Crop Recommendation System

AI 705: Recommendation Systems, International Institute of Information Technology Bangalore, 2024

Nisha Rathod
dept. Mtech CSE
MT2023195
rathod.nisha@iiitb.ac.in

Somya Malgudi
dept. Mtech CSE
MT2023165
somya.malgudi@iiitb.ac.in

Akshansh Jain
dept. Mtech CSE
MT2023030
akshansh.jain@iiitb.ac.in

***Abstract*—This paper introduces a crop recommendation system utilizing a bandit algorithm framework to optimize agricultural outputs. By leveraging multi-armed bandit algorithms, the system dynamically adapts to varying environmental conditions and historical crop performance data to recommend the most suitable crop types for specific plots of land.**

***Index Terms*—Content Based, UCB, Thomsons sampling, Epsilon greedy, Deep learning**

I. Introduction

In the realm of precision agriculture, optimizing crop selection based on real-time data and environmental conditions is crucial for enhancing productivity and sustainability. This paper introduces a crop recommendation system that employs multi-armed bandit algorithms to effectively address these challenges. This system continuously learns from environmental inputs and crop performance, providing tailored recommendations that aim to maximize yield and resource efficiency. Initial simulations demonstrate significant improvements in crop yield and resource utilization compared to traditional methods. The system's scalability and adaptability make it a promising tool for modern precision agriculture.

II. Data collection and Generation

A. Data Collection

Dataset was taken from kaggle (Crop Recommendation Dataset - ATHARVA INGLE). It consisted of following Data fields: 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 - rainfall in mm. For each unique crops there were 100 corresponding rows and there were only 22 unique rows, so we decided to generate more unique crops and compress data into single row.

B. Data Generation

We had 22 crops in our dataset, to generate more crops we first found correlation between columns. Now to make cohorts we found a threshold using IQR (0.29) and noticed that out data 2 cohorts.

Cohort 1: K-mean, P-mean
Cohort 2: humidity-mean, temperature-mean

```
imported_data = pd.read_csv("/content/imported_crops.csv")
print(imported_data.head())
```

```
        Name    K_mean    N_mean    P_mean  temperature_mean  humidity_mean  \
0    Broccoli -0.395067  0.072551 -0.299363         -0.730196      -2.464421
1       Kiwi  2.862921  1.770581  2.169387         -0.478527      -1.615034
2  Strawberry  2.785872 -0.024274  2.111003          0.211918       0.715227
3   Raspberry  0.101274  1.882509  0.076741         -0.042715      -0.144162
4  Blackberry  1.407498  0.950132  1.066535         -0.302464      -1.020819

    ph_mean  rainfall_mean
0 -0.741455       0.391420
1 -1.031437      -0.317673
2  0.089651      -1.090468
3  1.672384       1.003334
4  1.163085      -0.754451
```

```
populate_data = pd.read_csv("/content/populate_data.csv")
print(populate_data.head())
```

```
       Name    K_mean    N_mean    P_mean  temperature_mean  humidity_mean  \
0      Rice  2.589890 -0.680122 -0.739135          0.313570       1.058304
1     Wheat  1.225176  0.390698 -0.728476          0.004340       0.014647
2    Cotton  0.747869  0.391575 -0.717816         -0.368993      -1.245356
3      Jute  0.037714  0.308731 -0.707157          0.019014       0.064172
4  Sugarcane  0.568869  1.284250 -0.696498         -0.540578      -1.824458

    ph_mean  rainfall_mean
0  0.703327       1.395175
1  1.799890       0.326655
2 -0.164976       1.836870
3 -1.308706       1.988913
4  0.605424      -0.663376
```

Fig. 1. Imported and Local Crops data after processing

Now we fixed value of one of the columns from each cohorts and predicted for the other using Linear Regression. We have total for 100 local crops and 20 imported crops in our dataset now, and flexibility to increase this as much as we want.

C. Data Preprocessing

Now we have taken mean of min and max columns of each crop and made entry for it. Then we normalized each values in scale of -1 to 1. Categorical column 'label' was converted to numerical using One hot encoding. Finally we have made 2 datasets one for local crops and one for the crops that are imported in India. This division helps us promote agriculture of exotic crops which further down the line reduced the import rate. Fig. 1. shows head of both the csv files.

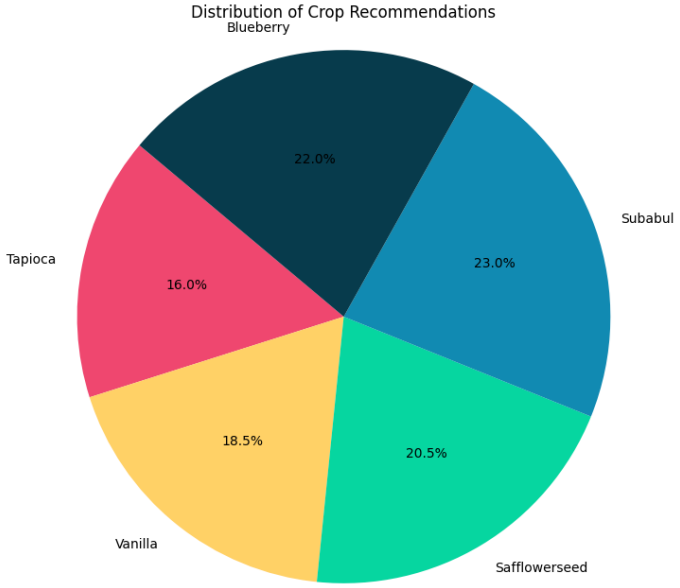
D. Farmer's Input

User profiling is done by taking soil content into consideration. After each season farmer gives rating to the crops sown. Our algorithm refines recommendations based on this temporal data. Fig. 2. shows sample input given by farmer when they first entered the system.

```
user_profile = get_user_input()
print(user_profile)
```

```
Enter K_mean value: 1.2
Enter N_mean value: 0.2
Enter P_mean value: 1.0
Enter temperature_mean value: 0.8
Enter humididty_mean value: -0.2
Enter ph_mean value: -0.8
Enter rainfall_mean value: 0.5
Finding best crops for you...
[ 1.2  0.2  1.   0.8 -0.2 -0.8  0.5]
```

Fig. 2. Farmers input



III. Content Based And Multi-Arm Bandit

A. Content Based

We use content based method to recommend 5 crops to farmer, this includes 4 local crops and 1 exotic crop. A content-based crop recommendation system leverages the power of data analysis to guide farmers towards optimal choices. By analyzing soil composition (nitrogen, phosphorus, potassium, and pH), climatic factors (temperature, humidity, and rainfall), the system calculates similarities between various crops. When a farmer enters their specific field data, the system identifies the closest matches based on these parameters. This shortlist is then refined by considering additional factors like market demands, profitability, and suitability for crop rotation practices. The final recommendation list provides the farmer with a data-driven selection of crops likely to thrive in their specific growing conditions, along with additional information to support informed decision-making.

```
recommendations = np.concatenate((recommendations_local, recommendations_imported))
print("Recommended crops:")
print(recommendations)
```

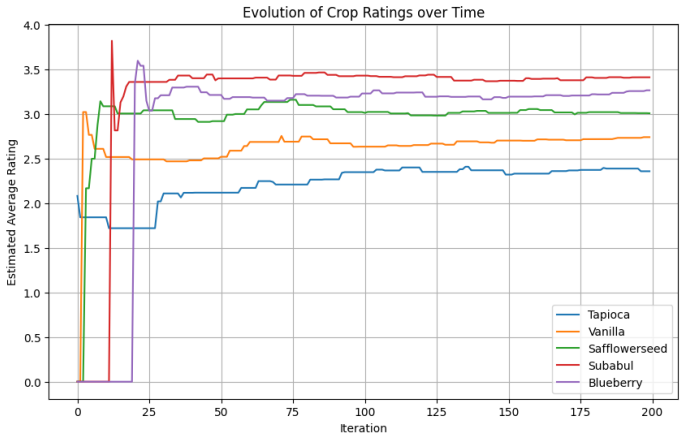
```
Recommended crops:
['Tapioca' 'Vanilla' 'Safflowerseed' 'Subabul' 'Blueberry']
```

