رواد مصر الرقمية

وزارة الاتصالات
وتكنولوجيا المعلومات

Crop Recommendation System: Data-Driven Agriculture

A solution designed to provide farmers with personalized recommendations for crop selection based on data analysis and machine learning.

Hanin Essam Sayed Mohamed

Date: Nov - 2024

Project Overview

Problem Statement:

- Precision agriculture is gaining importance to help farmers make informed decisions about crop selection. Currently, farmers often rely on traditional methods and general advice, which may not be suitable for their specific farm conditions. The lack of personalized recommendations leads to suboptimal crop yields and resource inefficiencies.

Solution Overview:

Develop a **Personalized Recommendation System** that suggest optimal crops based on multiple features:
- historical data, weather conditions, soil quality, and user interactions to suggest optimal crops.
- Leverages recommendation models to tailor suggestions to individual farmers' needs.

Unique Value Proposition:

- *Data-Driven Insights*
- *Advanced ML Techniques*
- *Scalability for Diverse Conditions*

PROJECT PIPLINE

Project Pipeline:

- **Data Collection** – Retrieve crop dataset from Kaggle and organize it for analysis.
- **Data Preprocessing** – Clean, encode, and scale data to prepare it for model training.
- **Model Development** – Train and evaluate GAN and Random Forest models for recommendations.
- **Experiment Tracking** – Use MLflow to monitor model performance and adjustments.
- **Model Deployment** – Deploy the final model and integrate with the frontend for recommendations.

DATASET

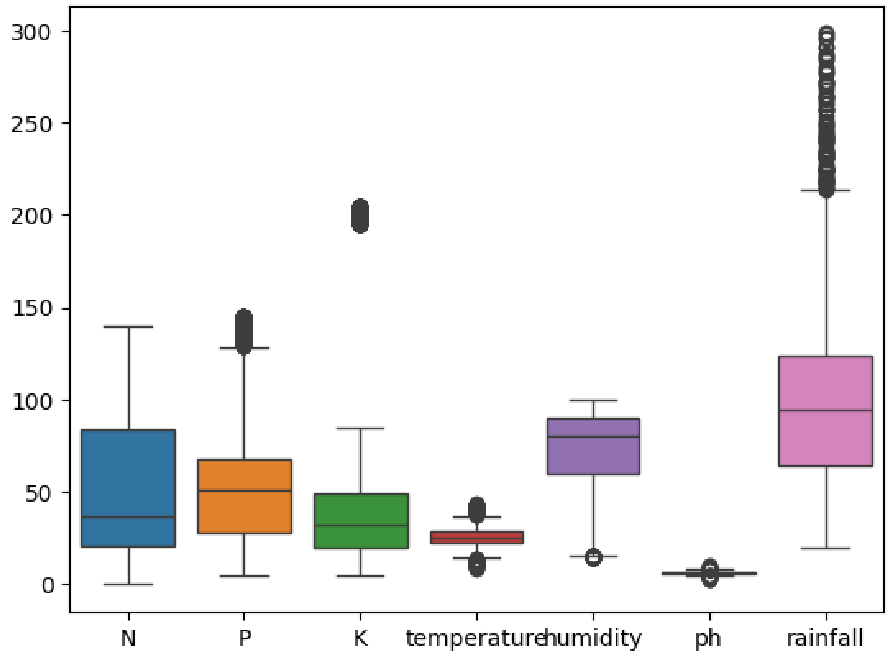
[9]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

N: ratio of Nitrogen content in soil
P (ratio of Phosphorous content in soil) - Important for energy transfer within the plant
K (ratio of Potassium content in soil) - Plays a significant role in water regulation
Temperature (temperature in degree Celsius)- Affects growth rate and metabolic processes. Different crops have optimal temperature ranges for best growth.
Humidity (relative humidity in %) - Influences transpiration and disease prevalence. High humidity can encourage fungal growth, while low humidity increases water stress.
pH (ph value of the soil)- Indicates soil acidity or alkalinity. Different plants thrive at different pH levels, as pH affects nutrient availability in the soil.
Rainfall (in mm) - Critical for soil moisture levels and crop hydration, with each crop type having specific water needs for optimal growth.

EDA

Outliers



No Missing Values

```
[15]: DF.isnull().sum()
```

[15]:

	0
N	0
P	0
K	0
temperature	0
humidity	0
ph	0
rainfall	0
label	0

dtype: int64

Dataset is Balanced

14]:

	count
label	
rice	100
maize	100
jute	100
cotton	100
coconut	100
papaya	100
orange	100
apple	100
muskmelon	100
watermelon	100
grapes	100
mango	100
banana	100
pomegranate	100
lentil	100

SPLITTING DATASET

```
[20]: X = DF[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
      y = DF[['crop']]

[21]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,shuffle=True,random_state=42, stratify=y)

[22]: X_temp = X_test
      y_temp = y_test
      X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, shuffle=True,random_state=42, stratify=y_temp)
```

BINARY ENCODING

[23]:
```
encoder = BinaryEncoder()
encoder.fit(y_train)
transformed_y_train = encoder.transform(y_train)
transformed_y_test = encoder.transform(y_test)
transformed_y_val = encoder.transform(y_val)
```

