Шифр: **Прогнозування врожайності**

Створення моделей нейронних мереж для прогнозування врожайності

ЗМІСТ

ВСТУП 3
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 4
2 РОБОТА З ДАНИМИ 5
2.1 Вибір мови програмування 5
2.2 Огляд вхідних даних 6
2.3 Обробка даних 9
3 СТВОРЕННЯ ТА АНАЛІЗ МОДЕЛЕЙ 11
3.1 Вибір моделей машинного навчання 11
3.2 Створення та навчання моделей за допомогою Python та бібліотек 17
ВИСНОВКИ 21
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 22

ВСТУП

Аграрний сектор є однією з найважливіших галузей економіки, яка забезпечує не тільки продовольчу безпеку, але і відіграє ключову роль у забезпеченні економічного зростання та стійкого розвитку України. Прогнозування урожайності у цій галузі виступає вирішальним завданням, яке дозволяє ефективно управляти виробництвом, оптимізувати використання ресурсів та забезпечувати стійкий розвиток аграрного сектору.

У контексті сучасних викликів, зокрема зміни клімату, нестабільності урожайності та ризику втрат, важливо розробляти та використовувати передові технології для прогнозування урожайності. Штучний інтелект (AI) стає важливим інструментом у цьому контексті, забезпечуючи можливість аналізувати великі обсяги даних, розпізнавати складні залежності та робити точні прогнози з високою ступенем достовірності.

У даній науковій роботі досліджується можливість застосування штучного інтелекту для прогнозування урожайності та аналізується вплив цих технологій на ефективність аграрного сектору. Для досягнення цієї мети використовуються як існуючі моделі машинного навчання, такі як: Decision Tree, Random Forest та Gradient Boosting, так і розроблена власна нейронна мережа на основі архітектури глибокого навчання.

Загальна мета цієї наукової роботи полягає у розробці та впровадженні передових технологій для прогнозування урожайності, що дозволить підвищити продуктивність та ефективність аграрного сектору, зменшити витрати та ризики виробництва, а також забезпечити стійкий розвиток сільського господарства в умовах сучасного глобального ринку.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Прогнозування врожайності з використанням штучного інтелекту та машинного навчання застосовується в різних країнах світу. Зокрема, ці технології використовуються в Україні, США, Китаї, Індії, Європейському Союзі та інших країнах з великим сільськогосподарським сектором. Вони знайшли широке застосування в різних культурах, від зернових до фруктових та овочевих культур. Прогнозування врожайності дозволяє господарствам ефективно планувати виробництво та розподіляти ресурси. Це допомагає знижувати витрати та оптимізувати використання добрив, поливу та інших ресурсів. Крім того, точні прогнози дозволяють уникнути можливих ризиків, таких як втрати врожаю через погіршення погодних умов або захворювання рослин.

У сучасному сільському господарстві використовуються різноманітні методи та моделі для прогнозування врожайності. До найпоширеніших методів відносяться статистичні моделі, які базуються на аналізі історичних даних, та моделі машинного навчання, які здатні працювати з великим обсягом даних та розпізнавати складні залежності між різними факторами, що впливають на врожайність. Також застосовуються супутникові знімки, вони дозволяють моніторити ріст та стан рослин на полі, виявляти проблемні зони, такі як стресові урочища або зараження хворобами, та аналізувати різноманітні фактори, які впливають на урожайність, такі як стан грунту, вологість, погодні умови тощо.

Для успішного прогнозування врожайності необхідно мати доступ до різноманітних видів даних, таких як історичні дані про врожаї, метеорологічні дані, дані про використання ресурсів та інші параметри. Ці дані дозволяють побудувати точні та надійні моделі прогнозування, які можуть бути використані для прийняття рішень у галузі сільського господарства.

2 РОБОТА З ДАНИМИ

2.3 Вибір мови програмування

При розробці програмного інструментарію для прогнозування врожайності було обрано мову програмування Python та деякі ключові бібліотеки, описані нижче.

Вибір Python обумовлений його широкою популярністю, простотою вивчення та використання, а також багатофункціональністю. Python має велику кількість бібліотек для обробки даних та машинного навчання, що робить його ідеальним вибором для розробки програмного інструментарію для прогнозування врожайності.

Бібліотеки:

Pandas: Використовується для роботи з даними, включаючи читання, фільтрацію, обробку та аналіз. За допомогою Pandas можна легко обробляти великі обсяги даних, виконувати агрегацію та групування даних для подальшого аналізу.

Matplotlib.pyplot: Використовується для візуалізації даних, створення графіків та діаграм. За допомогою Matplotlib можна побудувати різноманітні графіки, що допомагають зрозуміти структуру та закономірності в даних.

NumPy: Використовується для математичних обчислень та роботи з масивами даних. NumPy дозволяє виконувати швидкі та ефективні операції з числовими даними, що особливо важливо при роботі з великими обсягами даних.

TensorFlow: Використовується для реалізації нейронних мереж та моделей машинного навчання для прогнозування врожайності. TensorFlow надає потужні інструменти для створення та тренування моделей машинного навчання, що дозволяє побудувати складні прогностичні моделі.

