ltural-product-optimization-engine

March 19, 2024

1 Achieve Precision farming by optimizing the agricultural production.

Build a predictive model so as to suggest the suitable Crop to based on the available climatic and soil conditions.

2 Importing Libraries

[1]:
```
# for manipulation
import numpy as np
import pandas as pd

# for data visualisation
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# for interactivity
from ipywidgets import interact
```

[2]:
```
# read dataset
data=pd.read_csv('Agri.csv')
```

[3]:
```
# check shape of dataset
print("Shape of the dataset:",data.shape)
```

```
Shape of the dataset: (2200, 8)
```

[4]:
```
# check head of dataset
data.head()
```

[4]:

	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

[5]:
```
# to check missing values
data.isnull().sum()
```

[5]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

[6]:
```
# Let's check crops present in the dataset
data['label'].value_counts()
```

[6]:
```
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
```

[10]:
```
# Let's check summary of all Crops
print("Average Ratio of Nitrogen in the Soil:(0:.2f)",format(data['N'].mean()))
print("Average Ratio of Phosphorus in the Soil:(0:.2f)",format(data['P'].
 ↪mean()))
print("Average Ratio of Potassium in the Soil:(0:.2f)",format(data['K'].mean()))
print("Average temperature in celsius:(0:.2f)",format(data['temperature'].
 ↪mean()))
```

```python
print("Average relative humidity in percentage:(0:.2f)",format(data['humidity'].
 ↪mean()))
print("Average PH value of the soil:(0:.2f)",format(data['ph'].mean()))
print("Average Rainfall in mm:(0:.2f)",format(data['rainfall'].mean()))
```

```
Average Ratio of Nitrogen in the Soil:(0:.2f) 50.551818181818184
Average Ratio of Phosphorus in the Soil:(0:.2f) 53.36272727272727
Average Ratio of Potassium in the Soil:(0:.2f) 48.14909090909091
Average temperature in celsius:(0:.2f) 25.616243851779544
Average relative humidity in percentage:(0:.2f) 71.48177921778637
Average PH value of the soil:(0:.2f) 6.469480065256364
Average Rainfall in mm:(0:.2f) 103.46365541576817
```

[13]:
```python
# Let's check sumary statastics for each crop

@interact

def summary(crops=list(data['label'].value_counts().index)):
    x=data[data['label']==crops]
    print("-----------------------------------------")
    print("Statistics for Nitrogen")
    print("Minimum Nitrogen required:",x['N'].min())
    print("Average Nitrogen required:",x['N'].mean())
    print("Maximum Nitrogen required:",x['N'].max())
    print("------------------------------------------")
    print("Statistics for Phosphorus")
    print("Minimum Phosphorus required:",x['P'].min())
    print("Average Phosphorus required:",x['P'].mean())
    print("Maximum Phosphorus required:",x['P'].max())
    print("------------------------------------------")
    print("Statistics for Potassium")
    print("Minimum Potassium required:",x['K'].min())
    print("Average Potassium required:",x['K'].mean())
    print("Maximum Potassium required:",x['K'].max())
    print("------------------------------------------")
    print("Statistics for temperature")
    print("Minimum temperature required:",x['temperature'].min())
    print("Average temperature required:",x['temperature'].mean())
    print("Maximum temperature required:",x['temperature'].max())
    print("------------------------------------------")
    print("Statistics for humidity")
    print("Minimum humidity required:",x['humidity'].min())
    print("Average humidity required:",x['humidity'].mean())
    print("Maximum humidity required:",x['humidity'].max())
    print("------------------------------------------")
    print("Statistics for PH value")
    print("Minimum PH value required:",x['ph'].min())
```

```
    print("Average PH value required:",x['ph'].mean())
    print("Maximum PH value required:",x['ph'].max())
    print("-----------------------------------------")
    print("Statistics for rainfall")
    print("Minimum rainfall required:",x['rainfall'].min())
    print("Average rainfall required:",x['rainfall'].mean())
    print("Maximum rainfall required:",x['rainfall'].max())
    print("-----------------------------------------")
```

```
interactive(children=(Dropdown(description='crops', options=('rice', 'maize',␣
↪'jute', 'cotton', 'coconut', 'pa…
```

