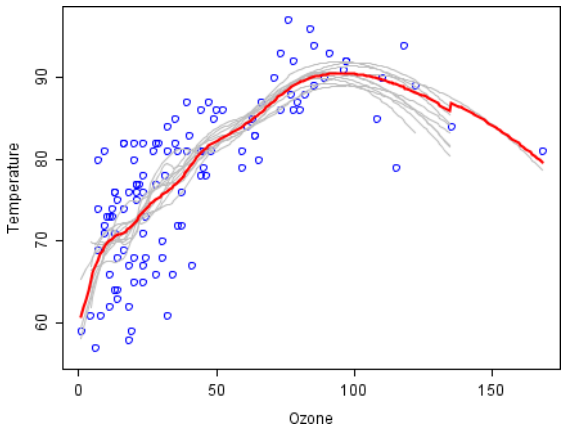
Title: Ensemble, Random Forest Classifier and Performance Measurement
Aim: To perform Ensemble (Voting), Random Forest classification and performance evaluation using Python
Objectives: To implement Ensemble (Voting), Random Forest Classifier and perform performance evaluation

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.

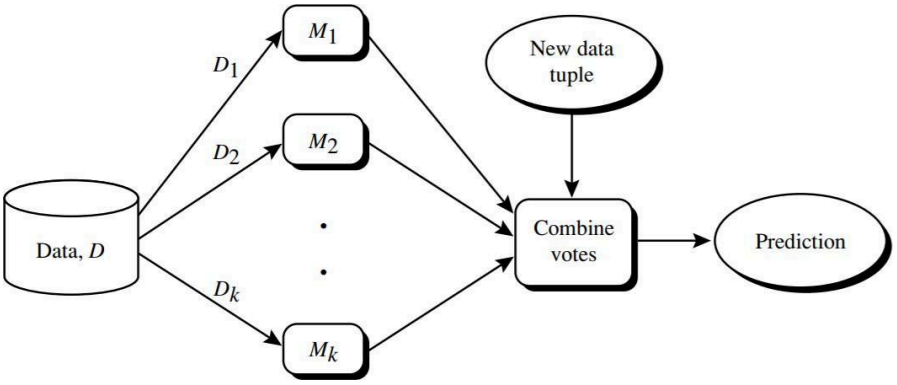
Hence, an ensemble method for classification is a composite model, made up of combination of classifiers.

Formal Definition:

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 ($1 \leq i \leq k - 1$) 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.



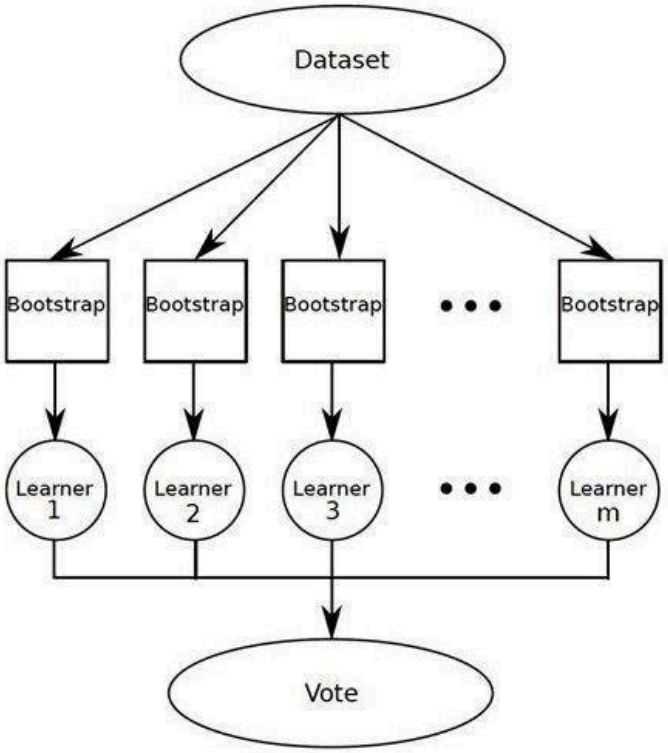
2. Bagging

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: Dataset

Platform:

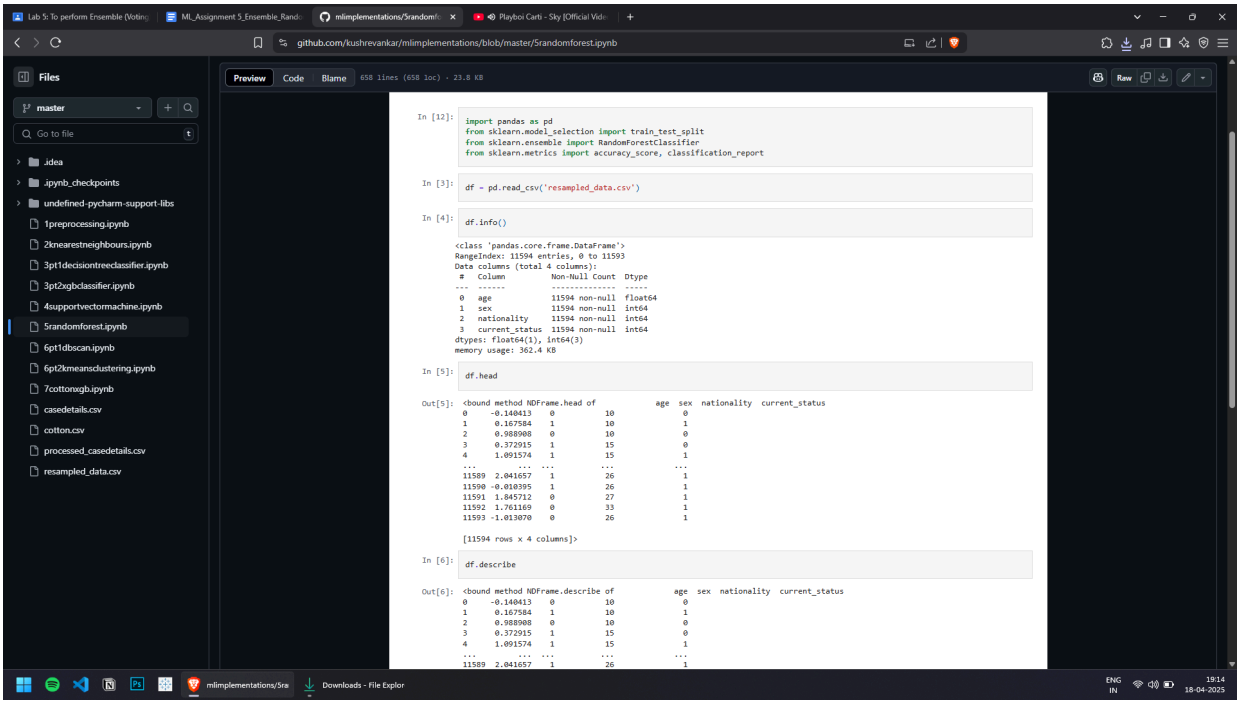
Output:

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

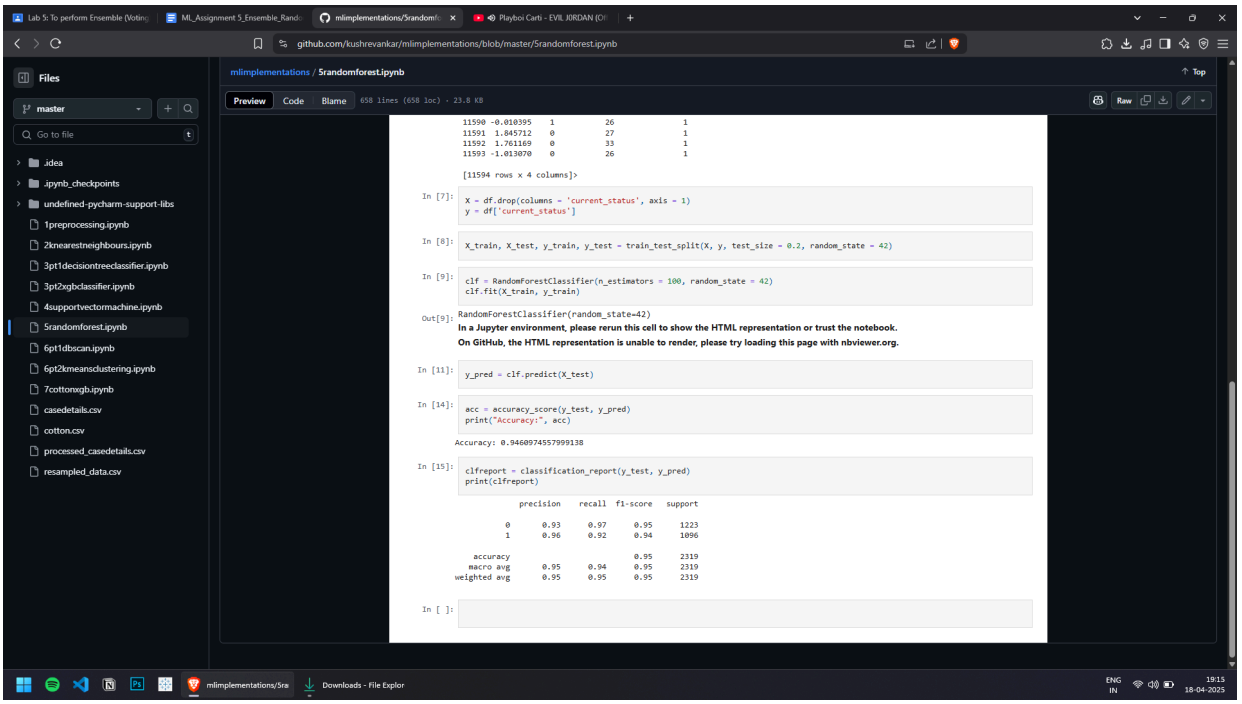
FAQS:

1. What is an ensemble model?
2. What are bagging, boosting and stacking?
3. Can we ensemble multiple models of same ML algorithm?
4. How can we identify the weights of different models?
5. What are the benefits of ensemble model?

Code & Output:



```
In [12]:
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

In [3]:
df = pd.read_csv('resampled_data.csv')

In [4]:
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11594 entries, 0 to 11593
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             11594 non-null  float64
 1   sex             11594 non-null  int64
 2   nationality     11594 non-null  int64
 3   current_status  11594 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 362.4 KB

In [5]:
df.head

Out[5]:
<bound method NDFrame.head of             age  sex  nationality  current_status
0     -0.140413    0           10              0
1      0.167584    1           10              1
2      0.988908    0           10              0
3      0.372915    1           15              0
4      1.091574    1           15              1
...         ...  ...          ...            ...
11589  2.041657    1           26              1
11590 -0.010395    1           26              1
11591  1.845712    0           27              1
11592  1.761169    0           33              1
11593 -1.013070    0           26              1

[11594 rows x 4 columns]>

In [6]:
df.describe

Out[6]:
<bound method NDFrame.describe of             age  sex  nationality  current_status
0     -0.140413    0           10              0
1      0.167584    1           10              1
2      0.988908    0           10              0
3      0.372915    1           15              0
4      1.091574    1           15              1
...         ...  ...          ...            ...
11589  2.041657    1           26              1
```



```
11590 -0.010395    1           26              1
11591  1.845712    0           27              1
11592  1.761169    0           33              1
11593 -1.013070    0           26              1

[11594 rows x 4 columns]>

In [7]:
X = df.drop(columns = 'current_status', axis = 1)
y = df['current_status']

In [8]:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

In [9]:
clf = RandomForestClassifier(n_estimators = 100, random_state = 42)
clf.fit(X_train, y_train)

Out[9]:
RandomForestClassifier(random_state=42)
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 [11]:
y_pred = clf.predict(X_test)

In [14]:
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

Accuracy: 0.9460974557999138

In [15]:
clfreport = classification_report(y_test, y_pred)
print(clfreport)

              precision    recall  f1-score   support

           0       0.93      0.97      0.95      1223
           1       0.96      0.92      0.94      1096

    accuracy                           0.95      2319
   macro avg       0.95      0.94      0.95      2319
weighted avg       0.95      0.95      0.95      2319

In [ ]:
```