Experiment No : 6 **Roll no :**

Aim : To study and implement random forest algorithm in machine learning.

Theory : Random Forest is an ensemble learning method primarily used for classification and regression tasks. It was introduced by Leo Breiman in 2001 as an improvement over decision trees and bagging methods. The fundamental idea behind Random Forest is to build a large number of decision trees during training and output the mode of the classes (for classification) or mean prediction (for regression) of the individual trees.

Random Forest reduces overfitting and improves accuracy by combining the predictions of multiple decision trees, each built on different subsets of the data and features. It is widely used due to its simplicity, robustness, and ability to handle high-dimensional data.

2. How Random Forest Works

The Random Forest algorithm follows these main steps:

Step 1: Bootstrapping (Data Sampling)

- From the original dataset of size N, multiple bootstrap samples (also of size N) are drawn **with replacement**. This means some data points may appear multiple times, while others may be left out.

Step 2: Building Decision Trees

- For each bootstrap sample, a **decision tree** is grown.
- At each node of the tree, a **random subset of features (m << total features)** is selected, and the best split is found only from this subset. This is a key difference from standard decision trees, which consider all features.
- Trees are grown **to maximum depth** without pruning (unlike traditional decision trees).

Step 3: Aggregation (Ensemble Prediction)

- **For classification**: Each tree votes for a class, and the majority vote determines the final prediction.
- **For regression**: The average of predictions from all trees is taken as the final output.

This process introduces two types of randomness:

1. Randomness in data (bootstrapping).
2. Randomness in feature selection at each split.

Together, these reduce correlation between individual trees and enhance the generalization ability of the model.

3. Mathematical Formulation

Let’s denote the training dataset as D={(xi,yi)}i=1ND = \{(x_i, y_i)\}_{i=1}^{N}. A Random Forest constructs TT decision trees, each built from a bootstrap sample Dt⊂DD_t \subset D. The prediction y^\hat{y} for a new input xx is:

Classification:

y^=mode{ht(x)}t=1T\hat{y} = \text{mode}\{h_t(x)\}_{t=1}^{T}

Regression:

y^=1T∑t=1Tht(x)\hat{y} = \frac{1}{T} \sum_{t=1}^{T} h_t(x)

Where ht(x)h_t(x) is the prediction of the ttht^{th} tree.

4. Advantages of Random Forest

- **High Accuracy**: Due to ensemble nature and reduced overfitting.
- **Handles High-Dimensional Data**: Effective even when the number of features is very large.
- **Robust to Outliers and Noise**: Since multiple trees average out errors.
- **Feature Importance**: Provides insight into the relative importance of each feature.
- **No Need for Scaling:** Works well without normalization of input features.

6. Applications of Random Forest

- **Medical Diagnosis**: Disease prediction using patient features.
- **Finance**: Credit scoring, fraud detection.
- **Marketing**: Customer segmentation, churn prediction.
- **Ecology**: Species classification, environmental modeling.
- **Natural Language Processing**: Text classification, sentiment analysis.

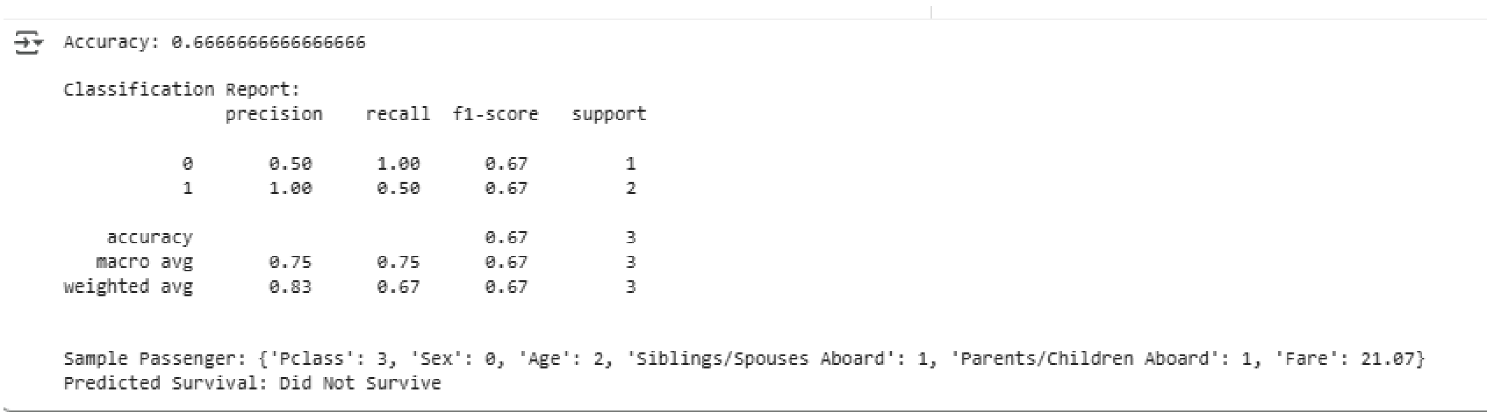
Source Code :

```
# Random Forest Example in Python

import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load dataset (Iris dataset for demo)
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target
```

```python
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create Random Forest model
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)

# Predict on test set
y_pred = rf_classifier.predict(X_test)

# Evaluate performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))

# Predict for a sample
sample = X_test.iloc[0:1]
prediction = rf_classifier.predict(sample)

print("\nSample Input:", sample.to_dict(orient="records")[0])
print("Predicted Class:", iris.target_names[prediction[0]])


url =
"https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titan
ic.csv"
response = requests.get(url)

with open('titanic.csv', 'wb') as f:
    f.write(response.content)

print("titanic.csv downloaded successfully.")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# -----------------------------
# Create a small Titanic dataset
# -----------------------------
data = {
    "Pclass": [3, 1, 3, 1, 3, 2, 3, 1, 3, 2],
    "Sex": ["male", "female", "female", "female", "male", "male",
"male", "female", "female", "male"],
    "Age": [22, 38, 26, 35, 35, 30, 27, 54, 2, 20],
    "Siblings/Spouses Aboard": [1, 1, 0, 1, 0, 0, 0, 0, 1, 0],
    "Parents/Children Aboard": [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    "Fare": [7.25, 71.28, 7.92, 53.10, 8.05, 10.50, 8.45, 51.86, 21.07,
13.00],
    "Survived": [0, 1, 1, 1, 0, 0, 0, 1, 1, 0]  # target column
}
```

```
df = pd.DataFrame(data)

# ------------------------------
# Preprocess
# ------------------------------
X = df.drop("Survived", axis=1)
y = df["Survived"]

# Encode 'Sex'
X["Sex"] = X["Sex"].map({"female": 0, "male": 1})

# ------------------------------
# Train/Test Split
# ------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# ------------------------------
# Random Forest Model
# ------------------------------
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Predictions
y_pred = rf.predict(X_test)

# ------------------------------
# Evaluation
# ------------------------------
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))

# ------------------------------
# Sample Prediction
# ------------------------------
sample = X_test.iloc[0:1]
prediction = rf.predict(sample)

print("\nSample Passenger:", sample.to_dict(orient="records")[0])
print("Predicted Survival:", "Survived" if prediction[0] == 1 else "Did
Not Survive")
```

Output :



```
Accuracy: 0.6666666666666666

Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3


Sample Passenger: {'Pclass': 3, 'Sex': 0, 'Age': 2, 'Siblings/Spouses Aboard': 1, 'Parents/Children Aboard': 1, 'Fare': 21.07}
Predicted Survival: Did Not Survive
```

Conclusion

Random Forest is a powerful, flexible machine learning model that leverages the strength of multiple decision trees to make robust predictions. Its use of randomization and ensemble learning makes it resistant to overfitting and highly accurate across a range of tasks. Despite its interpretability trade-offs, it remains a go-to method for many real-world data science applications.