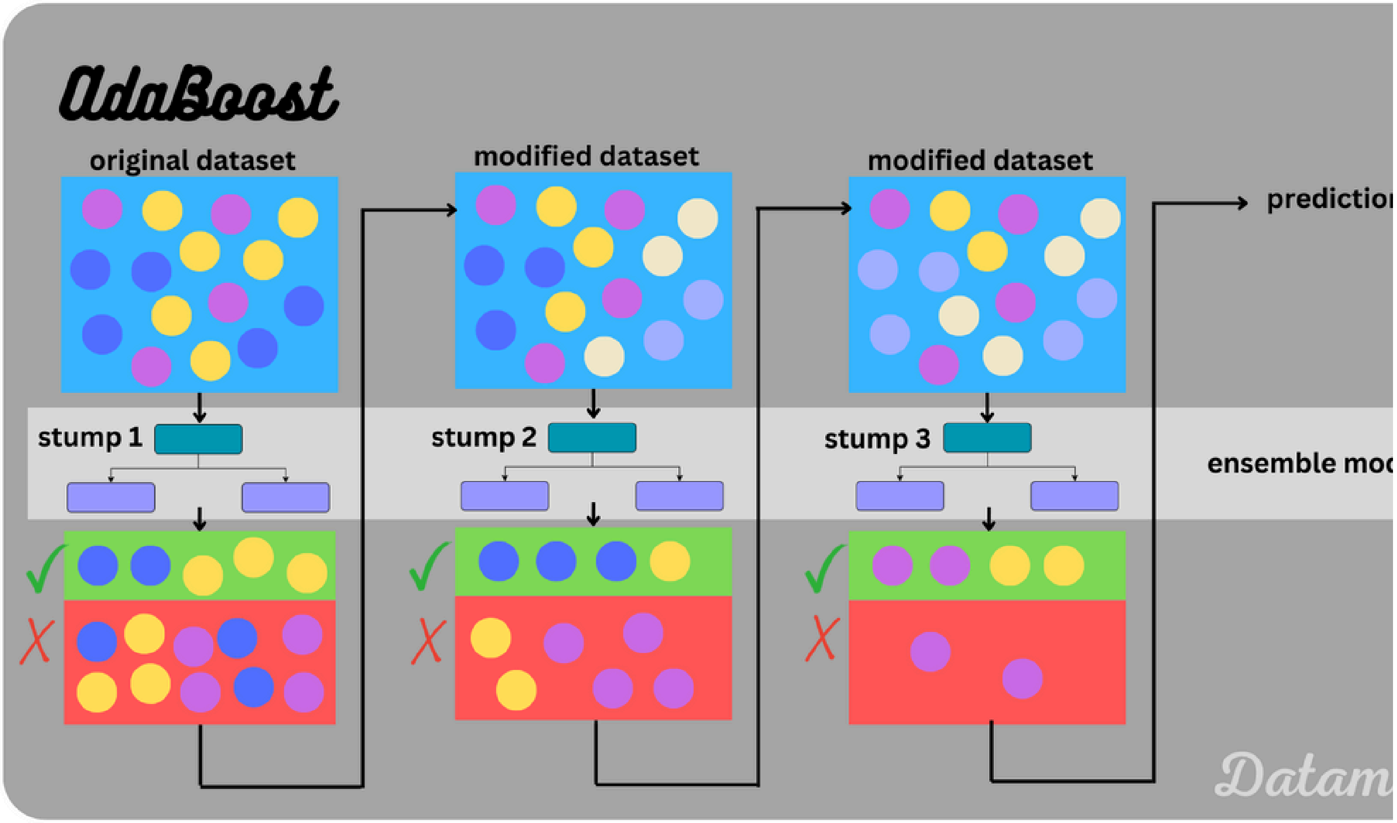
Boosting Algorithms

1. AdaBoost
2. Gradiant Boosting
   i)XG Boosting
   ii)LG Boosting

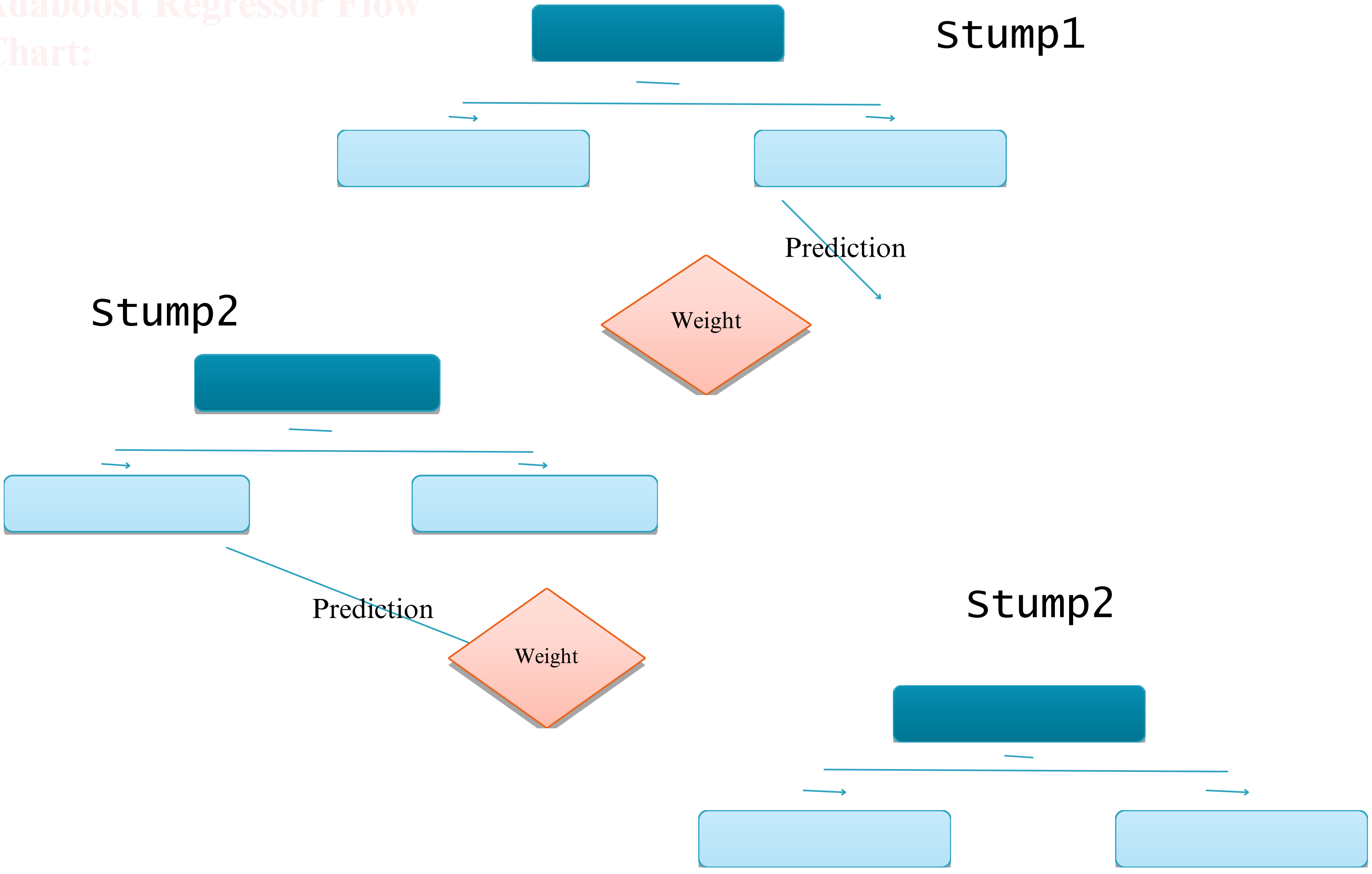
1) AdaBoost(Adaptive Boost)



Working Method:

- Adaboost Algorithm works same like boosting algorithm(ie. It do feature sampling in sequentially)
- It transforms weak learners to strong learners.
- AdaBoost is a Boosting algorithm, which means that the ensemble model is built sequentially and each new model builds on the results of the previous one, trying to improve its errors.
- The weights are determined in such a way that the wrongly predicted samples get higher weights than the correctly predicted samples.

Adaboost Regressor Flow Chart:



Stump1

Prediction

Weight

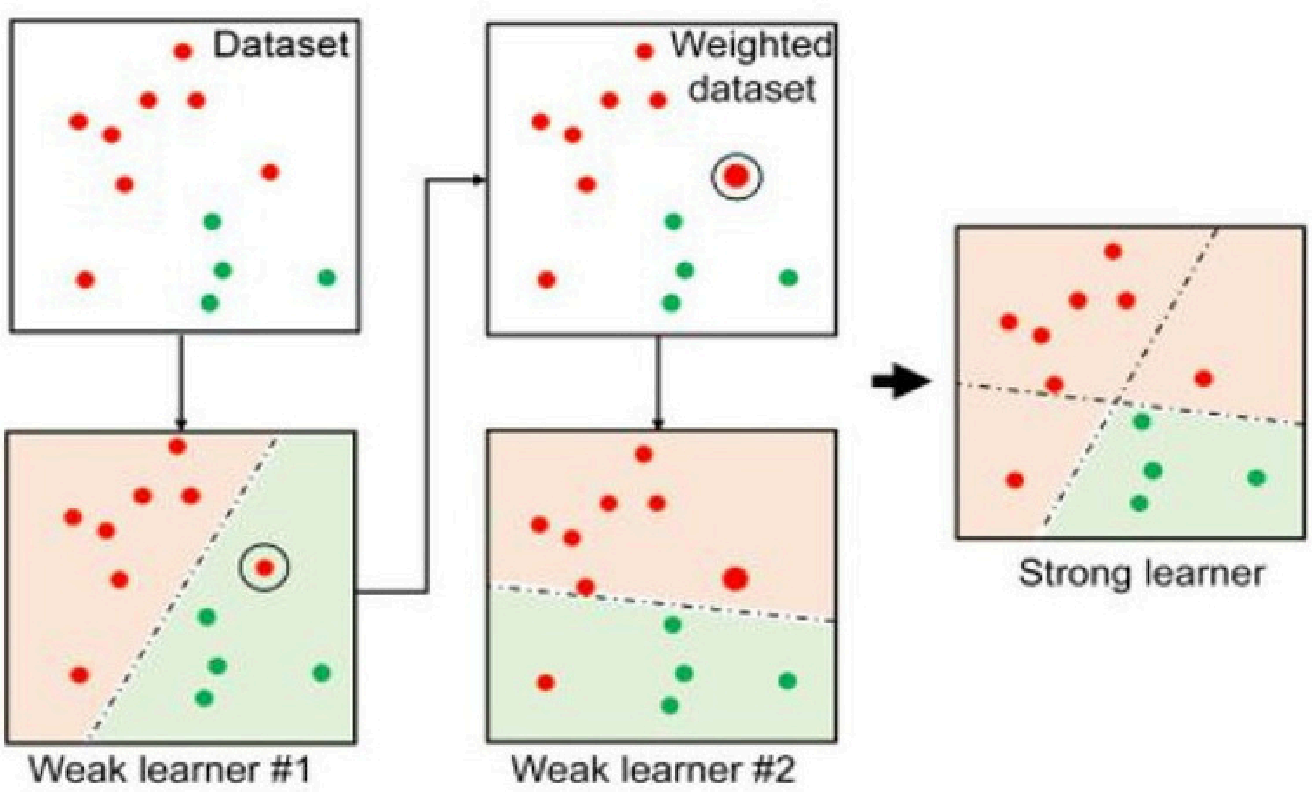
Stump2

Prediction

Weight

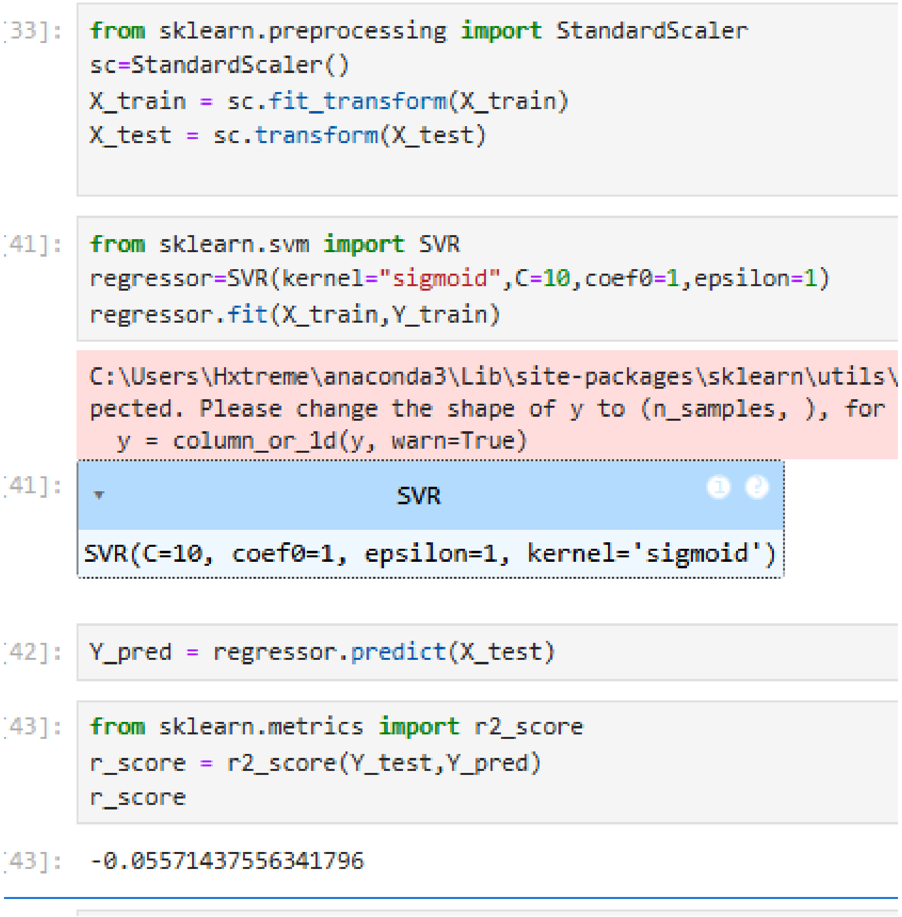
Stump2

Transformation of Weak learner into Strong learner:



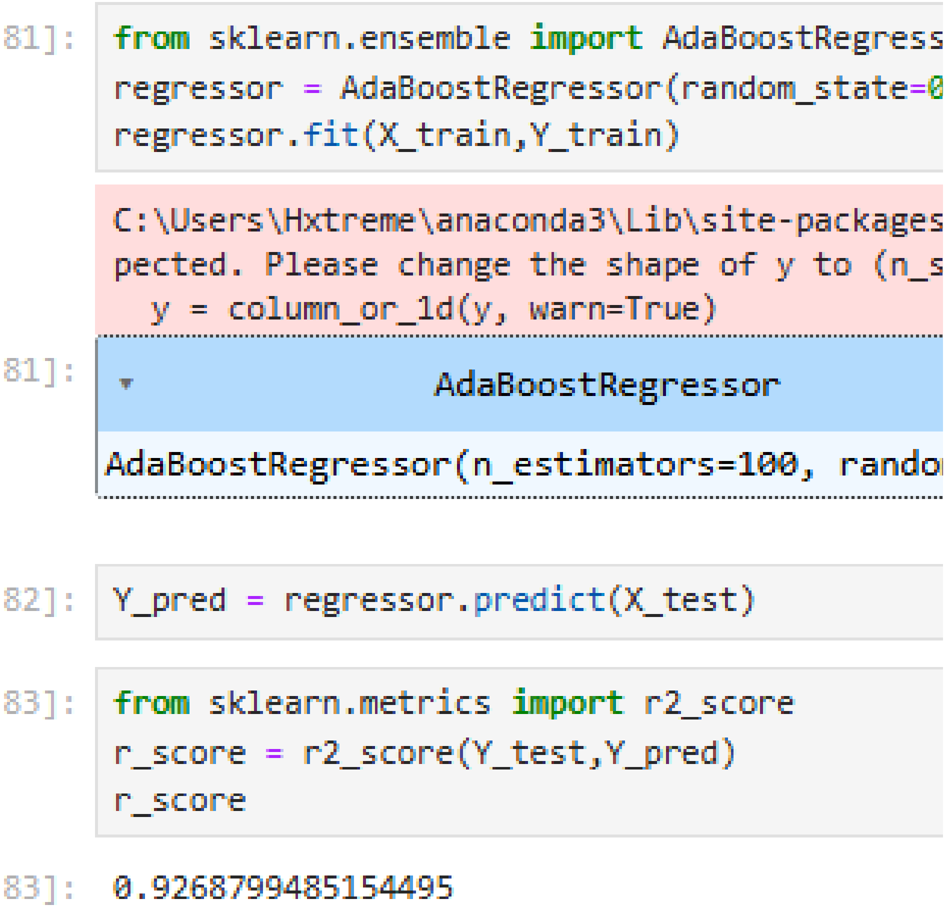
AdaBoost algorithm

Before applying



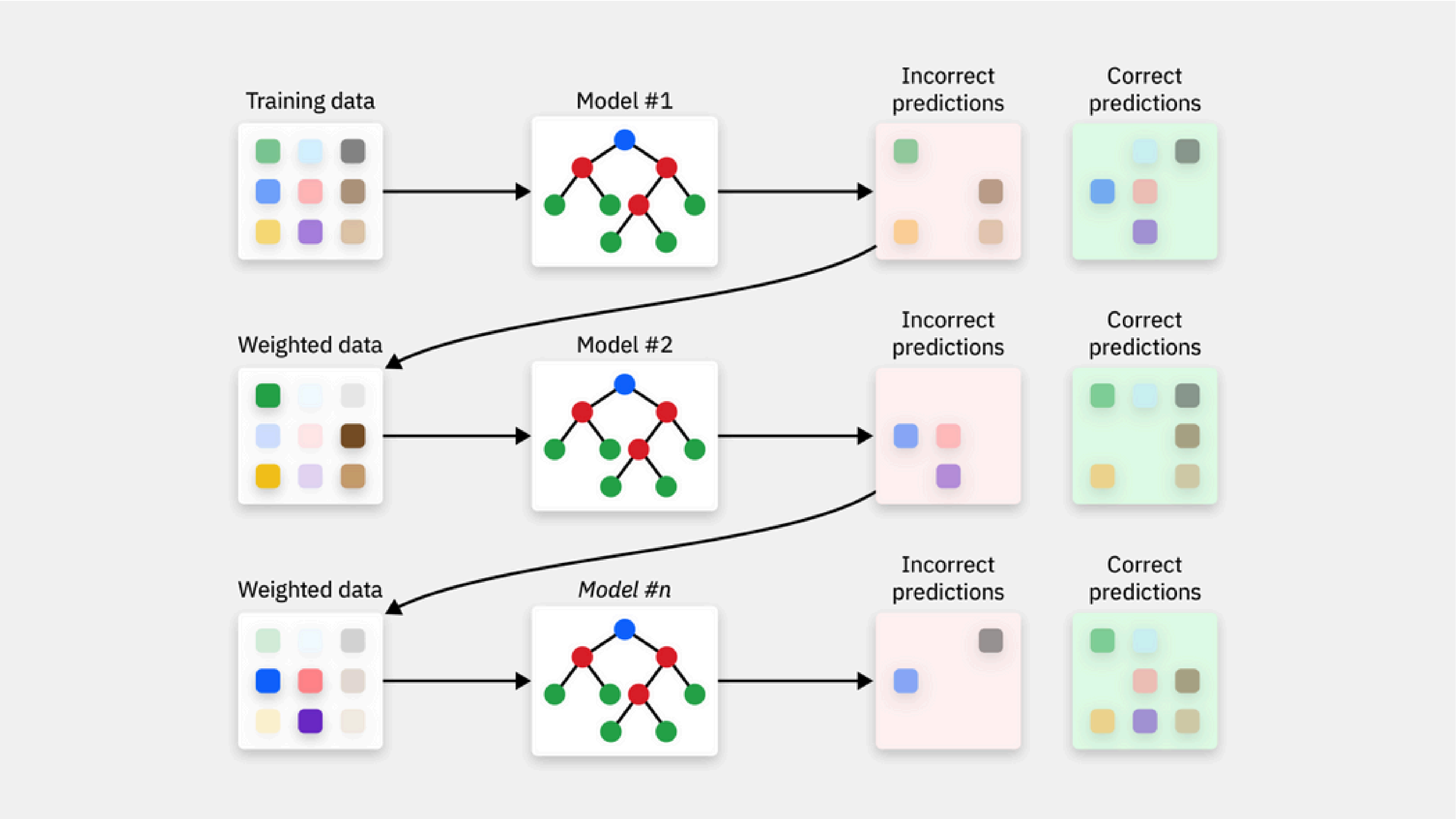
```
[33]: from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      X_train = sc.fit_transform(X_train)
      X_test = sc.transform(X_test)

[41]: from sklearn.svm import SVR
      regressor=SVR(kernel="sigmoid",C=10,coef0=1,epsilon=1)
      regressor.fit(X_train,Y_train)

      C:\Users\Hxtreme\anaconda3\Lib\site-packages\sklearn\utils\
      pected. Please change the shape of y to (n_samples, ), for
        y = column_or_1d(y, warn=True)

[41]:                    SVR
      SVR(C=10, coef0=1, epsilon=1, kernel='sigmoid')

[42]: Y_pred = regressor.predict(X_test)

