Ensemble Models

In machine learning

Introduction

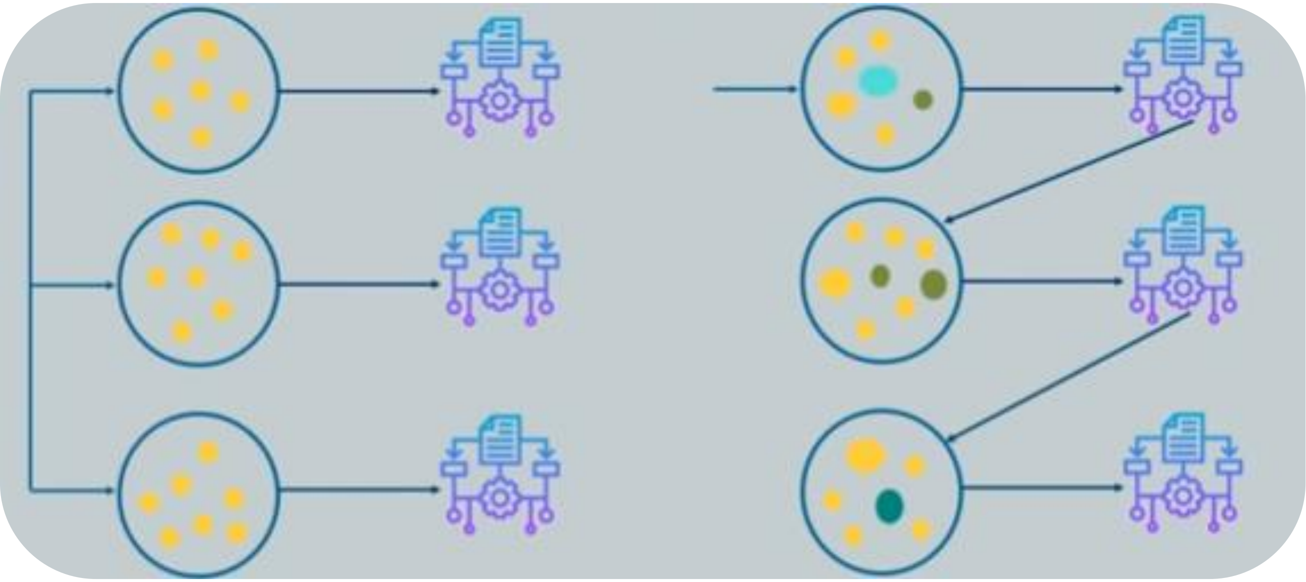
- Ensemble methods are learning algorithms
- Method of combining several **base** machine learning models

  Base learners – DT/ NBC/ Log. Reg.
  - To enhance the performance of the model
  - To improve the efficiency and accuracy of the model
- This is done to make a more robust system which **incorporates the predictions from all the base learners**.

Example base learners

- Logistic Regression
- Decision Tree
- Support vector Machine
- K-NN

- Ensemble models - application of group learning.
- Multiple modes are built by combining individual models together and used as a single model that is more accurate



Approaches to get the final output?

Following approaches are used to get the final output from all the base learners:

- **Averaging:**
  - It's defined as taking the average of predictions from models in case of regression problem or while predicting probabilities for the classification problem.

Model1	Model2	Model3	AveragePrediction
45	40	65	50

Approaches to get the final output?

- **Majority vote:**
  - It's defined as taking the prediction with maximum vote / recommendation from multiple models predictions while predicting the outcomes of a classification problem.

Model1	Model2	Model3	VotingPrediction
1	0	1	1

Approaches to get the final output?

- **Weighted average:**
  - In this, different weights are applied to predictions from multiple models then taking the average which means giving high or low importance to specific model output.

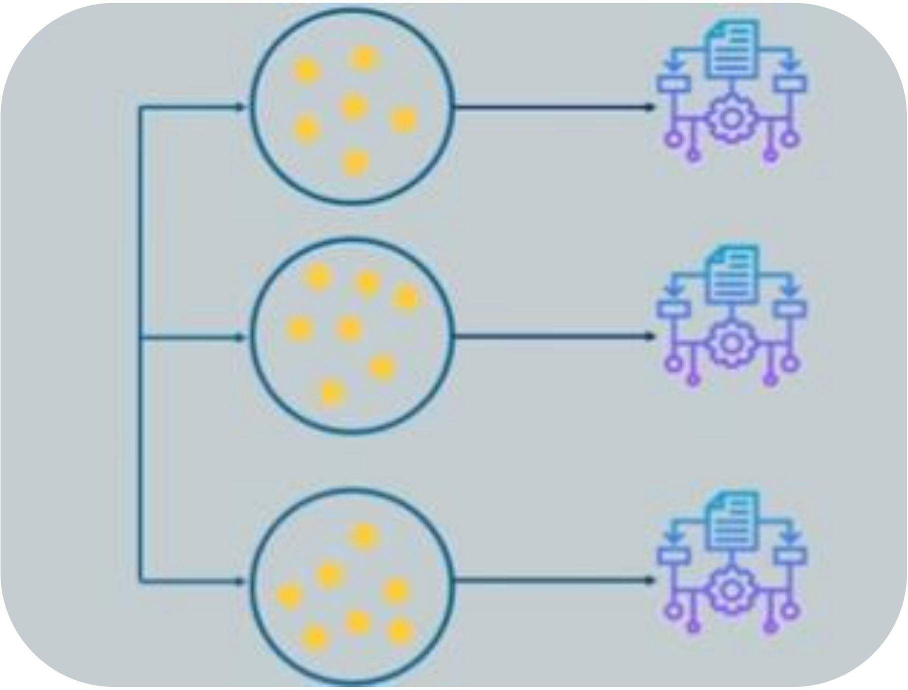
	Model1	Model2	Model3	WeightAveragePrediction
Weight	0.4	0.3	0.3	
Prediction	45	40	60	48

Ensemble Models

1. Bagging
2. Boosting

What is Bagging?

- Making models in parallel
- Averaging slightly different versions of the same model to improve accuracy.
- Various models are built in parallel on various samples and then these models vote to give final prediction



Why Bagging?

- There are TWO main sources of errors in modeling:
  - Errors due to Bias
  - Errors due to Variance

- Models with **low bias** are **more flexible** models
- Models with **high bias** are **less flexible** models
- Variance : how outputs are different for each model

- Models
  - Low Bias – High variance
- We need models with Low Bias and Low Variance

Ensemble Models (1): Bagging

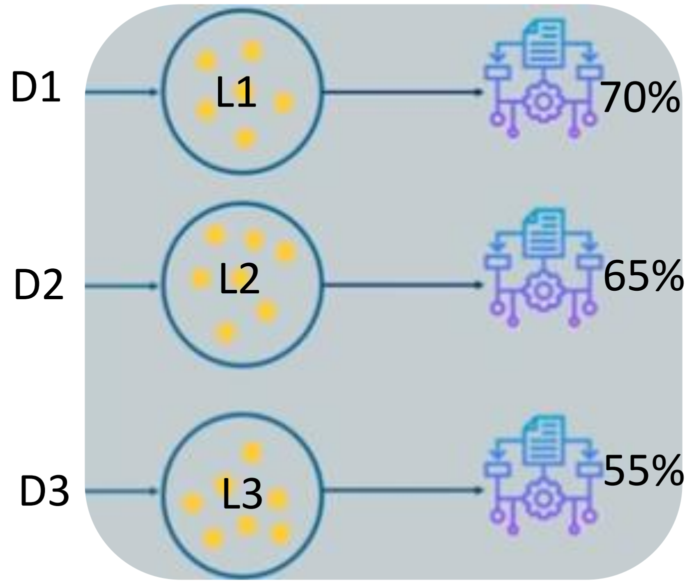
- Example: in random forest

1
2
3
4
5
6
7
8

Training Data (D)

D1	D2	D3
1	2	1
5	4	6
3	3	3
2	2	2
1	1	1

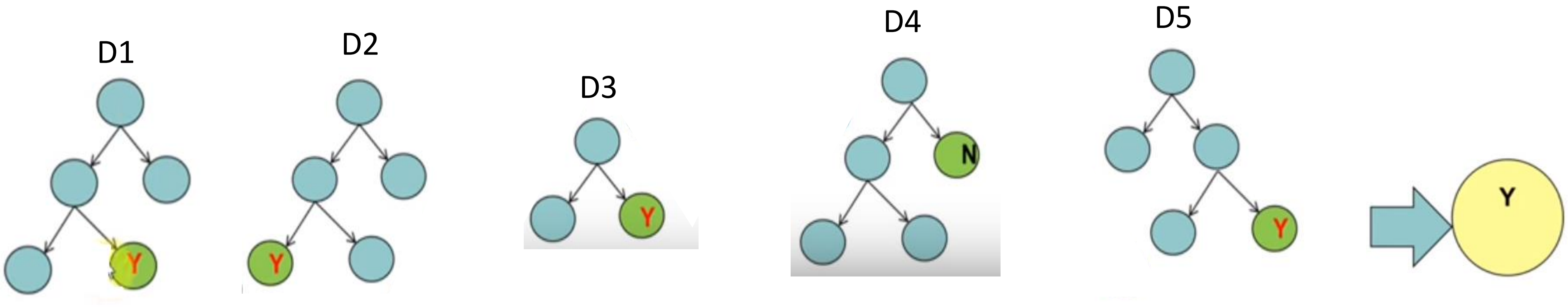
Sample D1, D2, D3 …



It is a sampling technique in which we choose **‘m’** observations or rows out of the original dataset of **‘n’** rows as well.

Ensemble Models (1): Bagging

- Example: in random forest (majority voting)



Ensemble Models (1): Bagging

- Bagging is also referred to as bootstrap aggregation.
- Designed to improve the stability and accuracy of machine learning algorithms.
- Used in statistical classification and regression.
- It also reduces variance and helps to avoid overfitting.

Bagging Example code

```
 Random Forest Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')

# Importing the mall dataset with pandas
import pandas as pd
data = pd.read_csv("drive/My Drive/Colab
Notebooks/DataSets/Social_Network_Ads.csv")
dataset = data
```

```
#dataset = pd.read_csv('train.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

```
# Splitting the dataset into the Training set
and Test set
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)
```

```python
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
```

```
# Fitting Random Forest Classification to the
Training set
from sklearn.ensemble import
RandomForestClassifier
classifier = RandomForestClassifier(n_estimators =
10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

