

# Agenda

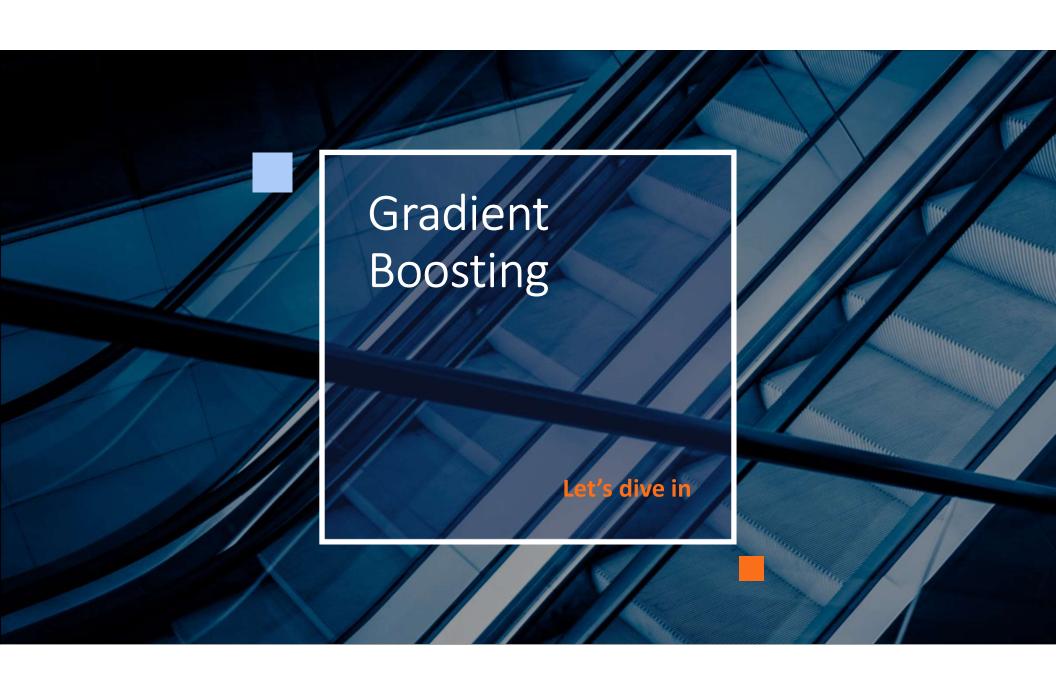
- Introduction
- AdaBoost
- XGBoost
- LG Boost
- Implementation outcome on Insurance prediction dataset



#### 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 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 <u>first weak learner will not be trained on the dataset</u>, 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")
                                                                   from sklearn.ensemble import GradientBoostingRegressor
dataset
dataset=pd.get dummies(dataset,drop first=True)
                                                                   from sklearn.datasets import make regression
dataset
dataset.columns
                                                                   from sklearn.metrics import r2 score
independent=dataset[['age', 'bmi', 'children', 'sex male',
'smoker yes']]
                                                                   gbr = GradientBoostingRegressor()
Independent
                                                                   gbr.fit(X train, y train)
dependent = dataset[['charges']]
dependent
                                                                   y pred1 = gbr.predict(X test)
from sklearn.model selection import
                                                                   print("Gradient Boosting - R2: ", r2 score(y test, y pred1))
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
```

12/21/2024 Annual Review

y\_train y test



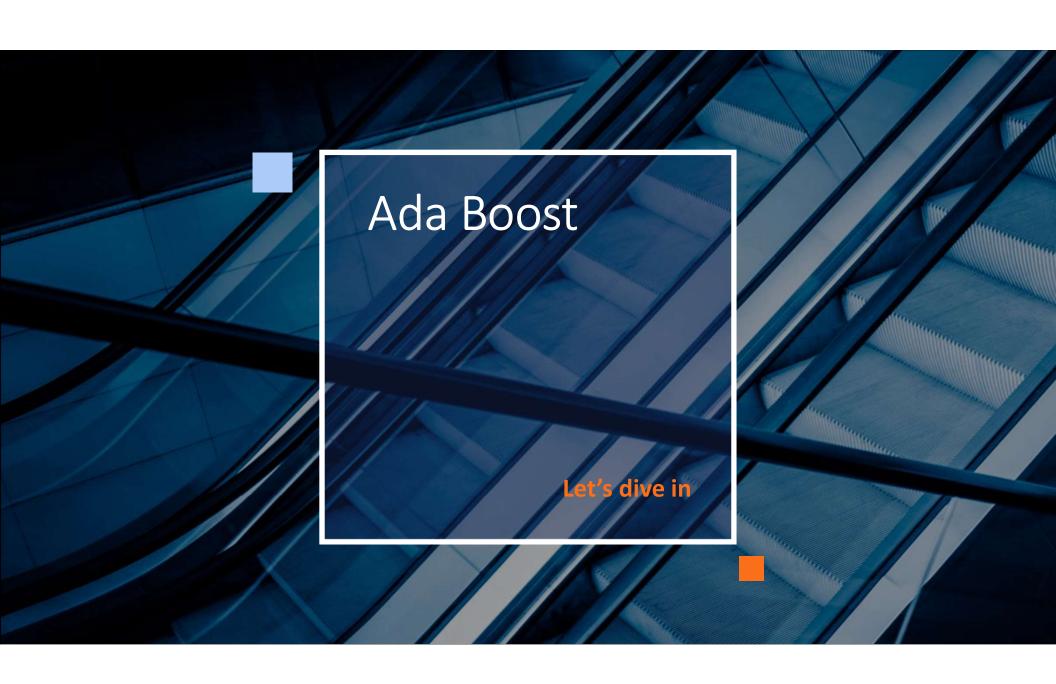
#### **XG Boosting Algorithm**

- XGBoost algorithm is the eXtreme Gradient Boosting algorithm
- difference between GradientBoosting is XGBoost is that XGbost 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))
```

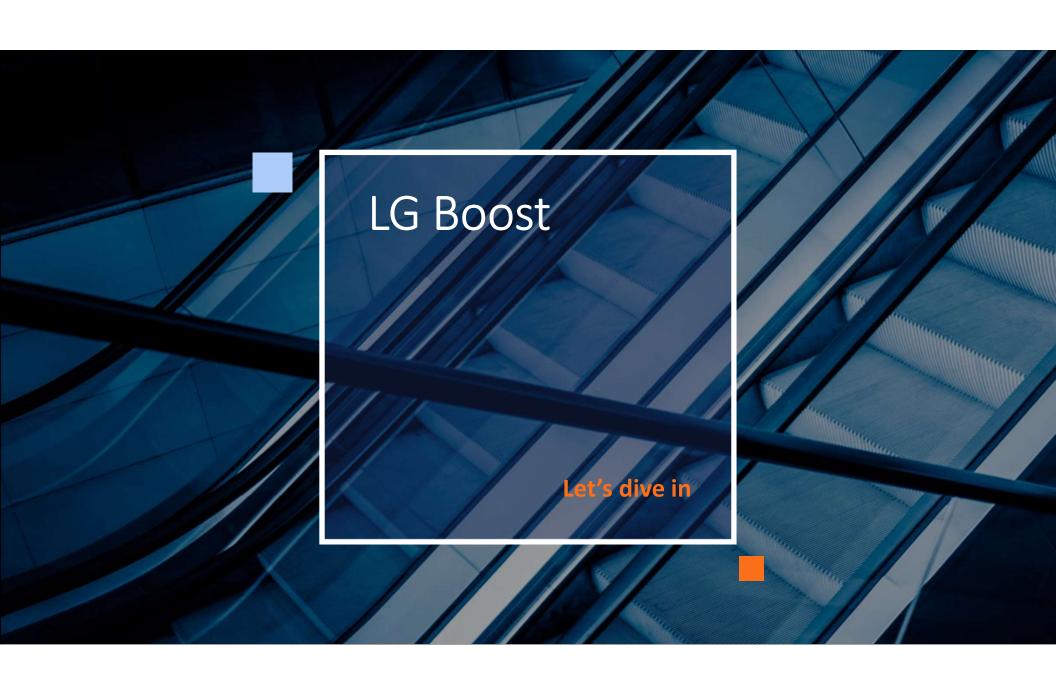


#### **Ada Boost Algorithm**

- 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 I 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
                                                               from sklearn.ensemble import AdaBoostRegressor
dataset = pd.read csv("insurance pre.csv")
dataset
                                                               from sklearn.metrics import r2 score
dataset=pd.get dummies(dataset,drop first=True)
dataset
                                                               adr = AdaBoostRegressor()
dataset.columns
independent=dataset[['age', 'bmi', 'children', 'sex male',
                                                               adr.fit(X train, y train)
'smoker yes']]
Independent
                                                               y pred3 = adr.predict(X test)
dependent = dataset[['charges']]
dependent
                                                               print("AdaBoost - R2: ", r2 score(y test, y pred3))
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
```



#### **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
                                                               !pip install lightgbm
dataset = pd.read csv("insurance pre.csv")
                                                               import lightgbm as lgb
dataset
                                                               from sklearn.metrics import r2 score
dataset=pd.get dummies(dataset,drop_first=True)
dataset
dataset.columns
                                                               from lightgbm import LGBMRegressor
independent=dataset[['age', 'bmi', 'children', 'sex male',
'smoker yes']]
                                                               lgr = LGBMRegressor()lgr.fit(X train, y train)
Independent
dependent = dataset[['charges']]
dependent
                                                               y pred5 = lgr.predict(X test)
from sklearn.model selection import
train test splitX train,X test,y train,y test=train test split(ind
                                                               print("LightGBM - R2: ", r2 score(y test, y pred5))
ependent, dependent, test size=0.30, random state=0)
X train
X test
y train
y test
```

# 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