Scikit-learn (sklearn): Забезпечує прості та ефективні інструменти для аналізу даних та моделювання, включаючи класифікацію, регресію, кластеризацію та зниження розмірності. Ця бібліотека заснована на NumPy, SciPy та matplotlib, що робить її високо інтегрованою в екосистему наукового обчислення в Python.

Ці бібліотеки допомагають забезпечити ефективну та точну роботу з даними, візуалізацію результатів та розробку моделей для прогнозування урожайності. Використання Python разом із зазначеними бібліотеками дозволяє створити потужний інструмент для аналізу та прогнозування врожайності культур.

2.2 Огляд вхідних даних

	Area	Item	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	hg/ha_yield
0	Albania	Maize	1990	1485	121.0	16.37	36613
1	Albania	Potatoes	1990	1485	121.0	16.37	66667
2	Albania	Rice, paddy	1990	1485	121.0	16.37	23333
3	Albania	Sorghum	1990	1485	121.0	16.37	12500
4	Albania	Soybeans	1990	1485	121.0	16.37	7000
5	Albania	Wheat	1990	1485	121.0	16.37	30197
6	Albania	Maize	1991	1485	121.0	15.36	29068
7	Albania	Potatoes	1991	1485	121.0	15.36	77818
8	Albania	Rice, paddy	1991	1485	121.0	15.36	28538
9	Albania	Sorghum	1991	1485	121.0	15.36	6667

Рисунок 2.1 – Отримані дані

Вхідними даними для дослідження є :

- Area – Країни
- Item – Види висаджуваних культур
- Year – Роки 1990-2013
- Average rain fall mm per year – Середня кількість опадів
- Pesticides tonnes – Кількість пестицидів
- Average temperature – Середня температура

- hg/ha_yield (Output) – Значення врожайності сільськогосподарських культур

Розглянемо кореляційну матрицю між даними для більш точного розуміння залежності даних.

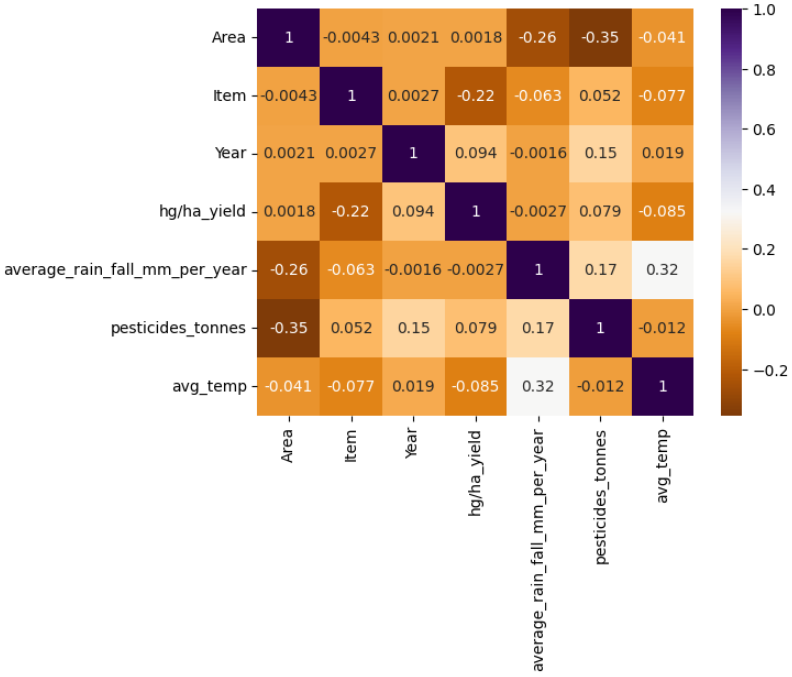

Рисунок 2.2 – Кореляційна матриця даних

Існує сильна кореляція між країною та кількістю пестицидів, країною та кількістю опадів, культурою та врожайністю.

Розглянемо статистику кількості основних елементів датасету.