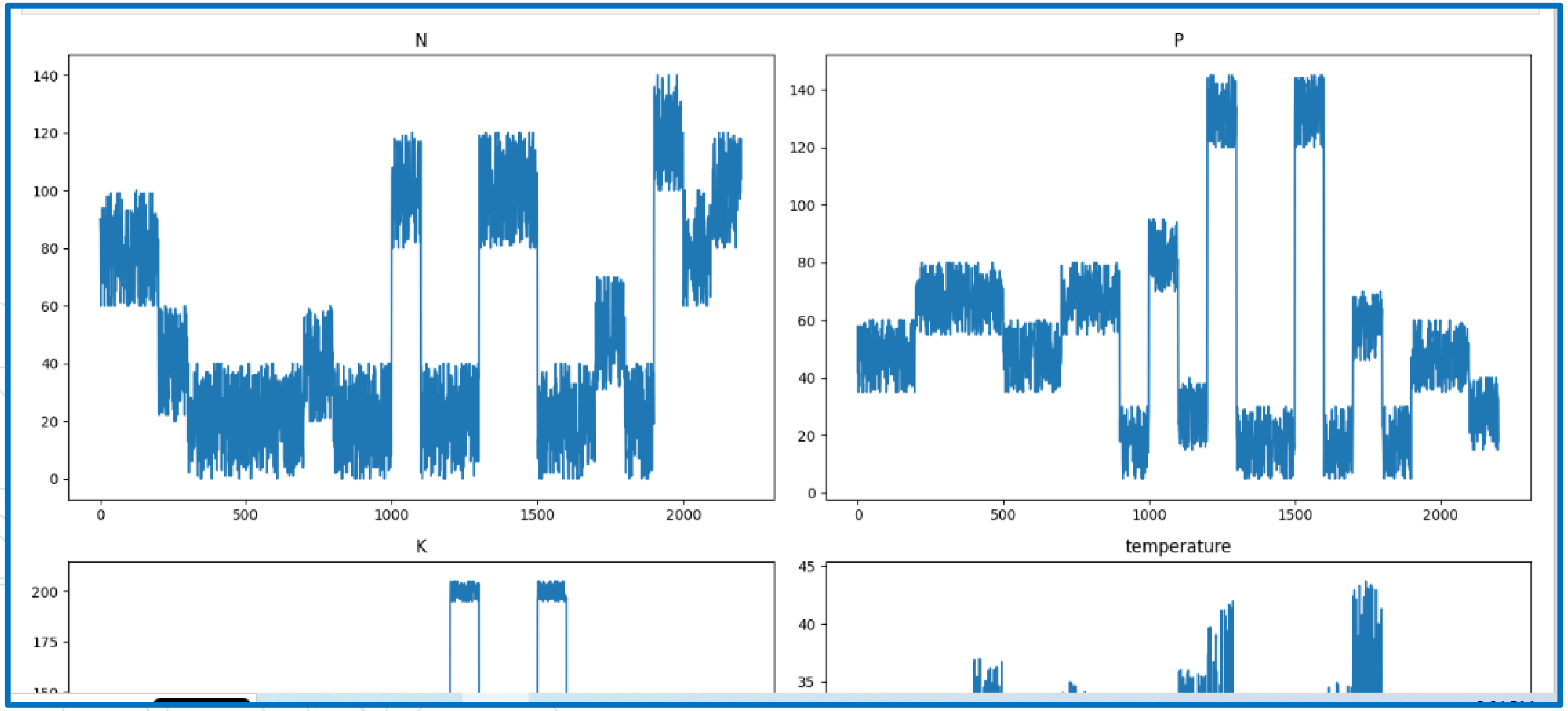
[28]:
```
transformed_y_train
```

[28]:

	crop_0	crop_1	crop_2	crop_3	crop_4
1607	0	0	0	0	1
1212	0	0	0	1	0
362	0	0	0	1	1
566	0	0	1	0	0
1671	0	0	0	0	1
...	...	...	...	...	...
808	0	0	1	1	1
422	0	1	1	0	1
2052	1	0	0	0	1
2083	1	0	0	0	1
1282	0	0	0	1	0

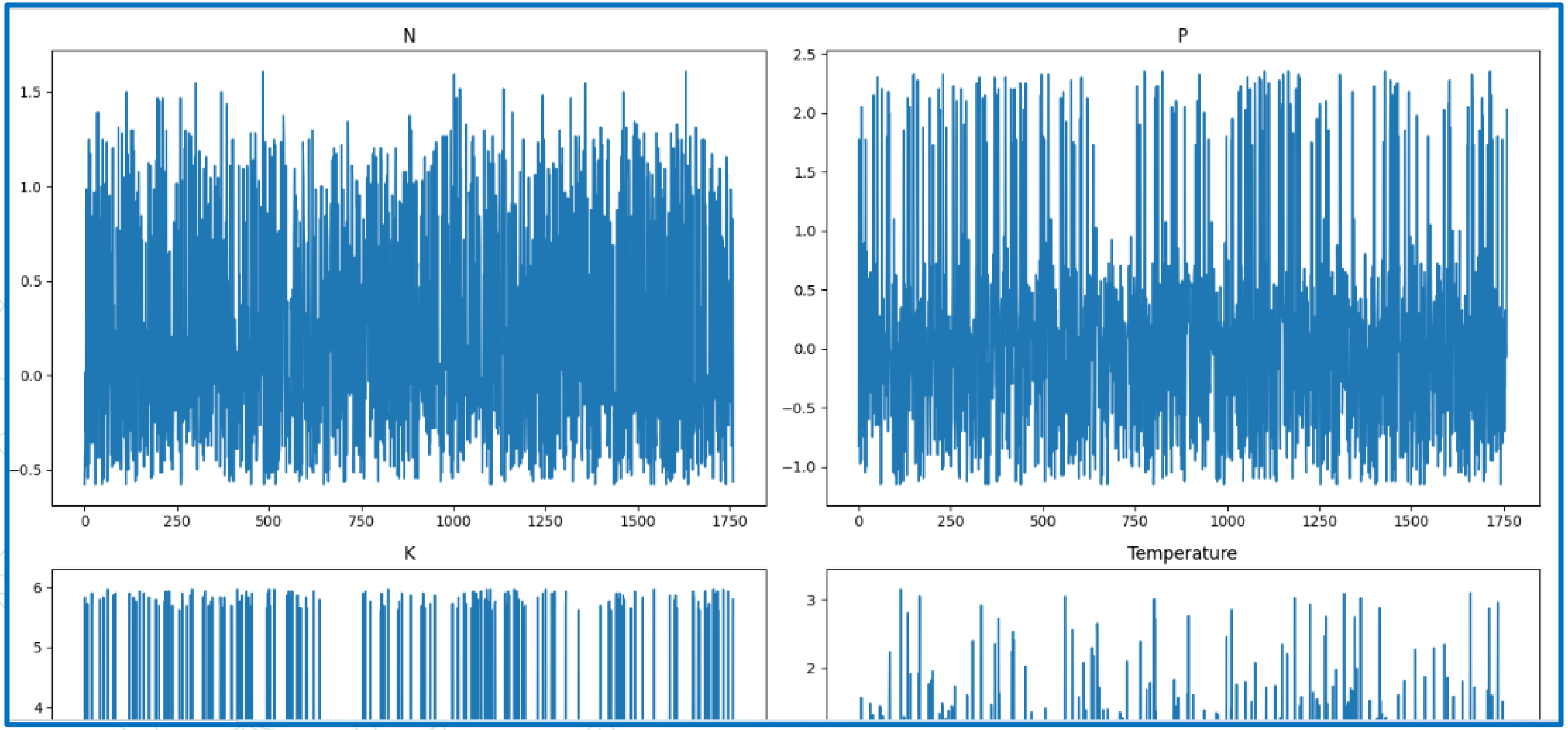
1760 rows × 5 columns

ROBUST SCALER



Before Scaling

ROBUST SCALER



After Scaling

MODEL DEVELOPMENT

Random Forest Model

```
n_estimators = 1000
criterion = "gini"
random_state = 42
max_depth = 10
# Log parameters
mlflow.log_param("n_estimators", n_estimators)
mlflow.log_param("criterion", criterion)
mlflow.log_param("random_state", random_state)
mlflow.log_param("max_depth", max_depth)


# Step 1: Train the Random Forest Model
rf_model = RandomForestClassifier(n_estimators=n_estimators, criterion=criterion,random_state=random_state,max_depth=max_depth)
rf_model.fit(transformed_x_train, transformed_y_train)
```

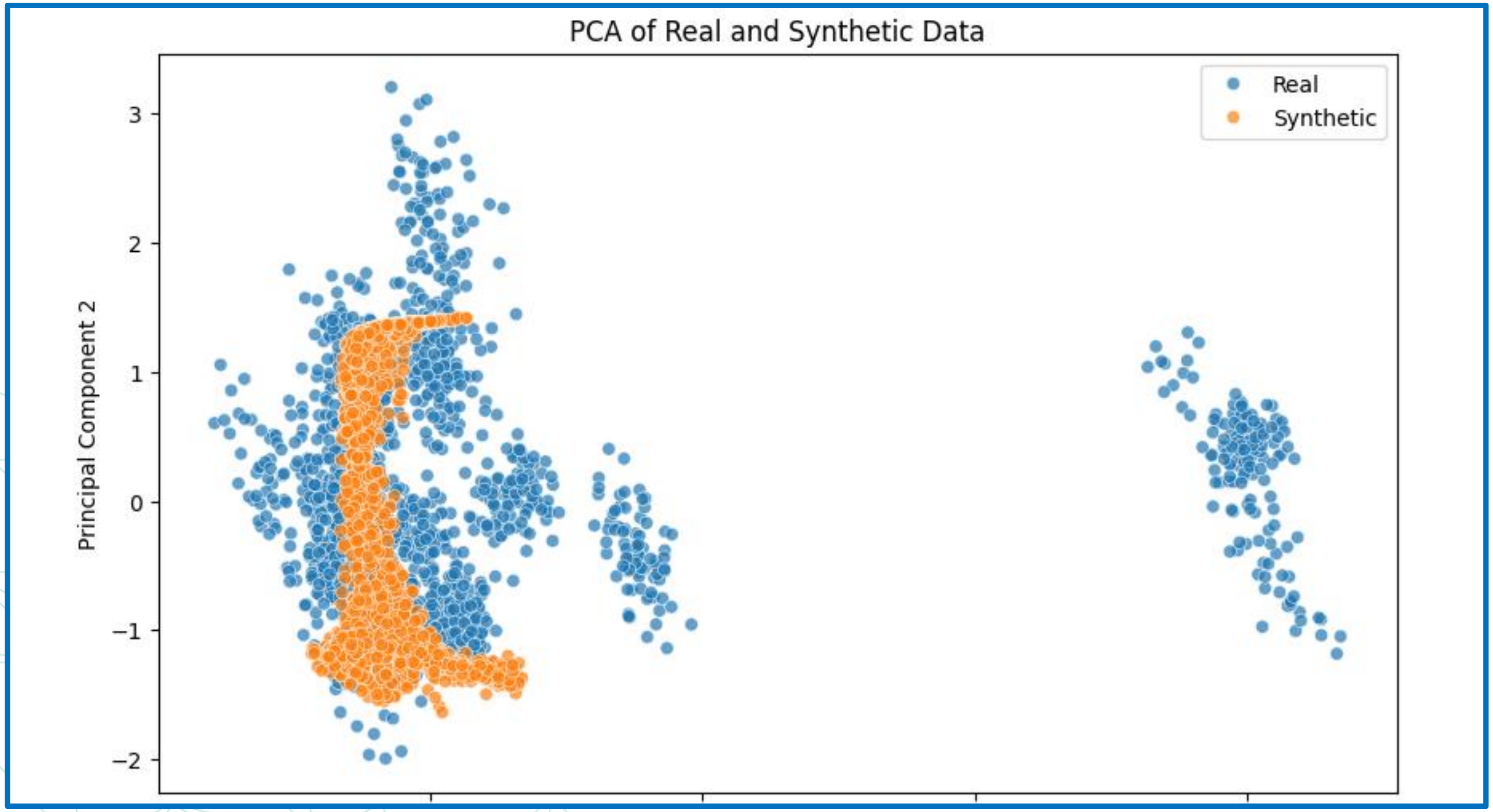
```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98        70
           1       1.00      0.99      0.99        80
           2       0.95      1.00      0.98        40
           3       1.00      1.00      1.00        20
           4       1.00      1.00      1.00        10

    accuracy                           0.99       220
   macro avg       0.99      0.99      0.99       220
weighted avg       0.99      0.99      0.99       220
```

GAN

```python
class Generator(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_dim, 256),  # Starting with fewer units
            nn.LeakyReLU(0.2),
            nn.Linear(256, 512),
            nn.LeakyReLU(0.2),
            nn.Linear(512, output_dim),
           # nn.LeakyReLU(0.2),
           # nn.Linear(1024, output_dim),
            nn.Tanh()  # Output synthetic crop features
        )

    def forward(self, x):
        return self.model(x)
```

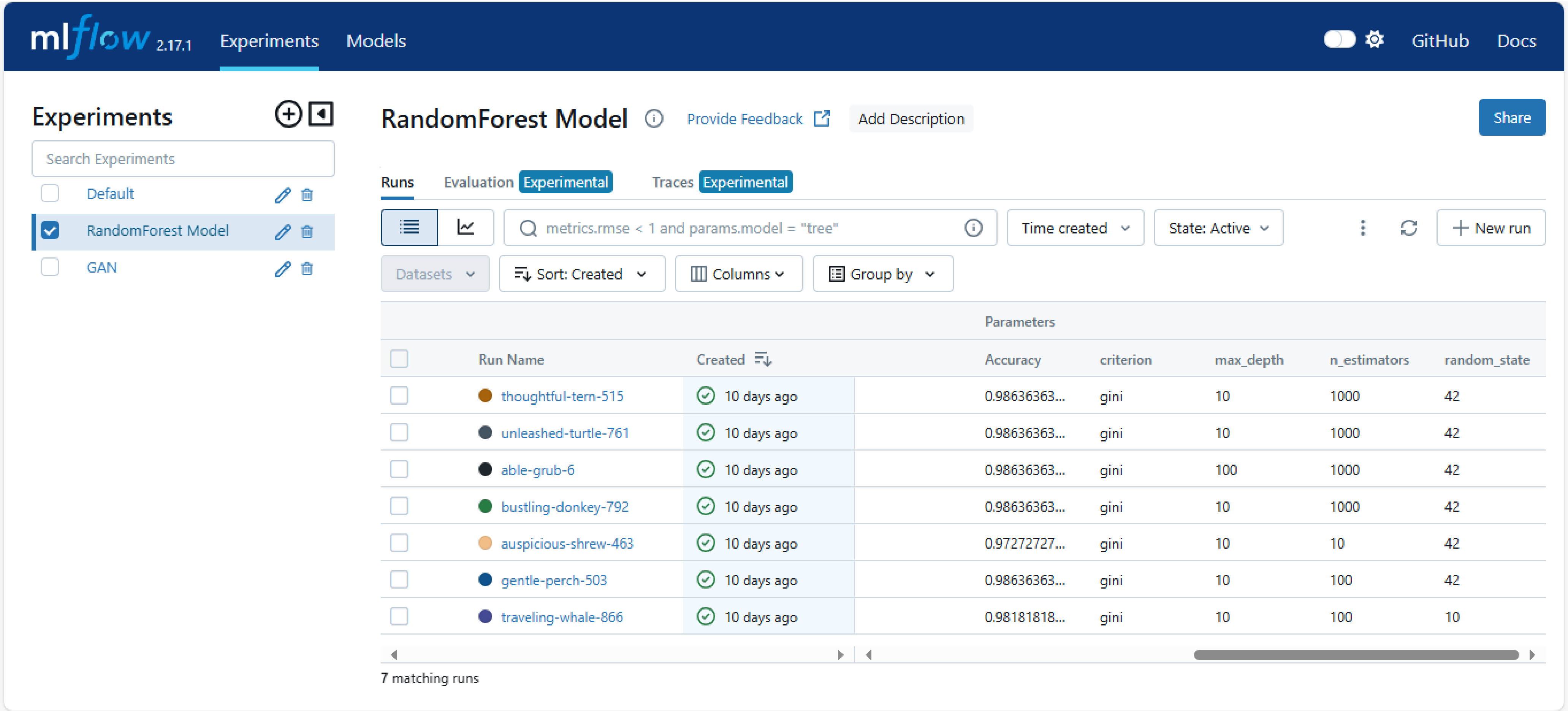
```python
class Discriminator(nn.Module):
    def __init__(self, input_dim):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            #nn.Linear(input_dim, 1024),
            #nn.LeakyReLU(0.2),
            nn.Linear(input_dim, 512),
            nn.LeakyReLU(0.2),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 1),
            nn.Sigmoid()  # Sigmoid output for binary classification
        )

    def forward(self, x):
        return self.model(x)
```

