

Experiment No.4

To implement ensemble learning using Bagging technique

Aim- To study, understand and implement a Bagging algorithm.

Theory-

In machine learning, for building solid and reliable models prediction accuracy is the key factor. Ensemble learning is a supervised machine-learning technique that combines multiple models to build a more powerful and robust model. The idea is that by combining the strengths of multiple models, we can create a model that is more robust and less likely to overfit the data. It can be used for both classifications and regression tasks.

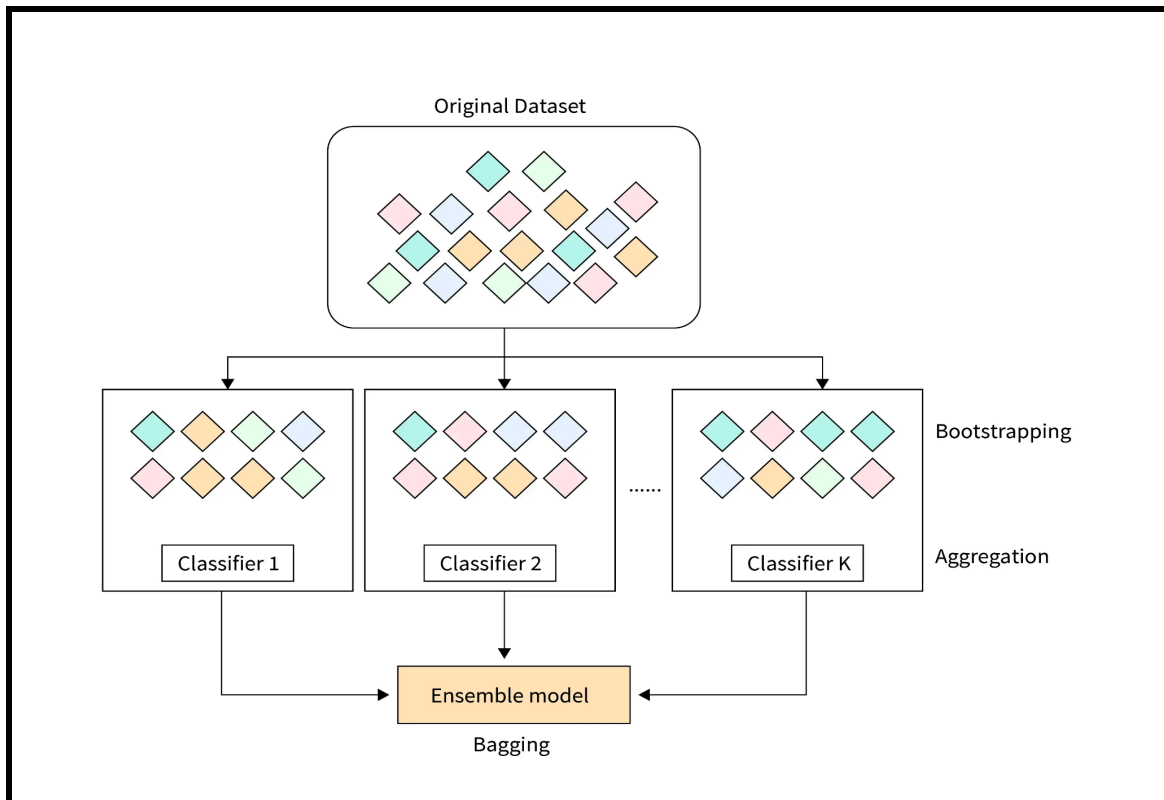
Ensemble learning techniques can be categorized in three ways:

- Bagging (Bootstrap Aggregating)
- Boosting
- Stacking (Stacked Generalization)

Bagging is a supervised machine-learning technique, and it can be used for both regression and classification tasks. In this article we will discuss the bagging classifier.

Bagging Classifier

Bagging (or Bootstrap aggregating) is a type of ensemble learning in which multiple base models are trained independently in parallel on different subsets of the training data. Each subset is generated using bootstrap sampling, in which data points are picked at random with replacement. In the case of the Bagging classifier, the final prediction is made by aggregating the predictions of the all-base model, using majority voting. In the case of regression, the final prediction is made by averaging the predictions of the all-base model, and that is known as bagging regression.



Bagging Classifier

Bagging helps improve accuracy and reduce overfitting, especially in models that have high variance.

Working of Bagging Classifier :

1. **Bootstrap Sampling:** In Bootstrap Sampling randomly 'n' subsets of original training data are sampled with replacement. This step ensures that the base models are trained on diverse subsets of the data, as some samples may appear multiple times in the new subset, while others may be omitted. It reduces the risks of overfitting and improves the accuracy of the model.
 - a. Let's break it down step by step:
 - b. Original training dataset: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
 - c. Resampled training set 1: [2, 3, 3, 5, 6, 1, 8, 10, 9, 1]
 - d. Resampled training set 2: [1, 1, 5, 6, 3, 8, 9, 10, 2, 7]
 - e. Resampled training set 3: [1, 5, 8, 9, 2, 10, 9, 7, 5, 4]
2. **Base Model Training:** In bagging, multiple base models are used. After the Bootstrap Sampling, each base model is **independently trained** using a specific learning

algorithm, such as decision trees, support vector machines, or neural networks on a different bootstrapped subset of data. These models are typically called “Weak learners” because they may not be highly accurate on their own. Since the base model is trained independently of different subsets of data. To make the model computationally efficient and less time-consuming, the base models can be trained in **parallel**.

3. **Aggregation:** Once all the base models are trained, it is used to make predictions on the unseen data i.e. the subset of data on which that base model is not trained. In the bagging classifier, the predicted class label for the given instance is chosen based on the majority voting. The class which has the majority voting is the prediction of the model.
4. **Out-of-Bag (OOB) Evaluation:** Some samples are excluded from the training subset of particular base models during the bootstrapping method. These “out-of-bag” samples can be used to estimate the model’s performance without the need for cross-validation.
5. **Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

Algorithm for the Bagging classifier:

Classifier generation:

Let N be the size of the training set.

for each of t iterations:

 sample N instances with replacement from the original training set.

 apply the learning algorithm to the sample.

 store the resulting classifier.

Classification:

for each of the t classifiers:

 predict class of instance using classifier.

return class that was predicted most often.

Steps to Perform Bagging :

- Consider there are n observations and m features in the training set. You need to select a random sample from the training dataset without replacement

- A subset of m features is chosen randomly to create a model using sample observations
- The feature offering the best split out of the lot is used to split the nodes
- The tree is grown, so you have the best root nodes
- The above steps are repeated n times. It aggregates the output of individual decision trees to give the best prediction

Advantages of Bagging in Machine Learning :

1. **Improved Predictive Performance:** Bagging Classifiers often outperforms single classifiers by reducing overfitting and increasing predictive accuracy. By combining multiple base models, it can better generalize to unseen data.
2. **Robustness:** Bagging reduces the impact of outliers and noise in the data by aggregating predictions from multiple models. This enhances the overall stability and robustness of the model.
3. **Reduced Variance:** Since each base model is trained on different subsets of the data, the aggregated model's variance is significantly reduced compared to an individual model.
4. **Parallelization:** Bagging allows for parallel processing, as each base model can be trained independently. This makes it computationally efficient, especially for large datasets.
5. **Flexibility:** Bagging Classifier is a versatile technique that can be applied to a wide range of machine learning algorithms, including decision trees, random forests, and support vector machines.

Applications of Bagging Classifier :

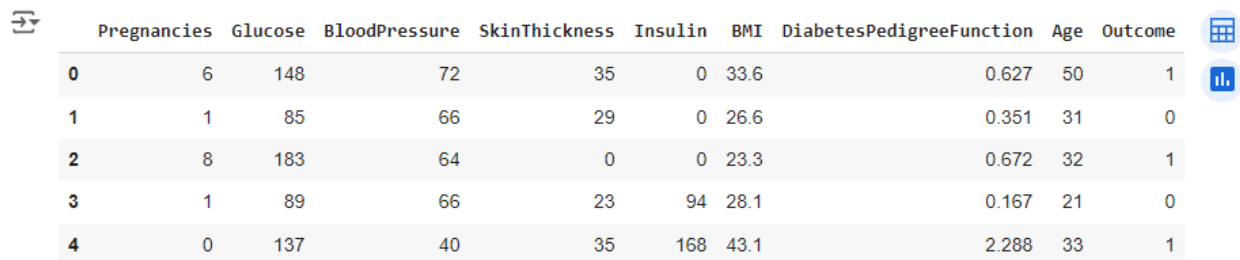
1. **Fraud Detection:** Bagging Classifier can be used to detect fraudulent transactions by aggregating predictions from multiple fraud detection models.
2. **Spam filtering:** Bagging classifier can be used to filter spam emails by aggregating predictions from multiple spam filters trained on different subsets of the spam emails.
3. **Credit scoring:** Bagging classifier can be used to improve the accuracy of credit scoring models by combining the predictions of multiple models trained on different subsets of the credit data.

4. **Image Classification:** Bagging classifiers can be used to improve the accuracy of image classification tasks by combining the predictions of multiple classifiers trained on different subsets of the training images.
5. **Natural language processing:** In NLP tasks, the bagging classifier can combine predictions from multiple language models to achieve better text classification results.

Program :

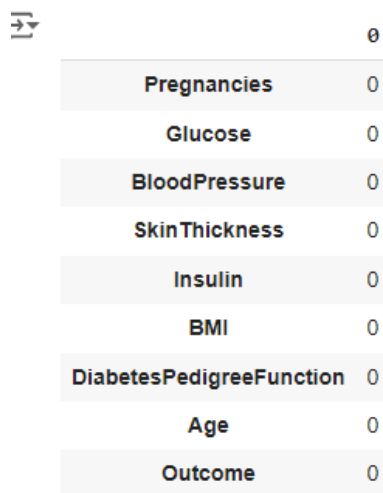
```
from google.colab import files
upload =files.upload()
```

```
import pandas as pd
df = pd.read_csv('diabetes.csv')
df.head()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.isnull().sum()
```



	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000

```
df.Outcome.value_counts()
```

	count
Outcome	
0	500
1	268

```
dtype: int64
```

```
x = df.drop("Outcome",axis = "columns")
y = df.Outcome
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
x_scaled[:3]
```

```
array([[ 0.63994726,  0.84832379,  0.14964075,  0.90726993, -0.69289057,
         0.20401277,  0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575,  0.53090156, -0.69289057,
        -0.68442195, -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, -1.28821221, -0.69289057,
        -1.10325546,  0.60439732, -0.10558415]])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,stratify = y,random_state = 10)
```

```
X_train.shape
```

```
(576, 8)
```

```
X_test.shape
```

(192, 8)

y_train.value_counts()

count	
Outcome	
0	375
1	201

dtype: int64

201/375

0.536

y_test.value_counts()

count	
Outcome	
0	125
1	67

dtype: int64

67/125

0.536

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
scores = cross_val_score(DecisionTreeClassifier(),x,y,cv = 5)
Scores
```

array([0.70779221, 0.67532468, 0.68181818, 0.79738562, 0.73202614])

scores.mean()

0.7188693659281894

```
from sklearn.ensemble import RandomForestClassifier
scores = cross_val_score(RandomForestClassifier(n_estimators = 40),x,y,cv = 5)
scores.mean()
```

0.7669637551990492

```
from sklearn.ensemble import BaggingClassifier
bag_model = BaggingClassifier(
    base_estimator = DecisionTreeClassifier(),
    n_estimators = 100,
    max_samples = 0.8,
    oob_score = True,
    random_state = 0
)
bag_model.fit(x_train,y_train)
bag_model.oob_score_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning:
`base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.
  warnings.warn(
```

0.7534722222222222

```
bag_model.fit(x_train,y_train)
bag_model.oob_score_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning:
`base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.
  warnings.warn(
```

0.7534722222222222

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

```
# Assuming 'bag_model' is your trained model
y_pred = bag_model.predict(x_test) # Generate predictions
```

```
# Now calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

0.7760416666666666

```
bag_model.score(x_test,y_test)
bag_model.oob_score_
```


0.7534722222222222

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
metrics.accuracy_score(y_test,y_pred)
```

0.7760416666666666

```
bag_model = BaggingClassifier(
    base_estimator = DecisionTreeClassifier(),
    n_estimators = 100,
    max_samples = 0.8,
    oob_score = True,
    random_state = 0
)
scores = cross_val_score(bag_model,x,y,cv = 5)
scores
```

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.4.

warnings.warn(

array([0.75324675, 0.72727273, 0.74675325, 0.82352941, 0.73856209])

scores.mean()

0.7578728461081402

```
from sklearn.metrics import classification_report
```

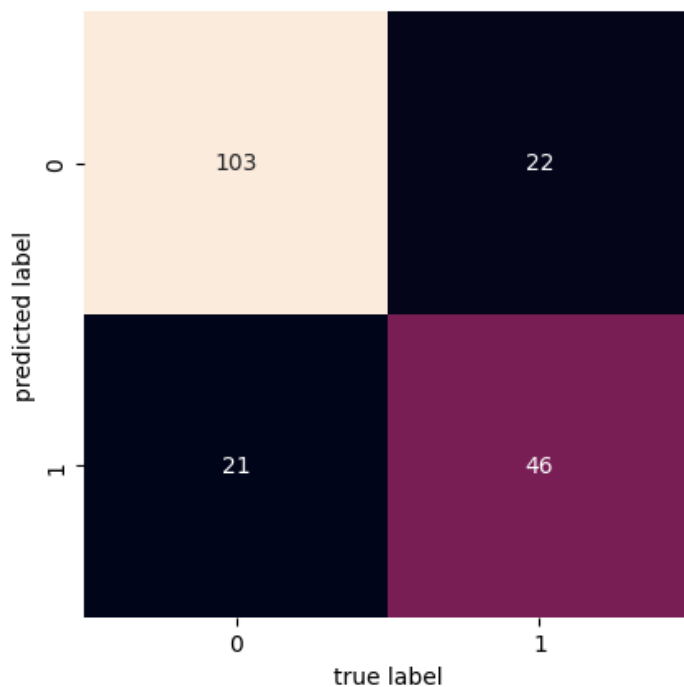
```
#y_pred = bag_model.predict(x_test)
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.82	0.83	125
1	0.68	0.69	0.68	67
accuracy			0.78	192
macro avg	0.75	0.76	0.75	192
weighted avg	0.78	0.78	0.78	192

```

from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
mat = confusion_matrix(y_test,y_pred)
sns.heatmap(mat,square = True,annot = True,fmt = 'd',cbar = False)
plt.xlabel('true label')
plt.ylabel('predicted label');

```



Conclusion :

Bagging Classifier, as an ensemble learning technique, offers a powerful solution for improving predictive performance and model robustness. Bagging Classifier avoids overfitting, improves generalization, and gives solid predictions for a wide range of applications by using the collective wisdom of numerous base models.