



Boosting

Algorithms

Study and implementation

By: Niyas

Agenda

- Introduction
- AdaBoost
- XGBoost
- LG Boost
- Implementation outcome on Insurance prediction dataset



Introduction

- Boosting algorithms are one of the **best-performing algorithms** among all the other ML algorithms with the best performance and higher accuracies
- All the boosting algorithms work on the basis of **learning from the errors of the previous model trained and tried avoiding the same mistakes made by the previously trained weak learning algorithm**





Gradient Boosting

Let's dive in



Gradient Boosting Algorithm

- Works on the principle of the **stagewise addition method**
- **Multiple weak learning algorithms are trained** and a **strong learner algorithm** is used as a **final model** from the addition of **multiple weak learning algorithms trained on the same dataset**.
- the **first weak learner will not be trained on the dataset**, it will simply return the **mean of the particular column**,
- **The residual for output of the first weak learner algorithm** will be calculated which will be **used as output or target column for next weak learning algorithm** which is to be trained.
- the **second weak learner will be trained** and the **residuals** will be calculated which will be used **as an output column again for the next weak learner**, this is how this **process will continue** until we reach **zero residuals**.
- In gradient boosting the dataset should be in **the form of numerical or categorical data** and the loss function using which the residuals are **calculated should be differential at all points**.

Gradient Boost Implementation

```
import pandas as pd
dataset = pd.read_csv("insurance_pre.csv")
dataset
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
dataset.columns
independent=dataset[['age', 'bmi', 'children', 'sex_male',
'smoker_yes']]
Independent
dependent = dataset[['charges']]
dependent
from sklearn.model_selection import
train_test_splitX_train,X_test,y_train,y_test=train_test_split(ind
ependent, dependent, test_size=0.30,random_state=0)
X_train
X_test
y_train
y_test
```

```
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.datasets import make_regression

from sklearn.metrics import r2_score

gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)

y_pred1 = gbr.predict(X_test)

print("Gradient Boosting - R2: ", r2_score(y_test, y_pred1))
```




XGBoost

Let's dive in



XG Boosting Algorithm

- XGBoost algorithm is the eXtreme Gradient Boosting algorithm
- difference between GradientBoosting is XGBoost is that XGBoost uses a regularization technique in it and is much faster
- performs better when there is a presence of numerical and categorical features in the dataset.

XG Boost Implementation

```
import pandas as pd
dataset = pd.read_csv("insurance_pre.csv")
dataset
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
dataset.columns
independent=dataset[['age', 'bmi', 'children', 'sex_male',
'smoker_yes']]
Independent
dependent = dataset[['charges']]
dependent
from sklearn.model_selection import
train_test_splitX_train,X_test,y_train,y_test=train_test_split(
independent, dependent, test_size=0.30,random_state=0)
X_train
X_test
y_train
y_test
```

```
!pip install xgboost
from xgboost import XGBRegressor
from sklearn.metrics import r2_score

xgr = XGBRegressor()xgr.fit(X_train, y_train)

y_pred2 = xgr.predict(X_test)

print("XGBoost - R2: ", r2_score(y_test, y_pred2))
```



Ada Boost

Let's dive in



Ada Boost Algorithm

- works on the principle of the stagewise addition method where multiple weak learners are used for getting strong learners
- Unlike Gradient Boosting in XGBoost, the alpha parameter calculated is related to the errors of the weak learner, here the value of the alpha parameter will be indirectly proportional to the error of the weak learner.
- Once the alpha parameter is calculated, the weightage will be given to the particular weak learners, here the weak learner that are doing mistakes will get more weightage to fill out the gap in error and the weak learners that are already performing well will get fewer weights as they are already a good model

Ada Boost Implementation

```
import pandas as pd
dataset = pd.read_csv("insurance_pre.csv")
dataset
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
dataset.columns
independent=dataset[['age', 'bmi', 'children', 'sex_male',
'smoker_yes']]
Independent
dependent = dataset[['charges']]
dependent
from sklearn.model_selection import
train_test_splitX_train,X_test,y_train,y_test=train_test_split(ind
ependent, dependent, test_size=0.30,random_state=0)
X_train
X_test
y_train
y_test
```

```
from sklearn.ensemble import AdaBoostRegressor

from sklearn.metrics import r2_score

adr = AdaBoostRegressor()

adr.fit(X_train, y_train)

y_pred3 = adr.predict(X_test)

print("AdaBoost - R2: ", r2_score(y_test, y_pred3))
```




LG Boost

Let's dive in



LG Boost Algorithm

- Expansion is Light Gradient Boosting Machine
- In LightGBM decision trees are grown leaf wise
- This means at a single time only one leaf from the whole tree will be grown
- the sampling of the data while training the decision tree is done by the method known as GOSS
- the variance of all the data samples is calculated and sorted in descending order
- Data samples having low variance are already performing well, so there will be less weightage given to the samples having low variance while sampling the dataset.

LG Boost Implementation

```
import pandas as pd
dataset = pd.read_csv("insurance_pre.csv")
dataset
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
dataset.columns
independent=dataset[['age', 'bmi', 'children', 'sex_male',
'smoker_yes']]
Independent
dependent = dataset[['charges']]
dependent
from sklearn.model_selection import
train_test_splitX_train,X_test,y_train,y_test=train_test_split(inde
pendent, dependent, test_size=0.30,random_state=0)
X_train
X_test
y_train
y_test
```

```
!pip install lightgbm
import lightgbm as lgb
from sklearn.metrics import r2_score

from lightgbm import LGBMRegressor

lgr = LGBMRegressor()lgr.fit(X_train, y_train)

y_pred5 = lgr.predict(X_test)
print("LightGBM - R2: ", r2_score(y_test, y_pred5))
```

Conclusion on Insurance charge prediction

Boosting Algorithm	R2	Insights
Gradient Boosting - R2:	0.8838177942251945	If you need more community support for the algorithm then use algorithms which was developed years back XGBoost or Gradient Boosting.
XGBoost - R2	0.8213337063789368	there is a need for regularization according to your dataset, then you can definitely use XGBoost
Adaboost - R2:	0.8537077260002826	to deal with any adaptable ada Boost perform better generally.
LightGBM - R2:	0.86603193419773	to deal with categorical data, then CatBoost and LightGBM perform very well on those types of datasets.

mentorniyasahamed@gmail.com