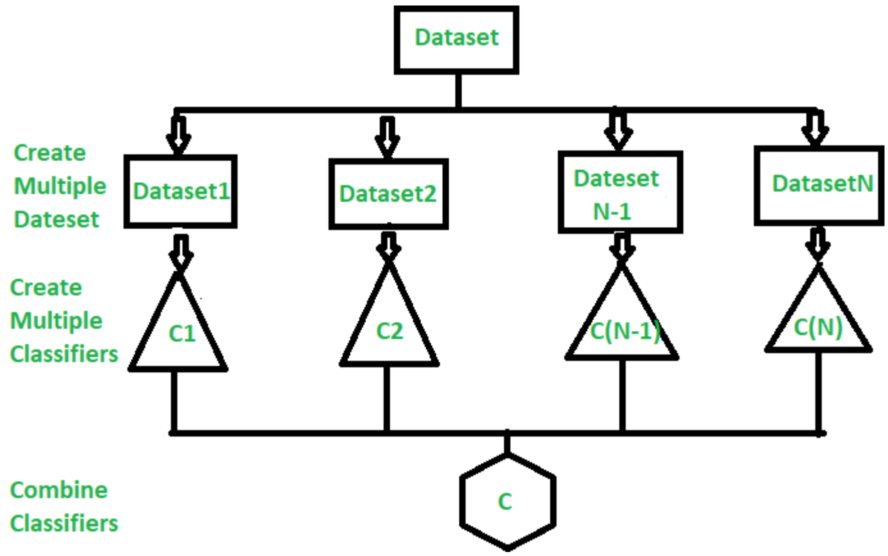
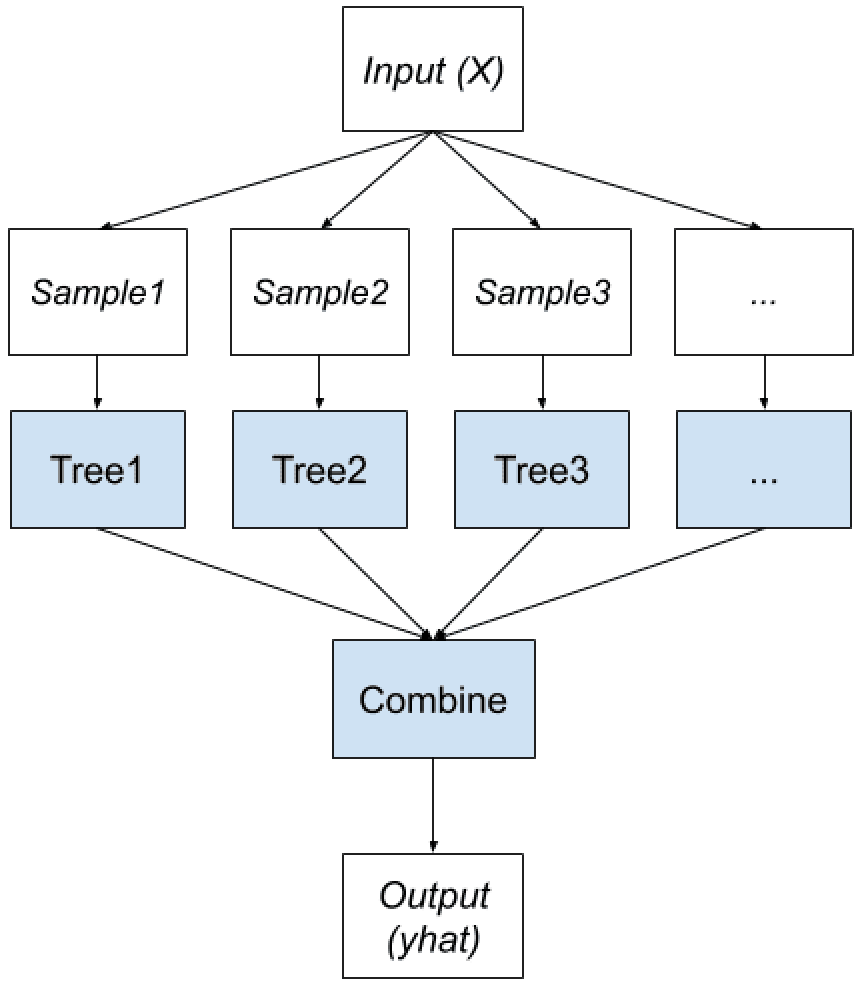
ML Experiment 5

