```
!pip install scikit-learn
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.inspection import permutation_importance
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix
```

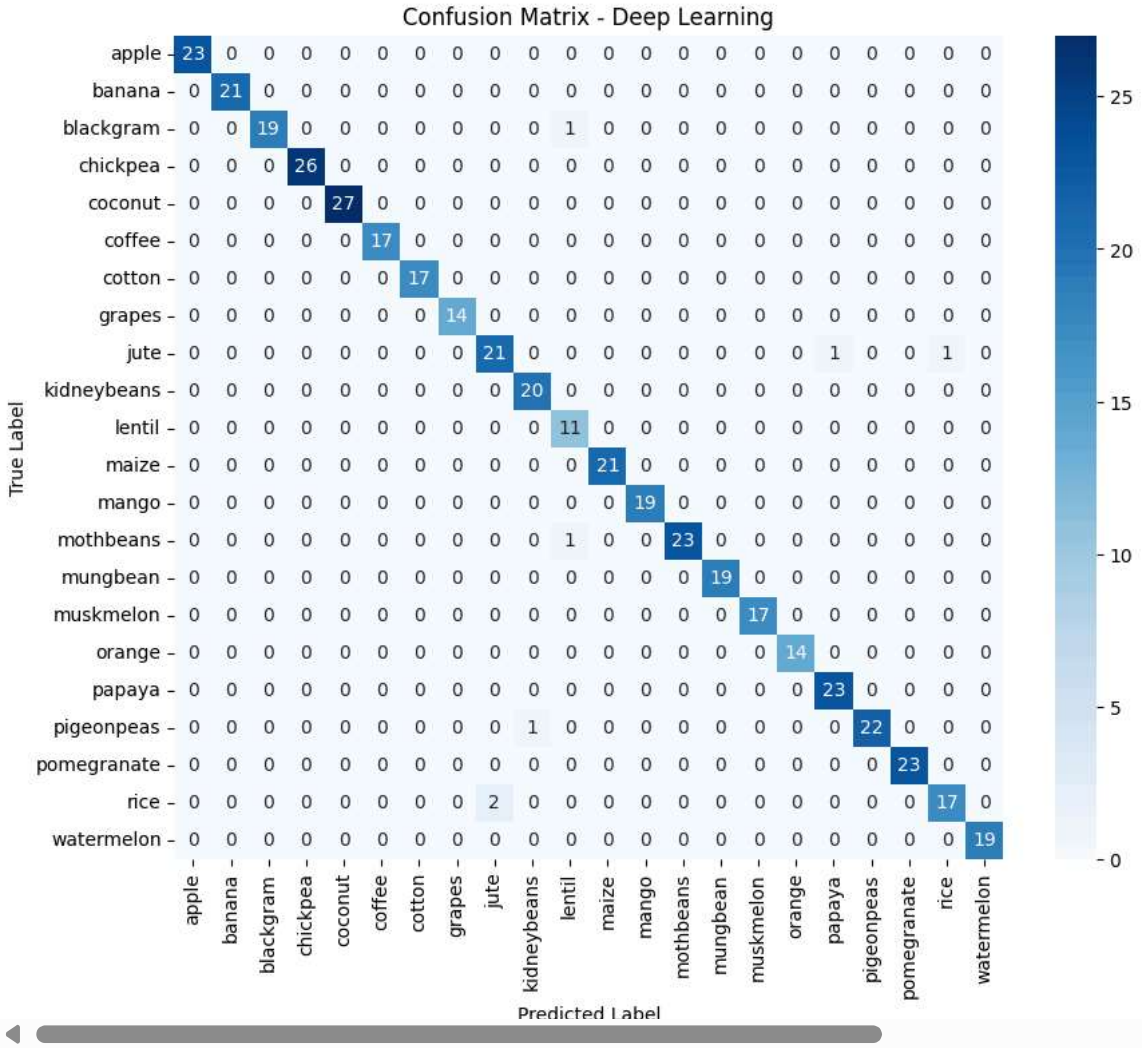
```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.metrics import accuracy_score
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.inspection import permutation_importance


file_path = '/content/Crop_recommendation.csv'
data = pd.read_csv(file_path)


X = data.drop(columns=['label'])
y = data['label']
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
y_categorical = to_categorical(y_encoded)
X_train, X_test, y_train, y_test = train_test_split(X, y_categorical, test_size=0.2, random_state=42)


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

def create_model():
    model = Sequential()
    model.add(Dense(128, input_dim=X_train_scaled.shape[1], activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model


model = create_model()
history = model.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_data=(X_test_scaled, y_test))

loss, accuracy = model.evaluate(X_test_scaled, y_test)
print(f"Test Accuracy: {accuracy}")

y_pred = model.predict(X_test_scaled)
y_pred_classes = y_pred.argmax(axis=-1)
y_true_classes = y_test.argmax(axis=-1)
```

```
accuracy_dl = accuracy_score(y_true_classes, y_pred_classes)
print(f"Deep Learning Model Accuracy: {accuracy_dl}")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 0.1585 - loss: 2.9128 - val_accuracy: 0.5727 - val_loss: 2.1864
Epoch 2/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.5860 - loss: 1.9496 - val_accuracy: 0.7818 - val_loss: 1.1251
Epoch 3/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7519 - loss: 1.0258 - val_accuracy: 0.8614 - val_loss: 0.6124
Epoch 4/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8413 - loss: 0.5873 - val_accuracy: 0.8955 - val_loss: 0.4278
Epoch 5/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8870 - loss: 0.4410 - val_accuracy: 0.9205 - val_loss: 0.3301
