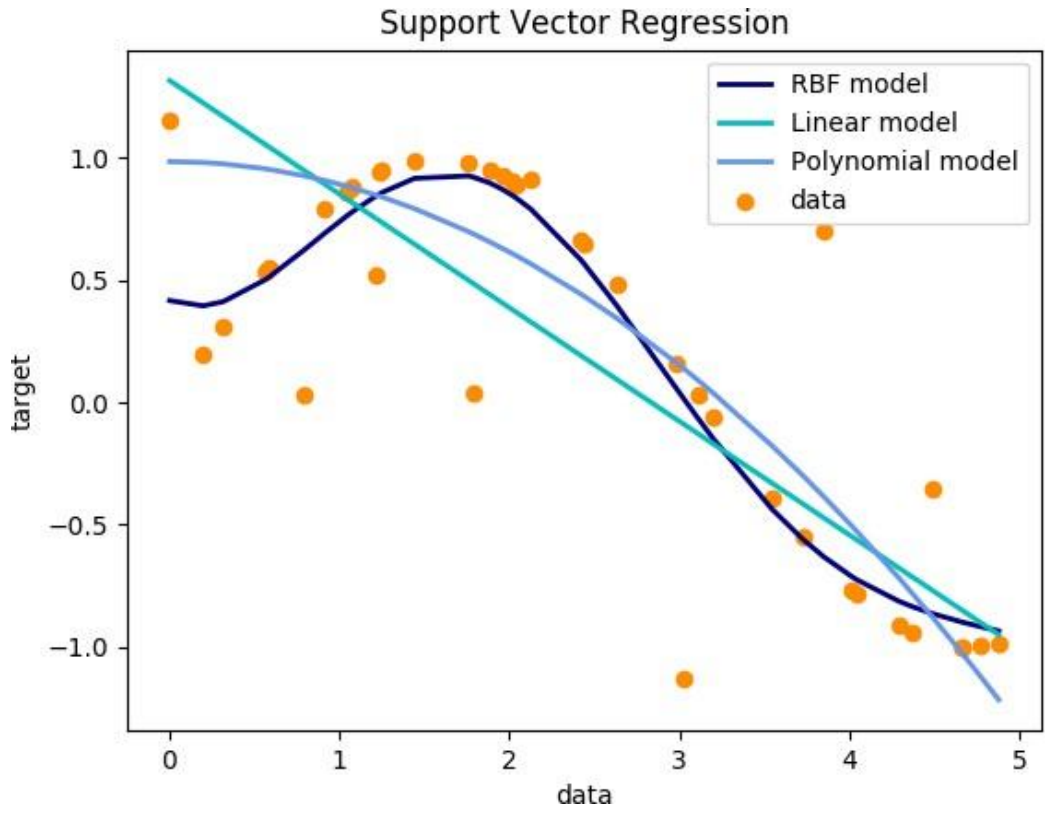

Рис. 1.4. Використання різних ядер в SVR: Linear, Polynomial, RBF (взято в [13]).

Важливим параметром для регресії є епсилон (ε) – це відстані навколо гіперплощини, на якій проведені граничні лінії. Він використовується для створення епсилон-трубки, в межах якої не застосовується штраф у функції втрати при навчанні з точками, та повністю залежить від цільових значень у навчальному наборі. Якщо епсилон перевищує діапазон цих значень, ми не можемо очікувати хорошого результату (рис. 1.5).

На відміну від інших методів регресії, які намагаються мінімізувати похибку між реальним і прогнозованим значенням, SVR намагається вмістити найкращу лінію в межах порогового значення (відстань між гіперплощиною та граничною лінією).

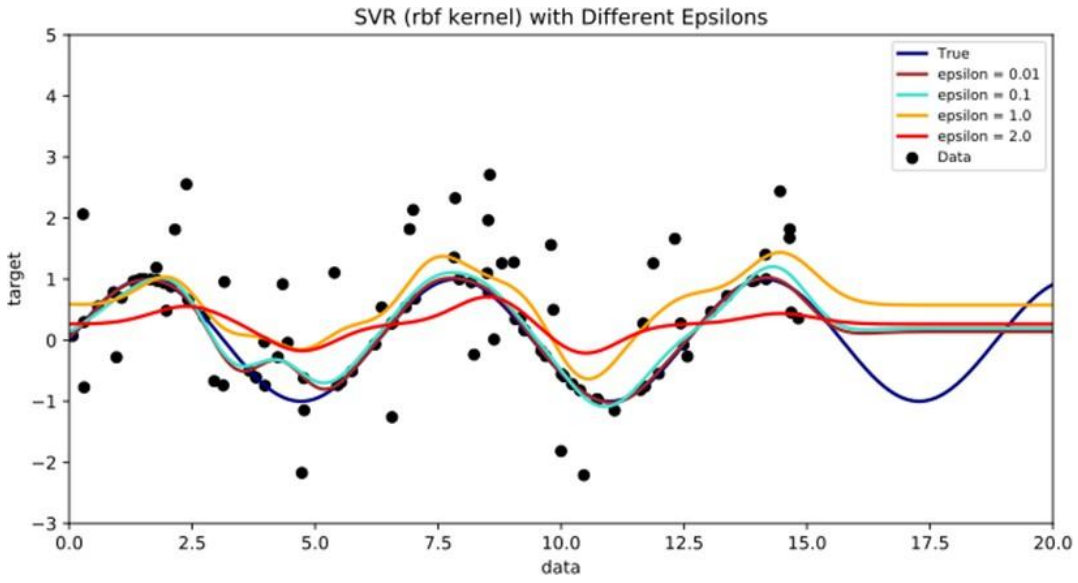

Рис. 1.5. Використання різних епсилон в SVR з RBF ядром (взято в [14]).

1.3. Постановка задачі дослідження

Метою випускної кваліфікаційної роботи є удосконалення методу корекції зміщення мінімальної і максимальної добових температур, що прогнозуються моделлю LDAPS.

Задля виконання мети були поставлені такі задачі:

1. Побудувати моделі прогнозування мінімальної і максимальної добових температур на основі регресії опорних векторів (SVR).
2. Підібрати параметри побудованих моделей для максимізації точності прогнозування температур.
3. Проаналізувати отримані значення метрик якості коефіцієнта детермінації (R^2) та середньоквадратичного відхилення (RMSE) з результатами вже проведеного дослідження та науково-виробничої практики.

РОЗДІЛ 2. МЕТОДИКА ДОСЛІДЖЕНЬ

2.1. Підготовка даних

Набір даних містить 25 атрибутів (табл. 2.1), що, окрім дати, є числовими значеннями. Після імпортування даних у вигляді таблиці в середовище виконання їх було перетворено на масиви, які, як і в статті [3], поділялись на навчальні та тестові вибірки. Як і в роботі [3], у дослідженнях цієї роботи використовувалась так звана методика «hindcast validation» (ретроспективна перевірка), суть якої полягає у перенавчанні моделі на всіх доступних даних кожного разу перед прогнозуванням наступного значення температури. Наприклад, для прогнозування максимальної та мінімальної температури повітря 1 серпня 2016 року модель навчається на даних від першого дня періоду дослідження (1 січня 2015 року) до 31 липня 2016 року. Також до усієї інформації, що використовувалась виконувалась стандартизація (виділення середнього значення та масштабування до одиничної дисперсії).