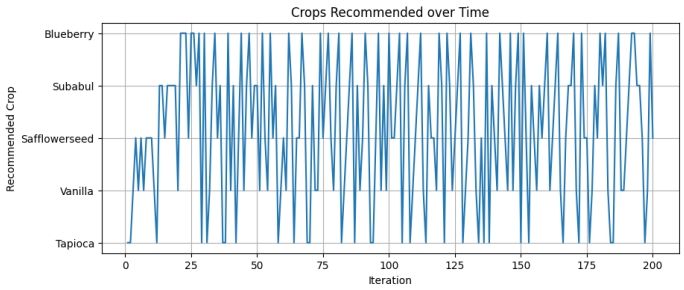
B. Multi-arm Bandit

The crops recommended from content based are passed on to MAB, where each crop is a arm and algorithm picks the most suitable arm.

a) Epsilon Greedy: Epsilon-greedy is an exploration-exploitation strategy for decision making. It balances choosing the known best option (exploitation) with exploring random options (exploration) to potentially discover better choices. With a small probability (epsilon), the algorithm takes a random action, helping it escape local optima and find potentially better solutions. We have fixed value of epsilon to be 0.3.

This pie chart shows how many times each crop was picked by epsilon greedy





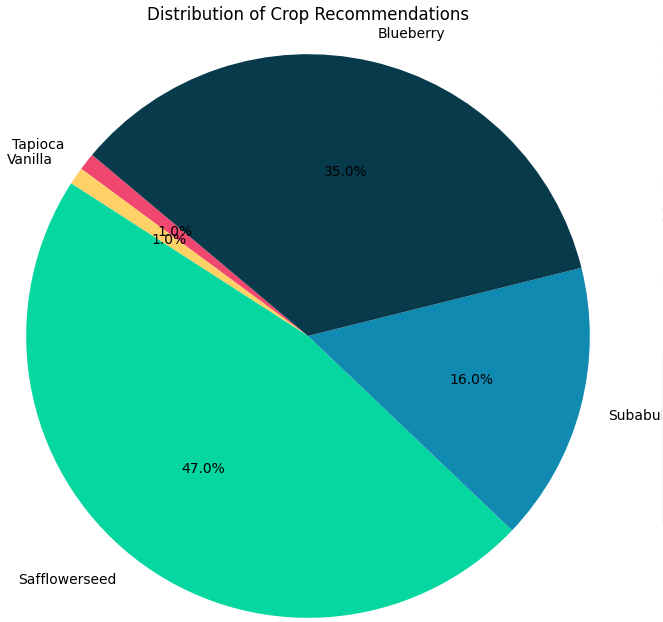
Above figure shows output of epsilon greedy the point to note here is the number of times each crop was recommended over 200 iterations:
Tapioca: 32
Vanilla: 37
Safflowerseed: 41
Subabul: 46
Blueberry: 44
Here Subabul is recommended most of the times we will later relate this to inference while doing regret analysis.

b) Thompsons Sampling: Thompson Sampling tackles the multi-armed bandit problem by balancing exploration and exploitation. It maintains probability distributions for each option's reward. It then samples from these distributions to choose an option, favoring those with potentially higher rewards (exploration). As it gathers data, the distributions update, focusing on options that consistently deliver good rewards (exploitation). The value of alpha and beta are set to 1 initially for this project. It is a Bayesian approach to the multi-armed bandit problem. Instead of directly keeping track of average rewards, it models belief about each crop's reward with a probability distribution. In each iteration, it does the following:

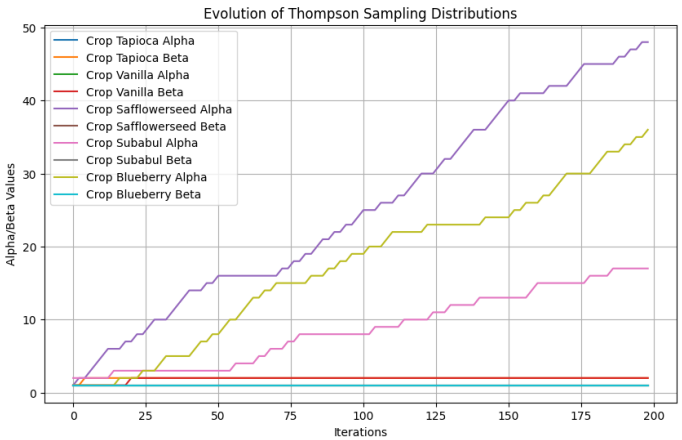
Sample: For each crop, it draws a random sample from its reward distribution.