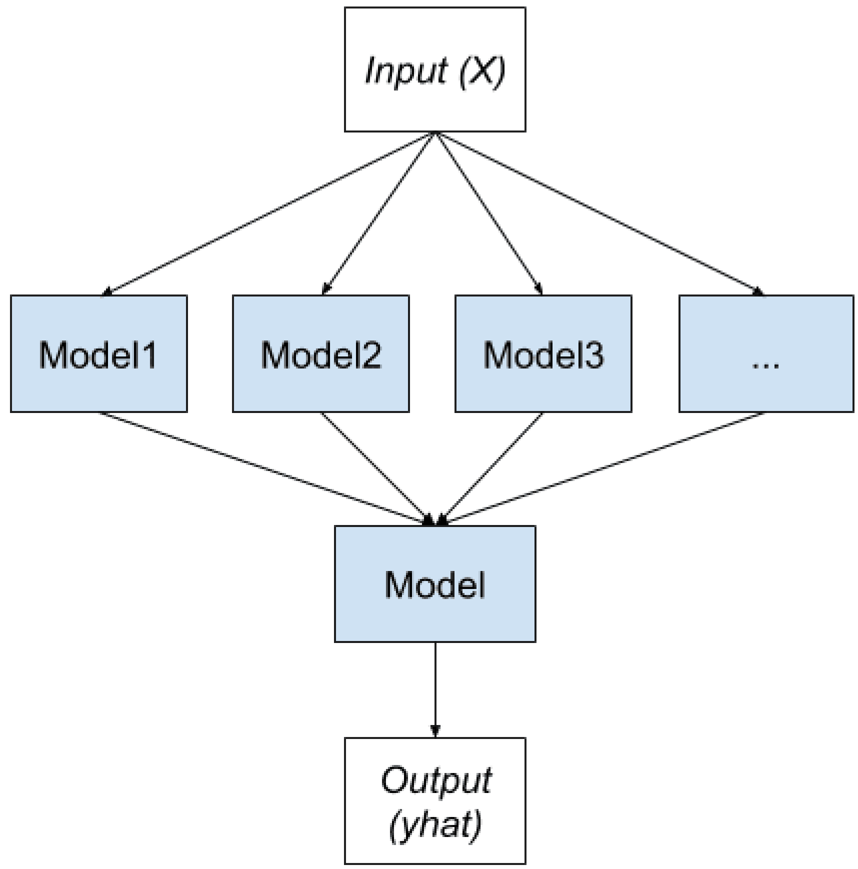
Program on Ensemble Learning

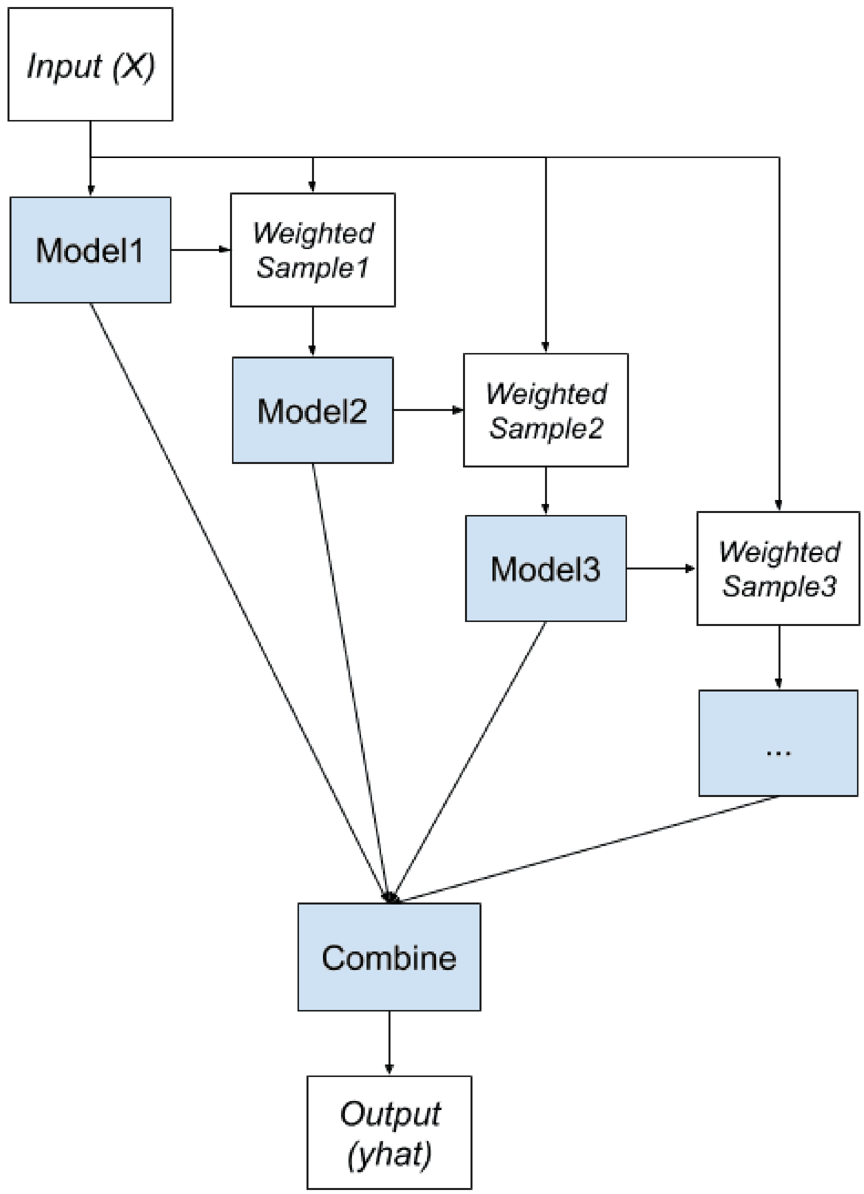
Name:	Pratik Daga
Roll No:	11
Learning Objective:	Implement the Ensemble Learning technique with appropriate data set and application
Learning Outcome:	Student are able to successfully implement Ensemble Learning
Course Outcome:	CSL701.4
Program Outcome:	(PO 3) Design/ development of solutions: Breadth and uniqueness of engineering problems i.e. the extent to which problems are original and to which solutions have previously been identified or codified (PO 12) Life Long Learning
Bloom's Taxonomy Level:	Analysis,Create
Theory:	Ensemble learning involves the creation and combination of multiple individual models, often referred to as base learners or weak learners, to form a stronger, more accurate final prediction. The idea is to leverage the complementary strengths of these models, leading to better overall performance. The key principle behind ensemble learning is the diversity among the base learners. Diversity can stem from differences in algorithms, training data subsets, or even initializations. These diverse models collectively tackle various aspects of the problem, capturing nuances that a single model might miss.

	 There are several popular ensemble techniques, each with its own way of combining the predictions of base learners: 1. Bagging (Bootstrap Aggregating): This technique involves training multiple instances of the same model on different subsets of the training data, typically obtained through bootstrapping. The final prediction is then averaged or majority-voted. 2. Boosting: Boosting focuses on iteratively improving the performance of base models. Weak learners are trained sequentially, and each subsequent learner gives more weight to the misclassified instances from previous learners. 3. Random Forests: A variation of bagging, random forests combine multiple decision trees by training them on bootstrapped samples and considering only a subset of features at each node split. The final prediction is an average or majority vote of the individual tree predictions. 4. Stacking: Stacking combines predictions from different models through a meta-model (often a simple linear regression or another machine learning algorithm). The base learners' outputs serve as features for the meta-model's training.