[18]:

```
#Let's compare average requirement of each crop with average conditions

@interact

def compare(conditions =␣
↪['N','P','K','temperature','humidity','ph','rainfall',]):
    print("Average value for:",conditions,"is (0:0.2f)",format(data[conditions].
↪mean()))
    print("-------------------------------------------------")
    print("Rice: (0:0.2f)",format(data[(data['label']=='rice')][conditions].
↪mean()))
    print("Maize: (0:0.2f)",format(data[(data['label']=='maize')][conditions].
↪mean()))
    print("Jute: (0:0.2f)",format(data[(data['label']=='jute')][conditions].
↪mean()))
    print("Coconut: (0:0.
↪2f)",format(data[(data['label']=='coconut')][conditions].mean()))
    print("Papaya: (0:0.2f)",format(data[(data['label']=='papaya')][conditions].
↪mean()))
    print("Orange: (0:0.2f)",format(data[(data['label']=='orange')][conditions].
↪mean()))
    print("Apple: (0:0.2f)",format(data[(data['label']=='apple')][conditions].
↪mean()))
    print("Muskmelon: (0:0.
↪2f)",format(data[(data['label']=='muskmelon')][conditions].mean()))
    print("Watermelon: (0:0.
↪2f)",format(data[(data['label']=='watermelon')][conditions].mean()))
    print("grapes: (0:0.2f)",format(data[(data['label']=='grapes')][conditions].
↪mean()))
    print("Mango: (0:0.2f)",format(data[(data['label']=='mango')][conditions].
↪mean()))
    print("Banana: (0:0.2f)",format(data[(data['label']=='banana')][conditions].
↪mean()))
```

```
    print("Pomegranate: {0:0.
↪2f}".format(data[(data['label']=='pomegranate')][conditions].mean()))
    print("lentil: {0:0.2f}".format(data[(data['label']=='lentil')][conditions].
↪mean()))
    print("Blackgram: {0:0.
↪2f}".format(data[(data['label']=='blackgram')][conditions].mean()))
    print("Mungbean: {0:0.
↪2f}".format(data[(data['label']=='mungbean')][conditions].mean()))
    print("Mothbeans: {0:0.
↪2f}".format(data[(data['label']=='mothbeans')][conditions].mean()))
    print("Pigeonpeas: {0:0.
↪2f}".format(data[(data['label']=='pigeonpeas')][conditions].mean()))
    print("Kidneybeans: {0:0.
↪2f}".format(data[(data['label']=='kidneybeans')][conditions].mean()))
    print("Chickpea: {0:0.
↪2f}".format(data[(data['label']=='chickpea')][conditions].mean()))
    print("Coffee: {0:0.2f}".format(data[(data['label']=='coffee')][conditions].
↪mean()))
```

```
interactive(children=(Dropdown(description='conditions', options=('N', 'P', 'K',␣
↪'temperature', 'humidity', 'p…
```

[20]:

```
@interact
def compare(conditions␣
↪=['N','P','K','temperature','humidity','ph','rainfall',]):
    print("crops with require greater than average", conditions ,'\n')
    print(data[data[conditions] > data[conditions].mean()]['label'].unique())
    print("--------------------------------------------")
    print("crops with require less than average", conditions ,'\n')
    print(data[data[conditions] <= data[conditions].mean()]['label'].unique())
```

```
interactive(children=(Dropdown(description='conditions', options=('N', 'P', 'K',␣
↪'temperature', 'humidity', 'p…
```