1	Station - Номер метеостанції: 1 to 25
2	Date - Поточний день: yyyy-mm-dd ('2013-06-30' to '2017-08-30')
3	Present_Tmax - Максимальна температура повітря від 0 до 21 год поточного дня (Â°C): 20 to 37.6
4	Present_Tmin - Мінімальна температура повітря від 0 до 21 год поточного дня (Â°C): 11.3 to 29.9
5	LDAPS_RHmin - Прогноз моделі LDAPS наступного -мінімальна відносна вологість (%): 19.8 to 98.5
6	LDAPS_RHmax - Прогноз моделі LDAPS наступного -максимальна відносна вологість (%):58.9 to 100
7	LDAPS_Tmax_lapse - LDAPS прогноз максимальної температури повітря на наступний день (Â°C): 17.6 to 38.5
8	LDAPS_Tmin_lapse - LDAPS прогноз мінімальної температури повітря на наступний день (Â°C): 14.3 to 29.6
9	LDAPS_WS - Прогноз моделі LDAPS середньої швидкості вітру на наступний день (m/s): 2.9 to 21.9
10	LDAPS_LH - Прогноз моделі LDAPS середнього прихованого теплового потоку на наступний день (W/m2): -13.6 to 213.4
11	LDAPS_CC1 - Прогноз моделі LDAPS за 1-й 6-годинний розріз середнього хмарного покриву наступного дня (0-5 h) (%): 0 to 0.97
12	LDAPS_CC2 - Прогноз моделі LDAPS за 2-й 6-годинний розріз середнього хмарного покриву наступного дня (6-11 h) (%): 0 to 0.97
13	LDAPS_CC3 - Прогноз моделі LDAPS за 3-й 6-годинний розріз середнього хмарного покриву наступного дня (12-17 h) (%): 0 to 0.98
14	LDAPS_CC4 - Прогноз моделі LDAPS за 4-й 6-годинний розріз середнього хмарного покриву наступного дня (18-23 h) (%): 0 to 0.97
15	LDAPS_PPT1 - Прогноз моделі LDAPS середньої кількості опадів за 1-й 6-годинний розріз на наступний день (0-5 h) (%): 0 to 23.7
16	LDAPS_PPT2 - Прогноз моделі LDAPS середньої кількості опадів за 2-й 6-годинний розріз на наступний день (6-11 h) (%): 0 to 21.6
17	LDAPS_PPT3 - Прогноз моделі LDAPS середньої кількості опадів за 3-й 6-годинний розріз на наступний день (12-17 h) (%): 0 to 15.8
18	LDAPS_PPT4 - Прогноз моделі LDAPS середньої кількості опадів за 4-й 6-годинний розріз на наступний день (18-23 h) (%): 0 to 16.7
19	lat - Широта (Â°): 37.456 to 37.645
20	lon - Довгота (Â°): 126.826 to 127.135
21	DEM - Висота (m): 12.4 to 212.3
22	Slope - Ухил (Â°): 0.1 to 5.2
23	Solar radiation - Сонячна радіація, що надходить за добу (wh/m2): 4329.5 to 5992.9
24	Next_Tmax - Максимальна температура повітря наступного дня (Â°C): 17.4 to 38.9
25	Next_Tmin - Мінімальна температура повітря наступного дня (Â°C): 11.3 to 29.8

Таблиця 2.1. Атрибути набору даних.

2.2. Програмні інструменти

Інструментом розробки програмного коду було обрано мову програмування Python. Розглянемо переваги та недоліки використання цього інструменту [15].

Переваги:

- Python підтримує як об'єктно-орієнтовану, так і процедурну моделі програмування;
- Це мова програмування загального призначення, яка є легкою для вивчення та швидко розвивається;
- Python має величезну кількість вбудованих бібліотек для аналізу даних, маніпулювання даними та машинного навчання;
- Python є крос-платформленим та безкоштовним.

Недоліки:

- Обмежена швидкість. Python часто призводить до повільного виконання в порівнянні з іншими мовами програмування;
- Проблеми з потоками.

Середовищем розробки обрано Google Colaboratory, сервіс на основі Jupyter Notebook. Він дозволяє будь-кому писати та виконувати довільний код на Python через браузер і особливо добре підходить для машинного навчання. Також Google Colaboratory не вимагає налаштування для використання. Це середовище працює на базі Ubuntu-18.04 та використовує процесор Intel Xeon з двома ядрами 2,2 ГГц і 12 ГБ оперативної пам’яті (з можливістю збільшення до 25ГБ).

Бібліотеки Python, що використовувались:

- Numpy – це бібліотека, яка надає підтримку багатовимірних масивів, різних похідних об’єктів, таких як замасковані масиви (можуть мати відсутні або недійсні записи) та матриці, а також набір підпрограм для швидких операцій з масивами, включаючи математичні, логічні, маніпуляції з формою, сортування, вибір, введення/виводу [16];

- Pandas – надає швидкі, гнучкі та виразні структури даних, розроблені для того, щоб зробити роботу над даними легкою та інтуїтивно зрозумілою (інструмент аналізу та маніпуляцій над даними). Добре підходить для роботи з табличними типами даних [17];
- Scikit-learn [18]:
  - StandartScaller – надає можливість стандартизації даних;
  - Metrics – надає функції обчислення метрик;
  - SVM – додає функціонал моделі SVR.
- SciPy [19]:
  - Optimize – надає функції для мінімізації (або максимізації) цільових функцій.

2.3. Метрика якості регресора

Як метрики якості були використані два показники точності: коефіцієнт детермінації (R^2) та квадратний корінь із середньоквадратичного відхилення (RMSE).

R^2 та RMSE вказують міру того, наскільки точно передбачувані моделлю дані співпадають з очікуваними. Максимальне значення R^2 дорівнює 1.

$$R^2 = 1 - \frac{\sum_1^n (y_i - y_{pr})^2}{\sum_1^n (y_i - y_{avr})^2}, \quad y_{avr} = \frac{1}{n}\sum_1^n y_i$$

$$RMSE = \sqrt{\sum_1^n \frac{(y_{pr} - y_i)^2}{n}},$$

де y_i – виміряне значення, y_{pr} – прогнозоване значення кожної моделі, а n – кількість зразків.

2.4. Підбір параметрів

Навчання SVR проведено двома способами з метою знаходження найкращого регресора для прогнозування як максимальної, так і мінімальної добових температур:

- Перший варіант – це навчання моделі по кожній станції окремо (не застосовувався в статті [3]). Для застосування даного методу, виключається вплив характеристик розташування (широта, довгота) та топографії (висота, ухил) станцій.
- Другій варіант – це навчання моделі по всім станціям разом. В даному методі використовуються усі атрибути навчальної вибірки.

В досліджуваних SVR моделях використовувалось ядро за замовчуванням (RBF). Параметри які варіювались: параметр регуляризації C, epsilon (визначає епсилон-трубку), коефіцієнт ядра gamma (визначає ширину ядра).