[43]: from sklearn.metrics import r2_score
      r_score = r2_score(Y_test,Y_pred)
      r_score

[43]: -0.05571437556341796
```

After applying

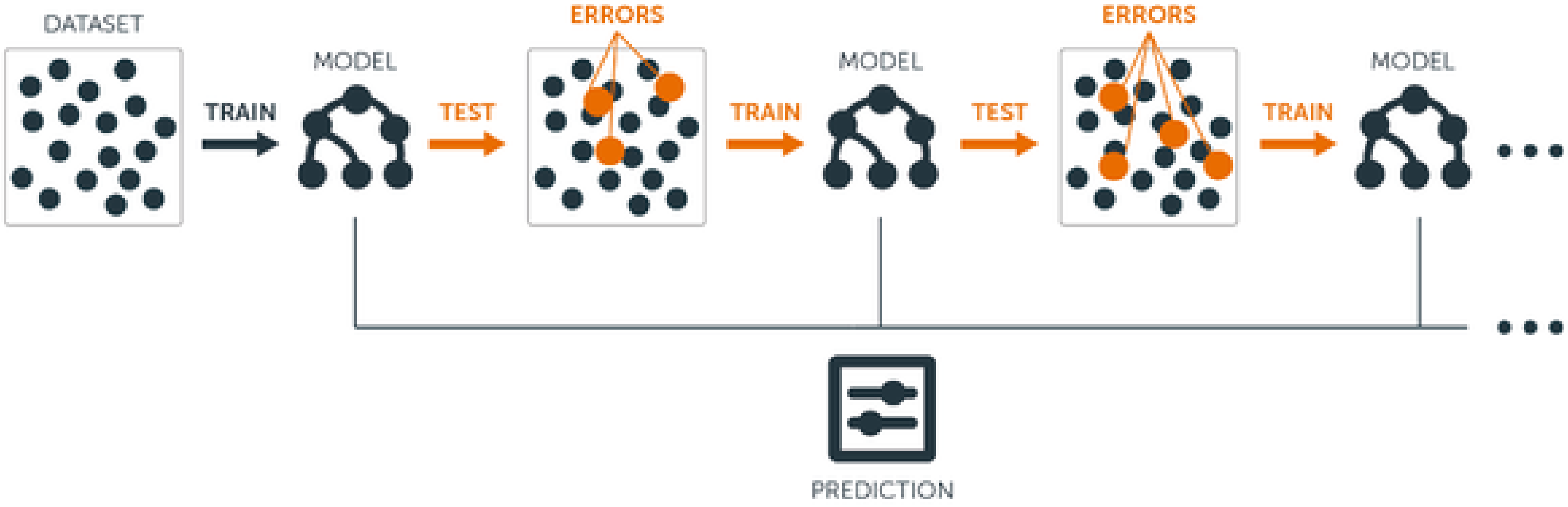


```
81]: from sklearn.ensemble import AdaBoostRegress
     regressor = AdaBoostRegressor(random_state=0
     regressor.fit(X_train,Y_train)

     C:\Users\Hxtreme\anaconda3\Lib\site-packages
     pected. Please change the shape of y to (n_s
       y = column_or_1d(y, warn=True)

81]:          AdaBoostRegressor
     AdaBoostRegressor(n_estimators=100, rando

82]: Y_pred = regressor.predict(X_test)

83]: from sklearn.metrics import r2_score
     r_score = r2_score(Y_test,Y_pred)
     r_score

83]: 0.9268799485154495
```

