Crop Recommendation Model Documentation Report - Team Intoloo

Step 1: Data Loading and Exploration

The dataset "Crop_Dataset.csv" was loaded into a Pandas DataFrame to understand its structure and features. The first few rows of the dataset were printed to provide an overview of the data. The dataset consists of various environmental factors such as nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, pH, rainfall, and total nutrients, along with corresponding crop labels and their encoded values.

Step 2: Handling Missing Values

No missing values were found in the dataset. Hence, no imputation or handling of missing values was necessary.

Step 3: Encoding Categorical Variables

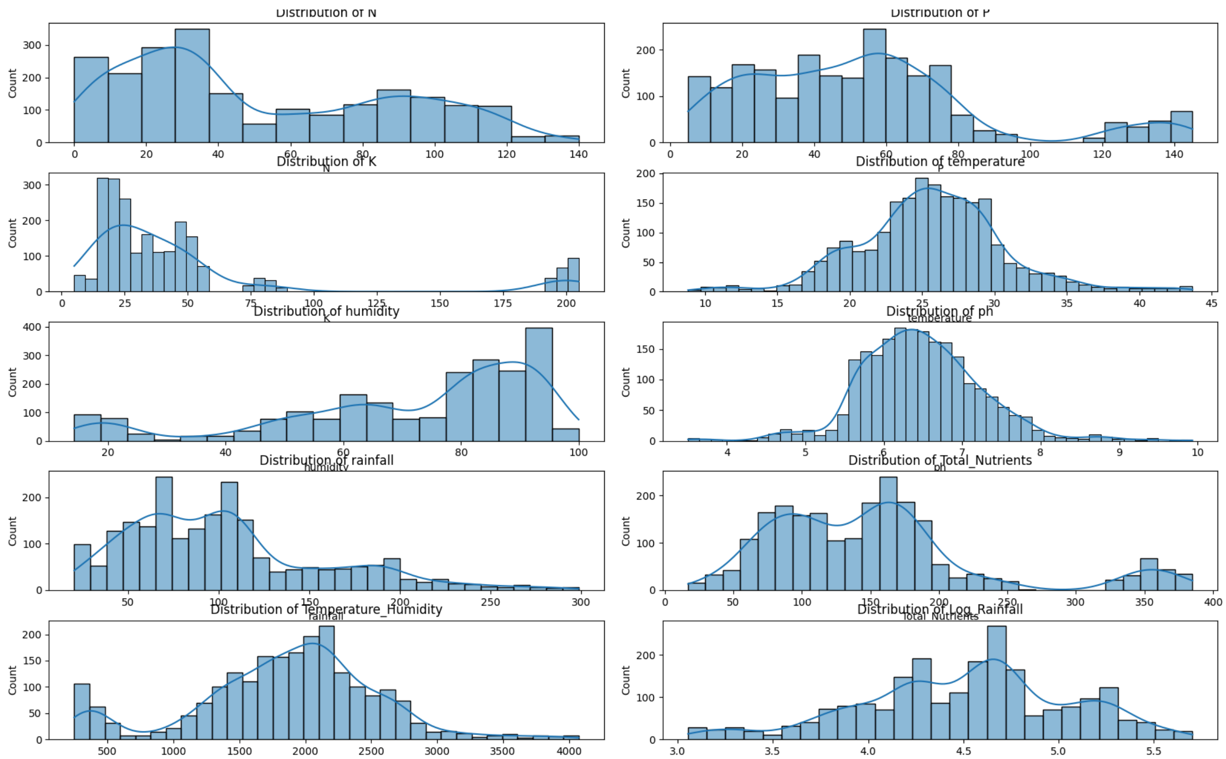
There were no categorical variables present in the dataset that required encoding. All features were numerical, making them suitable for model training without additional preprocessing.

Step 4: Scaling Numerical Features

Numerical features were scaled using the StandardScaler from scikit-learn to ensure that all features have the same scale. This step is crucial for many machine learning algorithms, especially those based on distance metrics or gradient descent.

Step 5: Data Visualization

Numerical features were visualized using histograms to understand their distributions. Additionally, a heatmap was generated to visualize the correlation between features, providing insights into potential relationships and dependencies. The distribution of crop labels was also visualized using a count plot to understand the class distribution in the dataset.