Algorithm :	**Bagging Ensemble** 

Stacking Ensemble



	Boosting Ensemble 

Data Set :	`fixed acidity,volatile acidity,citric acid,residual sugar,chlorides,free sulfur dioxide,total sulfur dioxide,density,pH,sulphates,alcohol,quality`
	`7.5,0.7,0,1.9,0.076,11,34,0.9978,3.51,0.56,9.4,5`
	`7.8,0.88,0,2.6,0.098,25,67,0.9968,3.2,0.68,9.8,5`
	`7.8,0.76,0.04,2.3,0.092,15,54,0.997,3.26,0.65,9.8,5`
	`11.2,0.28,0.56,1.9,0.075,17,60,0.998,3.16,0.58,9.8,6`
	`7.4,0.7,0,1.9,0.076,11,34,0.9978,3.51,0.56,9.4,5`
	`7.4,0.66,0,1.8,0.075,13,40,0.9978,3.51,0.56,9.4,5`

	7.9,0.6,0.06,1.6,0.069,15,59,0.9964,3.3,0.46,9.4,5 7.3,0.65,0,1.2,0.065,15,21,0.9946,3.39,0.47,10,7 7.8,0.58,0.02,2,0.073,9,18,0.9968,3.36,0.57,9.5,7 7.5,0.5,0.36,6.1,0.071,17,102,0.9978,3.35,0.8,10.5,5 6.7,0.58,0.08,1.8,0.097,15,65,0.9959,3.28,0.54,9.2,5 7.5,0.5,0.36,6.1,0.071,17,102,0.9978,3.35,0.8,10.5,5 5.6,0.615,0,1.6,0.089,16,59,0.9943,3.58,0.52,9.9,5 7.8,0.61,0.29,1.6,0.114,9,29,0.9974,3.26,1.56,9.1,5 8.9,0.62,0.18,3.8,0.176,52,145,0.9986,3.16,0.88,9.2,5 8.9,0.62,0.19,3.9,0.17,51,148,0.9986,3.17,0.93,9.2,5 8.5,0.28,0.56,1.8,0.092,35,103,0.9969,3.3,0.75,10.5,7 8.1,0.56,0.28,1.7,0.368,16,56,0.9968,3.11,1.28,9.3,5 7.4,0.59,0.08,4.4,0.086,6,29,0.9974,3.38,0.5,9,4 7.9,0.32,0.51,1.8,0.341,17,56,0.9969,3.04,1.08,9.2,6 8.9,0.22,0.48,1.8,0.077,29,60,0.9968,3.39,0.53,9.4,6 7.6,0.39,0.31,2.3,0.082,23,71,0.9982,3.52,0.65,9.7,5 7.9,0.43,0.21,1.6,0.106,10,37,0.9966,3.17,0.91,9.5,5 8.5,0.49,0.11,2.3,0.084,9,67,0.9968,3.17,0.53,9.4,5 6.9,0.4,0.14,2.4,0.085,21,40,0.9968,3.43,0.63,9.7,6 6.3,0.39,0.16,1.4,0.08,11,23,0.9955,3.34,0.56,9.3,5 7.6,0.41,0.24,1.8,0.08,4,11,0.9962,3.28,0.59,9.5,5 7.9,0.43,0.21,1.6,0.106,10,37,0.9966,3.17,0.91,9.5,5 7.1,0.71,0,1.9,0.08,14,35,0.9972,3.47,0.55,9.4,5 7.8,0.645,0,2,0.082,8,16,0.9964,3.38,0.59,9.8,6 6.7,0.675,0.07,2.4,0.089,17,82,0.9958,3.35,0.54,10.1,5 6.9,0.685,0,2.5,0.105,22,37,0.9966,3.46,0.57,10.6,6 8.3,0.655,0.12,2.3,0.083,15,113,0.9966,3.17,0.66,9.8,5 6.9,0.605,0.12,10.7,0.073,40,83,0.9993,3.45,0.52,9.4,6 5.2,0.32,0.25,1.8,0.103,13,50,0.9957,3.38,0.55,9.2,5 7.8,0.645,0,5.5,0.086,5,18,0.9986,3.4,0.55,9.6,6 7.8,0.6,0.14,2.4,0.086,3,15,0.9975,3.42,0.6,10.8,6 8.1,0.38,0.28,2.1,0.066,13,30,0.9968,3.23,0.73,9.7,7 5.7,1.13,0.09,1.5,0.172,7,19,0.994,3.5,0.48,9.8,4 7.3,0.45,0.36,5.9,0.074,12,87,0.9978,3.33,0.83,10.5,5 7.3,0.45,0.36,5.9,0.074,12,87,0.9978,3.33,0.83,10.5,5 8.8,0.61,0.3,2.8,0.088,17,46,0.9976,3.26,0.51,9.3,4 7.5,0.49,0.2,2.6,0.332,8,14,0.9968,3.21,0.9,10.5,6 8.1,0.66,0.22,2.2,0.069,9,23,0.9968,3.3,1.2,10.3,5 6.8,0.67,0.02,1.8,0.05,5,11,0.9962,3.48,0.52,9.5,5 4.6,0.52,0.15,2.1,0.054,8,65,0.9934,3.9,0.56,13.1,4 7.7,0.935,0.43,2.2,0.114,22,114,0.997,3.25,0.73,9.2,5 8.7,0.29,0.52,1.6,0.113,12,37,0.9969,3.25,0.58,9.5,5 6.4,0.4,0.23,1.6,0.066,5,12,0.9958,3.34,0.56,9.2,5 5.6,0.31,0.37,1.4,0.074,12,96,0.9954,3.32,0.58,9.2,5 8.8,0.66,0.26,1.7,0.074,4,23,0.9971,3.15,0.74,9.2,5 6.6,0.52,0.04,2.2,0.069,8,15,0.9956,3.4,0.63,9.4,6 6.6,0.5,0.04,2.1,0.068,6,14,0.9955,3.39,0.64,9.4,6 8.6,0.38,0.36,3,0.081,30,119,0.997,3.2,0.56,9.4,5