3 Distribution

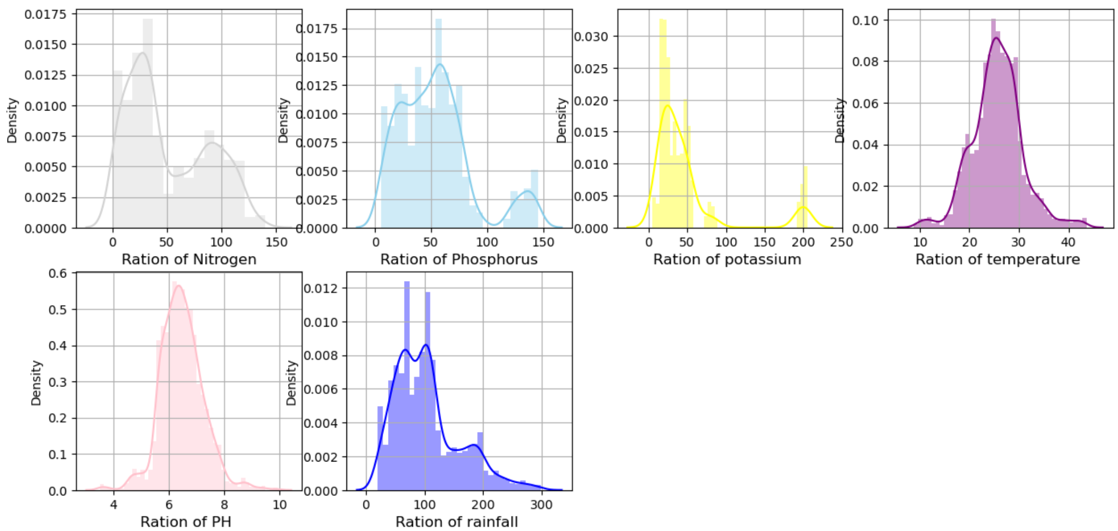
[25]:

```
# check the distribution of agricultural conditions
plt.rcParams['figure.figsize'] = (15,7)

plt.subplot(2, 4, 1)
sns.distplot(data['N'], color = 'lightgrey')
plt.xlabel('Ration of Nitrogen',fontsize = 12)
plt.grid()

plt.subplot(2, 4, 2)
sns.distplot(data['P'], color = 'skyblue')
```

```python
plt.xlabel('Ration of Phosphorus',fontsize = 12)
plt.grid()

plt.subplot(2, 4, 3)
sns.distplot(data['K'], color = 'yellow')
plt.xlabel('Ration of potassium ',fontsize = 12)
plt.grid()

plt.subplot(2, 4, 4)
sns.distplot(data['temperature'], color = 'purple')
plt.xlabel('Ration of temperature ',fontsize = 12)
plt.grid()

plt.subplot(2, 4, 5)
sns.distplot(data['ph'], color = 'pink')
plt.xlabel('Ration of PH ',fontsize = 12)
plt.grid()

plt.subplot(2, 4, 6)
sns.distplot(data['rainfall'], color = 'blue')
plt.xlabel('Ration of rainfall ',fontsize = 12)
plt.grid()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['N'], color = 'lightgrey')
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['P'], color = 'skyblue')
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:15: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['K'], color = 'yellow')
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:20: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['temperature'], color = 'purple')
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:25: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['ph'], color = 'pink')
C:\Users\Admin\AppData\Local\Temp\ipykernel_4444\2937747290.py:30: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['rainfall'], color = 'blue')
```



[29]:
```
#finding out some facts
print("some Interasting patterns")
print("---------------------------------------")
print("Crops which require very high ratio of Nitrogen content in the soil",data[data['N'] > 120]['label'].unique())
print("Crops which require very high ratio of Phosphorus content in the soil",data[data['P'] > 100]['label'].unique())
print("Crops which require very high ratio of Potassium content in the soil",data[data['K'] > 200]['label'].unique())
print("Crops which require very high rainfall",data[data['rainfall'] > 200]['label'].unique())
print("Crops which require very low temperature",data[data['temperature'] > 40]['label'].unique())
print("Crops which require very high temperature",data[data['temperature']>120]['label'].unique())
print("Crops which require very low humidity",data[data['humidity'] < 20]['label'].unique())
print("Crops which require very low PH",data[data['ph'] < 4]['label'].unique())
print("Crops which require very high PH",data[data['ph'] > 9]['label'].unique())
```

```
some Interasting patterns
---------------------------------------
Crops which require very high ratio of Nitrogen content in the soil ['cotton']
Crops which require very high ratio of Phosphorus content in the soil ['grapes'
'apple']
Crops which require very high ratio of Potassium content in the soil ['grapes'
'apple']
Crops which require very high rainfall ['rice' 'papaya' 'coconut']
Crops which require very low temperature ['grapes' 'papaya']
```

```
Crops which require very high temperature []
Crops which require very low humidity ['chickpea' 'kidneybeans']
Crops which require very low PH ['mothbeans']
Crops which require very high PH ['mothbeans']
```