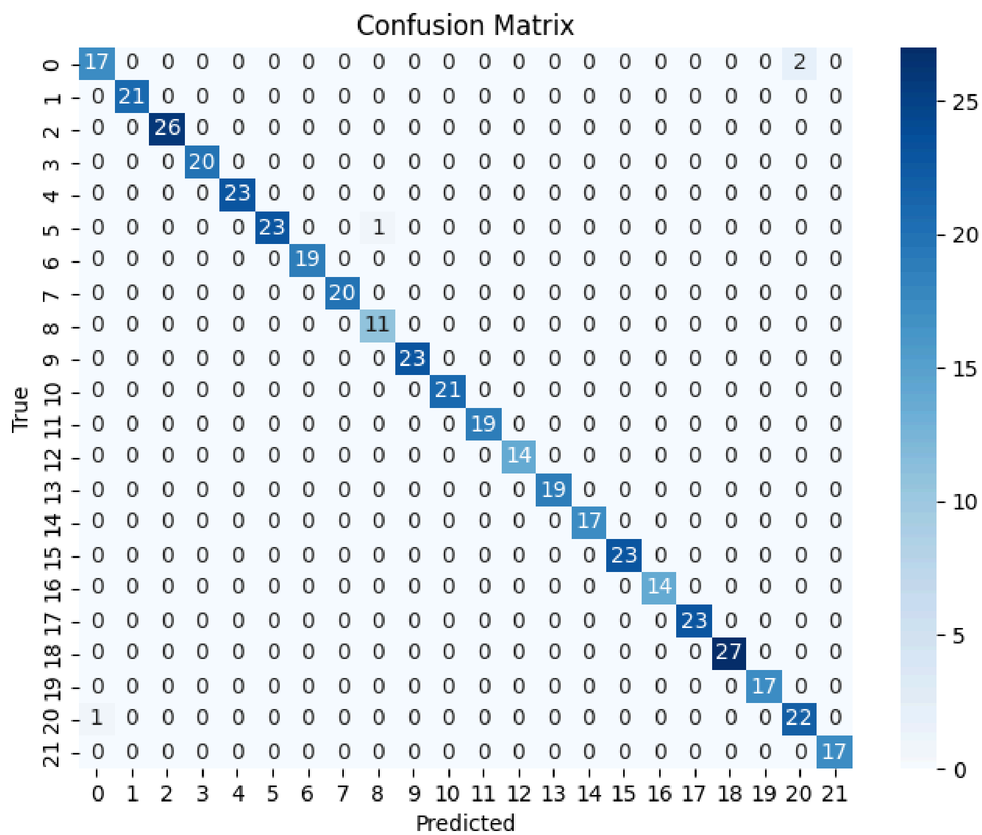
Step 6: Model Training

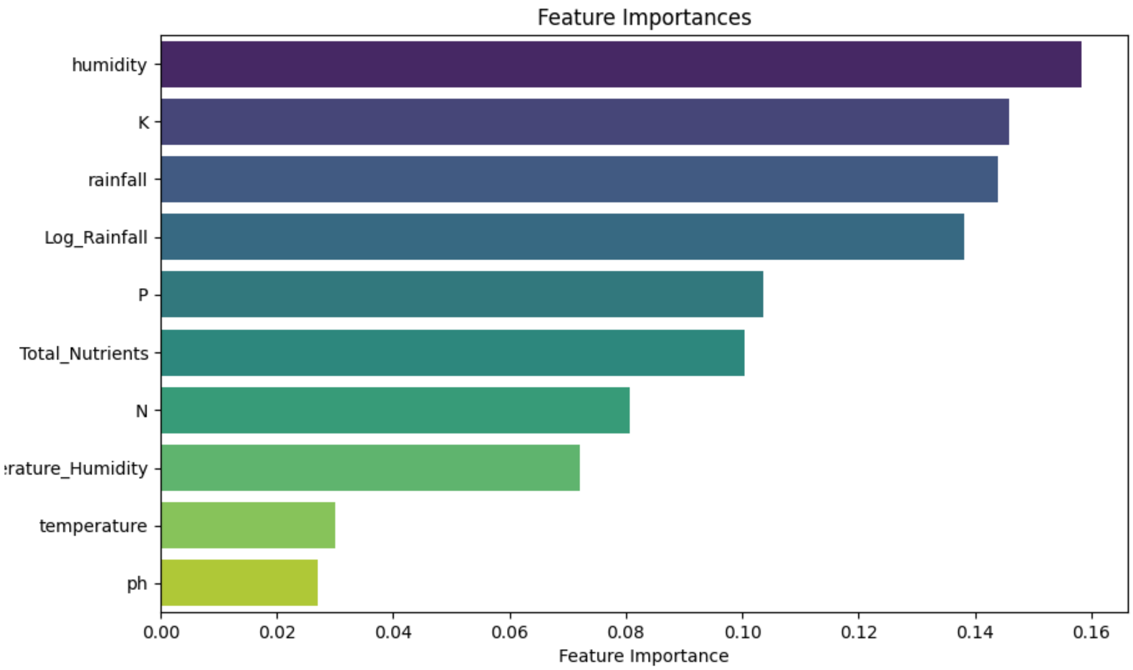
A Random Forest Classifier was chosen as the machine learning algorithm for building the predictive model. This choice was made due to the algorithm's ability to handle complex relationships and feature interactions effectively. The model was trained using the training data obtained after preprocessing.

```
(venv) (base) shakthiraveen@Shakthis-MacBook-Air task01 % python model.py
    N   P   K  temperature   humidity        ph    rainfall  Total_Nutrients  Temperature_Humidity  Log_Rainfall  Label  Label_Encoded
0  90  42  43    20.879744  82.002744  6.502985  202.935536              175           1712.196283      5.317804  wheat              0
1  85  58  41    21.770462  80.319644  7.038096  226.655537              184           1748.595734      5.427834  wheat              0
2  60  55  44    23.004459  82.320763  7.840207  263.964248              159           1893.744627      5.579595  wheat              0
3  74  35  40    26.491096  80.158363  6.980401  242.864034              149           2123.482908      5.496611  wheat              0
4  78  42  42    20.130175  81.604873  7.628473  262.717340              162           1642.720357      5.574878  wheat              0
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
N                         int64
P                         int64
K                         int64
temperature             float64
humidity                float64
ph                      float64
rainfall                float64
Total_Nutrients           int64
Temperature_Humidity    float64
Log_Rainfall            float64
Label                    object
Label_Encoded             int64
dtype: object
```

Step 7: Model Evaluation

The trained model's performance was evaluated using various metrics, including accuracy, precision, recall, and F1-score. The classification report provided detailed information about the model's performance for each class. Additionally, a confusion matrix was generated to visualize the model's predictions and identify any misclassifications.





Step 8: Model Saving

The trained Random Forest model was saved using the joblib library, allowing for easy reuse and deployment in future applications. This step ensures that the model can be accessed and utilized without needing to retrain it from scratch.

Conclusion

In conclusion, this documentation report outlines the process of building a crop recommendation model using machine learning techniques. By preprocessing the dataset, training a Random Forest classifier, evaluating its performance, and saving the trained model, we have created a robust solution for recommending suitable crops based on environmental conditions.

Appendix

```
# Importing necessary libraries
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 accuracy_score
import joblib

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

# Step 1: Load the dataset
data = pd.read_csv("Crop_Dataset.csv")

# Display the first few rows of the dataset to understand its structure
print(data.head())

# Step 2: Handle missing values (if any)
# Check for missing values in the dataset
print(data.isnull().sum())

# Since there are no missing values, we can proceed without any imputation

# Step 3: Encode categorical variables (if any)
# Check if there are any categorical variables that need encoding
print(data.dtypes)

# There are no categorical variables that need encoding in this dataset

# Step 4: Scale numerical features
# Separate features and labels
X = data.drop(['Label', 'Label_Encoded'], axis=1)
y = data['Label_Encoded']

# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Visualize numerical features
```

```python
numerical_features = ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall',
'Total_Nutrients', 'Temperature_Humidity', 'Log_Rainfall']

fig, ax = plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
for i, feature in enumerate(numerical_features):
   sns.histplot(data=data, x=feature, ax=ax[i//2, i%2], kde=True)
   ax[i//2, i%2].set_title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()

# Visualize correlation between features
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

# Visualize target variable distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='Label')
plt.title('Distribution of Crop Labels')
plt.show()



# Now, we have preprocessed the data. We can proceed to model training.

# Step 6: Model Training
# Choose a machine learning algorithm (Random Forest Classifier)
model = RandomForestClassifier(random_state=42)

# Train the model using the training data
model.fit(X_train, y_train)

# Step 7: Model Evaluation
# Evaluate the trained model's accuracy using the testing dataset
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the model: {accuracy}")

# Provide insights into model performance
# For classification tasks, accuracy is one of the metrics to evaluate the model's
performance.
```

```
# Additionally, you can explore other metrics like precision, recall, and F1-score for
each class.
# Compute precision, recall, and F1-score
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# numerical_features = ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall',
'Total_Nutrients', 'Temperature_Humidity', 'Log_Rainfall']

# Plot feature importances
feature_importances = model.feature_importances_
sorted_indices = np.argsort(feature_importances)[::-1]
sorted_features = [numerical_features[i] for i in sorted_indices]

plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importances[sorted_indices], y=sorted_features,
palette='viridis')
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importances')
plt.show()


# Now, let's proceed to save the trained model for future use.

# Step 8: Save the trained model
joblib.dump(model, 'crop_recommendation_model.joblib')
print("Model saved successfully!")
```