Epoch 6/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9015 - loss: 0.3429 - val_accuracy: 0.9318 - val_loss: 0.2772
Epoch 7/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9108 - loss: 0.2953 - val_accuracy: 0.9523 - val_loss: 0.2291
Epoch 8/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9367 - loss: 0.2396 - val_accuracy: 0.9341 - val_loss: 0.2151
Epoch 9/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9199 - loss: 0.2438 - val_accuracy: 0.9432 - val_loss: 0.1932
Epoch 10/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9399 - loss: 0.2037 - val_accuracy: 0.9477 - val_loss: 0.1707
Epoch 11/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9389 - loss: 0.1831 - val_accuracy: 0.9545 - val_loss: 0.1673
Epoch 12/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9434 - loss: 0.1656 - val_accuracy: 0.9523 - val_loss: 0.1529
Epoch 13/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9593 - loss: 0.1415 - val_accuracy: 0.9455 - val_loss: 0.1567
Epoch 14/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9588 - loss: 0.1404 - val_accuracy: 0.9568 - val_loss: 0.1321
Epoch 15/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9567 - loss: 0.1267 - val_accuracy: 0.9591 - val_loss: 0.1325
Epoch 16/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9499 - loss: 0.1380 - val_accuracy: 0.9545 - val_loss: 0.1352
Epoch 17/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9705 - loss: 0.1072 - val_accuracy: 0.9455 - val_loss: 0.1349
Epoch 18/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9583 - loss: 0.1165 - val_accuracy: 0.9614 - val_loss: 0.1208
Epoch 19/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9624 - loss: 0.1149 - val_accuracy: 0.9591 - val_loss: 0.1246
Epoch 20/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9606 - loss: 0.1070 - val_accuracy: 0.9614 - val_loss: 0.1222
Epoch 21/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9652 - loss: 0.1018 - val_accuracy: 0.9614 - val_loss: 0.1046
Epoch 22/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9662 - loss: 0.0940 - val_accuracy: 0.9614 - val_loss: 0.1028
Epoch 23/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9687 - loss: 0.0938 - val_accuracy: 0.9568 - val_loss: 0.1079
Epoch 24/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9732 - loss: 0.0877 - val_accuracy: 0.9591 - val_loss: 0.1040
Epoch 25/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9734 - loss: 0.0828 - val_accuracy: 0.9682 - val_loss: 0.1109
Epoch 26/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9615 - loss: 0.0946 - val_accuracy: 0.9614 - val_loss: 0.1054
Epoch 27/50
55/55 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9794 - loss: 0.0761 - val_accuracy: 0.9591 - val_loss: 0.1171
Epoch 28/50
```

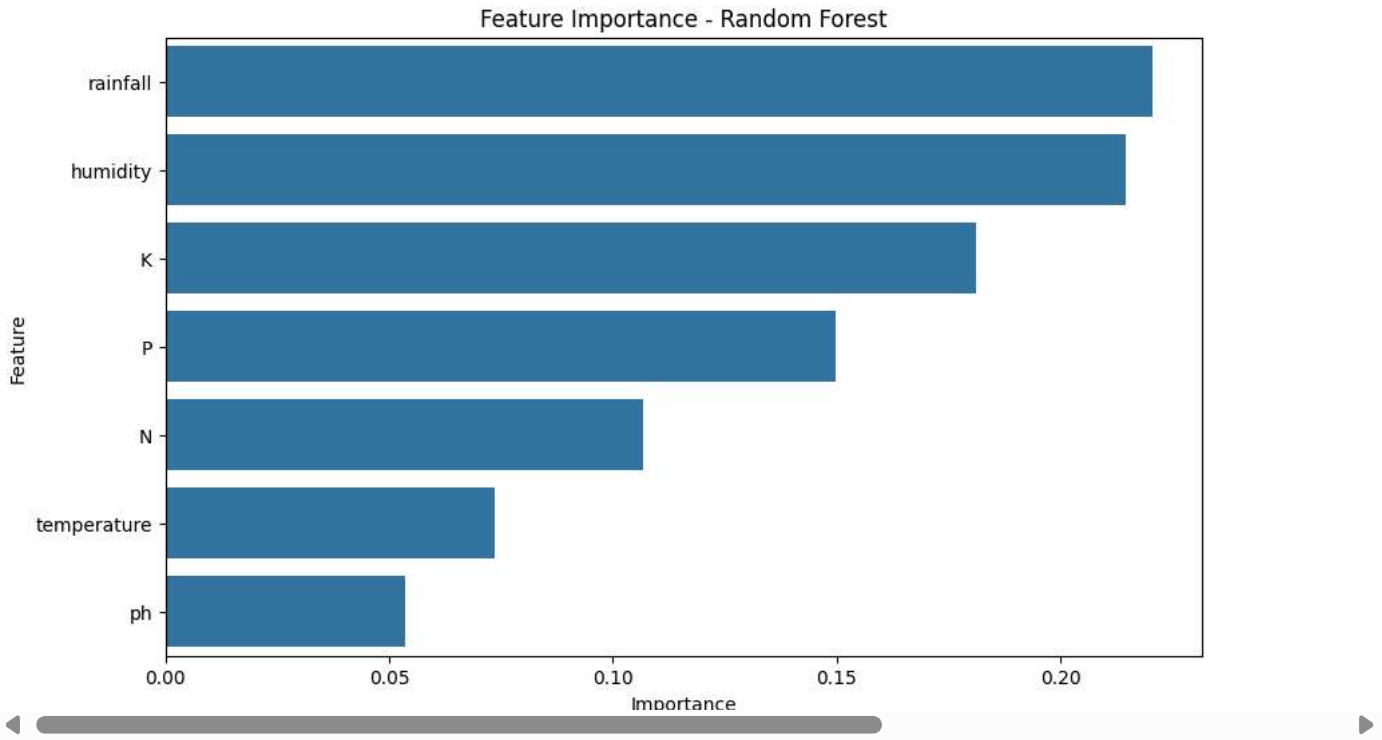
```
cm_dl = confusion_matrix(y_true_classes, y_pred_classes)

plt.figure(figsize=(10, 8))
sns.heatmap(cm_dl, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Deep Learning')
plt.show()
```




```
precision_dl = precision_score(y_true_classes, y_pred_classes, average='weighted')
recall_dl = recall_score(y_true_classes, y_pred_classes, average='weighted')
f1_dl = f1_score(y_true_classes, y_pred_classes, average='weighted')

print("\nDeep Learning Metrics:")
print(f"Precision: {precision_dl:.4f}")
print(f"Recall: {recall_dl:.4f}")
print(f"F1-score: {f1_dl:.4f}")
```

```
Deep Learning Metrics:
Precision: 0.9849
Recall: 0.9841
F1-score: 0.9842
```

```
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X, y)
importances_rf = rf_model.feature_importances_
feature_names = X.columns


feature_importance_df_rf = pd.DataFrame({'Feature': feature_names, 'Importance': importances_rf})
feature_importance_df_rf = feature_importance_df_rf.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df_rf)
plt.title('Feature Importance - Random Forest')
plt.show()
```



Start coding or generate with AI.