[32]:

```python
# to check which crops will grown as per season
print("summer crops")
print(data[(data['temperature'] > 30) & (data['humidity'] > 50)]['label'].
 ↪unique())
print("winter crops")
print(data[(data['temperature'] < 20) & (data['humidity'] > 30)]['label'].
 ↪unique())
print("rainy crops")
print(data[(data['rainfall'] > 200) & (data['humidity'] > 30)]['label'].
 ↪unique())
```

```
summer crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
winter crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
rainy crops
['rice' 'papaya' 'coconut']
```

[39]:

```python
# try to cluster out the crops

from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')

x= data.loc[:,␣
 ↪['N','P','K','temperature','humidity','humidity','ph','rainfall']].values
print(x.shape)

x_data = pd.DataFrame(x)
x_data.head()
```

```
(2200, 8)
```

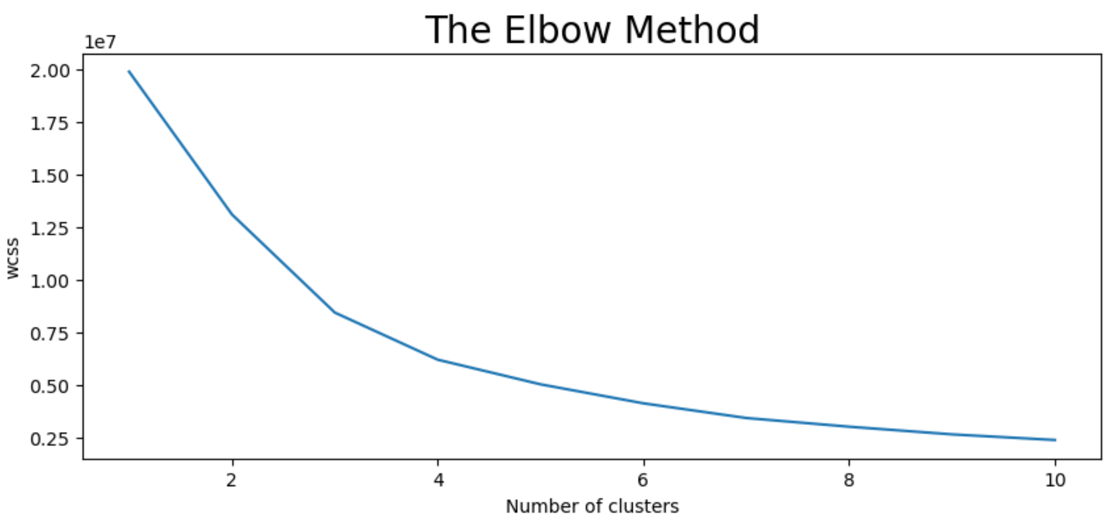
[39]:

	0	1	2	3	4	5	6	7
0	90.0	42.0	43.0	20.879744	82.002744	82.002744	6.502985	202.935536
1	85.0	58.0	41.0	21.770462	80.319644	80.319644	7.038096	226.655537
2	60.0	55.0	44.0	23.004459	82.320763	82.320763	7.840207	263.964248
3	74.0	35.0	40.0	26.491096	80.158363	80.158363	6.980401	242.864034
4	78.0	42.0	42.0	20.130175	81.604873	81.604873	7.628473	262.717340

[46]:

```python
# determine optimum number of clusters within datset
plt.rcParams['figure.figsize'] = (10,4)
```

```python
wcss=[]
for i in range(1,11):
    kn = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init =␣
 ↪10, random_state = 0)
    kn.fit(x)
    wcss.append(kn.inertia_)

# plot the results
plt.plot(range(1,11), wcss)
plt.title("The Elbow Method", fontsize = 20)
plt.xlabel("Number of clusters")
plt.ylabel("wcss")
plt.show()
```



