NAME : Mehul Uttam
PRN : 1032222936
TY CSE AIDS - A (A1 Batch)

ML Assignment-5

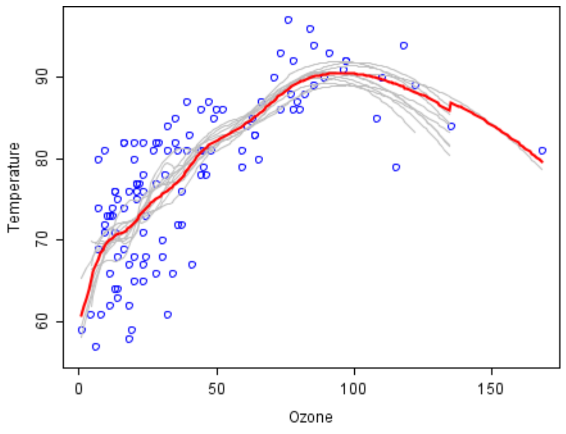
Title: Ensemble, Random Forest Classifier and Performance Measurement
Aim: To perform Ensemble (Bagging Boosting and Stacking), Random Forest classification and performance evaluation using Python
Objectives: To implement Ensemble (Bagging Boosting and Stacking), Random Forest Classifier and perform performance evaluation

Theory:

1. Ensemble Learning

The main principle behind ensemble methods is that a group of "weak learners/classifiers" can come together to form a "strong learner/classifier".

For example, given below is an analysis on the relationship between ozone and temperature.



The blue circles are the data points. Assumption is that they do model some function.

In the graph, each grey curve (weak learner) is an individual learner modelled on the data points, and is a fair approximation to the data. The red curve (the ensemble strong learner) is a combination of the weak learners. In contrast, it gives a much better approximation to the data.

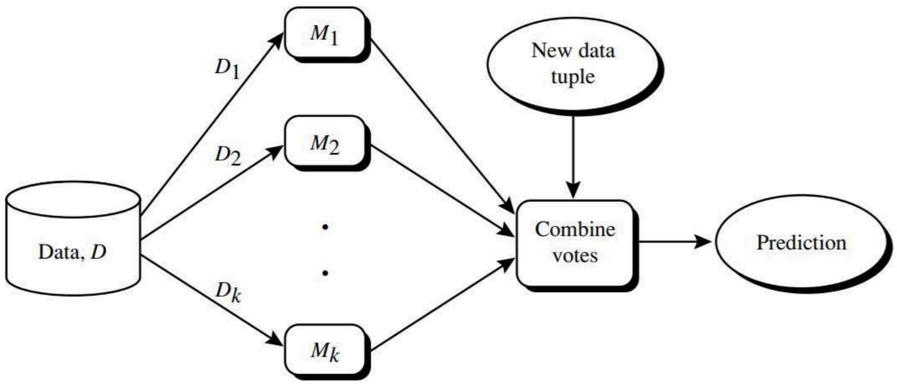
Hence, an ensemble method for classification is a composite model, made up of combination of classifiers.

<u>Formal Definition:</u>

An ensemble combines a series of *k* learned models (or *base classifiers*), *M1, M2,..., Mk* , with the aim of creating an improved composite classification model, M∗.

A given data set, D, is used to create k training sets, D*1, D2,..., Dk* , where Di (1 ≤ i ≤ k − 1) is used to generate classifier Mi .

Given a new data tuple to classify, the base classifiers each vote by returning a class prediction. The ensemble returns a class prediction based on the votes of the base classifiers. An ensemble tends to be more accurate than its base classifiers.