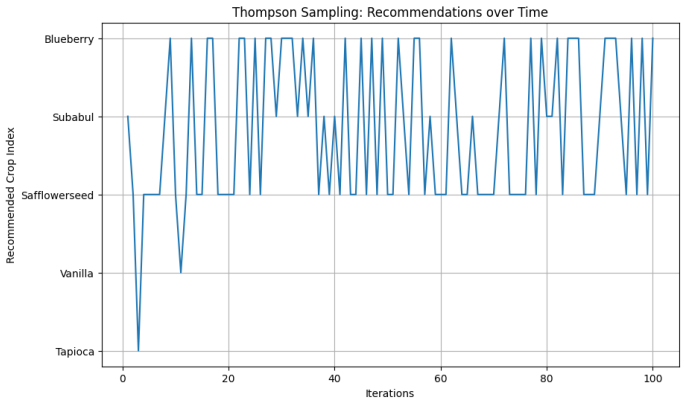
Select: It chooses the crop with the highest sampled reward.

Update: It updates the reward distribution based on the feedback received.



This shows how many times each crop was picked by thompsons sampling.





Above figure shows output of epsilon greedy the point to note here is the number of times each crop was recommended over 100 iterations:

Tapioca: 1

Vanilla: 1

Safflowerseed: 47

Subabul: 16

Blueberry: 35

Here Blueberry is recommended most of the times we will later relate this to inference while doing regret analysis.

c) Regret analysis: For this project we have assumed farmers ratings to be Gaussian distributed. So, for every recommended crop we have added a assumed mean so that we can have an optimal crop for the soil to calculate regret from in ou case Sababul has the highest mean rating so out of the 5 crops regret will be calculated compared to Sababul. Hence, Sababul should have least regret out of 5.

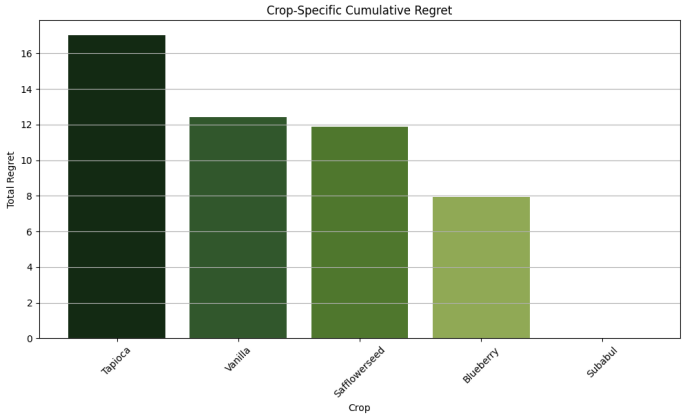
```
num_iterations = 200
means = [2.5, 2.8, 3.0, 3.5, 3.2]  # Mean ratings for each crop
std_dev = 0.5  # Standard deviation will be constant for each of them
num_crops = len(means)
crop_ratings_over_time = [[] for _ in range(num_crops)]
print(recommendations)
print(means)
```

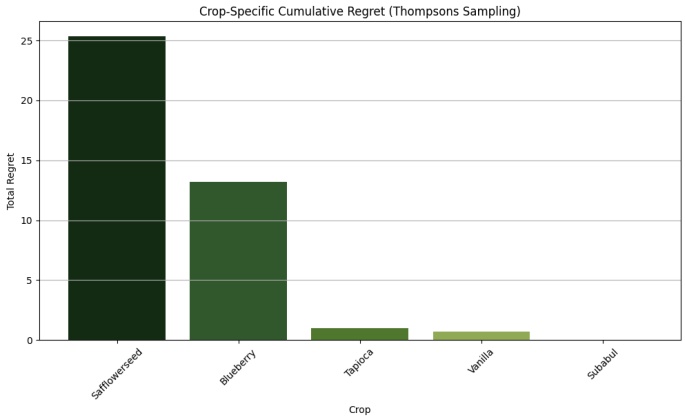
```
['Tapioca' 'Vanilla' 'Safflowerseed' 'Subabul' 'Blueberry']
[2.5, 2.8, 3.0, 3.5, 3.2]
```

In Epsilon greedy we can see that Sababul was recommended most of the times because Epsilon greedy picks up best crop with 0.3 probability.In this case our regret was linear. But in case of thompson sapling Sababul wasn't the one that was most recommended. Early in the sampling, the algorithm likely explored different crops more evenly because the initial lack of data causes high uncertainty in the estimates of their reward distributions. This uncertainty is represented by broader Beta distributions, which means that even a crop with a lower average rating can still occasionally show high potential yield (a high sample from the Beta distribution).