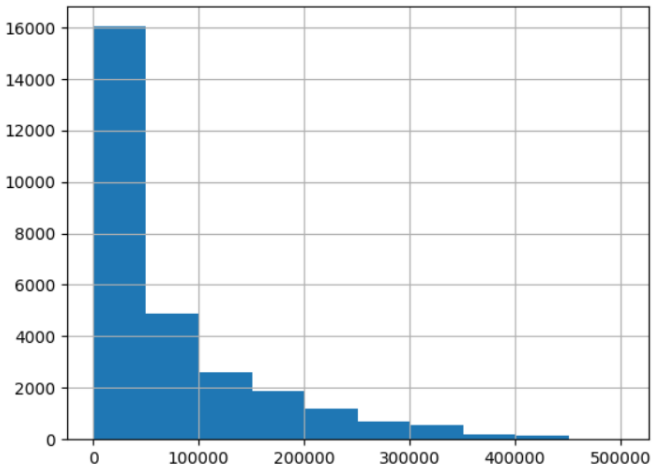

Рисунок 2.3 – Статистика врожаю

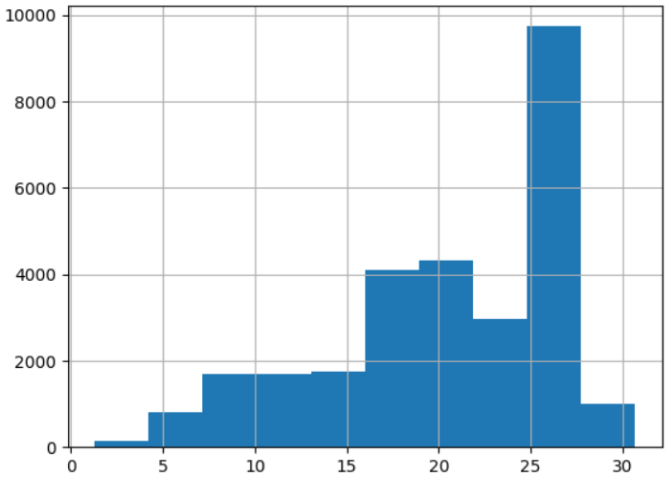

Рисунок 2.4 – Статистика температури

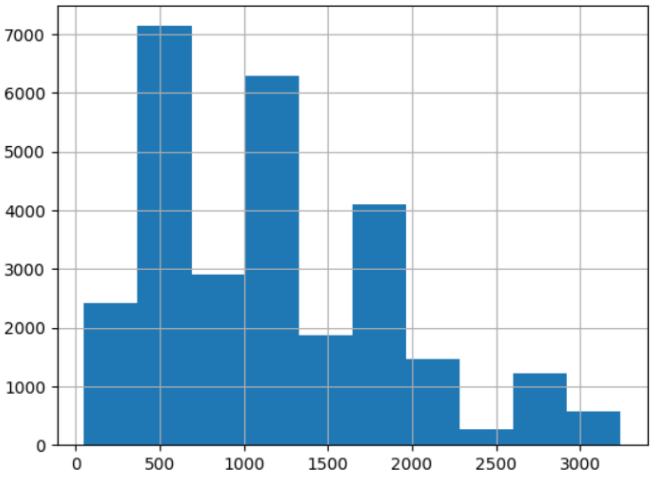

Рисунок 2.5 – Статистика опадів

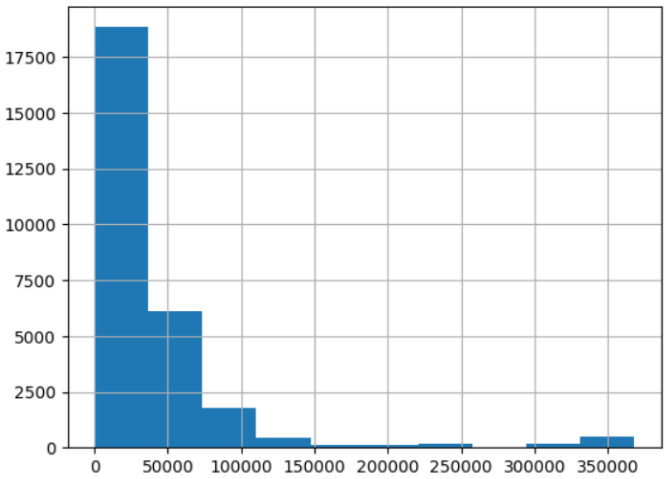

Рисунок 2.6 – Статистика пестицидів

2.3 Обробка даних

Обробка даних перед використанням для машинного навчання є критично важливим етапом, що включає в себе кілька ключових кроків:

1. **Очищення даних:** Першим кроком є виявлення та виправлення помилок в даних, таких як відсутні значення (NA або null), дублікати та помилкові значення. Це може включати заповнення відсутніх значень, видалення дублікатів та виправлення помилкових даних.
2. **Нормалізація та стандартизація:** Після очищення даних зазвичай застосовують нормалізацію (приведення даних до одного масштабу) та стандартизацію (центрування даних навколо середнього значення та шкальування їх за стандартним відхиленням). Це полегшує роботу моделей машинного навчання, які можуть бути чутливі до різниці в масштабах ознак.
3. **Інженерія ознак:** Додавання нових ознак на основі наявних даних може покращити прогностичні можливості моделей. Це може включати створення поліноміальних ознак, додавання інтеракційних та дискретизацію ознак.
4. **Обробка категоріальних ознак:** Якщо в даних присутні категоріальні ознаки (наприклад, типи культур), їх необхідно перетворити в числові формати, так що моделі машинного навчання можуть їх коректно обробляти.
5. **Видалення непотрібних ознак:** Іноді деякі ознаки можуть бути неінформативними або непотрібними для прогнозування, і вони можуть бути видалені перед використанням даних для навчання моделей.
6. **Розділення набору даних:** Набір даних зазвичай розділяють на навчальний та тестовий для оцінки продуктивності моделі на нових даних.