Підбір параметрів здійснювався як самостійно, так і автоматично за допомогою функції minimize пакету optimize (описано у підрозділі 2.2.). Цей підхід використовувався для знаходження змінних при яких мінімізується RMSE.

РОЗДІЛ 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1. Результати прогнозування за допомогою SVR.

Усі значення метрик були отримані за допомогою виразів, які описані в підрозділі 2.3., а принцип підбору ознак даних та параметрів для регресорів в підрозділі 2.4.

На усіх рисунках в даному підрозділі показано найкращі отримані значення R^2 та RMSE по рокам в залежності від методів навчання SVR.

Під час дослідження моделей для прогнозування максимальної добової температури найоптимальнішим виявився варіант з їх навчанням загально по всім станціям (рис. 3.1). Результати показали, що для тестової вибірки 2015 р. та періоду від 2015 р. по 2017 р. SVR цього типу однозначно перевершив варіант з навчанням по кожній станції окремо. На відміну від попередньо описаних років, 2016 р. та 2017 р. мають в обох випадках ідентичні R^2, тому перевага методики вирішується лише за допомогою RMSE. У 2016 р. RMSE моделі з навчанням по всім станціям разом становить 1.47 °C, що на 0.01 °C більше, ніж в іншому типові, але в 2017 р. спостерігається протилежна ситуація і тепер RMSE дорівнює 1.68 °C і є на 0.01 °C меншим.

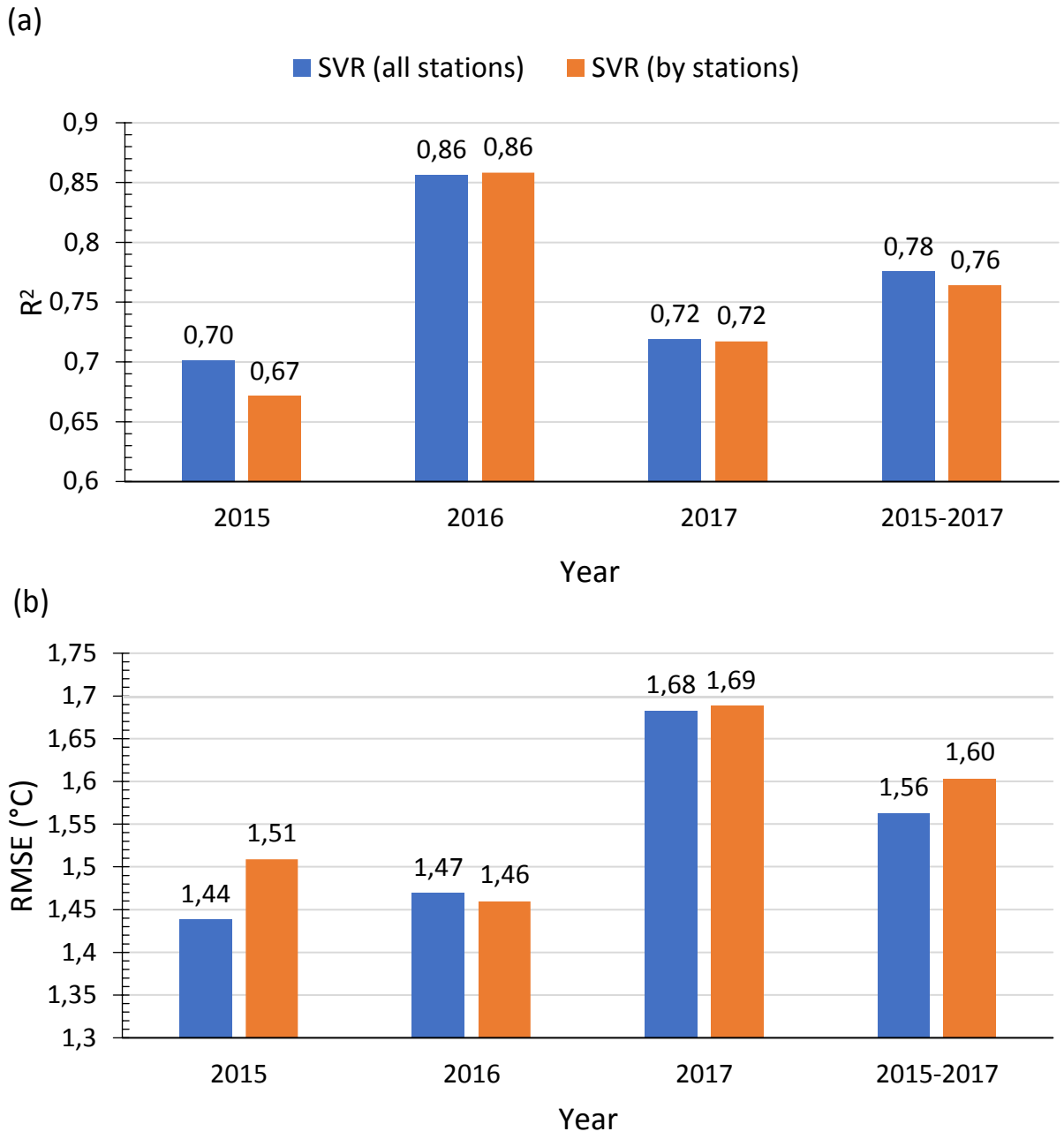

Рис. 3.1. Порівняльна характеристика коефіцієнта детермінації (a) та RMSE (b) для максимальної добової температури по роках в залежності від типу навчання SVR (загально по всім станціям “all stations”, по кожній станції окремо ”by stations”) з використанням hindcast validation.

Отримати наведені значення вдалось, використовуючи параметри у таблиці 3.1.

(a)

SVR (all stations)			
Year	C	gamma	epsilon
2015	113	$1.08 \cdot 10^{-3}$	0.373
2016	107	$1.16 \cdot 10^{-4}$	0.358
2017	121	$1.27 \cdot 10^{-4}$	0.266
2015-2017	98	$2.01 \cdot 10^{-4}$	0.317

(b)

SVR (by stations)			
Year	C	gamma	epsilon
2015	329	$2.04 \cdot 10^{-3}$	0.497
2016	111	$1.43 \cdot 10^{-4}$	0.289
2017	118	$1.64 \cdot 10^{-4}$	0.162
2015-2017	186	$1.15 \cdot 10^{-4}$	0.286

Таблиця 3.1. Параметри моделей SVR з навчанням по всім станціям разом (a) та навчанням по кожній станції окремо (b) до рис. 3.1.

На рис. 3.2 видно, що найкраща методика навчання регресорів для мінімальної добової температури – обернена до вибраної для максимальної добової температури, оскільки отримані значення метрик з методикою використання SVR для кожної станції окремо перевершують результати іншого способу в усіх випадках.