	7.6,0.51,0.15,2.8,0.11,33,73,0.9955,3.17,0.63,10.2,6 7.7,0.62,0.04,3.8,0.084,25,45,0.9978,3.34,0.53,9.5,5 10.2,0.42,0.57,3.4,0.07,4,10,0.9971,3.04,0.63,9.6,5 7.5,0.63,0.12,5.1,0.111,50,110,0.9983,3.26,0.77,9.4,5 7.8,0.59,0.18,2.3,0.076,17,54,0.9975,3.43,0.59,10,5 7.3,0.39,0.31,2.4,0.074,9,46,0.9962,3.41,0.54,9.4,6 8.8,0.4,0.4,2.2,0.079,19,52,0.998,3.44,0.64,9.2,5 7.7,0.69,0.49,1.8,0.115,20,112,0.9968,3.21,0.71,9.3,5 7.5,0.52,0.16,1.9,0.085,12,35,0.9968,3.38,0.62,9.5,7 7,0.735,0.05,2,0.081,13,54,0.9966,3.39,0.57,9.8,5 7.2,0.725,0.05,4.65,0.086,4,11,0.9962,3.41,0.39,10.9,5 7.2,0.725,0.05,4.65,0.086,4,11,0.9962,3.41,0.39,10.9,5 7.5,0.52,0.11,1.5,0.079,11,39,0.9968,3.42,0.58,9.6,5 6.6,0.705,0.07,1.6,0.076,6,15,0.9962,3.44,0.58,10.7,5 9.3,0.32,0.57,2,0.074,27,65,0.9969,3.28,0.79,10.7,5 8,0.705,0.05,1.9,0.074,8,19,0.9962,3.34,0.95,10.5,6 7.7,0.63,0.08,1.9,0.076,15,27,0.9967,3.32,0.54,9.5,6 7.7,0.67,0.23,2.1,0.088,17,96,0.9962,3.32,0.48,9.5,5 7.7,0.69,0.22,1.9,0.084,18,94,0.9961,3.31,0.48,9.5,5 8.3,0.675,0.26,2.1,0.084,11,43,0.9976,3.31,0.53,9.2,4 9.7,0.32,0.54,2.5,0.094,28,83,0.9984,3.28,0.82,9.6,5 8.8,0.41,0.64,2.2,0.093,9,42,0.9986,3.54,0.66,10.5,5 8.8,0.41,0.64,2.2,0.093,9,42,0.9986,3.54,0.66,10.5,5 6.8,0.785,0,2.4,0.104,14,30,0.9966,3.52,0.55,10.7,6 6.7,0.75,0.12,2,0.086,12,80,0.9958,3.38,0.52,10.1,5 8.3,0.625,0.2,1.5,0.08,27,119,0.9972,3.16,1.12,9.1,4 6.2,0.45,0.2,1.6,0.069,3,15,0.9958,3.41,0.56,9.2,5 7.8,0.43,0.7,1.9,0.464,22,67,0.9974,3.13,1.28,9.4,5 7.4,0.5,0.47,2,0.086,21,73,0.997,3.36,0.57,9.1,5 7.3,0.67,0.26,1.8,0.401,16,51,0.9969,3.16,1.14,9.4,5 6.3,0.3,0.48,1.8,0.069,18,61,0.9959,3.44,0.78,10.3,6 6.9,0.55,0.15,2.2,0.076,19,40,0.9961,3.41,0.59,10.1,5 8.6,0.49,0.28,1.9,0.11,20,136,0.9972,2.93,1.95,9.9,6 7.7,0.49,0.26,1.9,0.062,9,31,0.9966,3.39,0.64,9.6,5 9.3,0.39,0.44,2.1,0.107,34,125,0.9978,3.14,1.22,9.5,5 7,0.62,0.08,1.8,0.076,8,24,0.9978,3.48,0.53,9,5 7.9,0.52,0.26,1.9,0.079,42,140,0.9964,3.23,0.54,9.5,5 8.6,0.49,0.28,1.9,0.11,20,136,0.9972,2.93,1.95,9.9,6 8.6,0.49,0.29,2,0.11,19,133,0.9972,2.93,1.98,9.8,5 7.7,0.49,0.26,1.9,0.062,9,31,0.9966,3.39,0.64,9.6,5 5,1.02,0.04,1.4,0.045,41,85,0.9938,3.75,0.48,10.5,4 4.7,0.6,0.17,2.3,0.058,17,106,0.9932,3.85,0.6,12.9,6 6.8,0.775,0,3,0.102,8,23,0.9965,3.45,0.56,10.7,5 7,0.5,0.25,2,0.07,3,22,0.9963,3.25,0.63,9.2,5 7.6,0.9,0.06,2.5,0.079,5,10,0.9967,3.39,0.56,9.8,5 8.1,0.545,0.18,1.9,0.08,13,35,0.9972,3.3,0.59,9,6 8.3,0.61,0.3,2.1,0.084,11,50,0.9972,3.4,0.61,10.2,6 7.8,0.5,0.3,1.9,0.075,8,22,0.9959,3.31,0.56,10.4,6

	8.1,0.545,0.18,1.9,0.08,13,35,0.9972,3.3,0.59,9,6 8.1,0.575,0.22,2.1,0.077,12,65,0.9967,3.29,0.51,9.2,5 7.2,0.49,0.24,2.2,0.07,5,36,0.996,3.33,0.48,9.4,5 8.1,0.575,0.22,2.1,0.077,12,65,0.9967,3.29,0.51,9.2,5 7.8,0.41,0.68,1.7,0.467,18,69,0.9973,3.08,1.31,9.3,5 6.2,0.63,0.31,1.7,0.088,15,64,0.9969,3.46,0.79,9.3,5 8,0.33,0.53,2.5,0.091,18,80,0.9976,3.37,0.8,9.6,6 8.1,0.785,0.52,2,0.122,37,153,0.9969,3.21,0.69,9.3,5 7.8,0.56,0.19,1.8,0.104,12,47,0.9964,3.19,0.93,9.5,5 8.4,0.62,0.09,2.2,0.084,11,108,0.9964,3.15,0.66,9.8,5 8.4,0.6,0.1,2.2,0.085,14,111,0.9964,3.15,0.66,9.8,5 10.1,0.31,0.44,2.3,0.08,22,46,0.9988,3.32,0.67,9.7,6 7.8,0.56,0.19,1.8,0.104,12,47,0.9964,3.19,0.93,9.5,5 9.4,0.4,0.31,2.2,0.09,13,62,0.9966,3.07,0.63,10.5,6 8.3,0.54,0.28,1.9,0.077,11,40,0.9978,3.39,0.61,10,6 7.8,0.56,0.12,2,0.082,7,28,0.997,3.37,0.5,9.4,6 8.8,0.55,0.04,2.2,0.119,14,56,0.9962,3.21,0.6,10.9,6 7,0.69,0.08,1.8,0.097,22,89,0.9959,3.34,0.54,9.2,6 7.3,1.07,0.09,1.7,0.178,10,89,0.9962,3.3,0.57,9,5 8.8,0.55,0.04,2.2,0.119,14,56,0.9962,3.21,0.6,10.9,6 7.3,0.695,0,2.5,0.075,3,13,0.998,3.49,0.52,9.2,5 8,0.71,0,2.6,0.08,11,34,0.9976,3.44,0.53,9.5,5 7.8,0.5,0.17,1.6,0.082,21,102,0.996,3.39,0.48,9.5,5 9,0.62,0.04,1.9,0.146,27,90,0.9984,3.16,0.7,9.4,5 8.2,1.33,0,1.7,0.081,3,12,0.9964,3.53,0.49,10.9,5 8.1,1.33,0,1.8,0.082,3,12,0.9964,3.54,0.48,10.9,5 8,0.59,0.16,1.8,0.065,3,16,0.9962,3.42,0.92,10.5,7 6.1,0.38,0.15,1.8,0.072,6,19,0.9955,3.42,0.57,9.4,5 8,0.745,0.56,2,0.118,30,134,0.9968,3.24,0.66,9.4,5 5.6,0.5,0.09,2.3,0.049,17,99,0.9937,3.63,0.63,13,5 5.6,0.5,0.09,2.3,0.049,17,99,0.9937,3.63,0.63,13,5 6.6,0.5,0.01,1.5,0.06,17,26,0.9952,3.4,0.58,9.8,6 7.9,1.04,0.05,2.2,0.084,13,29,0.9959,3.22,0.55,9.9,6 8.4,0.745,0.11,1.9,0.09,16,63,0.9965,3.19,0.82,9.6,5 8.3,0.715,0.15,1.8,0.089,10,52,0.9968,3.23,0.77,9.5,5 7.2,0.415,0.36,2,0.081,13,45,0.9972,3.48,0.64,9.2,5 7.8,0.56,0.19,2.1,0.081,15,105,0.9962,3.33,0.54,9.5,5 7.8,0.56,0.19,2,0.081,17,108,0.9962,3.32,0.54,9.5,5 8.4,0.745,0.11,1.9,0.09,16,63,0.9965,3.19,0.82,9.6,5 8.3,0.715,0.15,1.8,0.089,10,52,0.9968,3.23,0.77,9.5,5 5.2,0.34,0,1.8,0.05,27,63,0.9916,3.68,0.79,14,6 6.3,0.39,0.08,1.7,0.066,3,20,0.9954,3.34,0.58,9.4,5 5.2,0.34,0,1.8,0.05,27,63,0.9916,3.68,0.79,14,6 8.1,0.67,0.55,1.8,0.117,32,141,0.9968,3.17,0.62,9.4,5 5.8,0.68,0.02,1.8,0.087,21,94,0.9944,3.54,0.52,10,5 7.6,0.49,0.26,1.6,0.236,10,88,0.9968,3.11,0.8,9.3,5 6.9,0.49,0.1,2.3,0.074,12,30,0.9959,3.42,0.58,10.2,6

