

Practical No. 7

Title: Implementation of Boosting Algorithms

Aim: Understanding basics of Boosting Algorithms

Introduction:

What is Boosting?

Boosting is an ensemble learning technique that builds a strong learner by combining multiple weak learners (usually decision trees with shallow depth). Instead of training models independently (like in bagging), boosting trains them sequentially, where each new model tries to correct the mistakes made by the previous ones.

Key Ideas Behind Boosting:

1. Sequential Learning: Models are trained one after the other.
 2. Focus on Errors: Each model tries to fix the errors of its predecessor.
 3. Weighted Voting: Final prediction is a weighted vote (or sum) of all the models.
 4. Boost Weak Learners: Even models that are only slightly better than random can be boosted into a strong predictor.
-

Types of Boosting Algorithms

1. AdaBoost (Adaptive Boosting)

- Starts with equal weights for all training instances.
- After each model, it increases the weights of misclassified points so that the next model focuses more on them.
- Final model is a weighted sum of all weak learners.

Base Learner: Usually decision stumps (1-level trees)

Sequence:

- Fit a model
- Compute error and adjust sample weights
- Fit next model on new weights

2. Gradient Boosting

- Works like gradient descent: each new model tries to minimize the error of the overall model using the gradient of a loss function.
- Models are trained on the residuals (errors) of the previous model.

Base Learner: Small decision trees

Sequence:

- Start with an initial prediction (e.g., mean)
- Compute residuals (errors)
- Fit a new model on the residuals
- Add new model to the ensemble

3. Stochastic Gradient Boosting

- Same as Gradient Boosting, but adds randomness:
 - Only a subset of data is used to train each new model.
- Helps reduce overfitting and improves generalization.

Exercise -

1. Implement Boosting Algorithms

```
#Boosting Algorithm: AdaBoost
```

```
import pandas as pd
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.model_selection import train_test_split
```

```
import warnings
```

```
url='https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv'
```

```
df = pd.read_csv(url)
```

```
df
```



	company	job	degree	salary_more_than_100k
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	business manager	bachelors	1
3	google	business manager	masters	1
4	google	computer programmer	bachelors	0
5	google	computer programmer	masters	1
6	abc pharma	sales executive	masters	0
7	abc pharma	computer programmer	bachelors	0
8	abc pharma	business manager	bachelors	0
9	abc pharma	business manager	masters	1
10	facebook	sales executive	bachelors	1
11	facebook	sales executive	masters	1
12	facebook	business manager	bachelors	1
13	facebook	business manager	masters	1
14	facebook	computer programmer	bachelors	1
15	facebook	computer programmer	masters	1

```
inp = df.drop(['salary_more_than_100k'],axis="columns")
```

```
trgt = df['salary_more_than_100k']
```

```
trgt
```



	salary_more_than_100k
0	0
1	0
2	1
3	1
4	0
5	1
6	0
7	0
8	0
9	1
10	1
11	1
12	1
13	1
14	1
15	1

```
dtype: int64
```

```

from sklearn.preprocessing import LabelEncoder
lbl_company = LabelEncoder()
lbl_job = LabelEncoder()
lbl_degree = LabelEncoder()


inp['company_new'] = lbl_company.fit_transform(inp['company'])
inp['job_new'] = lbl_job.fit_transform(inp['job'])
inp['degree_new'] = lbl_degree.fit_transform(inp['degree'])

```

```

inp_new = inp.drop(['company', 'job', 'degree'], axis='columns')
inp_new

```



	company_new	job_new	degree_new
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

```





from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inp_new, trgt, train_size=0.8)

```

```

model=AdaBoostClassifier(n_estimators=100, learning_rate=1)
model.fit(X_train, y_train)

```


  AdaBoostClassifier  

```
AdaBoostClassifier(learning_rate=1, n_estimators=100)
```

```

model.predict([[2,0,1]])

```

 /usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but AdaBoostClassifier will use the integer indices [0, 1, 2] to access the columns of the input X

```
warnings.warn(
array([1])
```