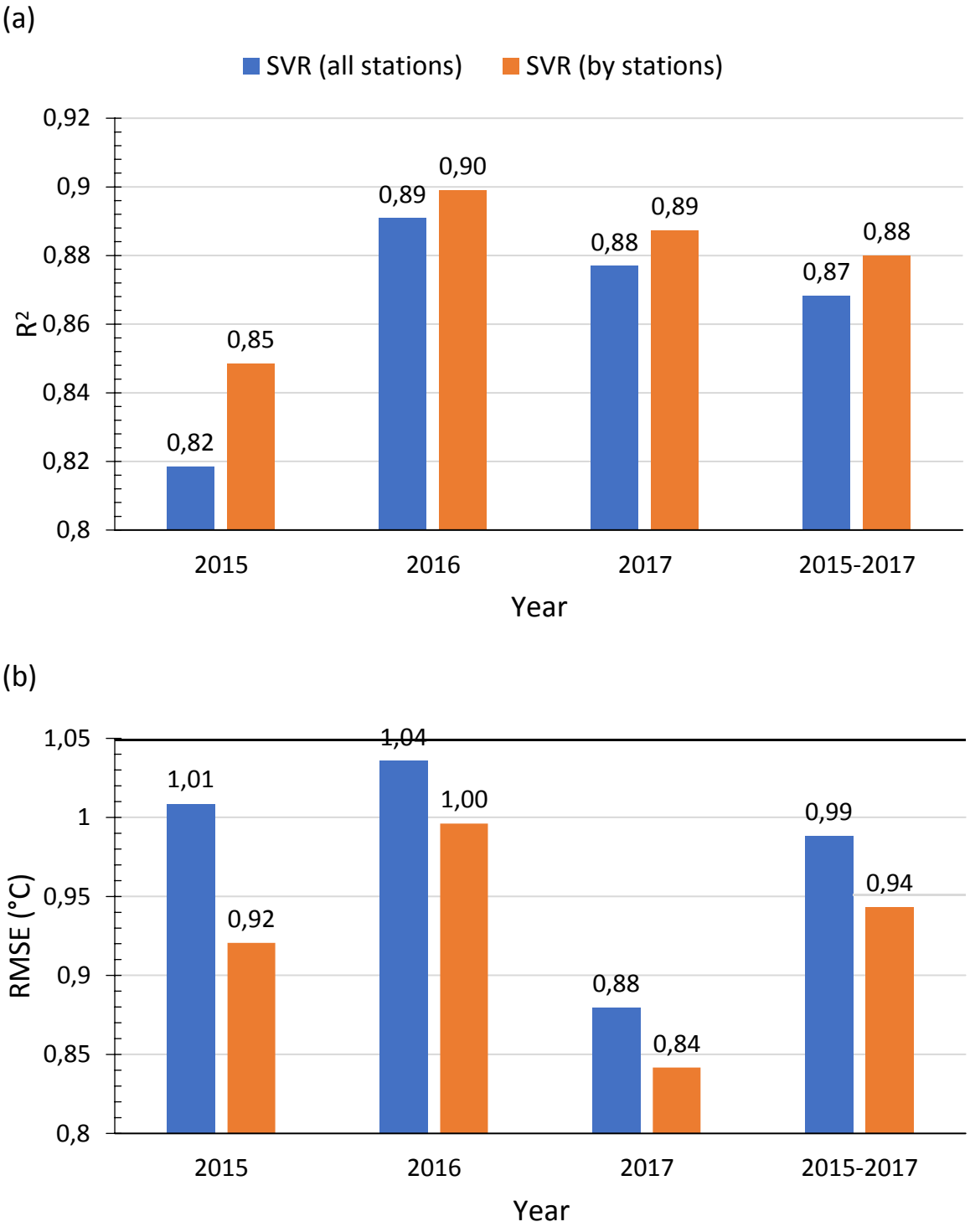

Рис. 3.2. Порівняльна характеристика коефіцієнта детермінації (a) та RMSE (b) для мінімальної добової температури по роках в залежності від типу навчання SVR (загально по всім станціям “all stations”, по кожній станції окремо ”by stations”) з використанням hindcast validation.

Отримати наведені значення вдалось, використовуючи параметри наведені у таблиці 3.2.

(a)

SVR (all stations)			
Year	C	gamma	epsilon
2015	364	$3.96 \cdot 10^{-4}$	0.424
2016	133	$3.55 \cdot 10^{-4}$	0.179
2017	128	$2.83 \cdot 10^{-4}$	0.247
2015-2017	288	$2.28 \cdot 10^{-4}$	0.066

(b)

SVR (by stations)			
Year	C	gamma	epsilon
2015	346	$1.59 \cdot 10^{-4}$	0.245
2016	365	$1.56 \cdot 10^{-4}$	0.155
2017	114	$1.09 \cdot 10^{-4}$	0.340
2015-2017	250	$1.38 \cdot 10^{-4}$	0.217

Таблиця 3.2. Параметри моделей SVR з навчанням по всім станціям разом (a) та навчанням по кожній станції окремо (b) до рис. 3.2.

3.2. Аналіз отриманих результатів

Щоб порівняти отримані щорічні результати hindcast validation SVR для максимальної (рис. 3.3) та мінімальної (рис. 3.4) добових температур використано моделі статті [3], такі як: LDAPS, SVR, MME. Значення досліджуваного методу у всіх випадках перевершили результати LDAPS, проте для інших моделей не все так однозначно.

Для максимальної добової температури в 2015 р. дослідження показало ефективність над SVR [3] та MME [3]. В 2016 р. та 2017 р. R^2 є меншим ніж в цих моделях(рис. 3.3 (a)) , але в обох випадках отриманий RMSE – менший ніж в SVR [3] (рис. 3.3 (b)).

