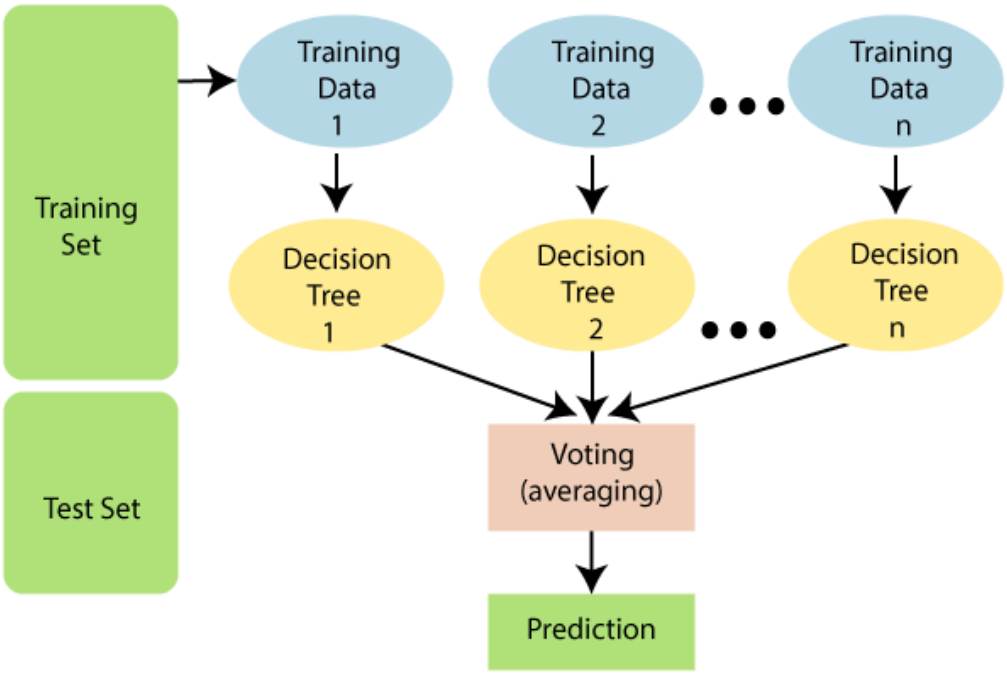
Cycle-11

Aim: To implement Random Forest Model

Solution:

Random Forest Model

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.
- It can be used for both Classification and Regression problems in ML.
- It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting



Source Code:

Step 1:Import required libraries and Load the dataset

```
#import required libraries and load the dataset
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_decision_regions
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

iris = datasets.load_iris()
X = iris.data[:, 2:]
y = iris.target
```

Step 2: Create training/ test data split

```
# Create training/ test data split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stratify=y)
```

Step 3: Create an instance of Random Forest Classifier & fit the model

```
# Create an instance of Random Forest Classifier

forest = RandomForestClassifier(criterion='gini',
                                n_estimators=5,
                                random_state=1,
                                n_jobs=2)

# Fit the model

forest.fit(X_train, y_train)
```

Step 4: Splitting Data into Training and Testing Datasets

```
# Measure model performance

y_pred = forest.predict(X_test)
print('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
```

Sample Output:

```
Accuracy: 0.978
```

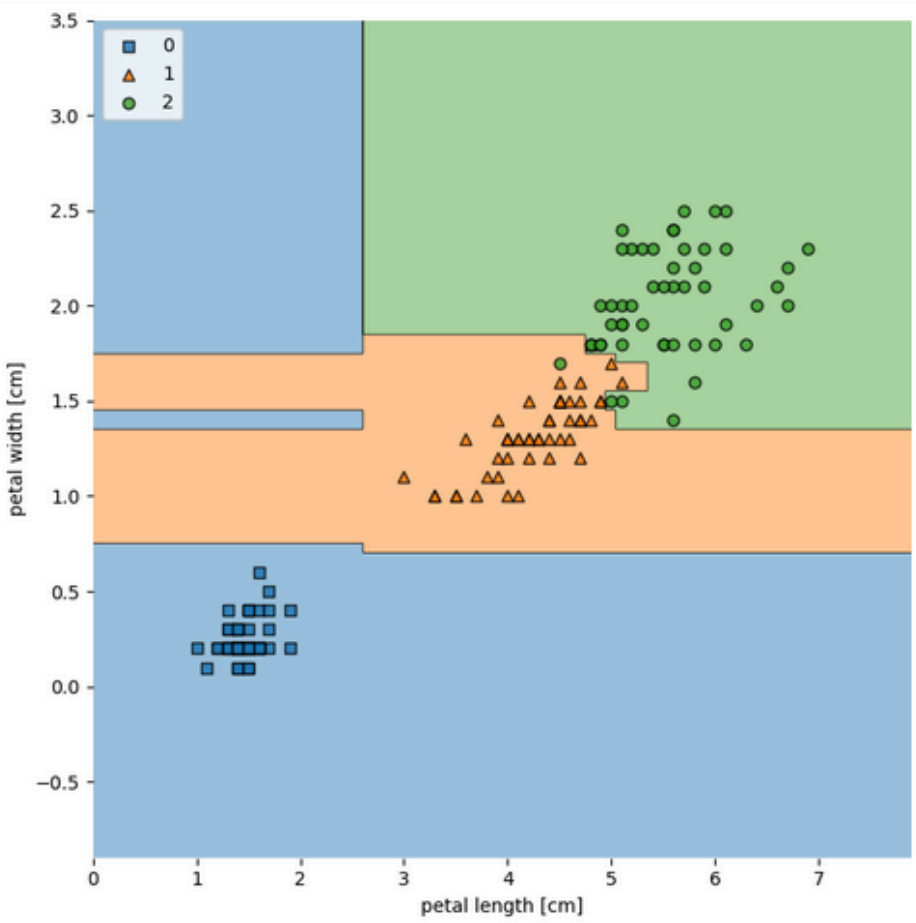
Step 5: Visualization of data

```
X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))

# plot_decision_regions function takes "forest" as classifier

fig, ax = plt.subplots(figsize=(7, 7))
plot_decision_regions(X_combined, y_combined, clf=forest)
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

Sample Output:


0
1
2
petal width [cm]
petal length [cm]
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7