```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("atharvaingle/crop-recommendation-dataset")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/atharvaingle/crop-recommendation-dataset?dataset_version_number=1...
100%|██████████| 63.7k/63.7k [00:00<00:00, 338kB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/atharvaingle/crop-recommendation-dataset/versions/1
```

```
import pandas as pd
df = pd.read_csv(path + "/Crop_recommendation.csv")
df.head()
```

	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

```
X = df.drop('label', axis=1) # Features
y = df['label'] # Labels
```

```
y = pd.get_dummies(y)#instead of beinga word output, out model will output a true or false for each type of fruit
y.head()
```

	apple	banana	blackgram	chickpea	coconut	coffee	cotton	grapes	jute	kidneybeans	...	mango	mothbeans	mungbean	muskmelon	or
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	F

5 rows × 22 columns

```
#scales all numerical values from 0 - 1 to make it easier for model to understand data
from sklearn.preprocessing import MinMaxScaler
import joblib
scaler = MinMaxScaler()
scaler.fit(X)

X = pd.DataFrame(scaler.transform(X), columns=X.columns)
joblib.dump(scaler, "/content/drive/MyDrive/Research/TSAFOLDER/saved_scaler.pkl")

X.head()
```

	N	P	K	temperature	humidity	ph	rainfall
0	0.642857	0.264286	0.190	0.345886	0.790267	0.466264	0.656458
1	0.607143	0.378571	0.180	0.371445	0.770633	0.549480	0.741675
2	0.428571	0.357143	0.195	0.406854	0.793977	0.674219	0.875710
3	0.528571	0.214286	0.175	0.506901	0.768751	0.540508	0.799905
4	0.557143	0.264286	0.185	0.324378	0.785626	0.641291	0.871231

```
#required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
```

```
#split out model into training, testing, and validation
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Split the training set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42)


X_train.shape
```

```
(1320, 7)
```

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.metrics import Precision, Recall

input_dim = 7
# Define the model
model = models.Sequential([
    layers.Input(shape=(input_dim,)),  # Replace 'input_dim' with the number of input features
    layers.Dense(128, activation='relu'),  # First hidden layer
    layers.Dropout(0.2),  # Dropout to reduce overfitting
    layers.Dense(64, activation='relu'),  # Second hidden layer
    layers.Dropout(0.2),
    layers.Dense(32, activation='relu'),  # Third hidden layer
    layers.Dense(22, activation='softmax')  # Output layer (regression task)
])

# Compile the model
model.compile(optimizer=optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy', Precision(name='precision'), Recall(name='recall')])

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 128)	1,024
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 32)	2,080
dense_7 (Dense)	(None, 22)	726

Total params: 12,086 (47.21 KB)
Trainable params: 12,086 (47.21 KB)

```
# Train the model
history = model.fit(
    X_train, y_train,  # Replace with your training data
    validation_data=(X_val, y_val),  # Replace with your validation data
    epochs=50,  # Adjust based on performance
    batch_size=16,  # Suitable for large datasets
    callbacks=[tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)]
)

# Evaluate the model on test data
test_loss, test_mae = model.evaluate(X_test, y_test)  # Replace with your test data
print(f"Test Loss: {test_loss}, Test accuracy: {test_mae}")



#gives accuracy of model
test_accracy = model.evaluate(X_test, y_test)
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9533 - loss: 0.1435 - precision: 0.9542 - recall: 0.9501
```

```
model.save("/content/drive/MyDrive/Research/TSAFOLDER/agriculture-model")
```

```
df.head()
```

	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

```
#required libraries
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import tensorflow as tf
#array of possible crops
crops = [
    "apple", "banana", "blackgram", "chickpea", "coconut", "coffee", "cotton",
    "grapes", "jute", "kidneybeans", "lentil", "maize", "mango", "mothbeans",
    "mungbean", "muskmelon", "orange", "papaya", "pigeonpeas", "pomegranate",
    "rice", "watermelon"
]
#scaler to change values from 0 -1
loaded_scaler = joblib.load("/content/drive/MyDrive/Research/TSAFOLDER/saved_scaler.pkl")
#load model from paht, you will need to change this
model = tf.keras.models.load_model("/content/drive/MyDrive/Research/TSAFOLDER/agriculture-model.h5")
#array of values in the order of nitrogen, phospherus, potassium, temp, humidity, ph, and rainfall
values = [74,54,35.4,20.87974371,82.00274423,7,202.9355362]
values = pd.DataFrame([values],
                  columns=['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall'])
values = pd.DataFrame(loaded_scaler.transform(X), columns=X.columns)

values = np.array(values)
values = values.reshape(1,7)
print(values)
output = model.predict(values)
output = np.argmax(output)
print(output)
crops[output]
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
[[0.64285714 0.26428571 0.19       0.34588614 0.79026683 0.46626364
  0.65645778]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 230ms/step
20
'rice'
```

```
X_test
```

	N	P	K	temperature	humidity	ph	rainfall
1451	0.721429	0.085714	0.210	0.593069	0.938733	0.416821	0.021904
1334	0.700000	0.021429	0.230	0.497956	0.842992	0.428373	0.104973
1761	0.421429	0.407143	0.220	0.990962	0.922659	0.534458	0.339742
1735	0.314286	0.392857	0.250	0.730414	0.890039	0.516399	0.281407
1576	0.214286	0.942857	0.975	0.404267	0.891779	0.326369	0.353489
...	...	...	...	...	...	...	...
59	0.707143	0.357143	0.150	0.370107	0.769692	0.466063	0.926001
71	0.478571	0.285714	0.165	0.398918	0.792226	0.590274	0.864657
1908	0.864286	0.300000	0.055	0.424105	0.758689	0.656029	0.187846
1958	0.828571	0.335714	0.070	0.405084	0.712913	0.405853	0.168382
482	0.035714	0.450000	0.075	0.293205	0.219879	0.406964	0.485575

440 rows × 7 columns

Double-click (or enter) to edit

```
#scales all numerical values from 0 - 1 to make it easier for model to understand data
from sklearn.preprocessing import MinMaxScaler
import joblib
scaler = MinMaxScaler()
scaler.fit(X)

X = pd.DataFrame(scaler.transform(X), columns=X.columns)
joblib.dump(scaler, "/content/drive/MyDrive/Research/TSAFOLDER/saved_scaler.pkl")

X.head()


loaded_scaler = joblib.load("/content/drive/MyDrive/Research/TSAFOLDER/saved_scaler.pkl")


values = [90,42,43,20.87974371,82.00274423,6.502985292000001,202.9355362]
values = pd.DataFrame([values],
                  columns=['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall'])
values = pd.DataFrame(loaded_scaler.transform(X), columns=X.columns)

values = np.array(values)
values = values.reshape(1,7)
values = loaded_scaler.transform(values)
print(values)
```

```
[[0.64285714 0.26428571 0.19       0.34588614 0.79026683 0.46626364
  0.65645778]]
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but MinMaxSc
  warnings.warn(
```

```
X = pd.DataFrame(loaded_scaler.transform(X), columns=X.columns)
X
```

	N	P	K	temperature	humidity	ph	rainfall
0	0.642857	0.264286	0.19	0.345886	0.790267	0.466264	0.656458