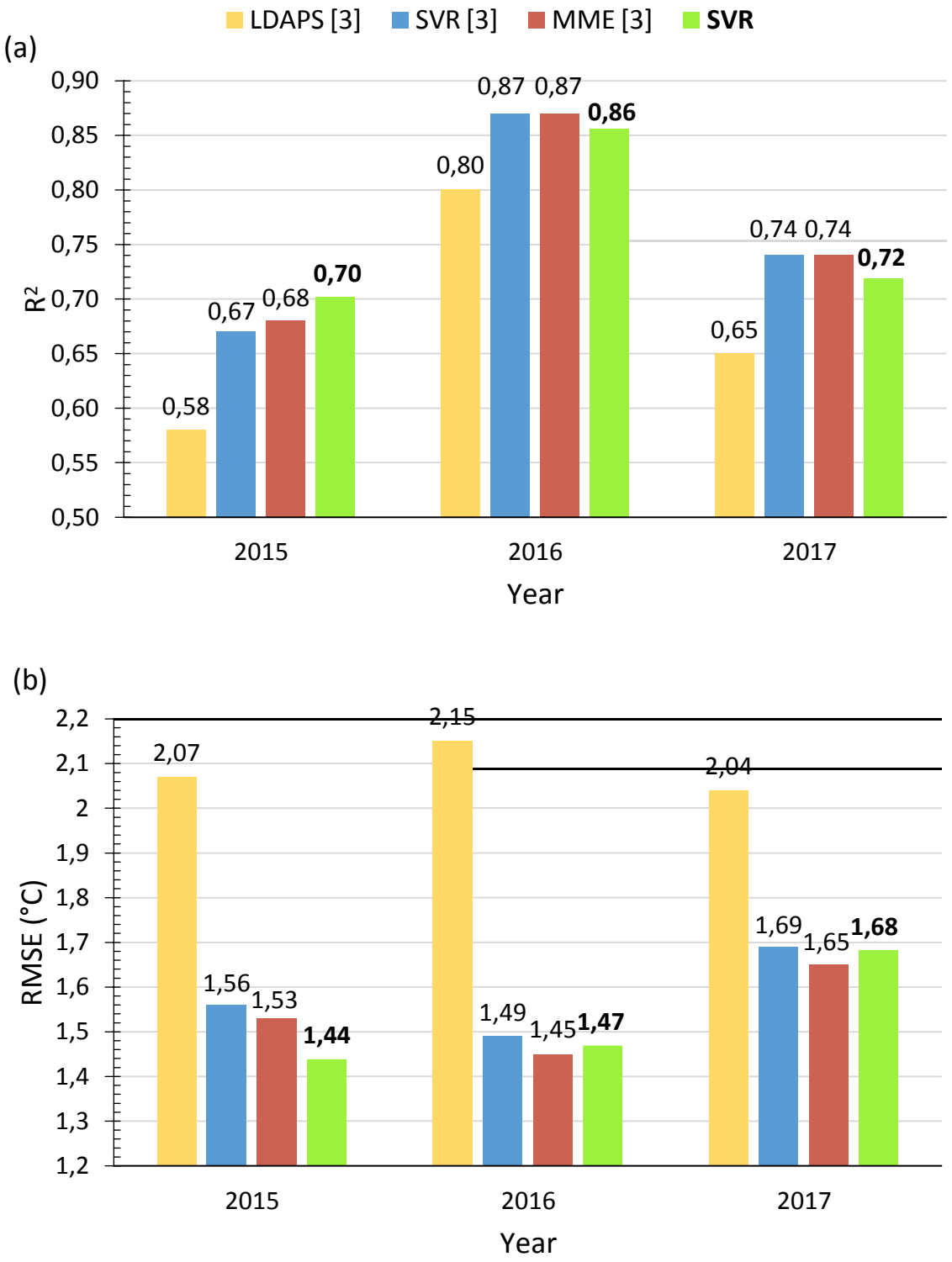

Рис. 3.3. Порівняльна характеристика коефіцієнта детермінації (a) та RMSE (b) для максимальної добової температур в щорічних результатах hindcast validation моделей LDAPS [3], SVR [3], MME [3], **SVR**.

Для мінімальної добової температури в 2015 р. дослідження також показало суттєву ефективність над SVR [3] та MME [3]. В 2016 р. та 2017 р. R^2 є більшим ніж SVR [3] та рівним з MME [3] (рис. 3.3 (a)). В цих випадках отриманий RMSE є кращим ніж SVR [3], проте перевершити значення MME [3] вдалось лише в 2016 р., оскільки в 2017 р. вони зрівнялись (рис. 3.3 (b)).

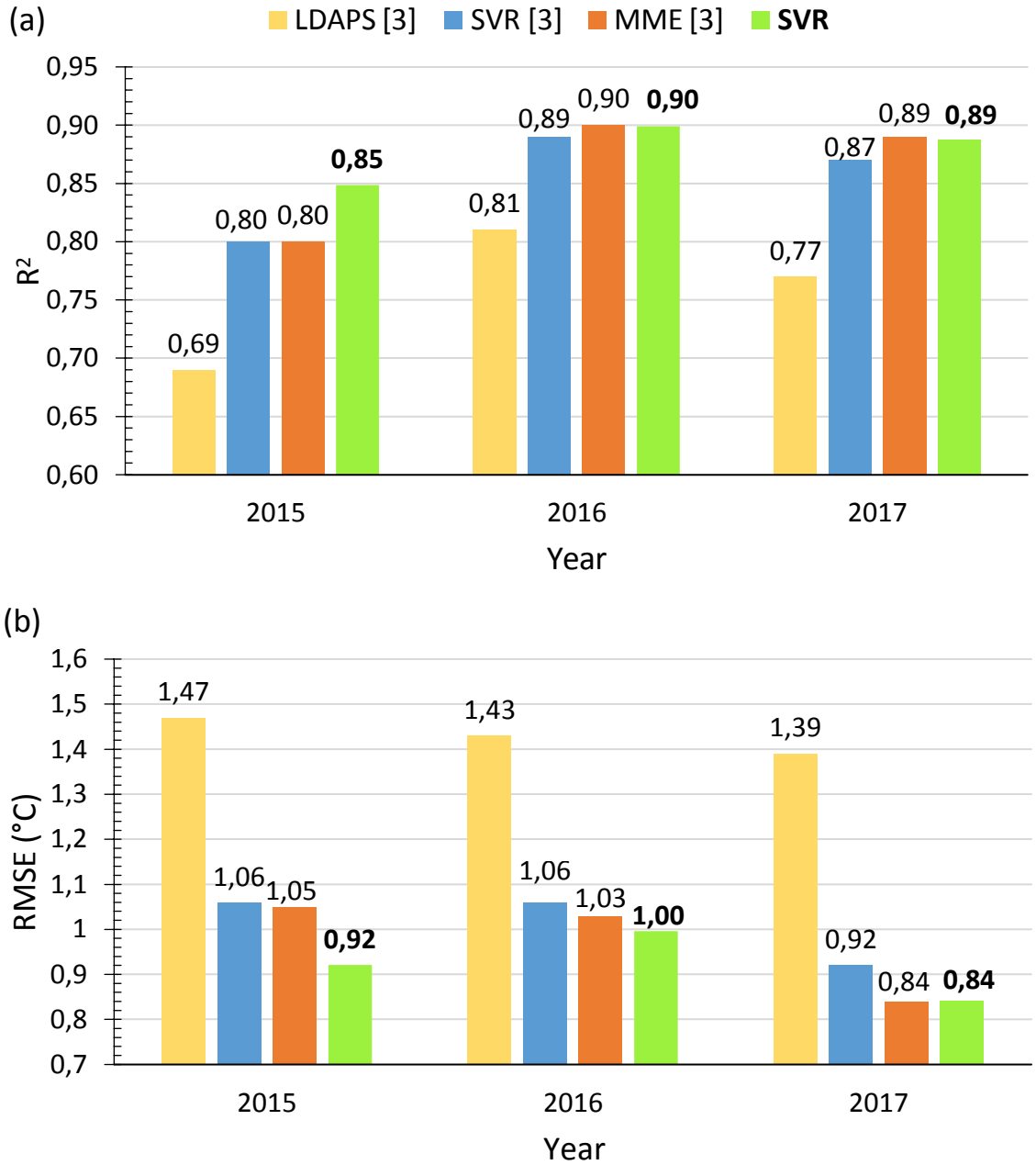

Рис. 3.4. Порівняльна характеристика коефіцієнта детермінації (a) та RMSE (b) для мінімальної добової температур в щорічних результатах hindcast validation моделей LDAPS [3], SVR [3], MME [3], **SVR**.

На рис. 3.5 показано порівняння отриманих значень метрик якості R^2 та RMSE розроблених SVR-регресорів з результатами моделей машинного навчання, наведених в статті [3], та методу k-найближчих сусідів (kNN), дослідженого під час навчальної практики. Часовий діапазон тестової вибірки, що використовувалась для цього рисунку – це загальний її період, тобто від 2015 р. по 2017 р.

Для мінімальної добової температури досліджені значення метрик R^2 та RMSE перевершили всі наведені методи. Вони становлять 0.88 та 0.94 °C, що на 0.01 та 0.04 °C краще попереднього лідера, тобто моделі MME.

Для максимальної добової температури отримане значення R^2 збіглося з найбільшим варіантом і дорівнює 0.78, проте RMSE перевершив усі результати моделей (SVR [3] включно), окрім ансамблю моделей MME.

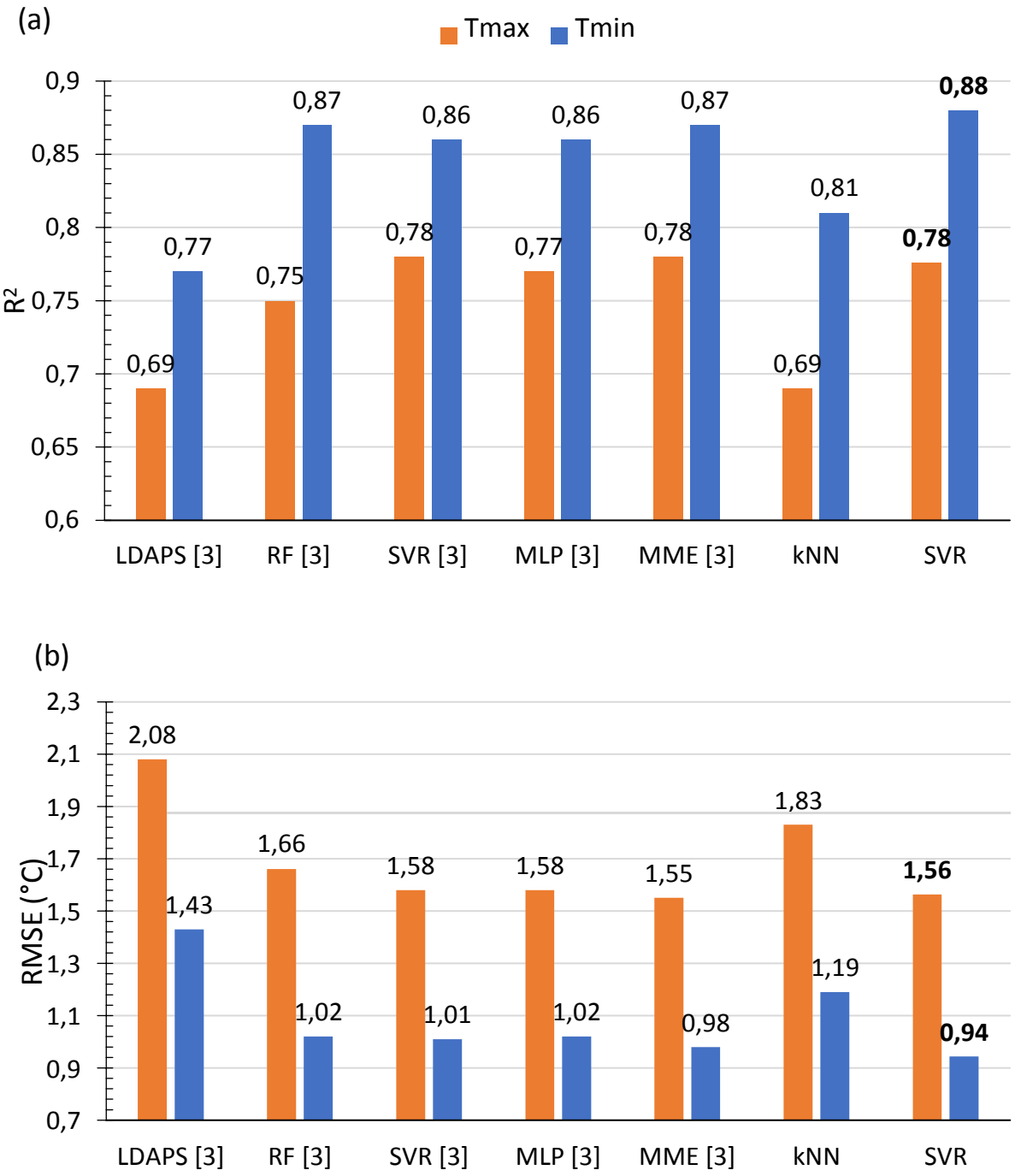

Рис. 3.5. Порівняльна характеристика коефіцієнта детермінації (a) та RMSE (b) для максимальної (Tmax) та мінімальної (Tmin) добових температур в моделях дослідження.

ВИСНОВКИ

В результаті виконання випускної кваліфікаційної роботи:

1. Побудовано моделі прогнозування мінімальної і максимальної добових температур на основі регресії опорних векторів (SVR) з використанням двох способів навчання: по всім станціям разом та по кожній станції окремо.
2. Встановлено перевагу методики навчання SVR-регресорів загально по всім станціям для максимальної добової температури та окремо по кожній станції для мінімальної добової температури.
3. Підібрано параметри для максимізації коефіцієнту детермінації (R^2) та мінімізації середньоквадратичного відхилення (RMSE) розроблених SVR моделей, що дозволило отримати точності прогнозування мінімальної та максимальної добових температур вищі, ніж наведені у літературі для окремих типів моделей (LDAPS, RF, SVR, MLP) без використання ансамблевого підходу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hayati M. Application of Artificial Neural Networks for Temperature Forecasting / M. Hayati, Z. Mohebi. // World Academy of Science, Engineering and Technology. – 2017. – С. 1.
2. Can deep learning beat numerical weather prediction? [Електронний ресурс] / M. G. Schultz, C. Betancourt, B. Gong та ін.]. – 2021. – Режим доступу до ресурсу: https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0097#RSTA20200097F1 (дата звернення: 20.10.2021).
3. Comparative Assessment of Various MachineLearning- Based Bias Correction Methodsfor Numerical Weather PredictionModel Forecasts of Extreme AirTemperatures in Urban Areas / C.Dongjin, Y. Cheolhee, I. Jungho, C. Dong-Hyun. // Earth and Space Science. – 2020. – С. 1–16. – Режим доступу до ресурсу: https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2019EA000740
4. A novel ensemble learning for post-processing of NWP Model's next-day maximum air temperature forecast in summer using deep learning and statistical approaches / C.Dongjin, Y. Cheolhee, S. Bokyung // ScienceDirect. – 2022. – С. 1–14. – Режим доступу до ресурсу: https://doi.org/10.1016/j.wace.2022.100410
5. A Hybrid Multi-Objective Optimizer-Based SVM Model for Enhancing Numerical Weather Prediction: A Study for the Seoul Metropolitan Area/ A. D. Mohanad, A. A. S Ahmed, H. A. Mohammed // Multidisciplinary Digital Publishing Institute. – 2021. – С. 1–17. – Режим доступу до ресурсу: https://doi.org/10.3390/su14010296
6. Андреас М. Введение в машинное обучение с помощью Python / М. Андреас, Г. Сара., 2017. 393с.
7. Vaishali A. What is Machine Learning? How Machine Learning Works and future of it? [Електронний ресурс] / Advani Vaishali . – 2021 – Режим доступу до ресурсу: https://www.mygreatlearning.com/blog/what-is-machine-learning/ (дата звернення: 10.11.2021).

8. Jason B. Difference Between Classification and Regression in Machine Learning [Електронний ресурс] / Brownlee Jason. – 2017. – Режим доступу до ресурсу: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/ (дата звернення: 17.11.2021).
9. Nearest Neighbors [Електронний ресурс] . – 2021. – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-nearest-neighbors (дата звернення: 20.11.2021).
10. Ashwin R. Unlocking the True Power of Support Vector Regression [Електронний ресурс] / Raj Ashwin // Towards Data Science. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0 (дата звернення: 08.02.2022).
11. A Feature-Weighted SVR Method Based on Kernel Space Feature / X. Minghua, W. Decheng, X. Lili // Multidisciplinary Digital Publishing Institute. – 2018. – С. 1–12. – Режим доступу до ресурсу: https://doi.org/10.3390/a11050062
12. Support Vector Machines [Електронний ресурс] . – 2022. – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/svm.html#svm-regression (дата звернення: 08.02.2022).
13. Support Vector Regression (SVR) using linear and non-linear kernels [Електронний ресурс] . – 2018. – Режим доступу до ресурсу: http://man.hubwiz.com/docset/Scikit.docset/Contents/Resources/Documents/auto_examples/svm/plot_svm_regression.html (дата звернення: 08.02.2022).
14. Support-vector machine [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Support-vector_machine (дата звернення: 15.04.2022).
15. Pros and Cons of Python in Machine Learning [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.zarantech.com/blog/pros-and-cons-of-python-in-machine-learning/ (дата звернення: 13.11.2021).

16. What is Numpy? [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://numpy.org/doc/stable/user/whatisnumpy.html#what-is-numpy (дата звернення: 20.11.2021).
17. Pandas package overview [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://pandas.pydata.org/docs/getting_started/overview.html#package-overview (дата звернення: 20.11.2021).
18. Scikit-learn [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://scikit-learn.org/stable/ (дата звернення: 20.11.2021).
19. SciPy [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://docs.scipy.org/doc/scipy/tutorial/general.html (дата звернення: 27.04.2022).