Program:	**Code block #1**
	
	`import pandas as pd`
	`import xgboost as xgb`
	`from xgboost import XGBClassifier`
	`from sklearn.model_selection import train_test_split from sklearn.metrics import mean_squared_error, log_loss from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier, VotingClassifier, BaggingRegressor`
	`from sklearn.linear_model import LinearRegression, LogisticRegression`
	
	Code block #2
	
	`df = pd.read_csv('winequality.csv')`
	
	Code block #3
	
	`df`
	
	Code block #4
	
	`target = df['quality']`
	
	Code block #5
	
	`train = df.drop('quality', axis = 1)`
	
	Code block #6
	
	`target.shape, train.shape`
	
	Code block #7
	
	`X_train, X_test, y_train, y_test = train_test_split(`
	`    train, target, test_size=0.20)`
	
	Code block #8
	
	`model_1 = LinearRegression()`
	`model_2 = xgb.XGBRegressor()`
	`model_3 = RandomForestRegressor()`
	
	Code block #9
	
	`model_1.fit(X_train, y_train)`

	Code block #10
	
	`model_2.fit(X_train, y_train)`
	
	Code block #11
	
	`model_3.fit(X_train, y_train)`
	
	Code block #12
	
	`pred_1  =  model_1.predict(X_test)`
	`pred_2  =  model_2.predict(X_test)`
	`pred_3 = model_3.predict(X_test)`
	
	Code block #13
	
	`pred_final = (pred_1+pred_2+pred_3)/3.0`
	
	Code block #14
	
	`# printing the mean squared error between real value and predicted value`
	`print(mean_squared_error(y_test, pred_final))`
	
	Code block #15
	
	`model_1 = LogisticRegression()`
	`model_2 = XGBClassifier()`
	`model_3 = RandomForestClassifier()`
	
	Code block #16
	
	`final_model = VotingClassifier( estimators=[('lr',`
	`    model_1), ('xgb', model_2),`
	`('rf', model_3)], voting='hard')`
	
	Code block #17
	
	`final_model.fit(X_train, y_train)`
	
	Code block #18
	
	`pred_final = final_model.predict(X_test)` **Code**
	
	block #19 `print(mean_squared_error(y_test,`
	
	`pred_final))`
	
	Code block #20

```
model =
BaggingRegressor(base_estimator=xgb.XGBRegressor())
```

Code block #21

```
model.fit(X_train, y_train)
```

Code block #22

```
pred = model.predict(X_test)
```

Code block #23

```
print(mean_squared_error(y_test, pred_final))
```

Outcome:

Block #3

```
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.5	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

Block #6

```
[35] target.shape, train.shape

     ((1599,), (1599, 11))
```

Average weighting

Block #9

```
model_1.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (statu
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
▾ LogisticRegression
LogisticRegression()
```

Block #10

```
model_2.fit(X_train, y_train)
```

```
▾ XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

Block #11

```
model_3.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

Block #14

```
# printing the mean squared error between real value and predicted value
print(mean_squared_error(y_test, pred_final))
```

```
0.374803252896761
```

Majority Voting

Block #17

```
final_model.fit(X_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

VotingClassifier
lr: LogisticRegression
xgb: XGBClassifier
rf: RandomForestClassifier
```

Block #19

```
print(mean_squared_error(y_test, pred_final))

0.4125
```

Bagging

Block #21

```
model.fit(X_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator` i
  warnings.warn(

BaggingRegressor
base_estimator: XGBRegressor
XGBRegressor
```

Block #23

```
print(mean_squared_error(y_test, pred_final))

0.4125
```

Conclusion :

In this experiment, we learnt about the ensemble learning techniques. We executed and plotted the corresponding tree related to it. We studied the training and implementation of the model.

References:

https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

https://www.geeksforgeeks.org/ensemble-classifier-data-mining/