GAN

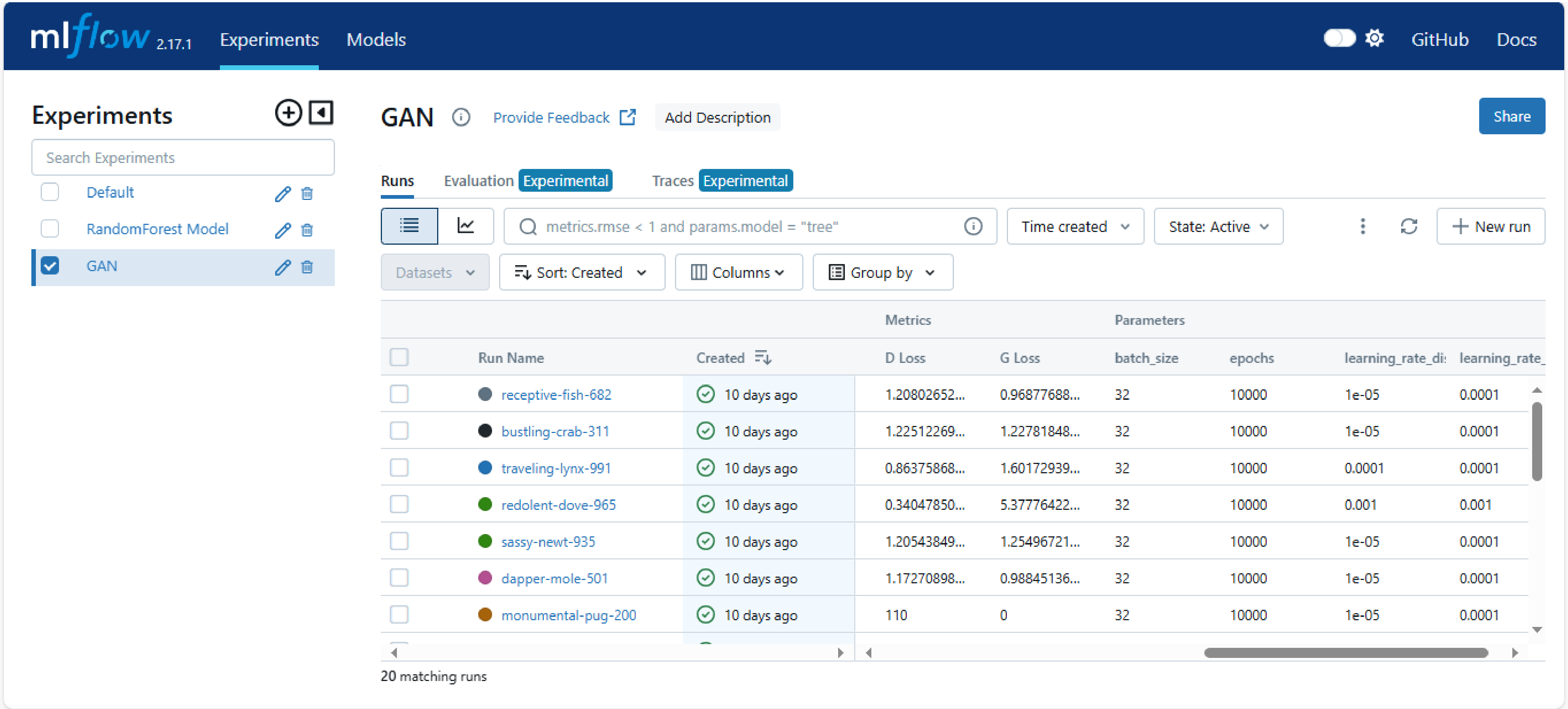

PCA of Real and Synthetic Data
Real
Synthetic
Principal Component 2
3
2
1
0
−1
−2