Ці кроки дозволяють підготувати дані для машинного навчання, забезпечуючи якість та ефективність моделей.

Результат основних кроків перед використання даних в моделях машинного навчання:

```
df_one = pd.get_dummies(df, columns=['Area',"Item"], dtype='int')
df_one.head()
```

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	hg/ha_yield	Area_Albania	Area_Algeria	Area_Angola
0	1990	1485	121.0	16.37	36613	1	0	0
1	1990	1485	121.0	16.37	66667	1	0	0
2	1990	1485	121.0	16.37	23333	1	0	0
3	1990	1485	121.0	16.37	12500	1	0	0
4	1990	1485	121.0	16.37	7000	1	0	0

5 rows × 116 columns

Рисунок 2.7 – Результат гарячого кодування

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
cols = ['Year','average_rain_fall_mm_per_year','pesticides_tonnes','avg_temp']
df_one[cols]= sc.fit_transform(df_one[cols])
df_one.head()
```

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	hg/ha_yield	Area_Albania	Area_Algeria	Area_Angola
0	-1.637075	0.473294	-0.616366	-0.661069	36613	1	0	0
1	-1.637075	0.473294	-0.616366	-0.661069	66667	1	0	0
2	-1.637075	0.473294	-0.616366	-0.661069	23333	1	0	0
3	-1.637075	0.473294	-0.616366	-0.661069	12500	1	0	0
4	-1.637075	0.473294	-0.616366	-0.661069	7000	1	0	0

5 rows × 116 columns

Рисунок 2.8 – Результат стандартизації

3 СТВОРЕННЯ ТА АНАЛІЗ МОДЕЛЕЙ

3.1 Вибір моделей машинного навчання

Прогнозування врожайності є завданням, яке можна розв'язати за допомогою методів регресії. Для роботи обрано три моделі машинного навчання: Decision Tree, Random Forest та Gradient Boosting. Нижче подано обґрунтування вибору кожної моделі та їх специфіку.

Decision Tree (Дерево рішень)— це модель прогнозування у формі дерева, яка використовується в статистиці, аналізі даних та машинному навчанні для рішення задач як класифікації, так і регресії. Модель дерева рішень ділить весь набір даних на менші підмножини, в той час як одночасно розвивається асоційоване дерево рішень. Кінцеве дерево включає кореневий вузол, внутрішні вузли та листові вузли.

Ключові компоненти:

- Кореневий вузол: Вузол, з якого починається дерево. Він містить увесь набір даних для аналізу.
- Розгалуження або внутрішні вузли: Точки рішення, де відбувається поділ на основі певної характеристики.
- Листові вузли: Кінцеві елементи дерева, які представляють результат або рішення.

Процес створення дерева рішень включає в себе поділ даних на підмножини на основі атрибутів. Вибір атрибуту для поділу в кожному вузлі заснований на статистичних мірках, таких як ентропія, індекс Джині або зниження помилок/варіативності. Процес продовжується до того моменту, поки не будуть виконані певні критерії зупинки, наприклад, коли всі елементи в листовому вузлі належать до одного класу або досягнуто максимальної глибини дерева.

Переваги:

- Інтуїтивність: Дерева рішень легко інтерпретувати та візуалізувати, що робить їх зрозумілими навіть для людей без технічного тла.
- Гнучкість: Підходять для рішення як задач класифікації, так і регресії.
- Впорання з нелінійністю: Можуть ефективно працювати з нелінійними відносинами між функціями та мітками.

Недоліки:

