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.

```
# All required libraries are imported here for you.
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import plotly.express as px
from plotly.subplots import make_subplots

# Load the dataset
crops = pd.read_csv("soil_measures.csv")
crops.head()
# Write your code here
```

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

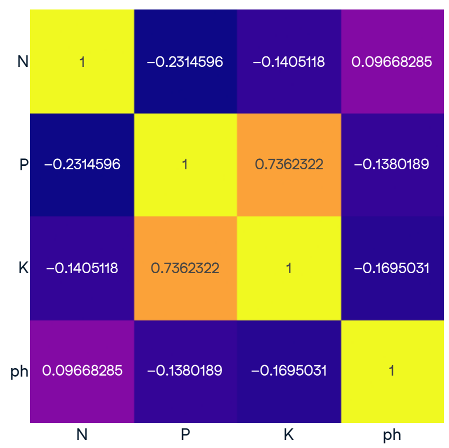
Rows: 5 Expand

```
crops.describe(include="all")
```

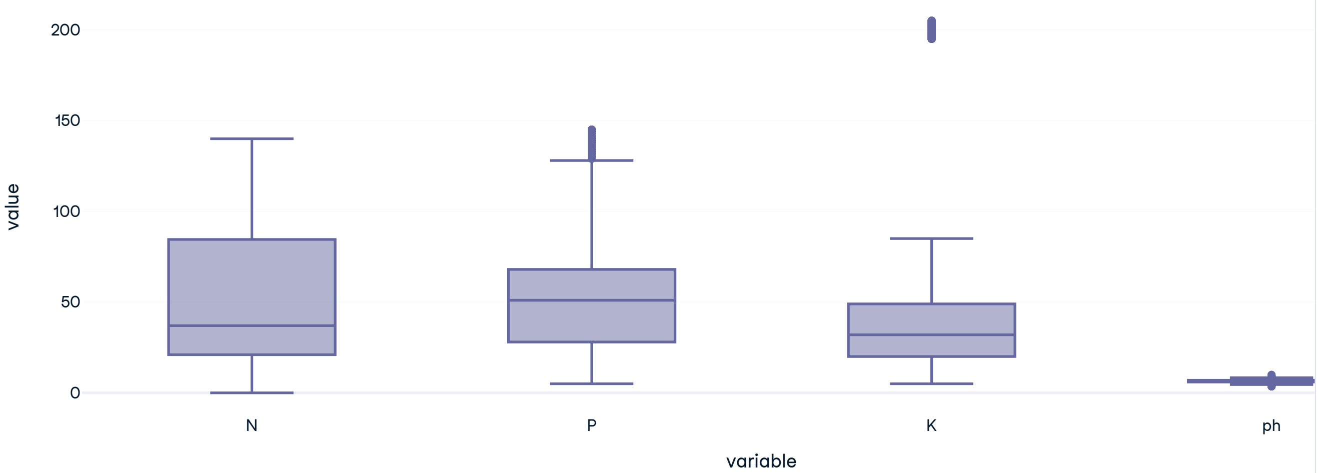
index	N	P	K	ph
count	2200	2200	2200	
unique				
top				
freq				
mean	50.5518181818	53.3627272727	48.1490909091	6.46948
std	36.9173338338	32.9858827386	50.6479305467	0.7739
min	0	5	5	3.5047
25%	21	28	20	5.97169
50%	37	51	32	6.425
75%	84.25	68	49	6.92364
max	140	145	205	9.935

Rows: 11 Expand

```
crops_numi = crops.drop("crop", axis = 1 )
correTIX= crops_numi.corr("pearson")
px.imshow(correTIX , text_auto=True)
```



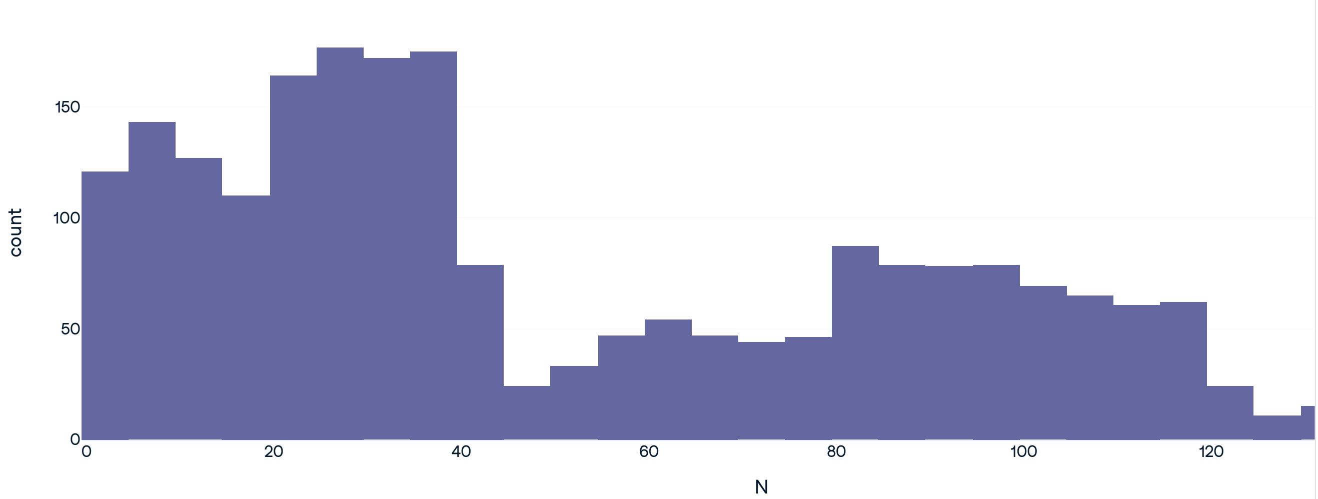

```
px.box(crops, hover_name="crop")
```

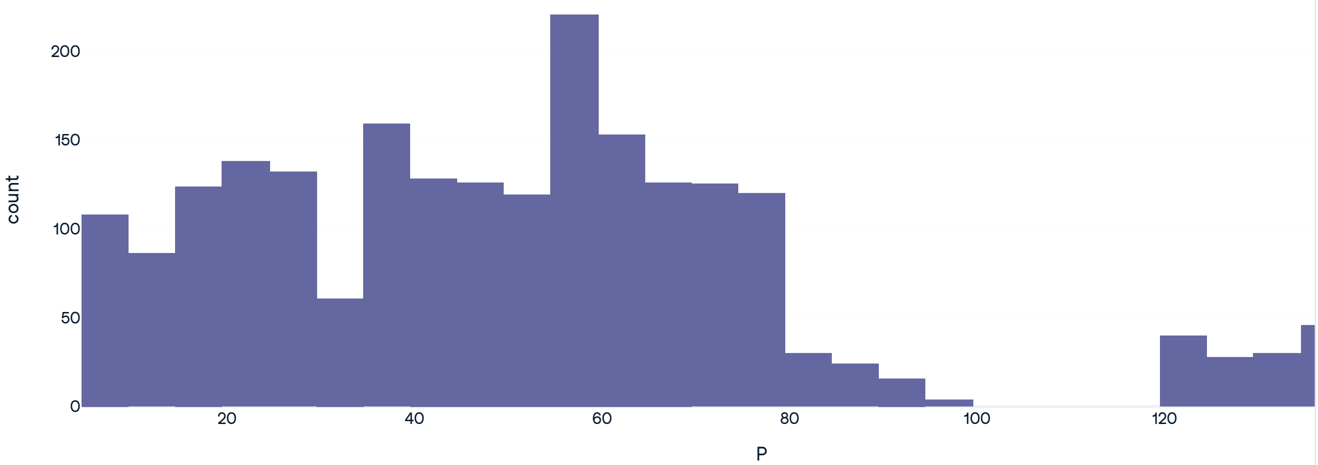


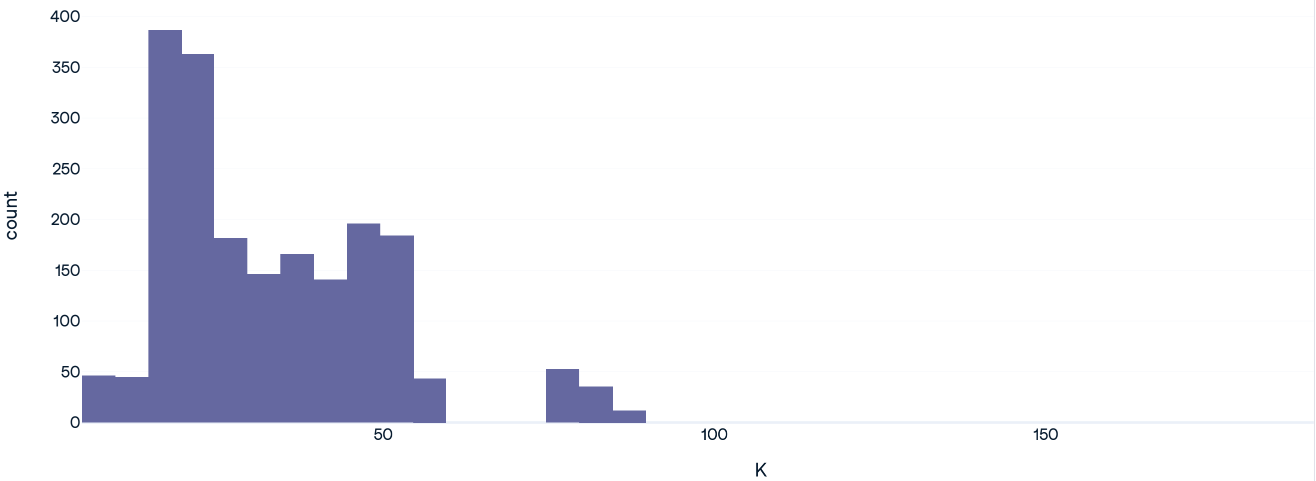
appel and grasp

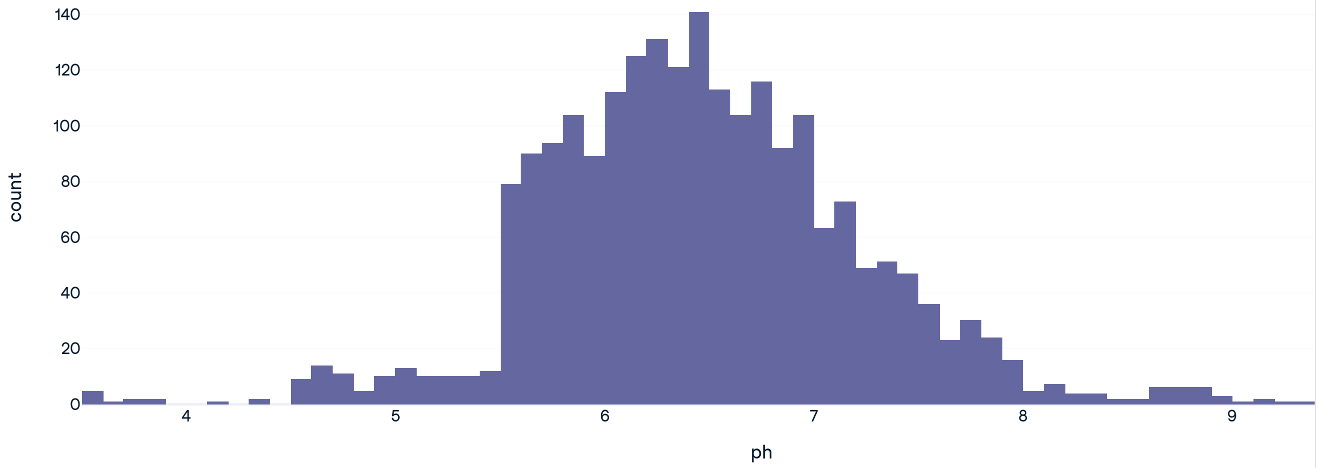
in K & P high oultiers cause its a need for poth apple and grasp thus we will capp the outliers instead of dropping.

```
Xs = ["N","P", "K" , "ph"]
for i in Xs :
    fig = px.histogram(crops_numi , x=i )
    fig.show()
```









Features engineering :

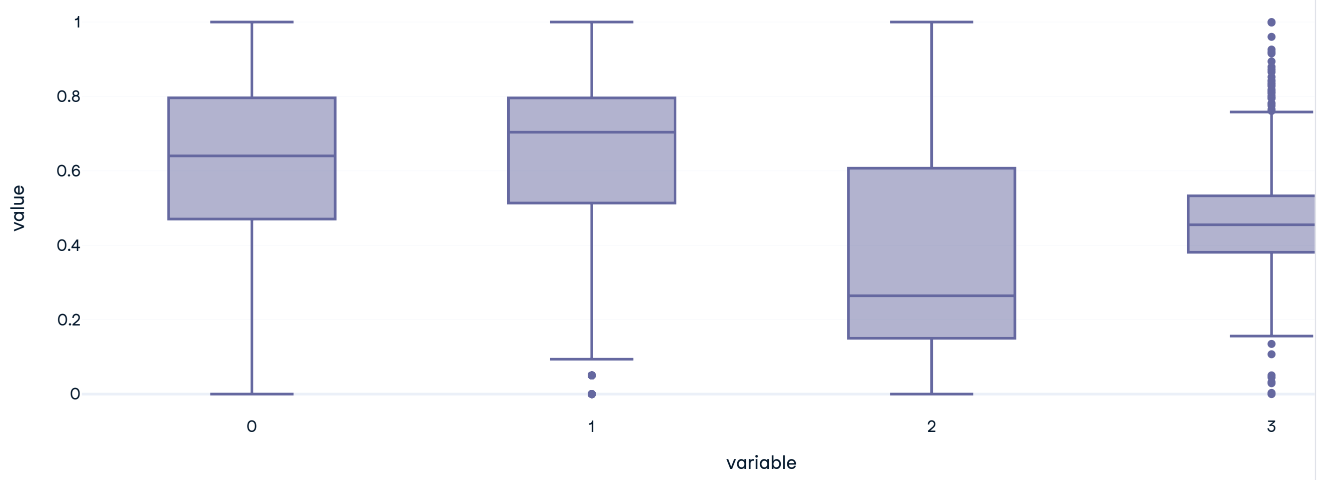
- N : normalization
- P : capping + logtransform + normalization
- K : capping + logtransform + normalization
- ph : normalization

In a pipeline :)

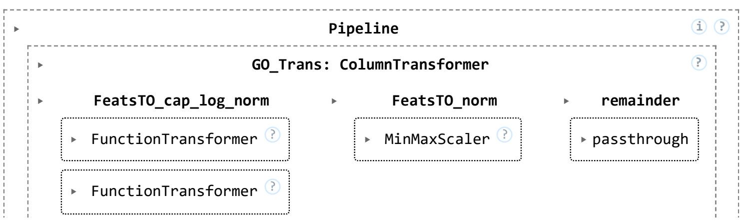
```
#splitting data
from sklearn.model_selection import train_test_split
X= crops_numi
y= crops["crop"]
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size= 0.2 , shuffle= True ,random_state= 42 , stratify= y )
```

```python
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

FeatsTO_cap_log_norm = ["K","P"]
FeatsTO_norm = ["N" , "ph"]

FeatsTO_cap_log_norm = Pipeline(
    steps= [
        ("clipping" , FunctionTransformer(lambda col : col.clip(upper = {"K" : 85  , "P" : 128 })) ) ,
        ("logging" , FunctionTransformer(np.log1p) ),
        ("normalizing" , MinMaxScaler(clip=True) )
        ])

FeatsTO_norm = Pipeline( steps=[
    ("normalizing" , MinMaxScaler(clip=True))
])

GO_Trans = ColumnTransformer(transformers=[
    ("FeatsTO_cap_log_norm" , FeatsTO_cap_log_norm, ["K","P"]),
    ("FeatsTO_norm" , FeatsTO_norm , ["N" , "ph"])
    ], remainder="passthrough"
)

GO_Pipe = Pipeline(steps=[
    ("GO_Trans" , GO_Trans),

    ("classifier", LogisticRegression())
])
```

```python
trans_data = GO_Trans.fit_transform(X_train)
px.box(trans_data)
```




```python
GO_Pipe.fit(X_train , y_train)
```



```
GO_Pipe.score(X_test,y_test)
```

```
0.6409090909090909
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV


GO2_Pipe = Pipeline(steps=[
    ("GO_Trans" , GO_Trans),

    ("classifier", KNeighborsClassifier())
])


param_grid = {
    'classifier__n_neighbors': [3, 5, 7, 9, 11, 13],
    'classifier__weights': ['uniform', 'distance'],
    'classifier__metric': ['euclidean', 'manhattan']
}

Grid_GO= GridSearchCV(GO2_Pipe , param_grid , scoring='accuracy' , cv=5 ,verbose=1)


Grid_GO.fit(X_train , y_train)
Grid_GO.score(X_test , y_test)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

```
0.740909090909091
```

```
print(f"best paramaters : {Grid_GO.best_params_}")
print(f"best score : {Grid_GO.best_score_}")
```