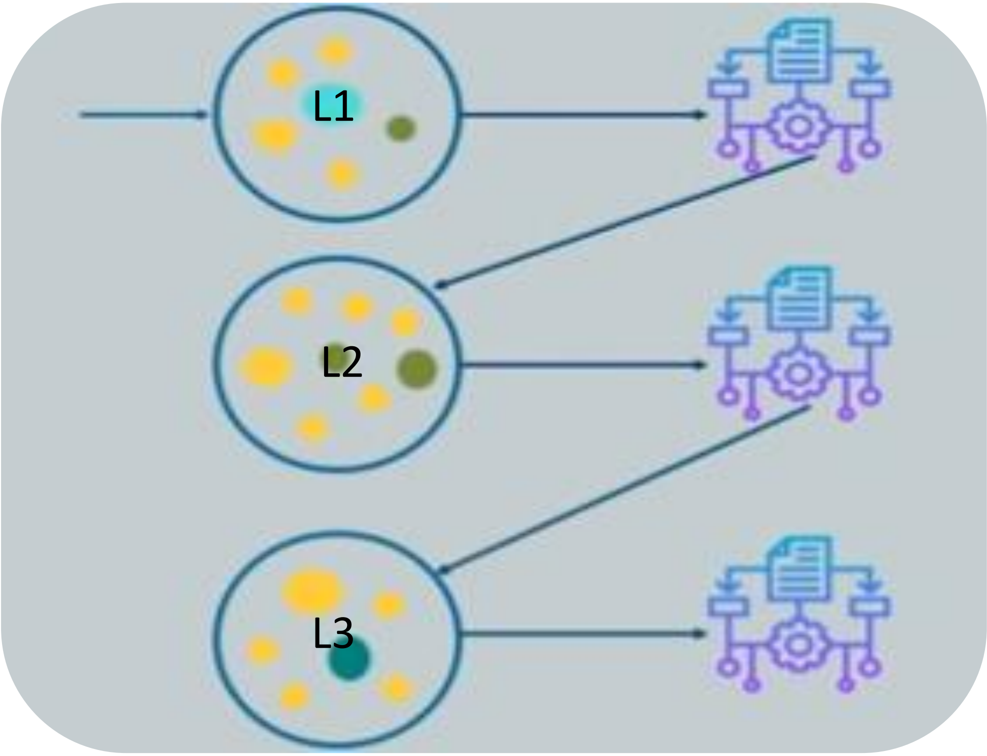
```python
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Boosting

Boosting

- A set of machine learning algorithms combined (weak learner) to form strong learners in order to increase the accuracy of the model.



Step-1: base learner reads data and assigns equal weight to each sample observation

Step-2: False predictions are assigned to the next base learner with a higher weightage on these incorrect predictions

Step-3: Repeat step 2 until the learner can correctly classify the output.

Types of Boosting

- Adaptive Boosting (AdaBoost)
- Gradient Boosting
- XGBoost

Adaptive Boosting

Step-1: Each data point is weighted equally for the first decision stump

Step-2: Misclassified data points are assigned higher weights

Step-3: A new decision stump is drawn by considering the data points with higher weights as more significant

Step-4: Again, if any data points are misclassified, they are given higher weight

Step-5: The process continues until all the observations are fall into the right class

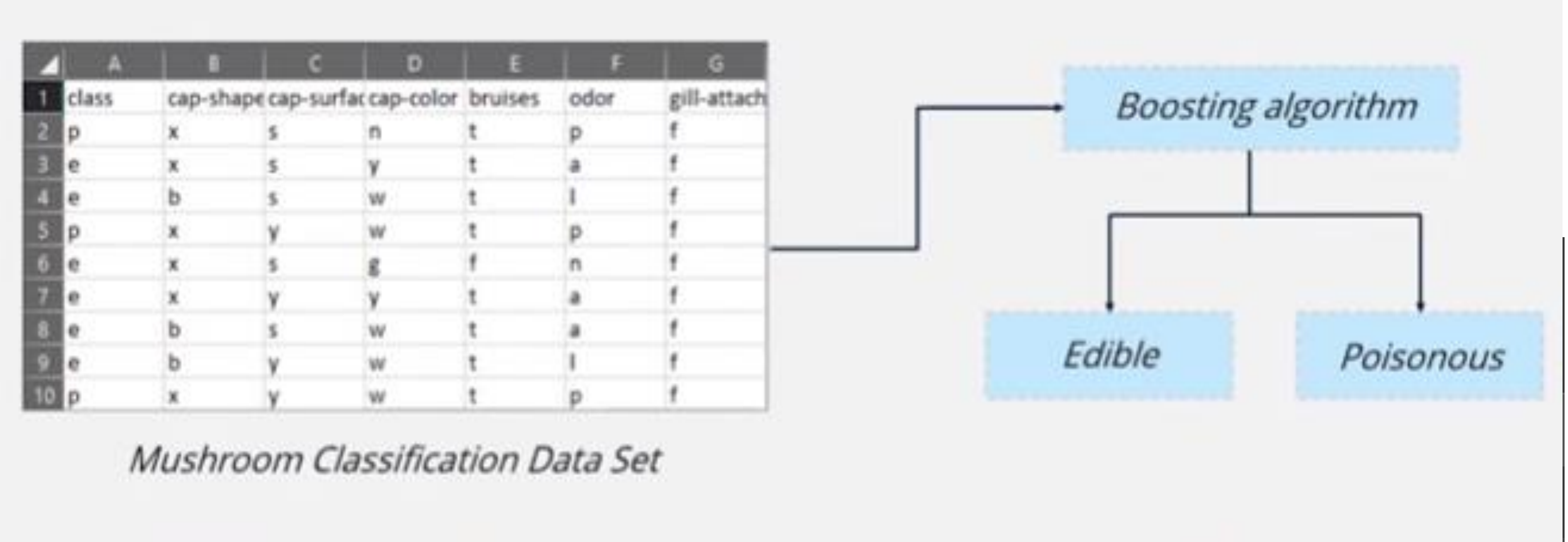
Gradient Boosting

- Base learners are generated sequentially in such a way that the present base learner is always more effective than the previous one.

XGBOOST

- Advanced version of Gradient boosting method that is designed to focus on computational speed and model efficiency.

Example



	A	B	C	D	E	F	G
1	class	cap-shape	cap-surfac	cap-color	bruises	odor	gill-attach
2	p	x	s	n	t	p	f
3	e	x	s	y	t	a	f
4	e	b	s	w	t	l	f
5	p	x	y	w	t	p	f
6	e	x	s	g	f	n	f
7	e	x	y	y	t	a	f
8	e	b	s	w	t	a	f
9	e	b	y	w	t	l	f
10	p	x	y	w	t	p	f

Mushroom Classification Data Set

716	e	b	s	w	t	l	f	c	b	k	...
5659	p	x	f	n	f	n	f	c	n	w	...

8124 rows × 23 columns

Ensemble Models (2): Boosting

- Models are built in series
- In each successive model, the weights are adjusted based on the learning of previous model.
- Boosting is used to reduce bias and variance in supervised learning
- It converts weak learners to strong ones.
- Algorithms that achieve hypothesis boosting quickly became known as "boosting".