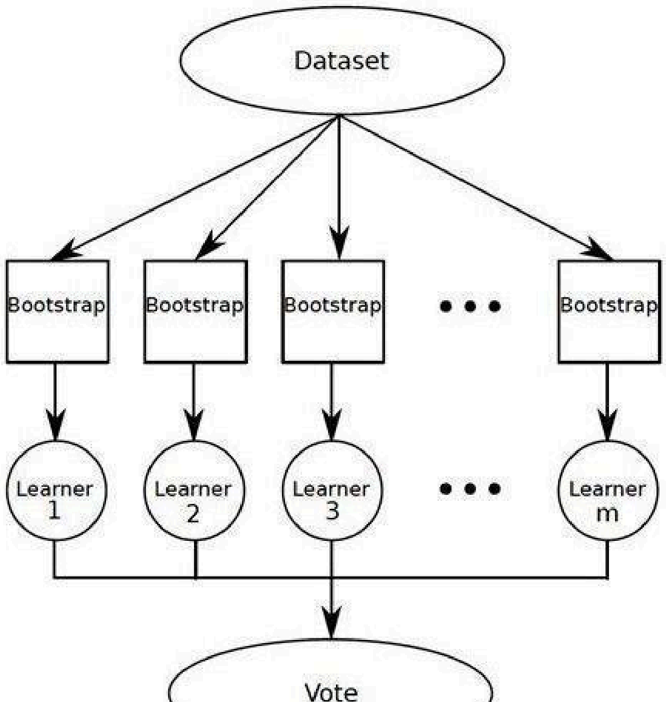
2. Bagging

Bootstrap Aggregation or bagging involves taking multiple samples from your training dataset (with replacement) and training a model for each sample.

The final output prediction is averaged across the predictions of all of the sub-models.

The three bagging models covered in this section are as follows:

1. Bagged Decision Trees
2. Random Forest



Note: Bagging is more robust to noisy data and less prone to overfitting.

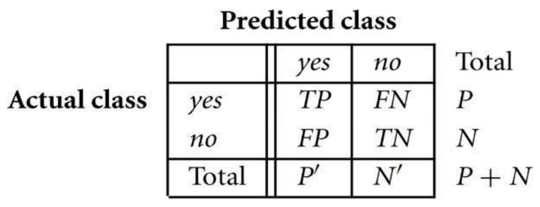
2.Random Forests

Random Forest Models can be thought of as **BAGG**ing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor.

3. Performance Metrics

a. Confusion Matrix

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes.



		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	*P′*	*N′*	*P* + *N*

Explanation of the terms associated with confusion matrix:

- **True Positives (TP)** – positive tuples correctly labelled by classifier
- **True Negatives (TN)** – negative tuples correctly labelled “negative” by classifier.
- **False Positives (FP)** – negative tuples mislabelled “positive” by classifier.
- **False Negatives (FN)** –positive tuples mislabelled “negative” by classifier.

We have used confusion_matrix function of sklearn.metrics to compute Confusion Matrix of ensemble classification model.

b. Accuracy

It is most common performance metric for classification algorithms. Defined as the ratio of number of correct predictions made to all predictions made.

Steps in Preprocessing of Data

1. Read the .csv file of dataset

2. Display few observations

3. Perform data preprocessing (handling missing data, etc)

4. Create the independent and dependent variables

5. Standardization of data

6. Split the data into training and test sets.

7. Training Decision tree classifier and RF classifier

8. Also apply bagging and boosting algorithm.

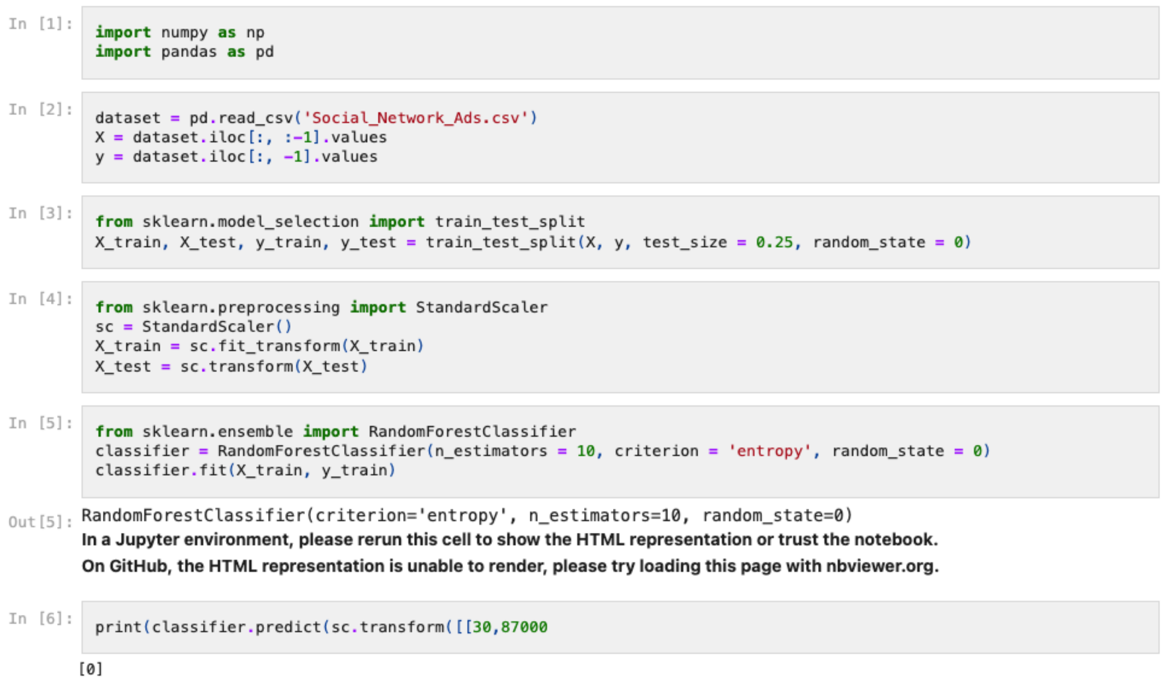
9. Fit the data in model to train it.

