Crop Recommendation System Project Machine Learning

A COURSE PROJECT REPORT

By

Mowleeshwaran V R[RA201003010565]
Subeshreenee K A[RA2011003010566]
Aravinda Lochanan G[RA2011003010546]

Under the guidance of

ABHIRAMI

In partial fulfilment for the Course

of

18CSC305J – ARTIFICIAL INTELLIGENCE

in SCHOOL OF COMPUTING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

APRIL 2023

TABLE OF CONTENTS

ABSTRACT

This abstract describes a crop recommendation engine that leverages machine learning algorithms to provide farmers with personalized recommendations on the most suitable crops to plant based on their specific soil and weather conditions. The engine utilizes data on soil properties, weather patterns, and crop history to generate insights on crop suitability and yield potential. The engine takes into account the farmer's preferences and goals to provide tailored recommendations. The effectiveness of the crop recommendation engine is evaluated using various metrics, including crop yield, revenue, and farmer satisfaction. The results demonstrate that the engine significantly improves the farmer's productivity and profitability while reducing the risk of crop failure.

INTRODUCTION

A crop recommendation engine is a powerful tool that uses advanced algorithms and machine learning techniques to help farmers make informed decisions about which crops to plant, when to plant them, and how to manage them for optimal yields. By analyzing a variety of factors such as soil type, climate data, and crop history, these systems can provide personalized recommendations tailored to the specific needs of individual farms. The goal of a crop recommendation engine is to help farmers maximize their productivity, reduce waste and ultimately increase their profits while promoting sustainable agricultural practices. With the increasing demand for food and the challenges posed by climate change, such systems are becoming an essential tool for modern agriculture.

METHODOLOGY

The methodology for a crop recommendation engine typically involves several steps:

1. Data Collection: The first step is to collect data from various sources, including weather stations, satellite imagery, soil sensors, and farmer input. This data is then processed and standardized to make it suitable for analysis.

2. Data Preprocessing: The collected data is preprocessed to clean and transform it into a suitable format for further analysis. This step involves removing outliers, filling missing values, and standardizing the data.

3. Feature Selection: The next step is to identify the most relevant features for the crop recommendation model. This involves analyzing the correlation between different variables and selecting the ones that have the highest impact on crop yield.

4. Model Training: Once the relevant features are identified, the model is trained using a machine learning algorithm such as decision trees, random forests, or neural networks. The model is trained to predict the optimal crop type and cultivation practices based on the input data.

5. Model Evaluation: The trained model is evaluated on a separate set of data to test its accuracy and generalizability. The model is fine-tuned based on the evaluation results to improve its performance.

6. Deployment: The final step is to deploy the model as a crop recommendation engine. Farmers can input their farm data and receive personalized recommendations for the optimal crop type, planting date, fertilization, and irrigation practices.

Overall, the methodology for a crop recommendation engine involves collecting and preprocessing data, selecting relevant features, training and evaluating a machine learning model, and deploying the model as a useful tool for farmers.

IMPLEMENTATION & RESULTS

CONCLUSION

In conclusion, a crop recommendation engine is a valuable tool for modern agriculture, as it provides personalized recommendations to farmers on the optimal crop type and cultivation practices based on the input data. With the increasing demand for food and the challenges posed by climate change, crop recommendation engines are becoming an essential tool for farmers to maximize their productivity, reduce waste, and increase their profits while promoting sustainable agricultural practices.

The project involves several steps, including data collection, pre-processing, feature selection, model training, evaluation, and deployment. Various tools and programming languages such as Python, scikit-learn, Flask, and cloud platforms like AWS and Google Cloud are used to build a crop recommendation engine.

Overall, the success of a crop recommendation project depends on the quality of data collected, the accuracy of the machine learning model, and the user-friendliness of the deployed system. By leveraging the power of machine learning and AI, a crop recommendation engine can help farmers make informed decisions, optimize their yields, and contribute to sustainable agriculture.

REFERENCES

- w3schools: https://www.w3schools.com/python/default.asp
- matplotlib: https://matplotlib.org/
- sklearn: https://scikit-learn.org/stable/
- pandas: https://pandas.pydata.org/

**************** Thank You ***************

Import Libraries

In [6]:
```
import pandas as pd
import ydata_profiling as pp
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading the dataset

In [7]:
```
crop = pd.read_csv('Crop_recommendation.csv')
crop.head()
```

Out[7]:

	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

Asq Six Question to yourself

In [8]:
```
crop.shape
```

Out[8]:
```
(2200, 8)
```

In [9]:
```
crop.isnull().sum()
```

Out[9]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [11]:
```
crop.duplicated().sum()
```

Out[11]:
```
0
```

In [10]:
```
crop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

In [11]:
```
crop.describe()
```

Out[11]:

	N	P	K	temperature	humidity	ph	rainf
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.0000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.4636
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.9583
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.2112
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.5516
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.8676
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.2675
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.5601

Exploring Data

In [12]:
```
corr = crop.corr()
```

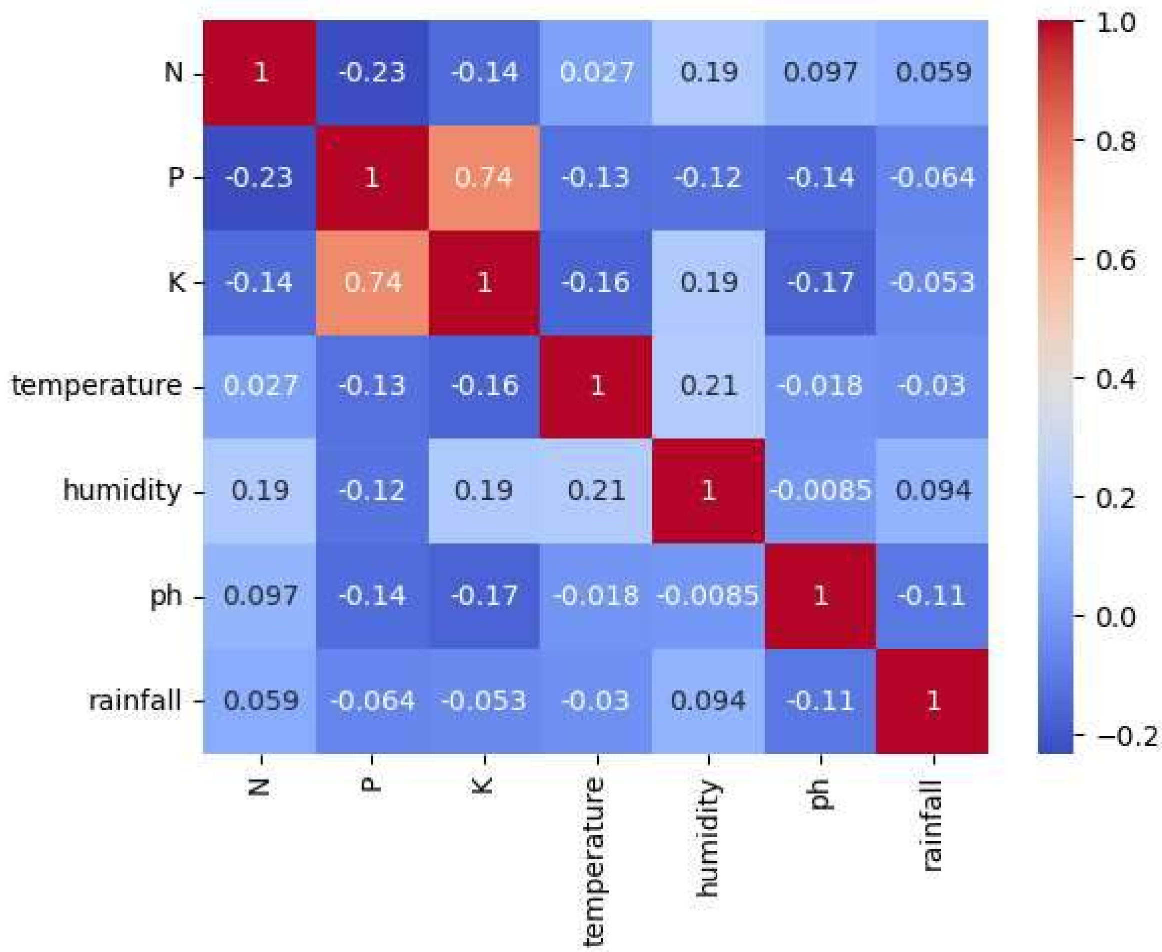
In [13]:
```
corr
```

Out[13]:

	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.231460	-0.140512	0.026504	0.190688	0.096683	0.059020
P	-0.231460	1.000000	0.736232	-0.127541	-0.118734	-0.138019	-0.063839
K	-0.140512	0.736232	1.000000	-0.160387	0.190859	-0.169503	-0.053461
temperature	0.026504	-0.127541	-0.160387	1.000000	0.205320	-0.017795	-0.030084
humidity	0.190688	-0.118734	0.190859	0.205320	1.000000	-0.008483	0.094423
ph	0.096683	-0.138019	-0.169503	-0.017795	-0.008483	1.000000	-0.109069
rainfall	0.059020	-0.063839	-0.053461	-0.030084	0.094423	-0.109069	1.000000

In [14]:
```
sns.heatmap(corr,annot=True,cbar=True, cmap='coolwarm', fmt='.2g')
```

Out[14]: <AxesSubplot:>



Encoding

In [15]:
```
crop['label'].value_counts()
```

Out[15]:
```
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
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
kidneybeans    100