- Схильність до перенавчання: Дерева рішень можуть створювати надто складні структури, які не генералізують добре дані, особливо при недостатньому обмеженні.
- Нестійкість: Малі зміни в даних можуть призвести до значних змін у структурі дерева.

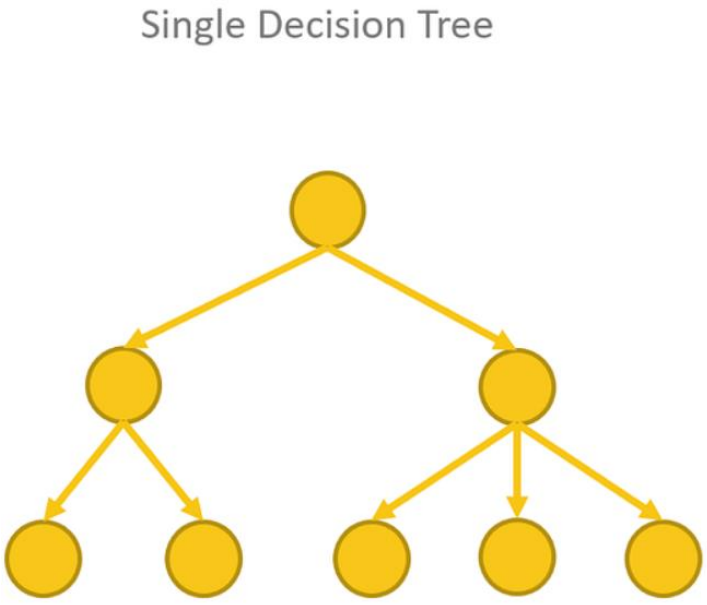

Рисунок 3.1 – Архітектура Decision Tree

Random Forest (Випадковий ліс) — це ансамбльний метод машинного навчання, що базується на ідеї об'єднання багатьох рішень дерев рішень для покращення точності прогнозування та зниження ризику перенавчання. Метод випадкового лісу використовує велику кількість дерев рішень, кожне з яких

навчається на випадковій підмножині набору даних та випадковій підмножині ознак. Результати цих дерев об'єднуються за допомогою голосування (у випадку класифікації) або усереднення (у випадку регресії), щоб отримати кінцевий прогноз.

Особливості випадкового лісу:

1. Випадковий вибір ознак: Кожен дерево навчається на випадковому підмножині ознак, що дозволяє зменшити кореляцію між деревами та зробити модель більш робастною до шуму та перенавчання.
2. Випадковий вибір об'єктів: Кожен дерево навчається на випадковій підмножині об'єктів з набору даних, що зменшує дисперсію моделі та поліпшує її узагальнення.
3. Багатодеревний ансамбль: Випадковий ліс використовує багато дерев, кожен з яких може бути слабим класифікатором або регресором, але колективно вони формують потужний ансамбль.
4. Висока точність: Випадковий ліс часто дає високу точність прогнозування на різних типах даних без потреби великої попередньої підготовки даних.
5. Стійкість до перенавчання: Завдяки випадковим підмножинам даних та ознак, випадковий ліс є стійким до перенавчання, що дозволяє ефективно працювати з великими наборами даних.

Переваги випадкового лісу:

- Добра обробка категоріальних даних: Може ефективно працювати з категоріальними та числовими даними без значного передпроцесингу.
- Автоматична обробка пропущених даних: Модель може обробляти дані з пропущеними значеннями без необхідності в додаткових кроків.
- Стійкість до шуму: Здатність моделі усереднювати результати багатьох дерев дозволяє зменшити вплив шуму на прогнози.

Недоліки випадкового лісу:

- Інтерпретовність: У порівнянні з одним деревом рішень, випадковий ліс складніше інтерпретувати через велику кількість дерев та їх взаємодію.
- Час навчання: Тренування великої кількості дерев може вимагати більше часу порівняно з іншими методами.

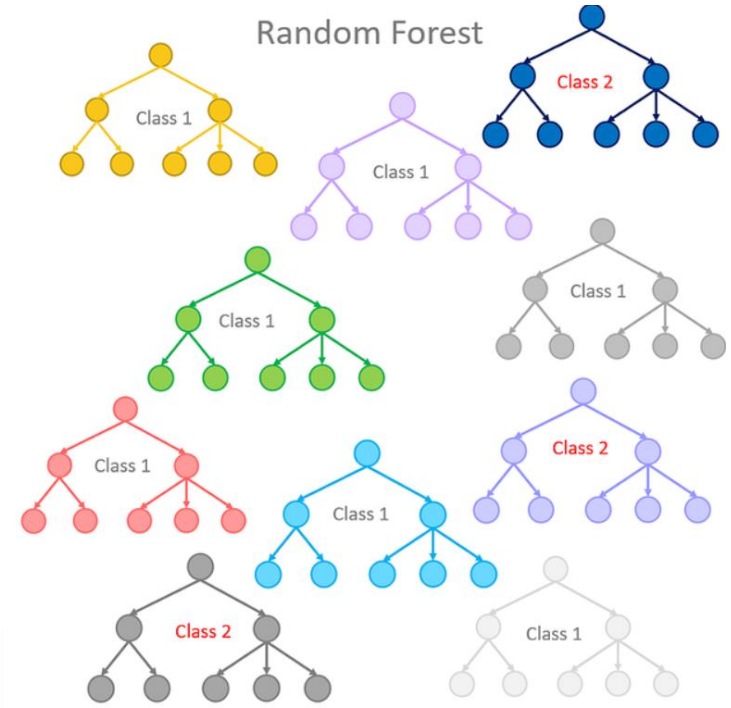

Рисунок 3.2 – Архітектура Random Forest

Gradient Boosting (Градієнтний бустінг) — це потужна техніка машинного навчання для регресійних і класифікаційних задач, яка працює на принципі послідовного покращення прогнозів шляхом врахування помилок попередніх моделей. В основі методу лежить ідея про те, що поєднання декількох "слабких" моделей може утворити одну "сильну" модель. У контексті посилення градієнта, ці "слабкі" моделі зазвичай представлені деревами рішень.

Принцип роботи:

1. Ініціалізація: Початкова модель будується на основі простого прогнозу (наприклад, середнього значення міток учбового набору).

