Name: Girish Nandanwar	Course: 22CT744 – Lab. Machine Learning
Roll No: A-56	Department: Computer Technology

Practical No.10

Aim: Build a crop recommendation model using tenserflow and keras

Theory:

1. Basic Concept

- The tree consists of:
    - Root Node: Represents the entire dataset and the first feature chosen for splitting.
    - Decision Nodes: Intermediate nodes where data is split based on certain feature thresholds.
    - Leaf Nodes: Represent the final class labels or output values.

The goal is to create branches that lead to the most homogeneous subsets possible.

---

2. How It Works

1. The algorithm evaluates all available features and selects the one that best separates the data.
2. It uses measures like Information Gain (Entropy-based) or Gini Index to decide the best split.
3. This process continues recursively until:
    - The data is perfectly classified, or
    - A stopping condition is reached (e.g., max depth or min samples per leaf).

---

3. Splitting Criteria

- Entropy: Measures impurity. Lower entropy = higher purity.

  Entropy=−∑pilog⁡2(pi)Entropy = -\sum p_i \log_2(p_i)Entropy=−∑pilog2(pi)

- Information Gain: Reduction in entropy after a dataset is split on an attribute.

  IG=Entropyparent−∑ninEntropychildiIG = Entropy_{parent} - \sum \frac{n_i}{n} Entropy_{child_i}IG=Entropyparent−∑nniEntropychildi

- Gini Index: Probability of misclassifying a sample.

  Gini=1−∑(pi)2Gini = 1 - \sum (p_i)^2Gini=1−∑(pi)2

---

4. Advantages

- Easy to understand and interpret.
- Requires little data preprocessing (no need for feature scaling).
- Can handle both categorical and numerical data.

---

5. Disadvantages

- Prone to overfitting, especially with small datasets.
- Small variations in data can result in a different tree structure.
- Biased toward features with more levels.

---

6. Evaluation Metrics

To assess performance:

- Accuracy: Ratio of correctly predicted observations to total observations.

- Confusion Matrix: Summarizes true positives, false positives, etc.
- Precision, Recall, F1-Score: For class-wise performance analysis.

Code:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

df = pd.read_csv("crop_data.csv")  # replace with your filename

print(df.head())

X = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y = df['label']

le = LabelEncoder()
y = le.fit_transform(y)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

```
model = Sequential([
    Dense(16, input_dim=X_train.shape[1], activation='relu'),
    Dense(32, activation='relu'),
    Dense(len(np.unique(y)), activation='softmax')  # Output layer = number of crops
])
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=50,
    batch_size=8,
    verbose=1
)
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"\nTest Accuracy: {accuracy*100:.2f}%")

sample = np.array([[90, 42, 43, 20.8, 82.0, 6.5, 202.9]])  # Example
sample_scaled = scaler.transform(sample)
prediction = model.predict(sample_scaled)
predicted_label = le.inverse_transform([np.argmax(prediction)])
print("\nPredicted Crop:", predicted_label[0])
```

Result:

```
Epoch 1/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.3045 - loss: 2.6384 - val_accuracy: 0.5182 - val_loss
Epoch 2/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6153 - loss: 1.4676 - val_accuracy: 0.7545 - val_loss
Epoch 3/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8199 - loss: 0.7548 - val_accuracy: 0.8250 - val_loss
Epoch 4/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8682 - loss: 0.4690 - val_accuracy: 0.8591 - val_loss
Epoch 5/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8841 - loss: 0.3471 - val_accuracy: 0.8955 - val_loss
Epoch 6/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9131 - loss: 0.2785 - val_accuracy: 0.9250 - val_loss
Epoch 7/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9318 - loss: 0.2299 - val_accuracy: 0.9182 - val_loss
Epoch 8/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9466 - loss: 0.1916 - val_accuracy: 0.9432 - val_loss
Epoch 9/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9557 - loss: 0.1606 - val_accuracy: 0.9523 - val_loss
Epoch 10/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9665 - loss: 0.1373 - val_accuracy: 0.9614 - val_loss
Epoch 11/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9710 - loss: 0.1205 - val_accuracy: 0.9614 - val_loss
Epoch 12/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9739 - loss: 0.1038 - val_accuracy: 0.9636 - val_loss
Epoch 13/50
...
Epoch 49/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9903 - loss: 0.0224 - val_accuracy: 0.9773 - val_loss
Epoch 50/50
220/220 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9926 - loss: 0.0218 - val_accuracy: 0.9659 - val_loss
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Conclusion:

TensorFlow and Keras together form a powerful combination for building and deploying machine learning and deep learning models.
TensorFlow provides a scalable backend for computations, while Keras simplifies the process of defining and training models, making deep learning more accessible to everyone—from beginners to AI researchers.