```
#Boosting Model in Scikit-learn
```

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import LabelEncoder
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import pandas as pd
data = pd.read_csv("drive/My Drive/Colab Notebooks/DataSets/mushrooms.csv")
data.head(5)
```

8124 rows × 23 columns

class	cap-shape	cap-surface	cap-color	...	odor	gill-attachment	gill-spacing	gill-size	gill-color

	target	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
6672	p	x	y	e	f	s	f	c	n	b	...
2269	e	x	f	g	t	n	f	c	b	p	...
716	e	b	s	w	t	l	f	c	b	k	...
5659	p	x	f	n	f	n	f	c	n	w	...

```
#Label Encoding
for label in dataset.columns:
    dataset[label] = 
LabelEncoder().fit(dataset[label]).transform(datas
et[label])

#print(dataset.info())
dataset
```

	target	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
6672	1	5	3	2	0	7	1	0	1	0	...
2269	0	5	0	3	1	5	1	0	0	7	...
716	0	0	2	8	1	3	1	0	0	4	...

```
#split dataset in features and target variable
X = dataset.drop(['target'], axis =1)
y = dataset['target']
```

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

```python
# Create Decision Tree classifer object

model = DecisionTreeClassifier(criterion = 'entropy', max_depth = 1)

AdaBoost = AdaBoostClassifier(estimator = model, n_estimators = 400, learning_rate= 1)
boostmodel = AdaBoost.fit(X_train, y_train)
```

```python
#Predict the response for test dataset
y_pred = boostmodel.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
prediction = metrics.accuracy_score(y_test, y_pred)
```

```
print("Accuracy:",prediction *100, '%')
```

Thank you