[52]:

```python
# implement K-means clustering
kn = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10,␣
 ↪random_state = 0)
y_means = kn.fit_predict(x)

a = data['label']
y_means = pd.DataFrame(y_means)
z = pd.concat([y_means, a], axis = 1)
z = z.rename(columns = {0: 'cluster'})

# check the cluster of each crop
print("Check the result of clustering \n")
print("Crops in First Cluster", z[z['cluster'] == 0]['label'].unique())
print("---------------------------------------------------")
```

```
print("Crops in second Cluster", z[z['cluster'] == 1]['label'].unique())
print("---------------------------------------------")
print("Crops in third Cluster", z[z['cluster'] == 2]['label'].unique())
print("---------------------------------------------")
print("Crops in Fourth Cluster", z[z['cluster'] == 3]['label'].unique())
```

```
Check the result of clustering

Crops in First Cluster ['pomegranate' 'orange' 'papaya' 'coconut']
---------------------------------------------
Crops in second Cluster ['watermelon' 'muskmelon' 'papaya']
---------------------------------------------
Crops in third Cluster ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute']
---------------------------------------------
Crops in Fourth Cluster ['grapes' 'apple']
```

[53]:

```
# split data set for predictive modelling
y = data ['label']
x = data.drop(['label'], axis = 1)

print("Shape of X:", x.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (2200, 7)
Shape of y: (2200,)
```

[56]:

```
# create training and testing sets for validation of results
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 = 0)

print("The shape of X-train",x_train.shape)
print("The shape of X-test",x_test.shape)
print("The shape of y_train",y_train.shape)
print("The shape of y_test",y_test.shape)
```

```
The shape of X-train (1760, 7)
The shape of X-test (440, 7)
The shape of y_train (1760,)
The shape of y_test (440,)
```

[57]:

```
#create predictive model
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train,y_train)
```

```python
y_pred = model.predict(x_test)
```

[58]:
```python
# evaluate the model performance
from sklearn.metrics import classification_report

#print the classification report data
cr = classification_report(y_test, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        18
      banana       1.00      1.00      1.00        18
   blackgram       0.86      0.82      0.84        22
    chickpea       1.00      1.00      1.00        23
     coconut       1.00      1.00      1.00        15
      coffee       1.00      1.00      1.00        17
      cotton       0.89      1.00      0.94        16
      grapes       1.00      1.00      1.00        18
        jute       0.84      1.00      0.91        21
 kidneybeans       1.00      1.00      1.00        20
      lentil       0.94      0.94      0.94        17
       maize       0.94      0.89      0.91        18
       mango       1.00      1.00      1.00        21
   mothbeans       0.88      0.92      0.90        25
    mungbean       1.00      1.00      1.00        17
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        23
      papaya       1.00      0.95      0.98        21
  pigeonpeas       1.00      1.00      1.00        22
 pomegranate       1.00      1.00      1.00        23
        rice       1.00      0.84      0.91        25
  watermelon       1.00      1.00      1.00        17

    accuracy                           0.97       440
   macro avg       0.97      0.97      0.97       440
weighted avg       0.97      0.97      0.97       440
```

[59]:
```python
# to check dataset
data.head()
```

[59]:

	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
```

[63]:
```
prediction = model.predict((np.array([[90,
                                       40,
                                       40,
                                       20,
                                       82,
                                       6.5,
                                       202]])))
print("The suggested crop for given climatic conditions is:",prediction)
```

```
The suggested crop for given climatic conditions is: ['rice']
```

[75]:
```
# check the model for banana
data[data['label'] == 'banana'].head()
```

[75]:

	N	P	K	temperature	humidity	ph	rainfall	label
1000	91	94	46	29.367924	76.249001	6.149934	92.828409	banana
1001	105	95	50	27.333690	83.676752	5.849076	101.049479	banana
1002	108	92	53	27.400536	82.962213	6.276800	104.937800	banana
1003	86	76	54	29.315908	80.115857	5.926825	90.109781	banana
1004	80	77	49	26.054330	79.396545	5.519088	113.229737	banana

[76]:
```
prediction = model.predict((np.array([[91,
                                       94,
                                       46,
                                       28.36,
                                       76.24,
                                       6.14,
                                       92.82]])))
print("The suggested crop for given climatic conditions is:", prediction)
```

```
The suggested crop for given climatic conditions is: ['banana']
```

[ ]: