Name :- Aniket Kumar

Sr.No :- 30

RollNo :- PF45

PRN No:- 1032171203.

AML LAB ASSIGNMENT - 05

Aim :- Implementation of Ensemble and Performance Measurement.

Title :- Ensemble learning method like Random forests, Bagging, and Boosting to improve accuracy.

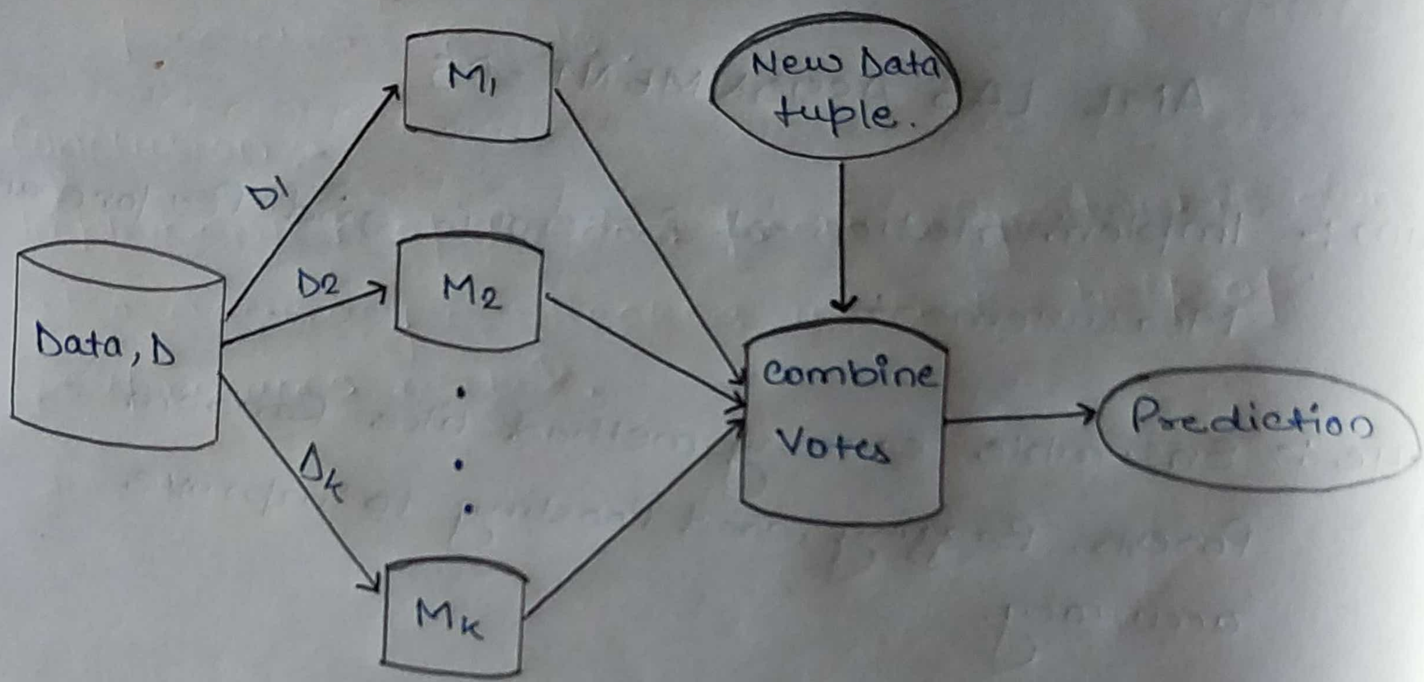
1. Ensemble Learning.

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

Definition.