2. Послідовне навчання: На кожному кроці будується нове дерево, яке покликане мінімізувати помилки попередніх моделей. Замість того, щоб просто виправляти помилки, посилення градієнта намагається зробити це найбільш ефективним шляхом, використовуючи градієнт функції втрат.
3. Оновлення моделі: Після кожного кроку модель оновлюється, додаючи до попереднього прогнозу прогноз нового дерева, помножений на коефіцієнт навчання (learning rate), що допомагає контролювати швидкість навчання моделі.
4. Зупинка: Процес продовжується до досягнення заздалегідь визначеної кількості ітерацій або поки помилка не зменшиться до прийнятного рівня.

Переваги:

- Висока точність: Посилення градієнта часто показує одні з найкращих результатів на різноманітних наборах даних.
- Гнучкість: Метод може використовуватись для рішення як задач класифікації, так і регресії.
- Автоматична обробка пропущених даних: Моделі, побудовані за допомогою посилення градієнта, можуть ефективно обробляти випадки з пропущеними даними.

Недоліки:

- Схильність до перенавчання: При недостатньому контролі параметрів модель може перенавчитися, особливо на невеликих наборах даних.
- Часові витрати: Через послідовний характер навчання моделей, посилення градієнта може вимагати більше часу для тренування порівняно з іншими методами.
- Підбір параметрів: Для досягнення найкращих результатів потребує тонкої настройки параметрів.

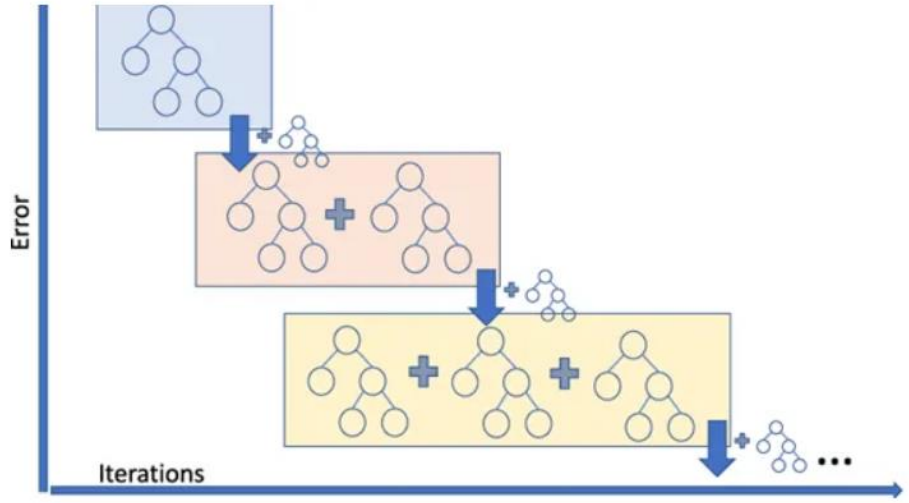

Рисунок 3.3 – Архітектура Gradient Boost

Також скористаємося багатошаровим персептроном (MLP, Multi-Layer Perceptron) для глубокого навчання. Він є розширенням концепції одношарового персептрона.

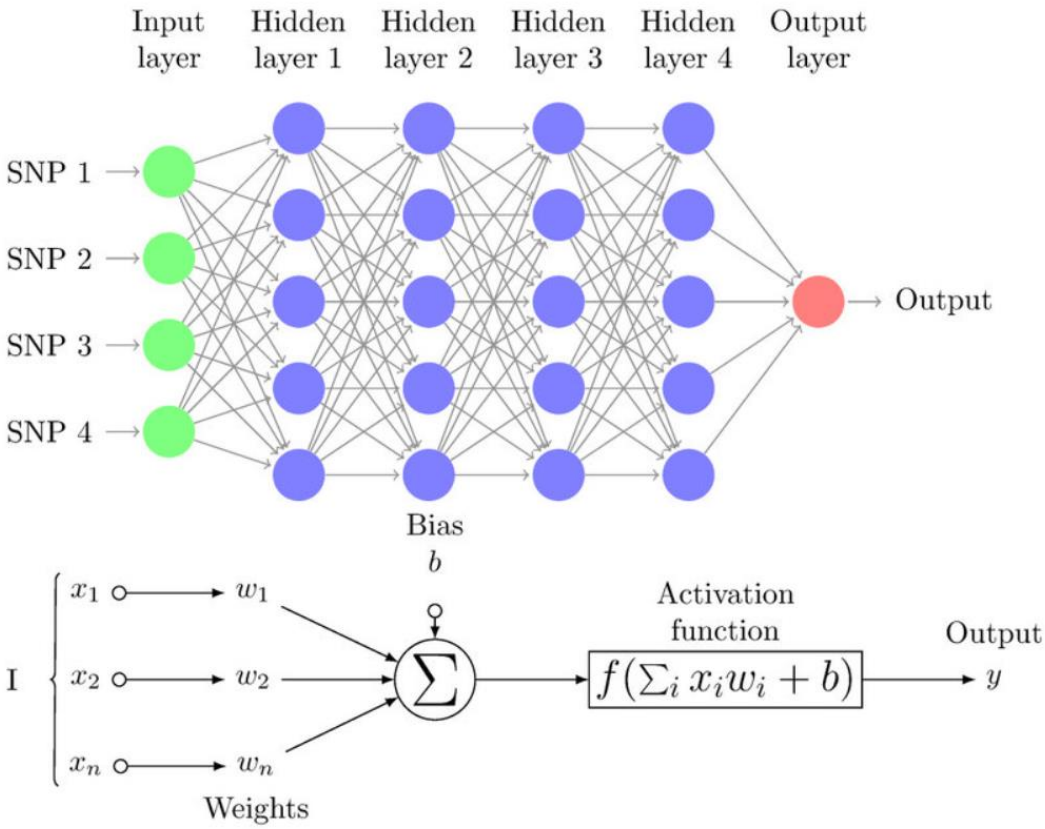

Рисунок 3.4 – Архітектура MLP

Принцип роботи:

1. Вхідний шар приймає вхідні дані і передає їх до наступного шару без зміни.
2. Приховані шари здійснюють основну обробку даних через вагові коефіцієнти і активаційні функції. Вони можуть виявляти складні взаємозв'язки між вхідними даними.
3. Активаційні функції в прихованих шарах дозволяють мережі вчитися і моделювати не лінійні завдання. Популярними виборами є ReLU (Rectified Linear Unit), сигмоїдна функція та гіперболічний тангенс.
4. Вихідний шар забезпечує результат роботи мережі, який може відповідати класифікації або прогнозуванню значення в задачах регресії.

Навчання багатошарового персептрону зазвичай здійснюється за допомогою алгоритму зворотного поширення помилки, де помилка між фактичним виходом мережі та очікуваним результатом розповсюджується назад через мережу, щоб коригувати ваги. Цей процес повторюється багато разів з метою мінімізації помилки.

Багатошарові персептрони застосовуються в широкому спектрі задач, включаючи розпізнавання образів, обробку мови, прогнозування фінансових тенденцій, управління роботами та багато інших. Завдяки здатності моделювати складні взаємозв'язки в даних, вони стали одними з основних інструментів у галузі машинного навчання та штучного інтелекту.

3.2 Створення та навчання моделей за допомогою Python та бібліотек

Розділимо дані для тестів та для навчання, навчимо моделі та виміряємо загальну точність.