10. Try to plot the decision boundary for training set and testing set.

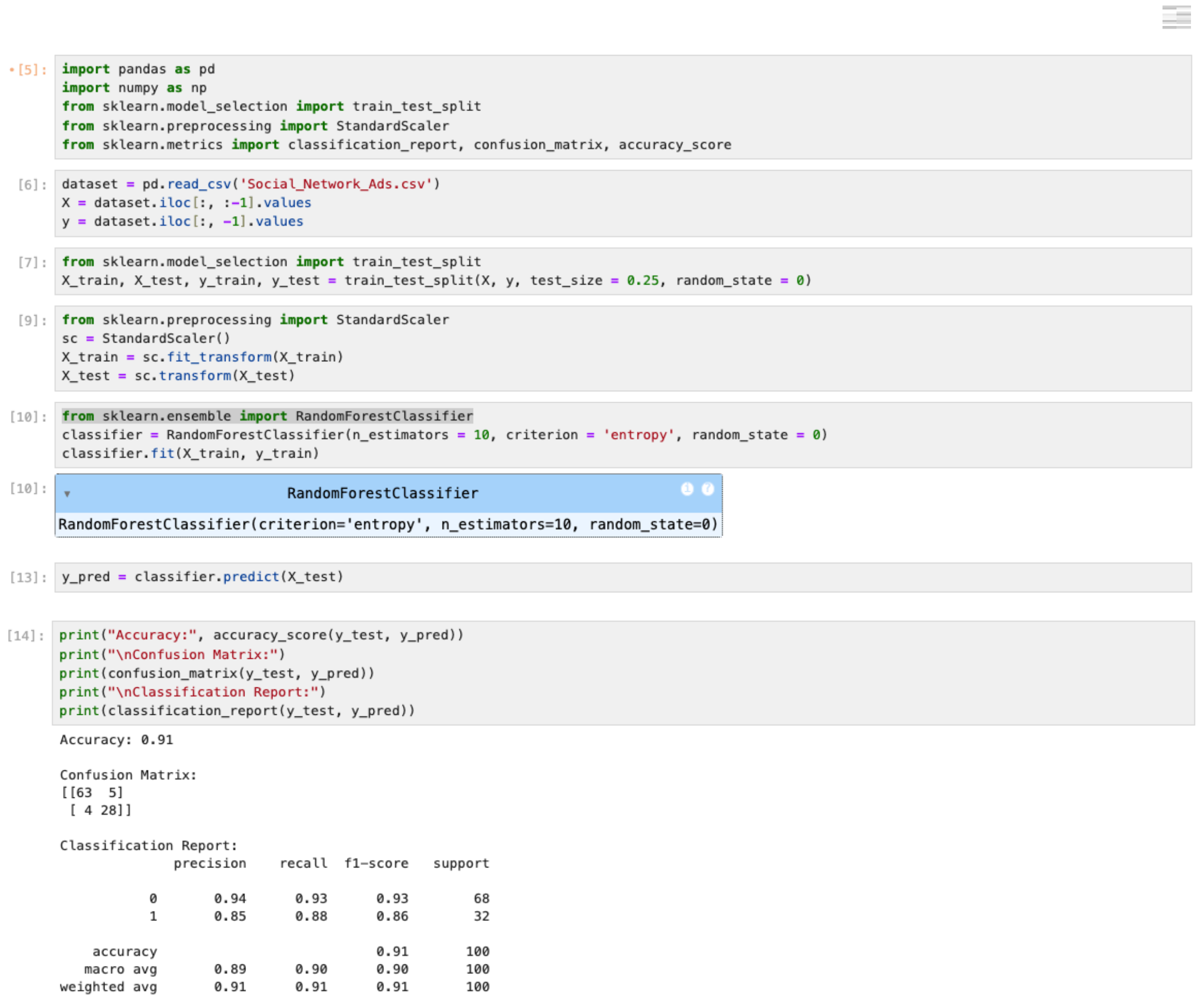
Input: https://github.com/psvm-tallman/ML-LAB/blob/main/Social_Network_Ads.csv