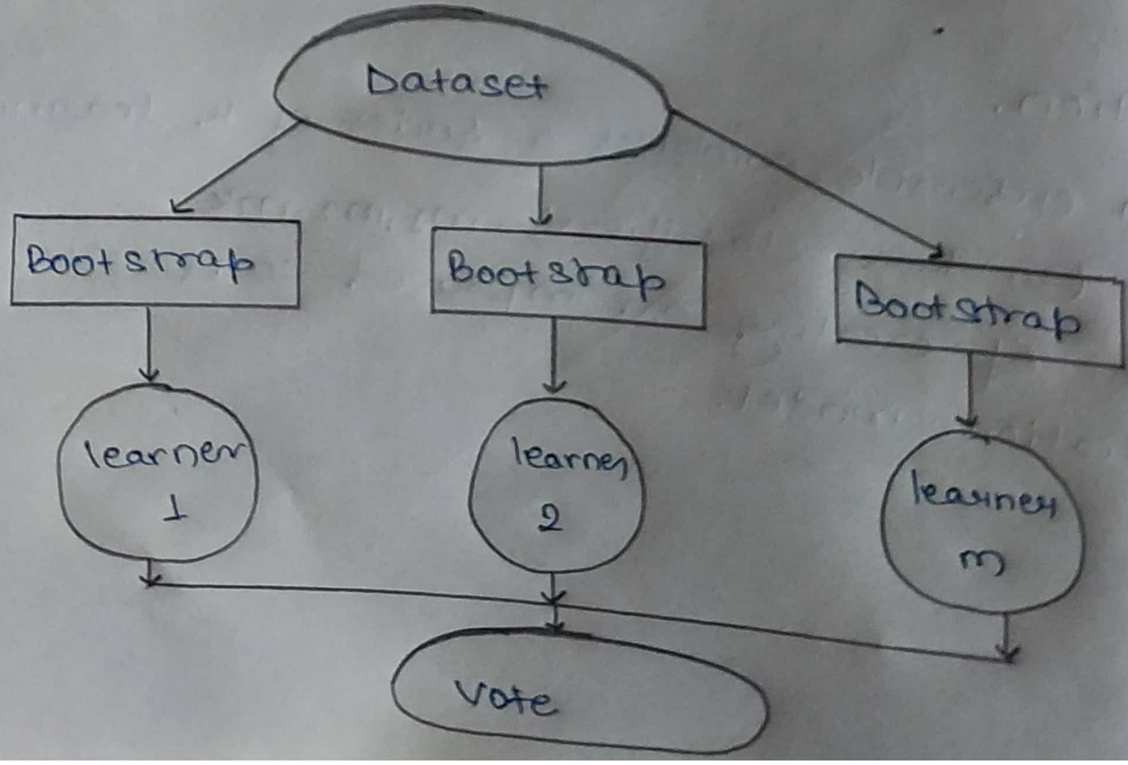
An ensemble combines a series of k learned models (or base classifiers), m1, m2, mk .... with the aim of creating an improved composite classification model.

A given dataset, D is used to create k training sets, D1, D2, D.., Dk where Di ($1 \leq i \leq k-1$) is used to generate classifier Mi



Bagging.

Bootstrap aggregation or bagging involves taking multiple samples from your training dataset and training a model for each sample.



3. Boosting and AdaBoost.

Boosting is an iterative technique which adjust the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice-versa.

Ada Boost is a popular boosting algorithm using Decision Trees for classification. Hence, it is a tree-based ensemble classifier.

Performance Metric

a. Confusion Matrix.

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

Predicted class

		Yes	no	Total
Actual class	Yes	TP	FN	P
	NO	FP	TN	N
	Total	P'	N'	P+N

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

b) Accuracy :-

It is the most common performance matrix for classification algorithm. Defined as the ratio of number of correct prediction made to all predictions made.

Conclusion :-

Implementation of Random forest algorithm is done also accuracy is checked with the help of performance matrix.

NAME : Aniket Kumar

S.no : 30

PRN : 1032171203

ROLL : PF45

Subject : AML Lab Assignment 5

In [1]:
```
pip install --upgrade scikit-learn
```

```
Requirement already up-to-date: scikit-learn in c:\users\aniket\anacond
a3\lib\site-packages (0.24.1)
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in c:\us
ers\aniket\anaconda3\lib\site-packages (from scikit-learn) (1.16.5)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in c:\use
rs\aniket\anaconda3\lib\site-packages (from scikit-learn) (0.13.2)
Requirement already satisfied, skipping upgrade: threadpoolctl>=2.0.0 i
n c:\users\aniket\anaconda3\lib\site-packages (from scikit-learn) (2.1.
0)
Requirement already satisfied, skipping upgrade: scipy>=0.19.1 in c:\us
ers\aniket\anaconda3\lib\site-packages (from scikit-learn) (1.4.1)
Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
import pandas as pd
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
```

```python
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import StackingClassifier
from matplotlib import pyplot
from matplotlib.pyplot import figure
figure(num=2, figsize=(16, 12), dpi=80, facecolor='w', edgecolor='k')
```

Out[9]:
```
<Figure size 1280x960 with 0 Axes>

<Figure size 1280x960 with 0 Axes>
```

In [10]:
```python
# get a stacking ensemble of models
def get_stacking():
  # define the base models
  level0 = list()
  level0.append(('lr', LogisticRegression()))
  level0.append(('knn', KNeighborsClassifier()))
  level0.append(('cart', DecisionTreeClassifier()))
  level0.append(('svm', SVC()))
  level0.append(('bayes', GaussianNB()))
  # define meta learner model
  level1 = LogisticRegression()
  # define the stacking ensemble
  model = StackingClassifier(estimators=level0, final_estimator=level1,
 cv=5)
  return model
```

In [11]:
```python
# get a list of models to evaluate
def get_models():
  models = dict()
  models['LogisticRegression'] = LogisticRegression()
  models['KNeighborsClassifier'] = KNeighborsClassifier()
  models['Decision tree'] = DecisionTreeClassifier()
  models['svm'] = SVC()
  models['GaussianNB'] = GaussianNB()
```

```
    models['stacking'] = get_stacking()
    return models
```

In [12]:

```
# evaluate a give model using cross-validation
def evaluate_model(model):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1
)
    scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jo
bs=-1, error_score='raise')
    return scores
```

In [15]:

```
# define dataset
data=pd.read_csv(r"C:\Users\Aniket\Desktop\Aml\5\winequality-red.csv")
data
```

Out[15]:

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

1599 rows × 12 columns

In [17]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in data.columns:
    data[i]=le.fit_transform(data[i])
```

In [18]:
```python
data
```

Out[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	27	76	0	10	39	11	28	342	63	18	
1	31	112	0	22	61	25	61	271	32	30	
2	31	88	4	17	55	15	48	287	38	27	
3	65	12	56	10	38	17	54	354	28	20	
4	27	76	0	10	39	11	28	342	63	18	
...	...	...	...	...	...	...	...	...	...	...	
1594	15	56	8	11	53	32	38	119	57	20	[illegible]
1595	12	47	10	15	25	40	45	135	64	38	[illegible]
1596	16	42	13	17	39	29	34	185	54	37	[illegible]
1597	12	65	12	11	38	32	38	164	69	33	[illegible]
1598	13	17	47	36	30	18	36	165	51	28	[illegible]

1599 rows × 12 columns

In [19]:
```python
X=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

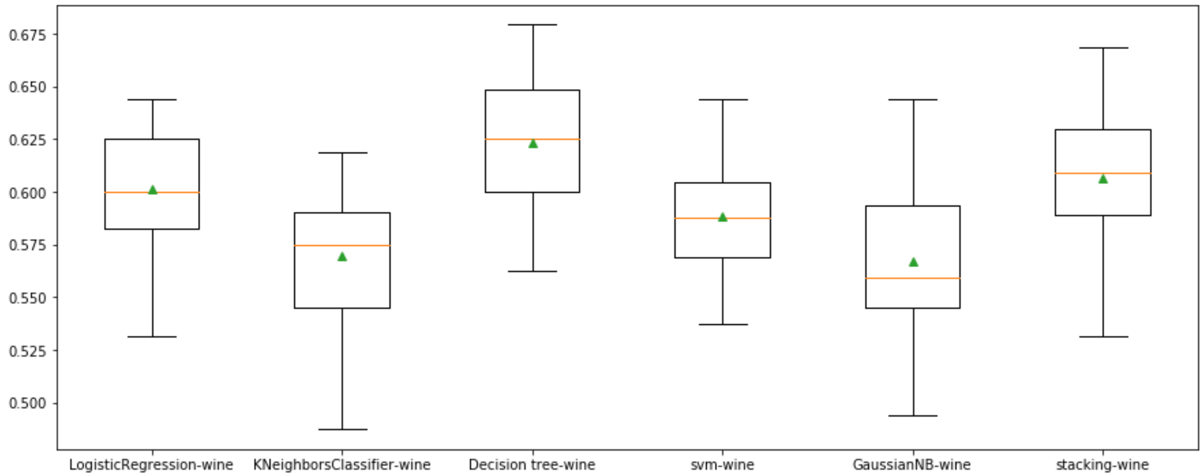
In [21]:
```python
# get the models to evaluate
models = get_models()
# evaluate the models and store results
```

```python
results, names, results1 = list(), list(),list()
for name, model in models.items():
  scores= evaluate_model(model)
  results.append(scores)
  names.append(name)
  print('>%s -> %.3f (%.3f)---Red-Wine-Quality-dataset' % (name, mean(s
cores), std(scores)))
```

```
>LogisticRegression -> 0.601 (0.027)---Red-Wine-Quality-dataset
>KNeighborsClassifier -> 0.570 (0.031)---Red-Wine-Quality-dataset
>Decision tree -> 0.623 (0.030)---Red-Wine-Quality-dataset
>svm -> 0.589 (0.025)---Red-Wine-Quality-dataset
>GaussianNB -> 0.567 (0.036)---Red-Wine-Quality-dataset
>stacking -> 0.606 (0.033)---Red-Wine-Quality-dataset
```

In [22]:

```python
pyplot.rcParams["figure.figsize"] = (15,6)
pyplot.boxplot(results, labels=[s+"-wine" for s in names], showmeans=Tr
ue)
pyplot.show()
```



In [ ]: