Experiment No. 6

Expt No. 6	**Implementation of Random Forest Algorithm**
Date :	

Aim:

To implement and understand the working of the Random Forest algorithm for classification tasks in Machine Learning using the digits dataset.

System Requirements:

- Operating System: Windows 8 or above / Linux / macOS
- Memory (RAM): Minimum 4 GB
- Processor: Minimum 2.33 GHz (Dual Core or higher)

Software/Tools Required:

Jupyter Notebook / Anaconda Navigator / Google Colaboratory / Spyder
Python 3.x with the following libraries:

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn

Purpose of the Experiment:

- To understand the application of the Random Forest algorithm for classification tasks using the digits dataset.
- To explore the process of splitting data into features and labels, training a Random Forest model, and evaluating its performance.

- To visualize model performance using metrics like accuracy, confusion matrix, and classification report.

<u>Expected Outcomes:</u>

- Understand the concept and principles of the Random Forest algorithm.
- Implement and train a Random Forest classifier on a multi-class dataset.
- Evaluate the performance of the model using appropriate metrics.
- Visualize how Random Forest aggregates decisions from multiple decision trees.

<u>Theory:</u>

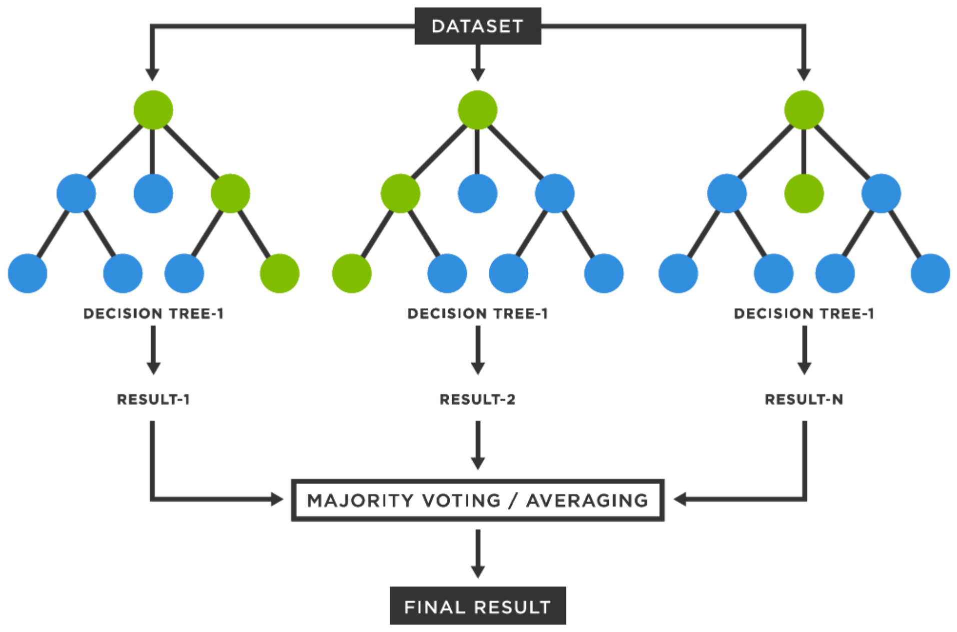
Random Forest is a supervised ensemble learning algorithm primarily used for classification and regression tasks. It constructs multiple decision trees during training and outputs the mode of the classes (for classification) or mean prediction (for regression) of the individual trees.

Key Features:

- Combines multiple decision trees to reduce overfitting and improve generalization.
- Uses bootstrapped sampling and random feature selection for tree diversity.
- Suitable for multi-class classification, like digits recognition.

Working of Random Forest:
1. Bootstrap Sampling: Randomly selects subsets of data to train individual trees.
2. Feature Randomness: Each tree considers a random subset of features for splitting nodes.
3. Tree Training: Each decision tree is trained independently.
4. Voting: For classification, the final output is the mode of all tree predictions.

In Random Forest, the final prediction depends on the type of task. For classification, each decision tree votes for a class, and the class with the majority of votes becomes the final prediction, improving accuracy and reducing errors from individual trees. For regression, each tree predicts a numerical value, and the final prediction is the average of all tree predictions, providing a more stable and reliable estimate.



Real-World Applications of Random Forest Algorithm

Here are four real-world applications of Random Forest Algorithm:

1. Handwritten digit recognition (MNIST, digits datasets).
2. Medical diagnosis for disease classification.
3. Credit risk analysis in financial institutions.
4. Fraud detection in transactional data.

1. Problem Statement

The objective of this experiment is to implement a Random Forest classifier to predict handwritten digits (0–9) based on pixel values. The digits.csv dataset is used for training and testing the model. The experiment involves splitting the data into features and labels, training the Random Forest model, evaluating its performance, and analyzing the results using accuracy metrics, confusion matrix, and classification report.

You can download the dataset from the following link:

https://github.com/25-Madhuri/Dataset/blob/main/digits.csv

Sample Code

Import Libraries

# Importing necessary libraries for data manipulation, visualization, and modeling

```
import pandas as pd
from sklearn.datasets import load_digits
```

#Load the Digits dataset from a CSV file

```
df = pd.read_csv('/content/digits.csv')
```

# List all attributes and methods of the digits object

```
dir(digits)
```

# Enable inline plotting for visualizations in Colab

```
%matplotlib inline
```

# Import the matplotlib library for plotting

```
import matplotlib.pyplot as plt
```

# Display sample images from the digits dataset
# Set the color map to grayscale

```
plt.gray()
```

# Plot the first 4 digit images

```
for i in range(4):
   plt.matshow(digits.images[i])
   plt.show()
```

# Convert the digits dataset features into a DataFrame for easier viewing

```
df = pd.DataFrame(digits.data)
```

# Display the first 5 rows of the DataFrame

```
df.head()
```

# Add the target labels (digits 0-9) to the DataFrame

```
df['target'] = digits.target
```

# Display the first 5 rows to verify the target column

```
df.head()
```

# Display the first 12 rows of the DataFrame to inspect the dataset

```
df[0:12]
```

# Split dataset into features and target

```
X = df.drop('target', axis='columns')
y = df['target']
```

# Import train_test_split for splitting dataset

```
from sklearn.model_selection import train_test_split
```

# Split data into training and testing sets (80% train, 20% test)

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

# Import RandomForestClassifier from sklearn

```
from sklearn.ensemble import RandomForestClassifier
```

# Initialize the Random Forest classifier with 20 trees

```
model = RandomForestClassifier(n_estimators=20)
```

```
# Train the model on the training data

model.fit(X_train, y_train)

# Evaluate the model's accuracy on the test set

accuracy = model.score(X_test, y_test)
print("Model Accuracy on Test Set:", accuracy)

# Make predictions on the test set using the trained model

y_predicted = model.predict(X_test)

# Import confusion_matrix from sklearn

from sklearn.metrics import confusion_matrix

# Compute the confusion matrix to evaluate model performance

cm = confusion_matrix(y_test, y_predicted)

# Display the confusion matrix

cm

# Import libraries for visualization

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn

# Plot confusion matrix as a heatmap for better visualization

plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix - Random Forest on Digits Dataset')
plt.show()
```

Observation

I. Record the following in observation table:

Sr.No	Metric	Description	Observations
1.	Shape of Dataset	Number of rows and columns in the dataset (df.shape)	
2.	Training and Testing Size	Size of training and testing sets after splitting the dataset	
3.	Model Accuracy	Accuracy score of Random Forest on test data	
4.	Sample Predictions	Example of predicted values for first few test samples	
5.	Classification Summary	Record: Number of Correct Predictions Number of Incorrect Predictions	

Exercise:

Use the famous Iris flower dataset from sklearn.datasets to predict flower species using Random Forest Classifier.

1. Measure prediction score using default n_estimators = 10.
2. Now fine-tune your model by changing the number of trees in your classifier and report the best score you can get along with the corresponding number of trees.
3. Dataset link: https://github.com/25-Madhuri/Dataset/blob/main/Iris.csv

Conclusion –
The Random Forest model accurately classifies handwritten digits by combining predictions from multiple decision trees. This ensemble approach improves accuracy and reduces errors, as shown by the confusion matrix and heatmap.

Viva Questions

Sr.No	Question	CO
1.	What is a Random Forest and how does it work?	**1ICPC406_3**
2.	How does Random Forest reduce overfitting compared to a single decision tree?	**1ICPC406_3**
3.	What are the advantages of using Random Forest for classification tasks?	**1ICPC406_3**
4.	How is the final prediction made in Random Forest?	**1ICPC406_3**
5.	Explain the role of bootstrapping and feature randomness in Random Forest.	**1ICPC406_3**
6.	How can model performance be evaluated for multi-class classification?	**1ICPC406_3**

References –

a. **Textbook –**

   i. Machine Learning with Python – An approach to Applied ML – Abhishek Vijayvargiya, BPB Publications, 1st Edition 2018

   ii. Machine Learning, Tom Mitchell, McGraw Hill Education, 1st Edition 1997

b. Online references –

   i. https://scikit-learn.org/stable/modules/ensemble.html#forest

   ii. https://www.datacamp.com/tutorial/random-forests-classifier-python