Platform: Jupyter Notebook

Output:



```
In [1]:  import numpy as np
         import pandas as pd

In [2]:  dataset = pd.read_csv('Social_Network_Ads.csv')
         X = dataset.iloc[:, :-1].values
         y = dataset.iloc[:, -1].values

In [3]:  from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

In [4]:  from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)

In [5]:  from sklearn.ensemble import RandomForestClassifier
         classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
         classifier.fit(X_train, y_train)

Out[5]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
        In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
        On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [6]:  print(classifier.predict(sc.transform([[30,87000

        [0]
```



```
[5]:  import pandas as pd
      import numpy as np
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

[6]:  dataset = pd.read_csv('Social_Network_Ads.csv')
      X = dataset.iloc[:, :-1].values
      y = dataset.iloc[:, -1].values

[7]:  from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

[9]:  from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X_train = sc.fit_transform(X_train)
      X_test = sc.transform(X_test)

[10]: from sklearn.ensemble import RandomForestClassifier
      classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
      classifier.fit(X_train, y_train)

[10]: RandomForestClassifier
      RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

[13]: y_pred = classifier.predict(X_test)

[14]: print("Accuracy:", accuracy_score(y_test, y_pred))
      print("\nConfusion Matrix:")
      print(confusion_matrix(y_test, y_pred))
      print("\nClassification Report:")
      print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.91

Confusion Matrix:
[[63  5]
 [ 4 28]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.93        68
           1       0.85      0.88      0.86        32

    accuracy                           0.91       100
   macro avg       0.89      0.90      0.90       100
weighted avg       0.91      0.91      0.91       100
```

Conclusion: Implementation of Random forest algorithm is done also accuracy is checked with the help of performance metric

FAQS:

1. What is an ensemble model?

- An **ensemble model** is a machine learning technique that combines the predictions of multiple base models (weak learners) to produce a more accurate and robust prediction. The idea is that by leveraging the strengths of different models and reducing their individual weaknesses, the ensemble performs better than any single model. Common ensemble techniques include bagging, boosting, and stacking.

2. What are bagging, boosting, and stacking?

- **Bagging (Bootstrap Aggregating)**:
  - Bagging is an ensemble method where multiple models are trained on different random subsets of the training data (generated using bootstrapping), and their predictions are aggregated (typically using majority voting or averaging).

    - **Example**: Random Forest is a popular bagging algorithm where decision trees are built on bootstrapped data.
    - **Advantage**: Reduces variance, making the model more robust and less prone to overfitting.

- **Boosting**:
    - Boosting is an iterative technique that trains models sequentially, with each model correcting the errors of its predecessor. The models are weighted based on their performance, and the final prediction is a weighted combination of all models.
    - **Example**: AdaBoost, Gradient Boosting, XGBoost.
    - **Advantage**: Focuses on improving weak learners and usually reduces bias, leading to better accuracy.

- **Stacking**:
    - Stacking involves training multiple models (base learners) and then using their predictions as inputs to a higher-level model (meta-learner), which makes the final prediction. The meta-learner aims to combine the strengths of the base learners.
    - **Example**: Logistic regression, SVM, or any other model can act as the meta-learner.
    - **Advantage**: Can combine different types of models and learn optimal ways to blend them for improved performance.

3. What are the benefits of ensemble models?

- **Increased Accuracy**: Ensemble methods typically outperform individual models, providing more accurate predictions.
- **Reduced Variance**: Bagging-based ensembles, like Random Forest, reduce variance, making the model more stable and robust to fluctuations in the data.
- **Reduced Bias**: Boosting methods focus on correcting the errors of weak models, reducing bias and improving performance.
- **Better Generalization**: Combining multiple models improves the generalization of the model to unseen data, reducing overfitting.
- **Flexibility**: Ensembles can combine models of different types, benefiting from the diverse strengths of various algorithms.