Report - Task - 01

Team - kodelabs

Table of Content

Approach Taken

1. Data Exploration and Preprocessing

The dataset was loaded using Pandas, and its features and structure were explored. There were no missing values in the dataset. Numerical features were scaled using StandardScaler to ensure they were on the same scale for the model. The categorical target variable (Label) had already been encoded into integers (Label_Encoded) in the dataset.

In [90]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
import joblib
```

In [91]:
```
data = pd.read_csv('Crop_Dataset.csv')
print(data.head())
```

```
    N   P   K  temperature   humidity        ph    rainfall  Total_Nutrients  \
0  90  42  43    20.879744  82.002744  6.502985  202.935536              175
1  85  58  41    21.770462  80.319644  7.038096  226.655537              184
2  60  55  44    23.004459  82.320763  7.840207  263.964248              159
3  74  35  40    26.491096  80.158363  6.980401  242.864034              149
4  78  42  42    20.130175  81.604873  7.628473  262.717340              162

   Temperature_Humidity  Log_Rainfall  Label  Label_Encoded
0           1712.196283      5.317804  wheat              0
1           1748.595734      5.427834  wheat              0
2           1893.744627      5.579595  wheat              0
3           2123.482908      5.496611  wheat              0
4           1642.720357      5.574878  wheat              0
```

In [92]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 12 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   temperature           2200 non-null   float64
 4   humidity              2200 non-null   float64
 5   ph                    2200 non-null   float64
 6   rainfall              2200 non-null   float64
 7   Total_Nutrients       2200 non-null   int64
 8   Temperature_Humidity  2200 non-null   float64
 9   Log_Rainfall          2200 non-null   float64
 10  Label                 2200 non-null   object
 11  Label_Encoded         2200 non-null   int64
dtypes: float64(6), int64(5), object(1)
memory usage: 206.4+ KB
None
```

In [93]:
```
print(data.isnull().sum())
```

```
N                       0
P                       0
K                       0
temperature             0
humidity                0
ph                      0
rainfall                0
Total_Nutrients         0
Temperature_Humidity    0
Log_Rainfall            0
Label                   0
Label_Encoded           0
dtype: int64
```

```
In [94]:  scaler = StandardScaler()
          numeric_columns = ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'Total_Nutrients', 'Temperature_Humidity', 'Lo
          data[numeric_columns] = scaler.fit_transform(data[numeric_columns])

          print(data.head())

          N         P         K  temperature  humidity        ph  rainfall  \
0  1.068797 -0.344551 -0.101688    -0.935587  0.472666  0.043302  1.810361
1  0.933329  0.140616 -0.141185    -0.759646  0.397051  0.734873  2.242058
2  0.255986  0.049647 -0.081939    -0.515898  0.486954  1.771510  2.921066
3  0.635298 -0.556811 -0.160933     0.172807  0.389805  0.660308  2.537048
4  0.743673 -0.344551 -0.121436    -1.083647  0.454792  1.497868  2.898373

   Total_Nutrients  Temperature_Humidity  Log_Rainfall  Label  Label_Encoded
0         0.287062             -0.203138      1.483789  wheat              0
1         0.399702             -0.151079      1.685576  wheat              0
2         0.086813              0.056511      1.963897  wheat              0
3        -0.038343              0.385081      1.811709  wheat              0
4         0.124359             -0.302501      1.955246  wheat              0
```

2. Model Training

A Random Forest classifier was chosen due to its robustness and effectiveness in classification tasks. The data was split into training and testing sets using an 80-20 split. The Random Forest classifier was instantiated and trained using the training set.

```
In [95]:  X = data.drop(columns=['Label', 'Label_Encoded'])
           y = data['Label_Encoded']

           X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [96]:  rf_classifier = RandomForestClassifier(random_state=42)
           rf_classifier.fit(X_train, y_train)

  Out[96]:  RandomForestClassifier
            RandomForestClassifier(random_state=42)

In [97]:  joblib.dump(rf_classifier, 'crop_recommendation_model.joblib')

  Out[97]:  ['crop_recommendation_model.joblib']
```

3. Model Evaluation

The model's performance was evaluated on the testing set. Metrics such as accuracy and the classification report (including precision, recall, and F1-score) were calculated to assess the model's performance.

```
In [98]:  y_pred = rf_classifier.predict(X_test)

           accuracy = accuracy_score(y_test, y_pred)
           print(f'Accuracy: {accuracy * 100:.2f}%')

           Accuracy: 99.09%
```

In [99]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.89      0.92        19
           1       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        26
           3       1.00      1.00      1.00        20
           4       1.00      1.00      1.00        23
           5       1.00      0.96      0.98        24
           6       1.00      1.00      1.00        19
           7       1.00      1.00      1.00        20
           8       0.92      1.00      0.96        11
           9       1.00      1.00      1.00        23
          10       1.00      1.00      1.00        21
          11       1.00      1.00      1.00        19
          12       1.00      1.00      1.00        14
          13       1.00      1.00      1.00        19
          14       1.00      1.00      1.00        17
          15       1.00      1.00      1.00        23
          16       1.00      1.00      1.00        14
          17       1.00      1.00      1.00        23
          18       1.00      1.00      1.00        27
          19       1.00      1.00      1.00        17
          20       0.92      0.96      0.94        23
          21       1.00      1.00      1.00        17

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

4. Joblib Model Creation and Prediction

The trained Random Forest classifier was saved as a joblib file for future use. To predict suitable crops for new environmental conditions, the model was loaded from the file and used to make predictions on new data. The predicted integer label was mapped back to the corresponding crop name using a dictionary.

In [100]:
```
model = joblib.load('crop_recommendation_model.joblib')

new_conditions = pd.DataFrame({
    'N': [20],
    'P': [15],
    'K': [10],
    'temperature': [35],
    'humidity': [30],
    'ph': [8.0],
    'rainfall': [50],
    'Total_Nutrients': [45],
    'Temperature_Humidity': [1050],
    'Log_Rainfall': [3.9]
})

new_conditions_scaled = scaler.transform(new_conditions)
new_prediction = model.predict(new_conditions_scaled)
```

Challenges Faced

1. Feature Scaling

Feature scaling is a crucial preprocessing step in machine learning, especially for algorithms that are sensitive to the scale of input features, such as distance-based algorithms (e.g., K-nearest neighbors) or gradient descent-based algorithms (e.g., linear regression, logistic regression, neural networks).

In our case, we have several numerical features like N, P, K, temperature, humidity, ph, rainfall, etc. These features may have different scales and units. For example, the values of N (Nitrogen content) could range from 0 to 200, while temperature may range from 0 to 50 degrees Celsius. If we don't scale these features, some features with larger magnitudes may dominate the learning process, leading to biased results.

By applying feature scaling, we transform the numerical features to a similar scale, typically ranging from -1 to 1 or 0 to 1. StandardScaler, which we used in our code, standardizes features by removing the mean and scaling to unit variance. This ensures that each feature contributes equally to the model's learning process, preventing any particular feature from having undue influence.

2. Model Selection

Model selection involves choosing the most appropriate algorithm for a given machine learning task. It's essential to select a model that can effectively capture the underlying patterns in the data and generalize well to unseen data.

In our case, we chose the Random Forest classifier. Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It's known for its robustness, scalability, and ability to handle high-dimensional data with a large number of features.

However, selecting the right model goes beyond just choosing a popular algorithm. It also involves tuning the model's hyperparameters to achieve the best performance. Hyperparameters are configuration settings that are external to the model and are set before the learning process begins. For Random Forest, hyperparameters include the number of trees in the forest, maximum depth of the trees, minimum samples required to split a node, etc.

In our code, we used default hyperparameters for the Random Forest classifier. However, in a real-world scenario, hyperparameter tuning through techniques like grid search or randomized search could be performed to find the optimal combination of hyperparameters that maximizes the model's performance on the dataset.

Insights Gained

The Random Forest classifier worked well for crop prediction, achieving high accuracy and reasonable performance in classifying different crop types based on environmental conditions. Proper preprocessing, including scaling and encoding features, was essential for effective model training and prediction.

Suggestions for Improving Model Performance

- Hyperparameter Tuning: Further tuning of the model's hyperparameters could improve performance.
- Alternative Models: Consider trying other machine learning models, such as Gradient Boosting or XGBoost, for potential improvements in prediction accuracy.
- Data Enrichment: Gathering more diverse data and features could help the model generalize better and improve performance.

Instructions for Running the Code

1. Load the Dataset: Ensure the dataset is in the correct location for the script.
2. Libraries: Install necessary libraries (Pandas, Scikit-learn, Joblib).
3. Run the Script: Execute the script to preprocess data, train the model, and evaluate its performance.
4. Use the Model: Utilize the saved model (crop_recommendation_model.joblib) to make predictions on new environmental conditions.