Practical 5

Problem Statement:

Implement Random Forest Classifier model to predict the safety of the car.

Dataset: car evaluation dataset

Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set

Objective:

1. To understand and implement the Random Forest Classifier
2. Predict the safety of cars based on given features.

Outcome:

After implementing the Random Forest Classifier on the Car Evaluation Dataset, students will adeptly utilize ensemble methods to predict car safety and evaluate model performance based on provided features.

Theory:

Random Forest Classifier is a supervised ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes for classification or the mean/average prediction of the individual trees for regression. It introduces randomness into the model when growing the trees and combines their outputs to improve performance, reduce overfitting, and give a more accurate prediction.

Key Characteristics:

1. Ensemble Method: It combines multiple decision trees to create a more robust and accurate model.

2. Bagging: Random Forest uses bootstrap aggregation, where subsets of the dataset are selected with replacement and used to train individual trees.

3. Feature Randomness: In each split test, only a random subset of the features is considered, introducing further diversity among the trees.

4. Reduced Overfitting: By averaging out biases, the combined prediction is often more accurate and less prone to overfitting.

5. Handle Missing Values: Random Forest can handle missing values and still produce accurate predictions.

6. Feature Importance: The model provides insights into which features have the most impact on predictions.

7. Versatility: It can be used for both regression and classification tasks.

However, like all models, Random Forest has its disadvantages, including the potential for increased complexity and longer computation times compared to simpler algorithms, especially when dealing with very large datasets.

Requirements:

1 Data Acquisition:

- Download the Car Evaluation Dataset from the provided link or Kaggle.

2. Data Exploration:

- Load the dataset using tools like pandas.
- Conduct a preliminary examination using methods like head(), describe(), and info().
- Identify any missing values or anomalies that need addressing.

3. Data Preprocessing:

- **Handling Missing Values**: If any, use techniques like imputation.
- **Encoding Categorical Variables**: Since Random Forest requires numerical input, use methods like Label Encoding or One-Hot Encoding.
- **Feature Scaling**: Although not always necessary for Random Forest, scaling (e.g., Min-Max Scaling, Standard Scaling) can be applied for consistent feature representation.
- **Train-Test Split**: Partition the data into training and test sets to evaluate the model's performance.

4. Model Building:

- Initialize the Random Forest Classifier using scikit-learn.
- Train the classifier on the training data.

5. Model Evaluation:

- Use the trained model to predict the safety of cars in the test set.
- Calculate and evaluate metrics such as accuracy, precision, recall, and the F1 score.
- (Optional) Visualize the results using a confusion matrix, ROC curve, etc.

6. Feature Importance Evaluation:

- Extract feature importance scores from the trained Random Forest model to understand which features have the most influence on predictions.

7. Hyperparameter Tuning (Optional but Recommended):

- Use tools like GridSearchCV or RandomizedSearchCV to find the best hyperparameters for the Random Forest Classifier, optimizing for better performance.

8. Model Refinement (Based on Evaluation and Hyperparameter Tuning):

- If the initial performance is unsatisfactory, refine the model. This can involve:
    a. Adjusting model hyperparameters.
    b. Incorporating more features or removing irrelevant ones.
    c. Trying advanced preprocessing techniques.

Algorithm:

Random Forest Classifier:

1. Input:

- Dataset D of size N.
- The number T of trees to be generated.

- The size m of the random subset of features to be considered at each split (usually $\sqrt{total^2 - feautures^2}$ for classification).
- Optional: Size n for the subsample of the training data, with or without replacement (if not provided, then n = N).

2. Output: A list of 'T' decision trees.

3. Procedure:

For i = 1 to T do:

- Create a bootstrap sample 'D_i' of size n by randomly selecting n samples from 'D' with replacement.
- Grow a decision tree 'Tree_i' on 'D_i' as follows:
    a. For each node of the tree, randomly select 'm' features without replacement.
    b. Split the node using the feature that provides the best split (according to some criterion, often Gini impurity or entropy for classification).
    c. Recurse for the split subsets until the maximum depth is reached, or another stopping criterion is met (e.g., minimum samples per leaf).
- Add 'Tree_i' to the list of trees.

4. Prediction: Given a new sample 'S':

a. For each tree Tree_i in the list of trees, get a prediction 'P_i'.
b. For classification, the final prediction is the mode of all the predictions {'P_1, P_2, ... P_T'}. For regression, it would be the mean.

Conclusion:

We have learned about Random Forest Classifier, being an ensemble method, provides more robust predictions than a single decision tree. Its capability to determine feature importance helps in understanding which features most influence predictions.

Oral Questions:

1. What is the primary reason to use Random Forest over a single decision tree?
2. How does increasing the number of trees in a Random Forest impact its performance?
3. Explain the concept of bootstrap sampling.
4. Why is feature randomness introduced during the tree-building process in Random Forest?