chickpea       100
coffee         100
Name: label, dtype: int64
```

In [16]:
```
crop_dict = {
    'rice': 1,
    'maize': 2,
    'jute': 3,
    'cotton': 4,
    'coconut': 5,
    'papaya': 6,
    'orange': 7,
    'apple': 8,
    'muskmelon': 9,
    'watermelon': 10,
    'grapes': 11,
    'mango': 12,
    'banana': 13,
    'pomegranate': 14,
    'lentil': 15,
    'blackgram': 16,
    'mungbean': 17,
    'mothbeans': 18,
    'pigeonpeas': 19,
    'kidneybeans': 20,
    'chickpea': 21,
    'coffee': 22
}
crop['label_num'] = crop['label'].map(crop_dict)
```

In [17]:
```
crop.drop('label',axis=1,inplace=True)
```

In [18]:
```
crop.head()
```

Out[18]:

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

Train Test Split

In [19]:
```
# Split the dataset into features and labels
X = crop.iloc[:, :-1]
y = crop.iloc[:, -1]
```

In [20]:
```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
```

Scale the features using MinMaxScaler

In [21]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Standarization

In [22]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Training Models

In [23]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

# create instances of all models
models = {
    'Linear Discriminant Analysis': LinearDiscriminantAnalysis(),
    'Logistic Regression': LogisticRegression(),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Bagging': BaggingClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'Extra Trees': ExtraTreeClassifier(),
}

from sklearn.metrics import accuracy_score
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f'{name}:\nAccuracy: {acc:.4f}')

# Selecting decistion tree model:
rdf = RandomForestClassifier()
rdf.fit(X_train,y_train)
y_pred = rdf.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
Linear Discriminant Analysis:
Accuracy: 0.9515
Logistic Regression:
Accuracy: 0.9470
Naive Bayes:
Accuracy: 0.9939
```

```
C:\Users\2002m\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\2002m\anaconda3\lib\site-packages\sklearn\neighbors\_classification.
py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtos
is`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdim
s` will become False, the `axis` over which the statistic is taken will be el
iminated, and the value None will no longer be accepted. Set `keepdims` to Tr
ue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)

