

**EXPERIMENT NO.**

Aim of the experiment: - To implement Ensemble Learning (Bagging/Boosting).

Course Outcome: - To implement ensemble techniques to combine predictions

Date of Conduction: - 23/08/2022

Date of Submission: - 20/09/2022

03

From different models.

Implementation

(05)

Understanding

(05)

Punctuality and Discipline

(05)

Total Marks (15)

Practical In charge

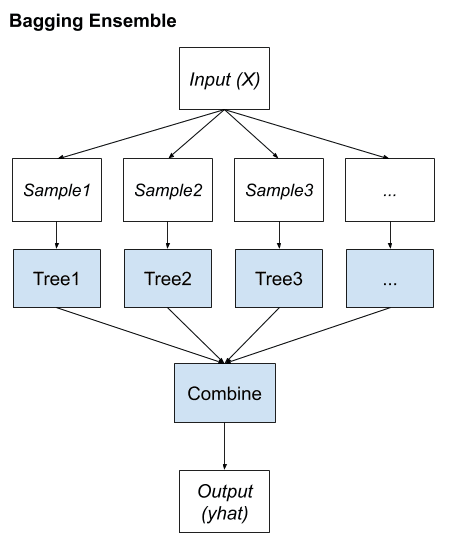
Theory:

Ensemble learning is a general meta-approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

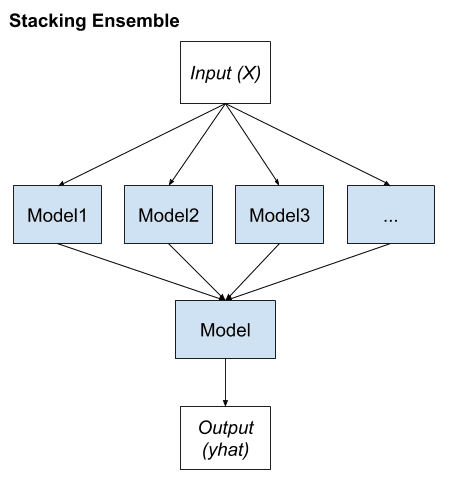
Although there are a seemingly unlimited number of ensembles that you can develop for your predictive modeling problem, there are three methods that dominate the field of ensemble learning. So much so, that rather than algorithms per se, each is a field of study that has spawned many more specialized methods.

The three main classes of ensemble learning methods are bagging, stacking, and boosting, and it is important to both have a detailed understanding of each method and to consider them on your predictive modeling project.

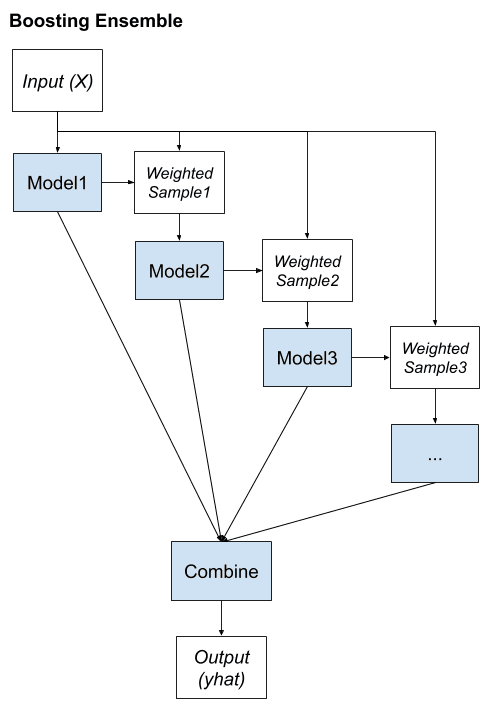
* Bagging involves fitting many decision trees on different samples of the same dataset and averaging the predictions.



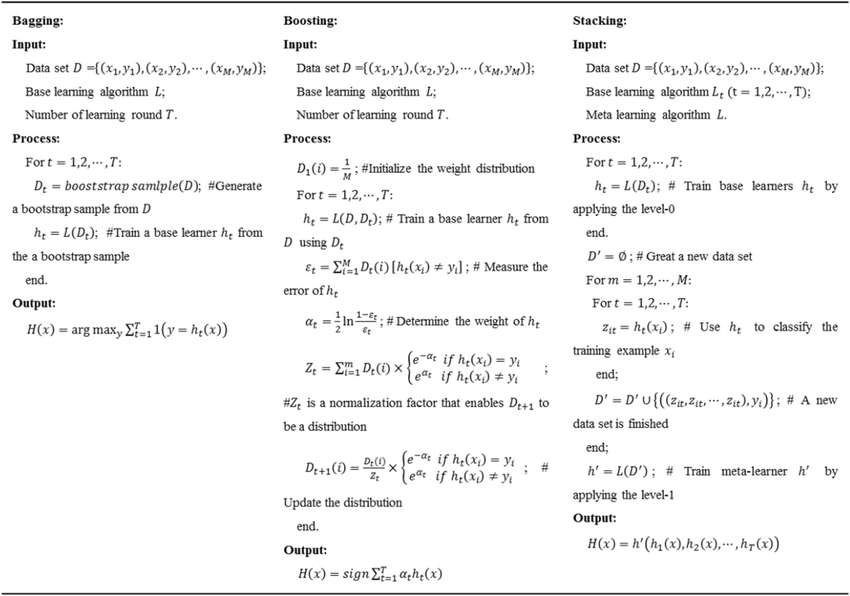
* Stacking involves fitting many different models types on the same data and using another model to learn how to best combine the predictions.



* Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.



**Algorithm:**



**Program:**

import pandas as pd

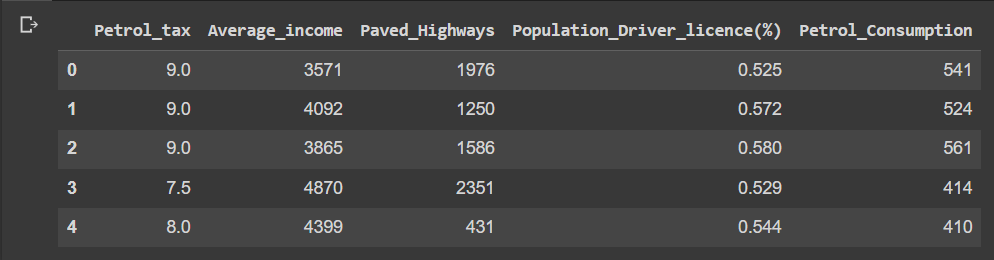
import numpy as np

import matplotlib.pyplot as plt

dataset = pd.read\_csv('petrol\_consumption.csv')

dataset.head()

**Output for dataset.head():**



X = dataset.iloc[:, 0:4].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators=20, random\_state=0)

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

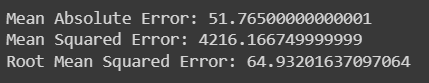
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

**Output:**

****

**Conclusion:**

Thus, we have successfully implemented random forest regression.

**Notebook Link:**

<https://colab.research.google.com/drive/1in1hokQeoiohUq7gareRx77nt_aehKHv#scrollTo=AXzdk4_9tPeG>