ДОДАТОК А

Код програми

```
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from IPython import display
from scipy.optimize import minimize
import math
import numpy as np
import pandas as pd
df =
pd.read_csv('https://drive.google.com/uc?id=19j5vrdPGLSD7TGRp8y6
czVsiQXTPEV3p&authuser=0&export=download')
df.dropna(inplace=True)
df = df[[d[5:7] != '06' for d in df.Date]]
df_X = df.loc[:, 'Present_Tmax':'Solar radiation']
df_Y = df.iloc[:, -2:]
# df
#@title Визначення функцій
do_scale_y = False

def create_XY(T, date_test_start='2015', date_test_end='2018',
station='all'):
    global X_train, X_test, y_train, y_test
    train_samples = df.Date < date_test_start
    test_samples = (date_test_start <= df.Date) & (df.Date <=
date_test_end)
    featurres = df_X.columns
    if station != 'all':
        train_samples = train_samples & (df.station == station)
        test_samples = test_samples & (df.station == station)
        featurres = [s for s in featurres if s not in ['lat',
'lon', 'DEM', 'Slope']]
    X_train = df_X[featurres][train_samples].values
    X_test = df_X[featurres][test_samples].values
    T_index = 0 if T == 'max' else 1
    y_train = df_Y[train_samples].values[:, T_index]
    y_test = df_Y[test_samples].values[:, T_index]

def train_and_predict(T, date_test_start='2015',
date_test_end='2018', station='all', return_metrics=True):
    global clf_pipeline

    create_XY(T, date_test_start, date_test_end, station)
    if len(X_test) == 0:
        return None
```

```
    clf_pipeline = make_pipeline(StandardScaler(), clf)

    if do_scale_y:
        scaler_y = StandardScaler()
        y_train_trans = scaler_y.fit_transform(y_train.reshape(-
1, 1)).flatten()
        clf_pipeline.fit(X_train, y_train_trans)
    else:
        clf_pipeline.fit(X_train, y_train)

    y_train_pred = clf_pipeline.predict(X_train)
    y_test_pred = clf_pipeline.predict(X_test)

    if do_scale_y:
        y_train_pred =
scaler_y.inverse_transform(y_train_pred.reshape(-1,
1)).flatten()
        y_test_pred =
scaler_y.inverse_transform(y_test_pred.reshape(-1, 1)).flatten()

    if return_metrics:
        metrics = pd.DataFrame([], columns=['R2_train',
'R2_test', 'RMSE_train', 'RMSE_test'])
        metrics.loc['all'] = r2_score(y_train, y_train_pred),
r2_score(y_test, y_test_pred), \
            mean_squared_error(y_train, y_train_pred,
squared=False), mean_squared_error(y_test, y_test_pred,
squared=False)
        return y_train, y_train_pred, y_test, y_test_pred,
metrics

    return y_train, y_train_pred, y_test, y_test_pred

def train_and_predict_by_stations(T, date_test_start='2015',
date_test_end='2018'):
    y_train_full = []
    y_train_pred_full = []
    y_test_full = []
    y_test_pred_full = []
    metrics = []
    for i in range(1, 26):
        y_train, y_train_pred, y_test, y_test_pred, m =
train_and_predict(T, date_test_start, date_test_end, i)

        y_train_full.append(y_train)
        y_train_pred_full.append(y_train_pred)
        y_test_full.append(y_test)
        y_test_pred_full.append(y_test_pred)

        metrics.append(m)
```

```
    y_train_full = np.concatenate(y_train_full)
    y_train_pred_full = np.concatenate(y_train_pred_full)
    y_test_full = np.concatenate(y_test_full)
    y_test_pred_full = np.concatenate(y_test_pred_full)

    metrics = pd.concat(metrics)
    metrics.index = list(range(1, 26))
    metrics.loc['avg'] = metrics.mean()
    metrics.loc['all'] = r2_score(y_train_full,
y_train_pred_full), \
                         r2_score(y_test_full,
y_test_pred_full), \
                         mean_squared_error(y_train_full,
y_train_pred_full, squared=False), \
                         mean_squared_error(y_test_full,
y_test_pred_full, squared=False)
    return metrics

def hindcast_validation(T, date_start='2015', date_end='2018'):
    dates = df.Date.unique()
    dates = dates[(date_start <= dates) & (dates <= date_end)]
    y_train_full = []
    y_train_pred_full = []
    y_test_full = []
    y_test_pred_full = []
    for date_test in dates:
        y_train, y_train_pred, y_test, y_test_pred =
train_and_predict(T, date_test, date_test, 'all', False)
        y_train_full.append(y_train)
        y_train_pred_full.append(y_train_pred)
        y_test_full.append(y_test)
        y_test_pred_full.append(y_test_pred)
        # print(date_test)
        # display.clear_output(wait=True)
    y_train_full = np.concatenate(y_train_full)
    y_train_pred_full = np.concatenate(y_train_pred_full)
    y_test_full = np.concatenate(y_test_full)
    y_test_pred_full = np.concatenate(y_test_pred_full)

    metrics = pd.DataFrame([], columns=['R2_train', 'R2_test',
'RMSE_train', 'RMSE_test'])
    metrics.loc['all'] = r2_score(y_train_full,
y_train_pred_full), r2_score(y_test_full, y_test_pred_full), \
        mean_squared_error(y_train_full, y_train_pred_full,
squared=False), mean_squared_error(y_test_full,
y_test_pred_full, squared=False)

    return metrics
```

```
def hindcast_validation_by_stations(T, date_start='2015',
date_end='2018'):
    dates = df.Date.unique()
    dates = dates[(date_start <= dates) & (dates <= date_end)]
    y_train_full = []
    y_train_pred_full = []
    y_test_full = []
    y_test_pred_full = []
    for i in range(1, 26):
        for date_test in dates:
            y = train_and_predict(T, date_start, date_end, i,
False)
            if y is None:
                continue
            y_train, y_train_pred, y_test, y_test_pred = y

            y_train_full.append(y_train)
            y_train_pred_full.append(y_train_pred)
            y_test_full.append(y_test)
            y_test_pred_full.append(y_test_pred)

            # print(i, date_test)
            # display.clear_output(wait=True)

    y_train_full = np.concatenate(y_train_full)
    y_train_pred_full = np.concatenate(y_train_pred_full)
    y_test_full = np.concatenate(y_test_full)
    y_test_pred_full = np.concatenate(y_test_pred_full)

    metrics = pd.DataFrame([], columns=['R2_train', 'R2_test',
'RMSE_train', 'RMSE_test'])
    metrics.loc['all'] = r2_score(y_train_full,
y_train_pred_full), \
                         r2_score(y_test_full,
y_test_pred_full), \
                         mean_squared_error(y_train_full,
y_train_pred_full, squared=False), \
                         mean_squared_error(y_test_full,
y_test_pred_full, squared=False)

    return metrics
do_scale_y = True
clf = SVR(C=100, gamma=0.0005, epsilon=0.2)
train_and_predict('min', '2015', '2018')[-1]
#Svr hindcast
##All year
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.4)
hindcast_validation('max', '2015', '2018')
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.4)
```

```
hindcast_validation('min', '2015', '2018')
do_scale_y = False
clf = SVR(C=80, gamma=0.0005, epsilon=0.4)
hindcast_validation_by_stations('max', '2015', '2018')
do_scale_y = True
clf = SVR(C=80, gamma=0.0005, epsilon=0.4)
hindcast_validation_by_stations('max', '2015', '2018')
do_scale_y = False
clf = SVR(C=170, gamma=0.0005, epsilon=0.3)
hindcast_validation_by_stations('min', '2015', '2018')
do_scale_y = True
clf = SVR(C=170, gamma=0.0005, epsilon=0.3)
hindcast_validation_by_stations('min', '2015', '2018')
##SVR Minimization
def minim_rmse(x):
  global iter, clf
  clf = SVR(C=x[0], gamma=x[1], epsilon=x[2])
  RMSE_test = hindcast_validation('min', '2015', '2018').iloc[0,
-1]
  iter += 1
  print(iter, x, RMSE_test)
  return RMSE_test
do_scale_y = True
iter = 0
#res = minimize(minim_rmse, [100, 0.0001, 0.38], method='Nelder-
Mead',
               #options={'xatol': 1e2,'fatol': 0.0001, 'disp':
True})
res = minimize(minim_rmse, [2.54313202e+02, 1.72914507e-04,
8.26307537e-02], method='Nelder-Mead',
               options={'xatol': 1e4,'fatol': 0.0001, 'disp':
True})
res
def minim_rmse(x):
  global iter, clf
  clf = SVR(C=x[0], gamma=x[1], epsilon=x[2])
  RMSE_test = hindcast_validation('max', '2015', '2018').iloc[0,
-1]
  iter += 1
  print(iter, x, RMSE_test)
  return RMSE_test
do_scale_y = True
iter = 0
res = minimize(minim_rmse, [100, 2.12282147e-04, 2.93556625e-
01], method='Nelder-Mead',
               options={'xatol': 1e4,'fatol': 0.0001, 'disp':
True})
res
do_scale_y = True
iter = 0
```

```
res = minimize(minim_rmse, [1.01322434e+02, 1.71521287e-03,
3.44810020e-01], method='Nelder-Mead',
               options={'xatol': 1e4,'fatol': 0.0001, 'disp':
True})
res
def f(x):
  global iter, clf
  clf = SVR(C=x[0], gamma=x[1], epsilon=x[2])
  R2_test = hindcast_validation('max', '2015', '2016').iloc[0,
1]
  iter += 1
  print(iter, x, R2_test)
  return - R2_test
do_scale_y = True
iter = 0
res = minimize(f, [1.01322434e+02, 1.71521287e-03, 3.44810020e-
01], method='Nelder-Mead',
               options={'fatol': 0.1, 'disp': True})
res
x_opt = [1.09765970e+02, 1.14347525e-03, 3.73544188e-01]
clf = SVR(C=x_opt[0], gamma=x_opt[1], epsilon=x_opt[2])
hindcast_validation('max', '2015', '2016')
def f(x):
  global iter, clf
  clf = SVR(C=x[0], gamma=x[1], epsilon=x[2])
  R2_test = hindcast_validation_by_stations('max', '2015',
'2016').iloc[0, 1]
  iter += 1
  print(iter, x, R2_test)
  return - R2_test
do_scale_y = True
iter = 0
res = minimize(f, [1.01322434e+02, 1.71521287e-03, 3.44810020e-
01], method='Nelder-Mead',
               options={'fatol': 0.1, 'disp': True})
res
##Each year
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.3)
hindcast_validation_by_stations('max', '2015', '2016')
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.4)
hindcast_validation_by_stations('min', '2015', '2016')
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.3)
hindcast_validation_by_stations('min', '2017', '2018')
do_scale_y = False
clf = SVR(C=120, gamma=0.0005, epsilon=0.4)
hindcast_validation_by_stations('max', '2017', '2018')
do_scale_y = False
clf = SVR(C=150, gamma=0.0005, epsilon=0.3)
```

```
hindcast_validation('max', '2015', '2016')
do_scale_y = False
clf = SVR(C=150, gamma=0.0005, epsilon=0.2)
hindcast_validation('min', '2015', '2016')
do_scale_y = False
clf = SVR(C=100, gamma=0.0005, epsilon=0.4)
hindcast_validation('min', '2017', '2018')
do_scale_y = False
clf = SVR(C=80, gamma=0.0005, epsilon=0.4)
hindcast_validation('max', '2017', '2018')
```