Sowing Success: How Machine Learning Helps Farmers Select the Best Crops

Measuring essential soil metrics such as nitrogen, phosphorous, potassium levels, and pH value is an important aspect of assessing soil condition. However, it can be an expensive and time-consuming process, which can cause farmers to prioritize which metrics to measure based on their budget constraints.

Farmers have various options when it comes to deciding which crop to plant each season. Their primary objective is to maximize the yield of their crops, taking into account different factors. One crucial factor that affects crop growth is the condition of the soil in the field, which can be assessed by measuring basic elements such as nitrogen and potassium levels. Each crop has an ideal soil condition that ensures optimal growth and maximum yield.

A farmer reached out to you as a machine learning expert for assistance in selecting the best crop for his field. They've provided you with a dataset called `soil_measures.csv`, which contains:

- `"N"`: Nitrogen content ratio in the soil
- `"P"`: Phosphorous content ratio in the soil
- `"K"`: Potassium content ratio in the soil
- `"pH"` value of the soil
- `"crop"`: categorical values that contain various crops (target variable).

Each row in this dataset represents various measures of the soil in a particular field. Based on these measurements, the crop specified in the `"crop"` column is the optimal choice for that field.

In this project, you will build multi-class classification models to predict the type of `"crop"` and identify the single most importance feature for predictive performance.

```python
# All required libraries are imported here for you.
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Load the dataset
crops = pd.read_csv("soil_measures.csv")

# Write your code here
crops.head()
```

index	N	P	K	ph
0	90	42	43	6.502985292
1	85	58	41	7.038096361
2	60	55	44	7.840207144
3	74	35	40	6.980400905
4	78	42	42	7.628472891

Rows: 5 Expand

```
crops.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 5 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   ph      2200 non-null   float64
 4   crop    2200 non-null   object 
dtypes: float64(1), int64(3), object(1)
memory usage: 86.1+ KB
```

```
crops.describe()
```

index ••• ⇅	N ••• ⇅	P ••• ⇅	K ••• ⇅	ph
count	2200	2200	2200	
mean	50.5518181818	53.3627272727	48.1490909091	
std	36.9173338338	32.9858827386	50.6479305467	
min	0	5	5	
25%	21	28	20	
50%	37	51	32	
75%	84.25	68	49	
max	140	145	205	

Rows: 8 ⤢ Expand

crops.isna.sum

```
# Check for missing values in the crops DataFrame
crops.isna().sum()
crops["crop"].value_counts()
```

ind... ••• ⇅	••• ⇅
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

Rows: 22 ⤢ Expand

```
#splitting the data
y= crops['crop']
```

```
# Create a dictionary to store the model performance for each feature
feature_performance = {}
```

```
# Fix: Use the correct column names as present in X_train/X_test
print("X_train columns:", X_train.columns)  # Debug: See actual column names

# Map the feature names to the actual column names in X_train/X_test
feature_map = {
    "N": "N",    # update if actual column name is different
    "P": "P",    # update if actual column name is different
    "K": "K",    # update if actual column name is different
    "ph": "ph"   # update if actual column name is different
}

feature_performance = {}

for feature in ["N", "P", "K", "ph"]:
    col_name = feature_map[feature]
    if col_name not in X_train.columns:
        print(f"Column '{col_name}' not found in X_train. Skipping.")
        continue
    log_reg = LogisticRegression(multi_class='multinomial')
    log_reg.fit(X_train[[col_name]], y_train)
    y_pred = log_reg.predict(X_test[[col_name]])
    f1 = metrics.f1_score(y_test, y_pred, average='weighted')
    feature_performance[feature] = f1
    print(f"F1.score for {feature}: {f1}")
```

```
X_train columns: Index(['N', 'P', 'K', 'ph'], dtype='object')
F1.score for N: 0.09149868209906838
F1.score for P: 0.14761942909728204
F1.score for K: 0.23896974566001802
F1.score for ph: 0.04532731061152114
```

```
best_predictive_feature = {"K":feature_performance['K']}
best_predictive_feature
```

```
{'K': 0.23896974566001802}
```

```
# getting dummies only for crop column
crops_dummy = pd.get_dummies(crops['crop'],drop_first=True)
crops_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   banana       2200 non-null   uint8
 1   blackgram    2200 non-null   uint8
 2   chickpea     2200 non-null   uint8
 3   coconut      2200 non-null   uint8
 4   coffee       2200 non-null   uint8
 5   cotton       2200 non-null   uint8
 6   grapes       2200 non-null   uint8
 7   jute         2200 non-null   uint8
 8   kidneybeans  2200 non-null   uint8
 9   lentil       2200 non-null   uint8
 10  maize        2200 non-null   uint8
 11  mango        2200 non-null   uint8
 12  mothbeans    2200 non-null   uint8
 13  mungbean     2200 non-null   uint8
 14  muskmelon    2200 non-null   uint8
 15  orange       2200 non-null   uint8
 16  papaya       2200 non-null   uint8
```

```
#crops_dummy = pd.concat([crops, crops_dummy], axis=1)
#crops_dummy = crops_dummy.drop("crop", axis=1)
#crops_dummy

crops_dummy = pd.concat([crops,crops_dummy],axis =1)
crops_dummy = crops_dummy.drop('crop',axis =1)
crops_dummy
```

… ↑↓	… ↑↓	… ↑↓	… ↑↓	ph … ↑↓	… ↑↓	b. … ↑↓	… ↑↓	… ↑↓	… ↑↓	… ↑↓	… ↑↓	… ↑↓	kid..
0	90	42	43	6.502985292	0	0	0	0	0	0	0	0	
1	85	58	41	7.038096361	0	0	0	0	0	0	0	0	
2	60	55	44	7.840207144	0	0	0	0	0	0	0	0	
3	74	35	40	6.980400905	0	0	0	0	0	0	0	0	
4	78	42	42	7.628472891	0	0	0	0	0	0	0	0	
5	69	37	42	7.073453503	0	0	0	0	0	0	0	0	
6	69	55	38	5.70080568	0	0	0	0	0	0	0	0	
7	94	53	40	5.718627178	0	0	0	0	0	0	0	0	
8	89	54	38	6.685346424	0	0	0	0	0	0	0	0	
9	68	58	38	6.336253525	0	0	0	0	0	0	0	0	
10	91	53	40	5.386167788	0	0	0	0	0	0	0	0	
11	90	46	42	7.50283396	0	0	0	0	0	0	0	0	
12	78	58	44	5.108681786	0	0	0	0	0	0	0	0	
13	93	56	36	6.98435366	0	0	0	0	0	0	0	0	
14	94	50	37	6.94801983	0	0	0	0	0	0	0	0	
15	60	48	39	7.042299069	0	0	0	0	0	0	0	0	

Rows: 2,200 ⤢ Expand

```
final_feature =['N','K','ph']
X_train, X_test, y_train, y_test = train_test_split(
    crops[final_features],
    crops["crop"],
    test_size=0.2,
    random_state=42
)
log_reg = LogisticRegression(max_iter=2000,multi_class='multinomial')
log_reg.fit(X_train,y_train)
y_pred = log_reg.predict(X_test)
f_error = f1_score(y_test,y_pred,average='weighted')
model_performance = f1_score(y_test, y_pred, average="weighted")
print(model_performance)
```

```
0.558010495235685
```