EXPERIMENT TRACKING

RANDOMFOREST MODEL

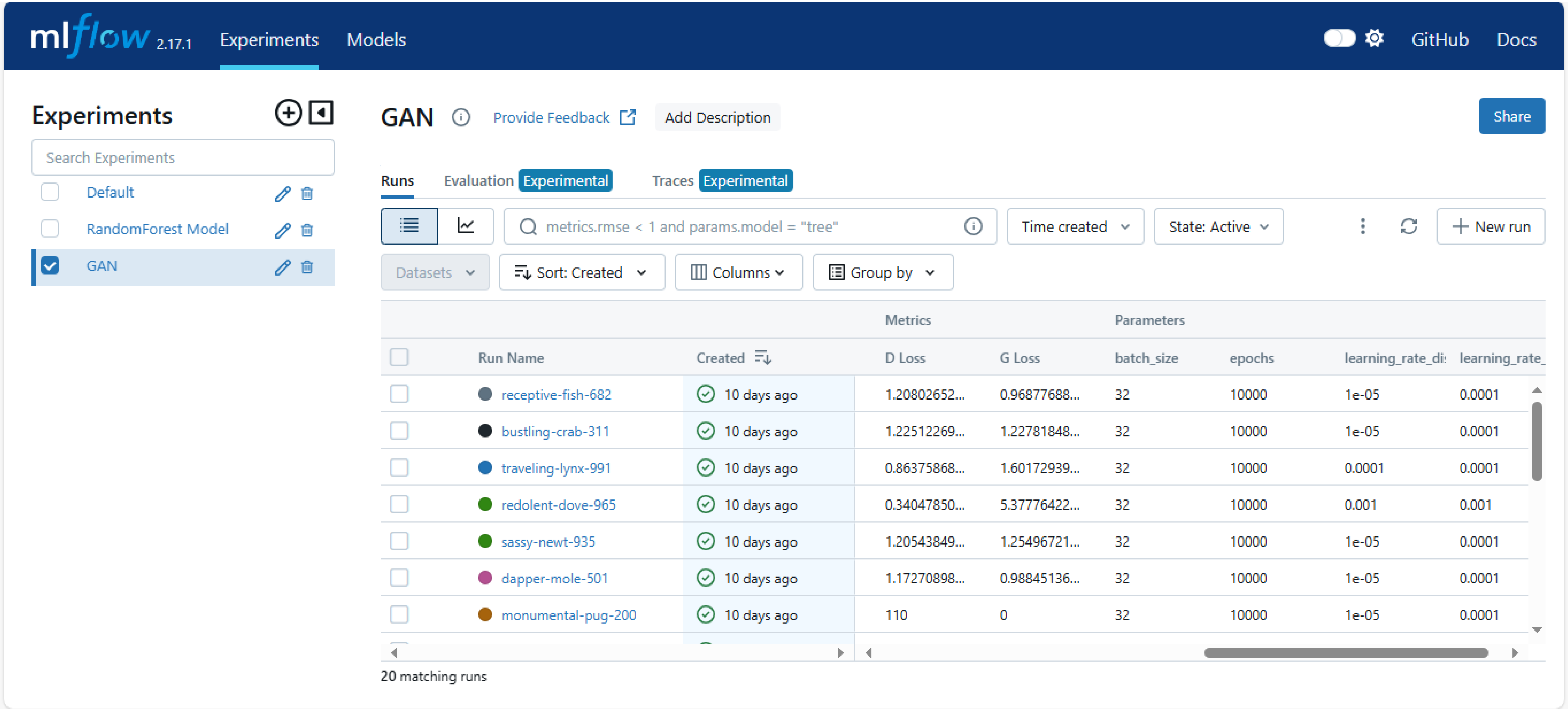


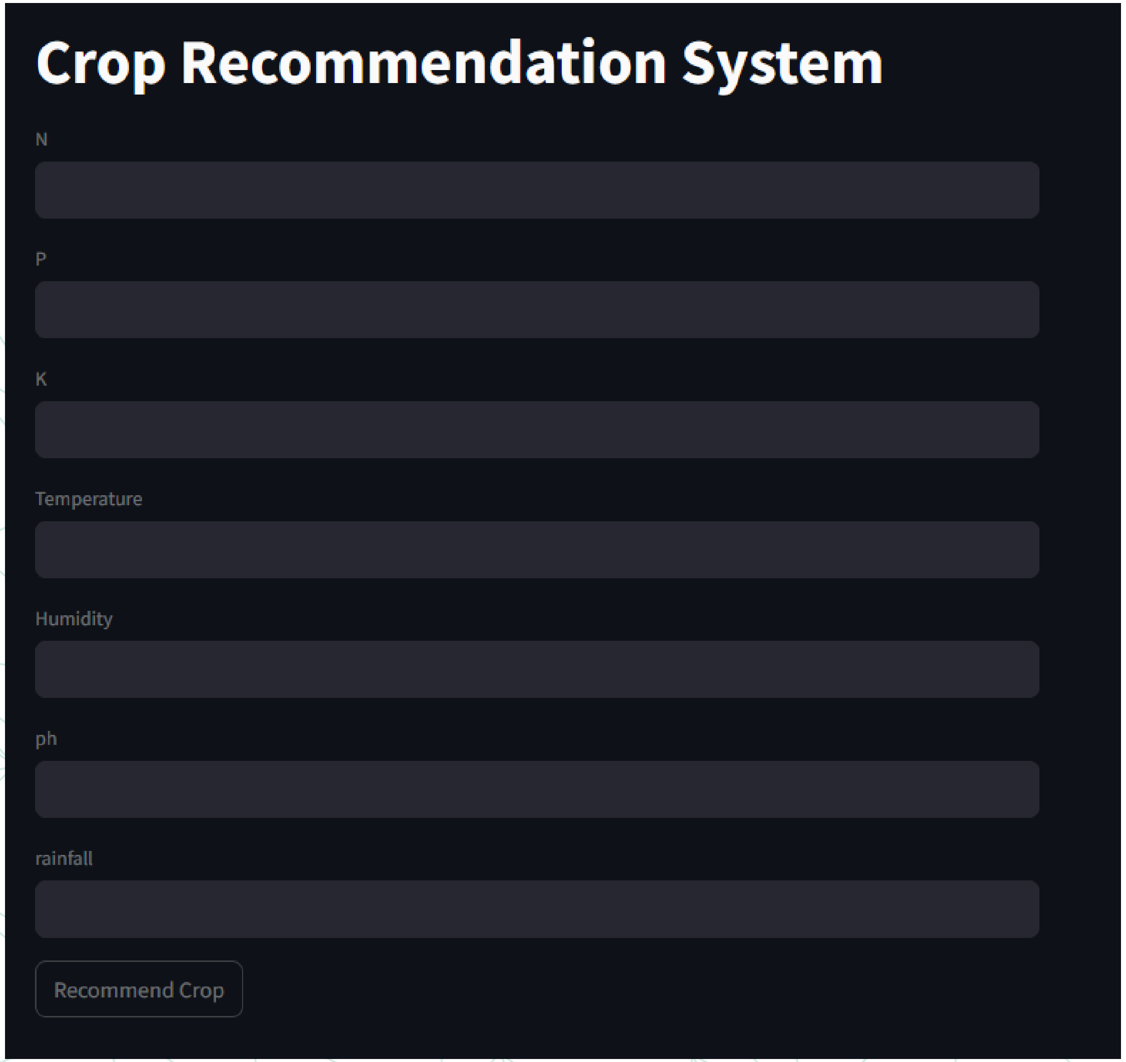
GAN


mlflow 2.17.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
RandomForest Model
GAN
GAN
Provide Feedback
Add Description
Share
Runs
Evaluation Experimental
Traces Experimental
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
New run
Datasets
Sort: Created
Columns
Group by
Metrics
Parameters
Run Name
Created
D Loss
G Loss
batch_size
epochs
learning_rate_di:
learning_rate_
receptive-fish-682 10 days ago 1.20802652... 0.96877688... 32 10000 1e-05 0.0001
bustling-crab-311 10 days ago 1.22512269... 1.22781848... 32 10000 1e-05 0.0001
traveling-lynx-991 10 days ago 0.86375868... 1.60172939... 32 10000 0.0001 0.0001
redolent-dove-965 10 days ago 0.34047850... 5.37776422... 32 10000 0.001 0.001
sassy-newt-935 10 days ago 1.20543849... 1.25496721... 32 10000 1e-05 0.0001
dapper-mole-501 10 days ago 1.17270898... 0.98845136... 32 10000 1e-05 0.0001
monumental-pug-200 10 days ago 110 0 32 10000 1e-05 0.0001
20 matching runs

MODEL DEPLOYMENT

GAN


mlflow 2.17.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
RandomForest Model
GAN
GAN
Provide Feedback
Add Description
Share
Runs
Evaluation Experimental
Traces Experimental
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
New run
Datasets
Sort: Created
Columns
Group by
Metrics
Parameters
Run Name | Created | D Loss | G Loss | batch_size | epochs | learning_rate_di: | learning_rate_
receptive-fish-682 | 10 days ago | 1.20802652... | 0.96877688... | 32 | 10000 | 1e-05 | 0.0001
bustling-crab-311 | 10 days ago | 1.22512269... | 1.22781848... | 32 | 10000 | 1e-05 | 0.0001
traveling-lynx-991 | 10 days ago | 0.86375868... | 1.60172939... | 32 | 10000 | 0.0001 | 0.0001
redolent-dove-965 | 10 days ago | 0.34047850... | 5.37776422... | 32 | 10000 | 0.001 | 0.001
sassy-newt-935 | 10 days ago | 1.20543849... | 1.25496721... | 32 | 10000 | 1e-05 | 0.0001
dapper-mole-501 | 10 days ago | 1.17270898... | 0.98845136... | 32 | 10000 | 1e-05 | 0.0001
monumental-pug-200 | 10 days ago | 110 | 0 | 32 | 10000 | 1e-05 | 0.0001
20 matching runs



Crop Recommendation System UI

N: ratio of Nitrogen content in soil
P (ratio of Phosphorous content in soil)
K (ratio of Potassium content in soil)
Temperature (temperature in degree Celsius
Humidity (relative humidity in %)
pH (ph value of the soil)
Rainfall (in mm)

DEPLOYED ON STREAMLIT

End Users + Features

Primary User Personas:

- **Farmers** – Individuals seeking tailored crop recommendations based on their soil and environmental conditions to optimize yields.
- **Agricultural Consultants** – Professionals aiming to provide data-driven advice to farmers for crop selection and management practices.
- **Educational Institutions** – Organizations looking to educate students and communities about sustainable agriculture practices and crop optimization.

Future Key Features and Their Impact:

Data Analytics and Reporting

User Needs: In-depth insights into soil health, crop performance, and climate trends.

Problem Solved: Empowers farmers and consultants to make informed decisions, enhancing crop yields and sustainability through data-driven strategies.

Collaborative Tools

User Needs: Efficient sharing of data and recommendations among farmers, consultants, and educational institutions.

Problem Solved: Facilitates communication and knowledge sharing, enabling collaborative efforts in crop management and education.

Mobile Accessibility

User Needs: Access to recommendations and data while on the field or during farm activities.

Problem Solved: Ensures that farmers and consultants can stay connected to real-time insights and recommendations, enhancing responsiveness and management capabilities.

Data Structure

Data Structure
Database Architecture:
Type: Using **CSV File.**

Key Entities and Relationships:
- **Soil Nutrients** – Columns include **N**, **P**, **K** (Nitrogen, Phosphorus, Potassium).
- **Environmental Factors** – Columns include **Temperature**, **Humidity**, and **Rainfall**.
- **Soil Condition** – Column **pH** reflects soil acidity/alkalinity.
- **Relationships**: Each crop's recommended nutrient profile relates directly to environmental factors and soil conditions. Data is aggregated per crop type, allowing recommendations based on historical and climate data.

Data Flow:

- **Data Collection Source**: Kaggle dataset Crop Recommendation Dataset.
- **Data Characteristics**:
    - **Columns**: N, P, K, Temperature, Humidity, pH, Rainfall.
    - **Shape**: *[e.g., 2200 rows x 7 columns]*.
    - **Balance**: The dataset is *[balanced]* for different crop types, ensuring varied but comprehensive recommendations.

Data Storage and Access:
- **Storage**: CSV format
- **Access**: Machine learning models retrieve data for training, testing, and recommendation generation.

Techniques:
- **Feature Scaling**: Normalize temperature, humidity, pH, and nutrient levels.
- **Preprocessing**: Address any null values and adjust imbalanced data if necessary.

Programming Languages & Frameworks

Main Programming Languages:

1. Python – Primary language for data analysis, machine learning, and model building.
2. Torch – Used for model development and deep learning.

Frameworks and Tools:

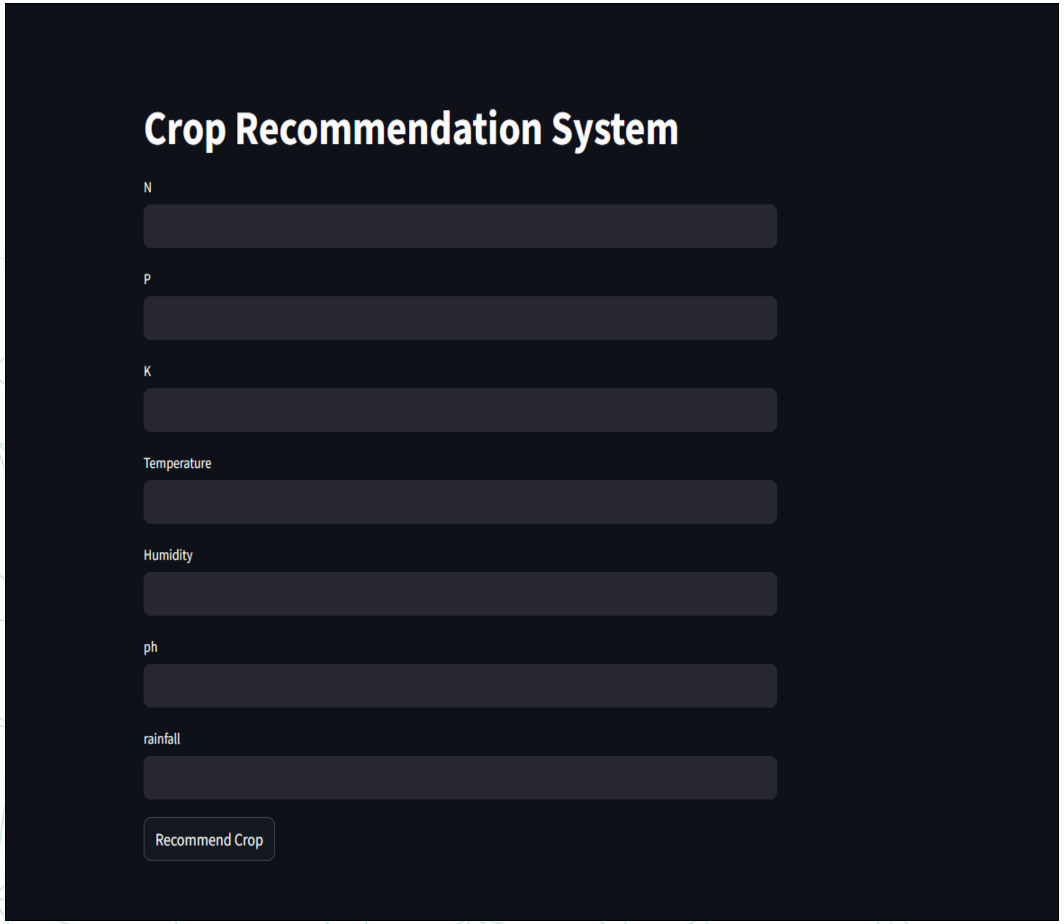
1. GANs (Generative Adversarial Networks) – For creating synthetic data to enhance model training.
2. Google Colab/Jupyter Notebook – Cloud-based platform for code execution and model training.
3. MLflow – Tracking and managing machine learning experiments.
4. Streamlit – Building an interactive front end for model deployment and user interaction.

Libraries:

- Data Manipulation: numpy, pandas
- Data Visualization: matplotlib, seaborn
- Encoding and Scaling: category_encoders (BinaryEncoder), sklearn.preprocessing (RobustScaler)
- Machine Learning: sklearn (RandomForestClassifier, model evaluation metrics)
- Deep Learning: tensorflow, torch (with modules like torch.nn, torch.optim)
- Recommender System (optional): cosine_similarity, surprise (SVD for collaborative filtering)

Supporting Technologies:

- Cloud Platform – Google Colab acts as the cloud environment for collaborative work and large model computations.



Live Application

رواد مصر الرقمية

وزارة الاتصالات وتكنولوجيا المعلومات

Project Team & Roles

Name	Role
Abdelrahman Osama Mohamed Nabih	Week 1: Data Collection and Preprocessing
Abdelrhman Walaa Hussein	Week 2: ML and Recommendation Modeling
Hanin Essam Sayed Mohamed	Week 3: - Advanced Techniques - Refine ML - GANs - Deployment
Mostafa Mohamed Youssef	Week 4: - MLOps - Reviewing code - Final Presentation - Test deployment

رواد مصر الرقمية

وزارة الاتصالات وتكنولوجيا المعلومات

Project Team & Roles cont'd

Project Pipeline:

- Week 1-2: Data Collection and Preprocessing
- Week 3-4: Model Development and Experiment Tracking
- Week 5: Model Deployment and Testing
- Week 6: Adjustments, Presentation and Project Completion

Thank YOU!

Any Questions?

Reach out to the team:

he106400@gmail.com
moustafa.youssuf@gmail.com
abdelrahmanosama@aucegypt.edu
abdelrahmanwalaa96@gmail.com