```
from sklearn.model_selection import train_test_split
x_train , x_test, y_train , y_test = train_test_split(X, y ,test_size=0.2, random_state=42)
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

Рисунок 3.5 – Код розділу та додавання моделей

```
dt = DecisionTreeRegressor()
dt.fit(x_train , y_train)
y_pred = dt.predict(x_test)
score = r2_score(y_test,y_pred)
score
```

```
0.9667660384529735
```

Рисунок 3.6 – Навчання та результат

```
gbr=GradientBoostingRegressor(n_estimators=300, max_depth=5, random_state=42)
gbr.fit(x_train , y_train)
y_pred = gbr.predict(x_test)
score = r2_score(y_test,y_pred)
score
```

```
0.961781084793516
```

Рисунок 3.7 – Навчання та результат

```
rf =RandomForestRegressor(n_estimators=300, max_depth=5, random_state=42)
rf.fit(x_train , y_train)
y_pred = rf.predict(x_test)
score = r2_score(y_test,y_pred)
score
```

```
0.8117897437053598
```

Рисунок 3.8 – Навчання та результат

Тепер створимо Multi-Layer Perceptron для глибокого навчання.

```
model = Sequential([
    Dense(256, activation='relu', input_shape=(x_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error', metrics=[R2Score()])
history = model.fit(x_train, y_train, epochs=15, batch_size=15, validation_data=(x_test, y_test), verbose=1)
```

Рисунок 3.9 – Код та архітектура створеного Multi-Layer Perceptron

Кількість нейронів, шарів, епох та batch_size вибрано для максимізації ефективності у вигляді кращого часу та кращої точності.

```
Epoch 1/15
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_shape`/`input_dim` argument
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - loss: 5463232512.0000 - r2_score: 0.2442 - val_loss: 863460736.0000 - val_r2_score: 0.8814
Epoch 2/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - loss: 798402368.0000 - r2_score: 0.8893 - val_loss: 508304160.0000 - val_r2_score: 0.9302
Epoch 3/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - loss: 437161888.0000 - r2_score: 0.9400 - val_loss: 317553440.0000 - val_r2_score: 0.9564
Epoch 4/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 291889152.0000 - r2_score: 0.9596 - val_loss: 230110832.0000 - val_r2_score: 0.9684
Epoch 5/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 218351600.0000 - r2_score: 0.9704 - val_loss: 205009968.0000 - val_r2_score: 0.9718
Epoch 6/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 175278672.0000 - r2_score: 0.9753 - val_loss: 181611712.0000 - val_r2_score: 0.9750
Epoch 7/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - loss: 177004304.0000 - r2_score: 0.9757 - val_loss: 184004688.0000 - val_r2_score: 0.9747
Epoch 8/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - loss: 156333360.0000 - r2_score: 0.9783 - val_loss: 167791296.0000 - val_r2_score: 0.9769
Epoch 9/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - loss: 153204400.0000 - r2_score: 0.9784 - val_loss: 152917232.0000 - val_r2_score: 0.9790
Epoch 10/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 132992560.0000 - r2_score: 0.9811 - val_loss: 148584096.0000 - val_r2_score: 0.9796
Epoch 11/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - loss: 138053584.0000 - r2_score: 0.9809 - val_loss: 152814592.0000 - val_r2_score: 0.9790
Epoch 12/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - loss: 132834304.0000 - r2_score: 0.9815 - val_loss: 139730320.0000 - val_r2_score: 0.9808
Epoch 13/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 119086256.0000 - r2_score: 0.9834 - val_loss: 159458944.0000 - val_r2_score: 0.9781
Epoch 14/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 116315872.0000 - r2_score: 0.9839 - val_loss: 139680800.0000 - val_r2_score: 0.9808
Epoch 15/15
1695/1695 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 124157128.0000 - r2_score: 0.9830 - val_loss: 126716712.0000 - val_r2_score: 0.9826
```

✓ 1 мин. 44 сек. выполнено в 01:40

Рисунок 3.10 – Процес навчання

```
predictions = model.predict(x_test)
r2 = r2_score(y_test, predictions)
print(f'R2 score: {r2}')
```

```
89/89 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
R2 score: 0.9825876205059674
```

Рисунок 3.11 –Результат навчання

3.3 Аналіз результатів

Таблиця 3.1

Модель	Точність
Multi-Layer Perceptron	0.98
Decision Tree	0.96
Gradient Boosting	0.96
Random Forest	0.81

Кращою виявилася модель глибокого навчання Multi-Layer Perceptron, що не дивно. Гіршою стала модель Random Forest. Взагалі моделі показали дуже гарні результати.

ВИСНОВКИ

Загальною метою цієї роботи було розроблення та застосування ефективних методів прогнозування врожайності з метою підвищення продуктивності та стабільності сільськогосподарського виробництва.

У роботі було досліджено та реалізовано методи прогнозування врожайності за допомогою штучного інтелекту, зокрема застосовано різні моделі машинного навчання, такі як дерева рішень, градієнтне підсилення, випадковий ліс та модель глибокого навчання – багатошаровий персептрон. Порівняння різних моделей машинного навчання показало, що кожна з них має свої переваги та недоліки, і вибір моделі повинен залежати від конкретних умов та потреб сільськогосподарських підприємств.

Найкращою моделлю для рішення завдання прогнозування врожайності став багатошаровий персептрон.

Результати цієї роботи можуть бути використані для покращення управління сільськогосподарським виробництвом, збільшення врожайності та зниження ризиків фінансових втрат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Francois Chollet: «Deep Learning with Python»
2. [Електронний ресурс] Режим доступу: URL: https://www.python.org/
3. [Електронний ресурс] Режим доступу: URL: https://scikit-learn.org/stable/
4. [Електронний ресурс] Режим доступу: URL: https://www.tensorflow.org/
5. [Електронний ресурс] Режим доступу: URL: https://eos.com/ru/products/crop-monitoring/custom-solutions/yield-prediction/
6. Vidhya, K.; George, S.; Suresh, P.; Brindha, D.; Jebaseeli, T.J. Agricultural Farm Production Model for Smart Crop Yield Recommendations Using Machine Learning Techniques. Eng. Proc. 2023, 59, 20. https://doi.org/10.3390/engproc2023059020
7. Satish Kumar Kalhotra, S. K. S. K. K., Prakash, K. ., Mishra, M. K. ., & Kumar, M. S. A. K. . (2023). A Study of Crop Yield Prediction Using Machine Learning Approaches. Journal of Advanced Zoology, 44(S5), 1351–1354. https://doi.org/10.17762/jaz.v44iS-5.1263
8. Talaat, F.M. Crop yield prediction algorithm (CYPA) in precision agriculture based on IoT techniques and climate changes. Neural Comput & Applic 35, 17281–17292 (2023). https://doi.org/10.1007/s00521-023-08619-5
9. Barriguinha, A.; de Castro Neto, M.; Gil, A. Vineyard Yield Estimation, Prediction, and Forecasting: A Systematic Literature Review. Agronomy 2021, 11, 1789. https://doi.org/10.3390/agronomy11091789
10. Nyéki, A.; Neményi, M. Crop Yield Prediction in Precision Agriculture. Agronomy 2022, 12, 2460. https://doi.org/10.3390/agronomy12102460

11. Halder, M., Datta, A., Siam, M.K.H., Mahmud, S., Sarkar, M.S., Rana, M.M. (2023). A Systematic Review on Crop Yield Prediction Using Machine Learning. In: Nguyen, T.D.L., Verdú, E., Le, A.N., Ganzha, M. (eds) Intelligent Systems and Networks. ICISN 2023. Lecture Notes in Networks and Systems, vol 752. Springer, Singapore. https://doi.org/10.1007/978-981-99-4725-6_77