Over time, as more samples are drawn, Thompson Sampling should theoretically converge towards choosing the best crop more frequently, assuming the reward (rating) distributions are stationary and well-modeled. However, random fluctuations and the inherent randomness in the Beta sampling can result in periods where suboptimal crops are chosen more often, especially visible in a limited number of iterations like 200. If Safflowerseed happened to yield

ratings above your threshold early on, it would lead to a more rapid increase in its alpha parameter (indicating more positive outcomes), even if by chance. This would make its Beta distribution shift towards higher values, making it more likely to be chosen in subsequent iterations. Conversely, if Sababul had slightly worse outcomes or less dramatic positive ratings, it might not have boosted its parameters as much, leading to it being chosen less frequently.





C. Upper Confidence Bound

Upper Confidence Bound (UCB) tackles the challenge of balancing established user preferences with the potential for discovering hidden gems. It achieves this by assigning a score to each item. This score combines an item's historical value (like purchase rate) with a bonus that encourages exploration of less-recommended items. As user interactions accumulate, UCB refines its scores, favoring items that consistently generate positive responses. This approach prevents the system from getting stuck on popular but potentially sub-optimal choices, promoting exploration and potentially leading to the discovery of new favorites for users.

UCB offers a deterministic approach, meaning it calculates a single score for each item based on a clear formula. This allows for easier analysis and control of the exploration-exploitation balance. Thompson Sampling, on the other hand, relies on random sampling from probability distributions, introducing an element of chance in the selection process.

In general, UCB tends to converge on optimal recommendations faster than Thompson Sampling, especially when dealing with a large number of items. This is because UCB prioritizes items with a high exploration bonus initially, leading to quicker exploration of the item space.

Thompson Sampling can be more sensitive to noise in user data, such as random purchases or clicks. This can lead to the algorithm getting stuck on sub-optimal recommendations due to a single positive interaction. UCB, with its focus on average reward and exploration bonus, can be less susceptible to this issue.

The UCB score provides a clear interpretation of an item's value, combining its historical performance and exploration potential. This allows for easier debugging and understanding of the recommendation process. Thompson Sampling's reliance on probability distributions can be less interpretable.

If user preferences change rapidly, Thompson Sampling's ability to adapt to new data through its probabilistic nature might be beneficial.

```python
# Farmer preferences with mean values for various factors
farmer_preferences = {
    'K_mean': -2,
    'N_mean': -1,
    'P_mean': 0.3,
    'temperature_mean': -1.1,
    'humidity_mean': -0.2,
    'ph_mean': 1,
    'rainfall_mean': 2,
    'budget': 1000  # Add farmer's budget
}
```

In previously discussed algorithms we didn't consider budget as a factor for farmers so this time we added it in user profiling then we apply UCB. From a linear regression perspective, each crop can be seen as a separate regression problem where we try to predict the reward (yield or profit) based on a set of features (such as soil quality, weather conditions, etc.). The Linear UCB algorithm extends this by incorporating uncertainty estimates into the decision-making process.

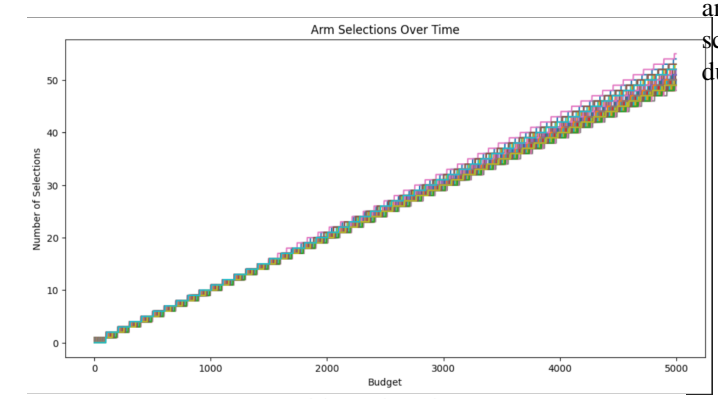
Features: Each crop is associated with a feature vector. These features represent the characteristics of the crop and its environment, which are used as inputs to the regression model.

