Implementation of Second Module

Weekly Date

14/10/24 to 19/10/24

Model Training and Accuracy Evaluation:

1. Model Selection

- **Decision Tree Classifier**:
    - This model splits data based on the most important features to make decisions, resulting in a tree-like structure.
    - It's easy to interpret and understand, but it can easily overfit, meaning it might perform poorly on new data.
- **Random Forest Classifier**:
    - This is a collection (or "forest") of many decision trees.
    - Each tree in the forest is built using random subsets of the data, and the final prediction is based on the majority vote from all trees.
    - Random Forest usually works better because it reduces the risk of overfitting by averaging predictions.

2. Model Training

- Both models were trained using the **training data** (X_train, y_train):
    - **X_train** contains the input features (like ph, NPK, etc.),
    - **y_train** contains the actual labels (the target value we want to predict, like "yes" or "no").
- The fit() function trains both models by finding the best splits (for decision trees) or creating multiple trees (for Random Forest).

python

```
dt_model.fit(X_train, y_train)  # Train Decision Tree
rf_model.fit(X_train, y_train)  # Train Random Forest
```

3. Predictions and Accuracy Calculation

- After training, we use both models to make predictions on **test data** (X_test).
- Then, we compare the predicted values to the actual values (y_test) to calculate **accuracy**, which tells us how well the model performed.

Accuracy = (Number of correct predictions) / (Total number of predictions)

python

```
dt_predictions = dt_model.predict(X_test)  # Predictions from Decision Tree
rf_predictions = rf_model.predict(X_test)  # Predictions from Random Forest
```

- **Accuracy Score**: This tells us the percentage of correct predictions:
  - **Decision Tree**: 5.40%
  - **Random Forest**: 6.81%

Although both models have low accuracy, **Random Forest** performed slightly better.

4. Reasons for Choosing Random Forest Over Decision Tree

- **Reduces Overfitting**: Random Forest creates multiple trees, which helps reduce the model's reliance on specific data points and improves generalization to new data.
- **Higher Accuracy**: Combining predictions from multiple trees usually results in better overall performance.
- **Noise Tolerance**: Random Forest is more robust to small changes or errors in the data.
- **Works with Large Data**: Random Forest handles large datasets and high-dimensional features better than a single Decision Tree.
- **Feature Importance**: Random Forest can identify which features are most important for predictions.

- **Handles Missing Values**: It can still work well even if some data points are missing.
- **Outlier Resistance**: Random Forest is less influenced by extreme values or outliers because it averages over multiple trees.

Conclusion:

Even though both models are based on decision trees, **Random Forest** is generally the better choice because it:

- Reduces overfitting,
- Handles large and noisy data better, and
- Provides higher accuracy compared to a single Decision Tree.

Implementation Details

```
[21] predicted_fertilizer = predict_fertilizer(district="Kolhapur", soil_color="Black", nitrogen=75, phosphorus=50, potassium=100, pH=6.5, rainfall
     print("\n Predicted Fertilizer:", predicted_fertilizer)
```

```
 Predicted Fertilizer: Urea
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

```
from sklearn.metrics import accuracy_score

# Predict on test data
dt_predictions = dt_model.predict(X_test)
rf_predictions = rf_model.predict(X_test)

# Calculate accuracy
dt_accuracy = accuracy_score(y_test,
rf_accuracy = accuracy_score(y_test, rf_predictions)

print(f" Decision Tree Accuracy: {dt_accuracy * 100:.2f}%")
print(f" Random Forest Accuracy: {rf_accuracy * 100:.2f}%")
```

ndarray: rf_predictions

ndarray with shape (5184,)

```
 Decision Tree Accuracy: 5.40%
 Random Forest Accuracy: 6.81%
```

Implementation Details

```
Decision Tree Accuracy: 5.40%
Random Forest Accuracy: 6.81%
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2732: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2732: UserWarning: X has feature names, but RandomForestClassifier was fitted without feature names
  warnings.warn(
```

1.Decision Tree Accuracy: 5.40%

2.Random Forest Accuracy: 6.81%

Random Forest is preferred because it gives more accurate results and reduces overfitting.

Why Random Forest Over Decision Tree?

1. **Reduces Overfitting** – Decision Trees overfit; Random Forest averages multiple trees for better generalization.
2. **Higher Accuracy** – Majority voting in Random Forest improves prediction accuracy.
3. **Handles Noise Better** – Less sensitive to small data changes compared to Decision Trees.
4. **Works Well on Large Datasets** – Scales better with high-dimensional data.
5. **Feature Importance** – Identifies key parameters influencing predictions.
6. **Handles Missing Values** – More robust compared to single Decision Trees.
7. **Resistant to Outliers** – Averaging across trees minimizes their impact.