```
best paramaters : {'classifier__metric': 'euclidean', 'classifier__n_neighbors': 7, 'classifier__weights': 'distance'}
best score : 0.7698863636363636
```

RandomForest

```
from sklearn.ensemble import RandomForestClassifier

GO3_Pipe = Pipeline(steps=[
    ("GO_Trans" , GO_Trans),

    ("classifier", RandomForestClassifier())
])


param_grid_rf = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [10, 20, None],
    'classifier__min_samples_leaf': [1, 2, 4],
    'classifier__min_samples_split': [2, 5]
}

Grid2_GO = GridSearchCV( GO3_Pipe , param_grid_rf , cv = 5 , scoring = 'accuracy' , verbose=1)

Grid2_GO.fit(X_train , y_train)
Grid2_GO.score(X_test , y_test)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
```

```
0.7909090909090909
```

تحليل أداء نماذج التصنيف للتنبؤ بنوع المحصول

1. ملخص المشروع

تم اختبار ثلاثة نماذج رئيسية مع تطبيق تقنيات المعالجة المسبقة وضبط المعاملات. يهدف هذا التحليل إلى تقييم ومقارنة أداء نماذج تعلم آلي مختلفة في مهمة التنبؤ بنوع المحصول بناءً على خصائص التربة (N, P, K, ph) للوصول إلى أفضل دقة ممكنة.

2. المنهجية

تم اتباع عملية منهجية موحدة لضمان المقارنة العادلة:

- **المعالجة المسبقة:** تم بناء `Pipeline` متكامل باستخدام `ColumnTransformer` لمعالجة الميزات العددية وتطبيق عمليات التحجيم (`Scaling`) والتحويل (`Transformation`).
- **النماذج المختبرة:**
  1. Logistic Regression
  2. K-Nearest Neighbors (KNN)
  3. Random Forest Classifier
- **التحسين:** تم استخدام أداة `GridSearchCV` للبحث المنهجي عن أفضل توليفة من المعاملات الفائقة (Hyperparameters) للنماذج المتقدمة.

3. مقارنة النتائج النهائية

النموذج (Model)	الدقة النهائية (Final Accuracy)	أبرز الاستنتاجات
1. Logistic Regression	**~64.0%**	**خط أساس أولي.** أداؤه المنخفض يثبت أن الحدود الفاصلة بين الأصناف ليست خطية على الإطلاق
2. KNN (بعد الضبط)	**~74.1%**	**تحسن ملحوظ.** يثبت أن النماذج غير الخطية أنسب، ولكنه لا يزال محدودًا بسبب حساسيته للمسافات
3. Random Forest (بعد الضبط)	🏆 **~79.1%**	**الأداء الأمثل.** يوضح أن النماذج الشجرية هي الأنسب للتعامل مع تعقيد وتفاعل الميزات في البيانات

4. الاستنتاجات حول طبيعة البيانات

كشفت عملية النمذجة عن رؤى عميقة حول طبيعة البيانات:

- **العلاقات غير خطية:** الفجوة الكبيرة في الأداء بين `LogisticRegression` والنموذجين الآخرين هي أقوى دليل على أن العلاقة بين خصائص التربة ونوع المحصول معقدة وغير خطية.
- **تفاعل الميزات:** تفوق `Random Forest` وهو ما تبرع النماذج الشجرية في التقاطه (e.g., تأثير `K` يعتمد على قيمة `ph`) بشكل خاص يشير إلى وجود تفاعل بين الميزات.
- **وجود نمط قوي:** الوصول إلى دقة تقترب من 80% يثبت أن الميزات تحتوي على إشارة قوية ونمط واضح يمكن للنموذج الصحيح أن يتعلمه ويتنبأ به بدقة عالية.

5. الخلاصة

بعد ضبط معاملاته أنه الحل الأمثل لهذه المشكلة، محققًا دقة تصل إلى **79.1%**. أثبت نموذج **Random Forest Classifier** الرحلة من نموذج خطي بسيط إلى نموذج شجري متقدم لم تكن فقط بحثًا عن دقة أعلى، بل كانت عملية استكشافية كشفت عن الطبيعة المعقدة وغير الخطية للبيانات، وهو ما لم يكن ممكنًا اكتشافه بالاعتماد فقط على أدوات التحليل الأولي مثل مصفوفة الارتباط.