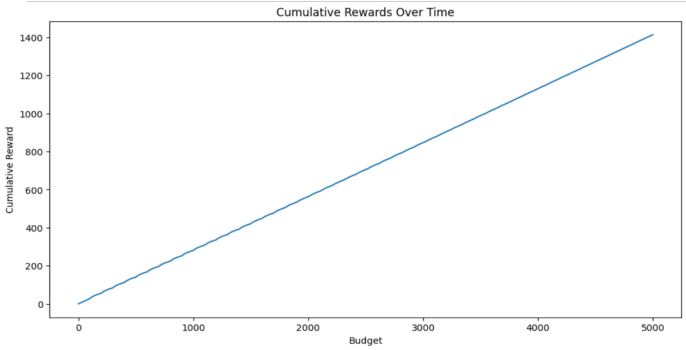
Reward Prediction: The regression model aims to predict the reward (yield or profit) associated with each crop based on its features. In this implementation, the estimated parameters (thhat) represent the coefficients of the linear regression model.

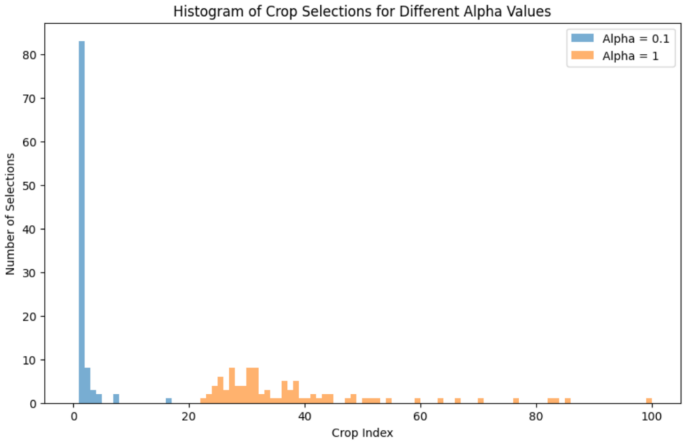
Exploration-Exploitation Trade-off: The algorithm balances exploration and exploitation by selecting crops with the highest Upper Confidence Bound (UCB) values. The UCB values incorporate both the estimated mean reward and a measure of uncertainty. This uncertainty is crucial for exploration, allowing the algorithm to explore crops with uncertain rewards to gather more information.

Parameter Updates: The algorithm updates the parameters (A and b) based on the observed rewards, similar to how parameters are updated in online linear regression. These

updates are designed to improve the accuracy of reward predictions over time.







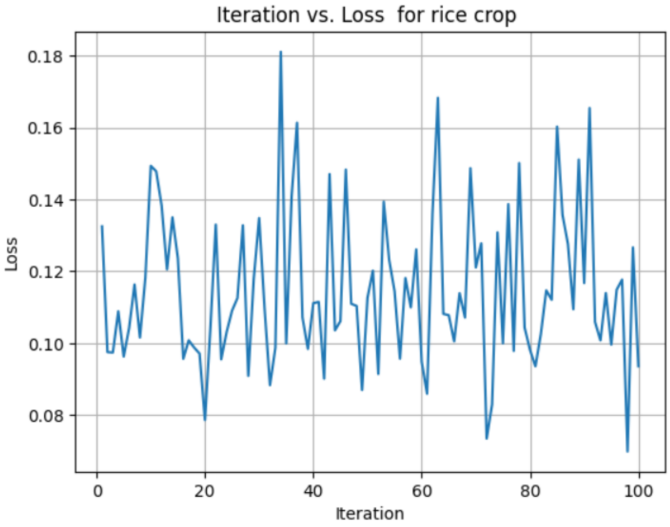
Overall, the goal is to maximize the cumulative reward obtained by the farmer over a series of crop selections, taking into account both the estimated rewards and uncertainties associated with each crop.

D. Deep Learning

So, finally we are trying to predict yield of the crop we do this by deep learning. We have a three layered neural network where the output is likelihood of recommending a crop. Deep learning can revolutionize crop recommendation systems by going beyond simple feature analysis. These powerful algorithms can learn complex, non-linear relationships between soil properties, climatic data, historical crop yields, and even satellite imagery. This allows the system to capture intricate patterns and identify subtle interactions that might be missed by traditional methods. As a result, deep learning models can generate highly personalized recommendations, optimizing crop selection for a farmer's specific conditions and potentially boosting agricultural yields. There is a vast scope but we were not able to implement these functionalities due to time constraints.

ph	rainfall	rice	maize	chickpea	...	Walnuts	Betel nuts	Saffron	Black pepper	Vanilla	Cocoa (for chocolate)	Tobacco	Opium poppy (for pharmaceuticals)	Rubber
6.502985	202.935536	1.000000	0.478097	0.345581	...	0.414586	0.118429	0.125272	0.372220	0.413866	0.295368	0.237662	0.168658	0.406253
7.038096	226.655537	1.000000	0.970426	0.116768	...	0.660926	0.289212	0.201984	0.728562	0.783968	0.755196	0.328535	0.159942	0.035712
7.840207	263.964248	1.000000	0.445445	0.952519	...	0.662534	0.243655	0.333669	0.414658	0.197860	0.282600	0.278283	0.437898	0.259250
6.980401	242.864034	1.000000	0.369750	0.306424	...	0.728671	0.657179	0.540246	0.198410	0.694863	0.022463	0.126366	0.410381	0.099195
7.628473	262.717340	1.000000	0.254595	0.534189	...	0.438542	0.210944	0.002217	0.208929	0.317599	0.265559	0.880377	0.000640	0.393815
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6.036764	196.676977	0.673648	0.257931	0.339002	...	0.098185	0.153412	0.203208	0.019298	0.476018	0.037724	0.303925	0.087877	0.912975
4.774558	217.397726	0.297328	0.213116	0.240650	...	0.074088	0.303383	0.372145	0.333923	0.196423	0.328375	0.396961	0.135815	0.392952
4.760295	52.777676	0.476840	0.238219	0.253580	...	0.213546	0.919879	0.650971	0.472679	0.295223	0.794290	0.646794	0.456419	0.030335
6.992872	112.977350	0.144716	0.043518	0.286779	...	0.695738	0.129894	0.424553	0.431950	0.420007	0.557357	0.019879	0.226291	0.808680
7.754700	77.738711	0.103499	0.977100	0.064997	...	0.250292	0.220798	0.358268	0.807187	0.279254	0.467349	0.371204	0.398340	0.011195

Since we need many number of classes and huge data to train deep learning backbone. So, we have made column for each crops as shown figure above and values are soil content of corresponding crop. Then we train our network for each crop separately and output is the predicted yield.




```
recommendation
✓ 0.0s
{'grapes': 0.53705894947052,
 'pomegranate': 0.5000537037849426,
 'Tomatoes': 0.4943333864212036,
 'Bananas': 0.4888961911201477,
 'Carrots': 0.47675299644470215,
 'Strawberry': 0.24196287989616394}
```

Deep learning excels at identifying complex patterns in massive datasets, uncovering hidden insights that might elude traditional methods. In recommendation systems, for instance, they can tailor suggestions to a user's specific preferences. Deep learning systems learn and improve over time as they are exposed to more data. This allows them to adapt to changing environments and refine their performance continuously.

ACKNOWLEDGMENT

We would like to extend our deepest gratitude to Professor Raghuram Bharadwaj for his invaluable guidance and insightful teachings in the fields of machine learning and recommendation systems. His expertise and mentorship have been fundamental to our understanding and interest in these areas.

We are also immensely grateful to our teaching assistants, whose dedication and assistance throughout the course were crucial to our learning experience. Their patience and support have greatly enhanced our ability to grasp complex concepts and apply them effectively.

REFERENCES

[1] Interquartile Range (IQR) - Statistics By Jim Interquartile Range (IQR) - Statistics By Jim
[2] Crop Recommendation Dataset by Atharva Ingle Crop Recommendation Dataset by Atharva Ingle
[3] Crop Rotations by Rodale Institute Crop Rotations by Rodale Institute
[4] AI 705: Recommendation Systems by Prof. Raghuram Bharadwaj