Support Vector Machine:
Accuracy: 0.9636
K-Nearest Neighbors:
Accuracy: 0.9773
Decision Tree:
Accuracy: 0.9848
Random Forest:
Accuracy: 0.9924
Bagging:
Accuracy: 0.9894
AdaBoost:
Accuracy: 0.2106
Gradient Boosting:
Accuracy: 0.9833
Extra Trees:
Accuracy: 0.9227
0.9924242424242424
```

Predictive System

In [25]:

```
# Define function to make predictions
def predict_crop(N, P, K, temperature, humidity, pH, rainfall):
    # Create a numpy array with the input values
    input_values = np.array([[N, P, K, temperature, humidity, pH, rainfall]])

    # Use the model to make a prediction
    prediction = rdf.predict(input_values)

    # Return the predicted crop label
    return prediction[0]
```

In [26]:
```
N = 21
P = 26
K = 27
tem = 27.003155
humidity = 47.675254
ph = 5.699587
rainfall = 95.851183

pred = predict_crop(N, P, K, tem, humidity, ph, rainfall)

if pred == 1:
    print("Rice is the best crop to be cultivated right there")
elif pred == 2:
    print("Maize is the best crop to be cultivated right there")
elif pred == 3:
    print("Jute is the best crop to be cultivated right there")
elif pred == 4:
    print("Cotton is the best crop to be cultivated right there")
elif pred == 5:
    print("Coconut is the best crop to be cultivated right there")
elif pred == 6:
    print("Papaya is the best crop to be cultivated right there")
elif pred == 7:
    print("Orange is the best crop to be cultivated right there")
elif pred == 8:
    print("Apple is the best crop to be cultivated right there")
elif pred == 9:
    print("Muskmelon is the best crop to be cultivated right there")
elif pred == 10:
    print("Watermelon is the best crop to be cultivated right there")
elif pred == 11:
    print("Grapes is the best crop to be cultivated right there")
elif pred == 12:
    print("Mango is the best crop to be cultivated right there")
elif pred == 13:
    print("Banana is the best crop to be cultivated right there")
elif pred == 14:
    print("Pomegranate is the best crop to be cultivated right there")
elif pred == 15:
    print("Lentil is the best crop to be cultivated right there")
elif pred == 16:
    print("Blackgram is the best crop to be cultivated right there")
elif pred == 17:
    print("Mungbean is the best crop to be cultivated right there")
elif pred == 18:
    print("Mothbeans is the best crop to be cultivated right there")
elif pred == 19:
    print("Pigeonpeas is the best crop to be cultivated right there")
elif pred == 20:
    print("Kidneybeans is the best crop to be cultivated right there")
elif pred == 21:
    print("Chickpea is the best crop to be cultivated right there")
elif pred == 22:
    print("Coffee is the best crop to be cultivated right there")
else:
```

```
    print("Sorry, we could not determine the best crop to be cultivated with t
```

```
Mango is the best crop to be cultivated right there

C:\Users\2002m\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but RandomForestClassifier was fitted wi
th feature names
  warnings.warn(
```

In [27]:

```
X_train
```

Out[27]:

	N	P	K	temperature	humidity	ph	rainfall
1102	21	26	27	27.003155	47.675254	5.699587	95.851183
1159	29	35	28	28.347161	53.539031	6.967418	90.402604
141	60	44	23	24.794708	70.045567	5.722580	76.728601
1004	80	77	49	26.054330	79.396545	5.519088	113.229737
2	60	55	44	23.004459	82.320763	7.840207	263.964248
...	...	...	...	...	...	...	...
1638	10	5	5	21.213070	91.353492	7.817846	112.983436
1095	108	94	47	27.359116	84.546250	6.387431	90.812505
1130	11	36	31	27.920633	51.779659	6.475449	100.258567
1294	11	124	204	13.429886	80.066340	6.361141	71.400430
860	32	78	22	23.970814	62.355576	7.007038	53.409060

1540 rows × 7 columns