2)Gradiant Boost

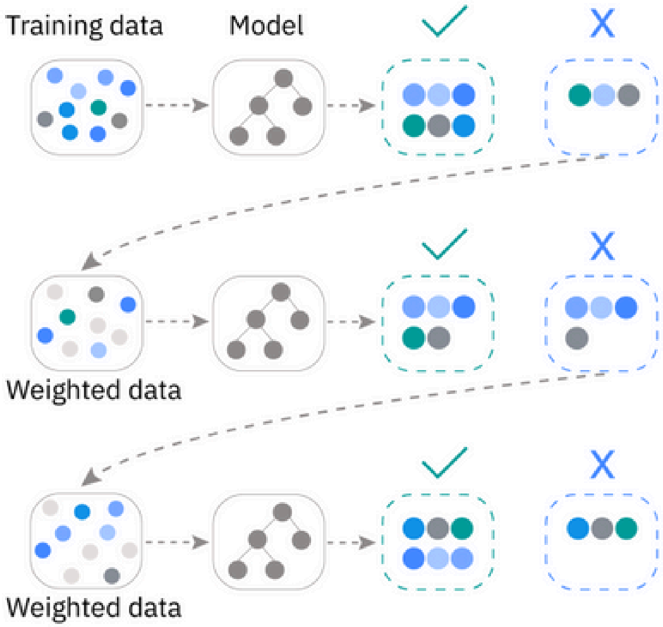


Flow Diagram for Gradiant Boost:



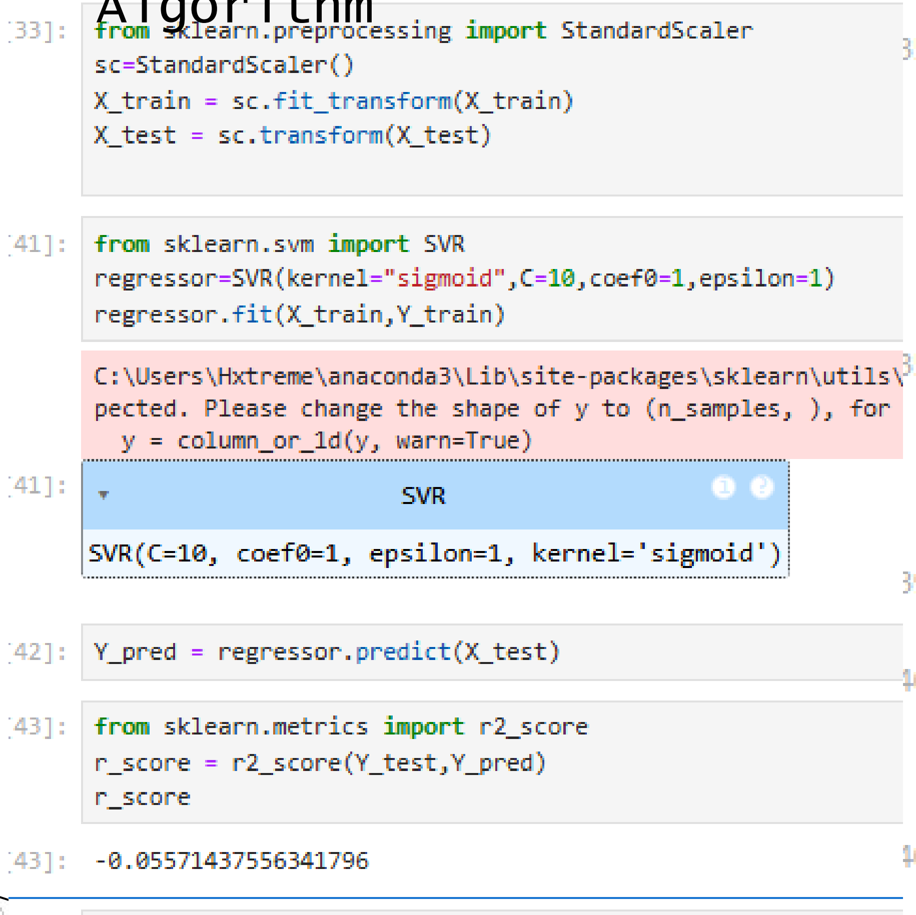
About Gradiant Boosting:

- Gradient boosting is a machine learning technique that combines multiple weak prediction models into a single ensemble.
- The model improves sequentially but not incremental weight.
- These weak models are typically decision trees, which are trained sequentially to minimize errors and improve accuracy.
- By combining multiple decision tree regressors or decision tree classifiers, gradient boosting can effectively capture complex relationships between features.


Training data
Model
✓
X
✓
X
Weighted data
✓
X
Weighted data

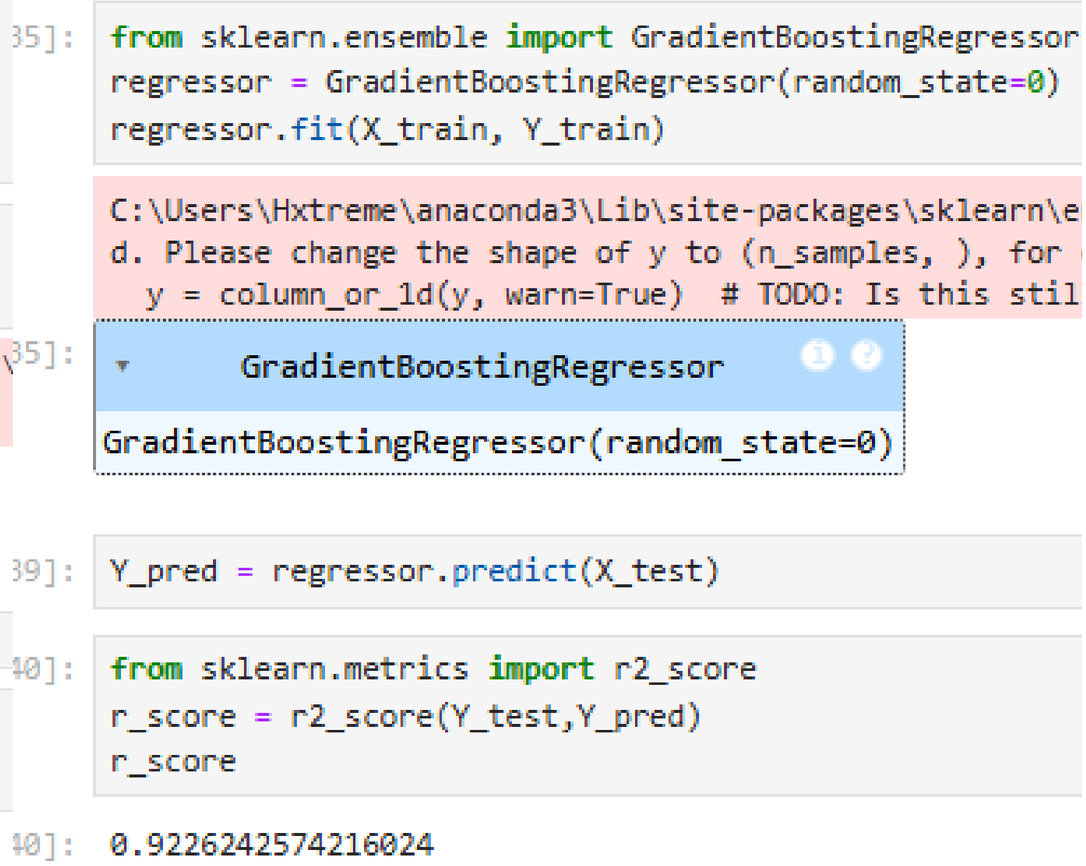
Gradiant Boosting Regressor

Before Boosting Algorithm



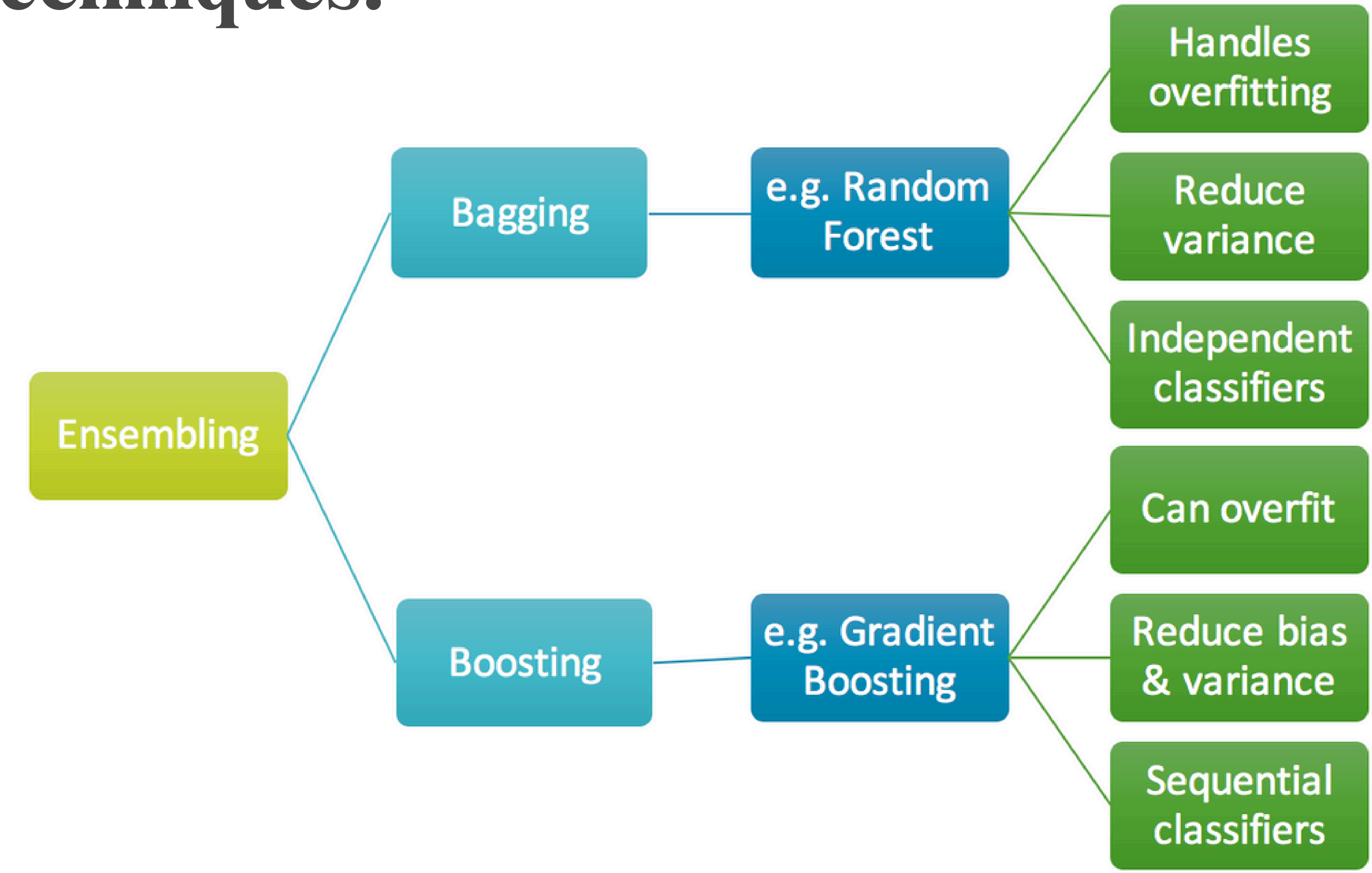
```
33]: from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)

41]: from sklearn.svm import SVR
     regressor=SVR(kernel="sigmoid",C=10,coef0=1,epsilon=1)
     regressor.fit(X_train,Y_train)

     C:\Users\Hxtreme\anaconda3\Lib\site-packages\sklearn\utils\
     pected. Please change the shape of y to (n_samples, ), for
       y = column_or_1d(y, warn=True)

41]: SVR
     SVR(C=10, coef0=1, epsilon=1, kernel='sigmoid')

42]: Y_pred = regressor.predict(X_test)

43]: from sklearn.metrics import r2_score
     r_score = r2_score(Y_test,Y_pred)
     r_score

43]: -0.05571437556341796
```

After Boosting Algorithm



```
35]: from sklearn.ensemble import GradientBoostingRegressor
     regressor = GradientBoostingRegressor(random_state=0)
     regressor.fit(X_train, Y_train)

     C:\Users\Hxtreme\anaconda3\Lib\site-packages\sklearn\e
     d. Please change the shape of y to (n_samples, ), for
       y = column_or_1d(y, warn=True)  # TODO: Is this stil

35]: GradientBoostingRegressor
     GradientBoostingRegressor(random_state=0)

39]: Y_pred = regressor.predict(X_test)

40]: from sklearn.metrics import r2_score
     r_score = r2_score(Y_test,Y_pred)
     r_score

40]: 0.9226242574216024
```

Comparison between two ensembling Techniques:



Ensembling

Bagging

e.g. Random Forest

Handles overfitting

Reduce variance

Independent classifiers

Boosting

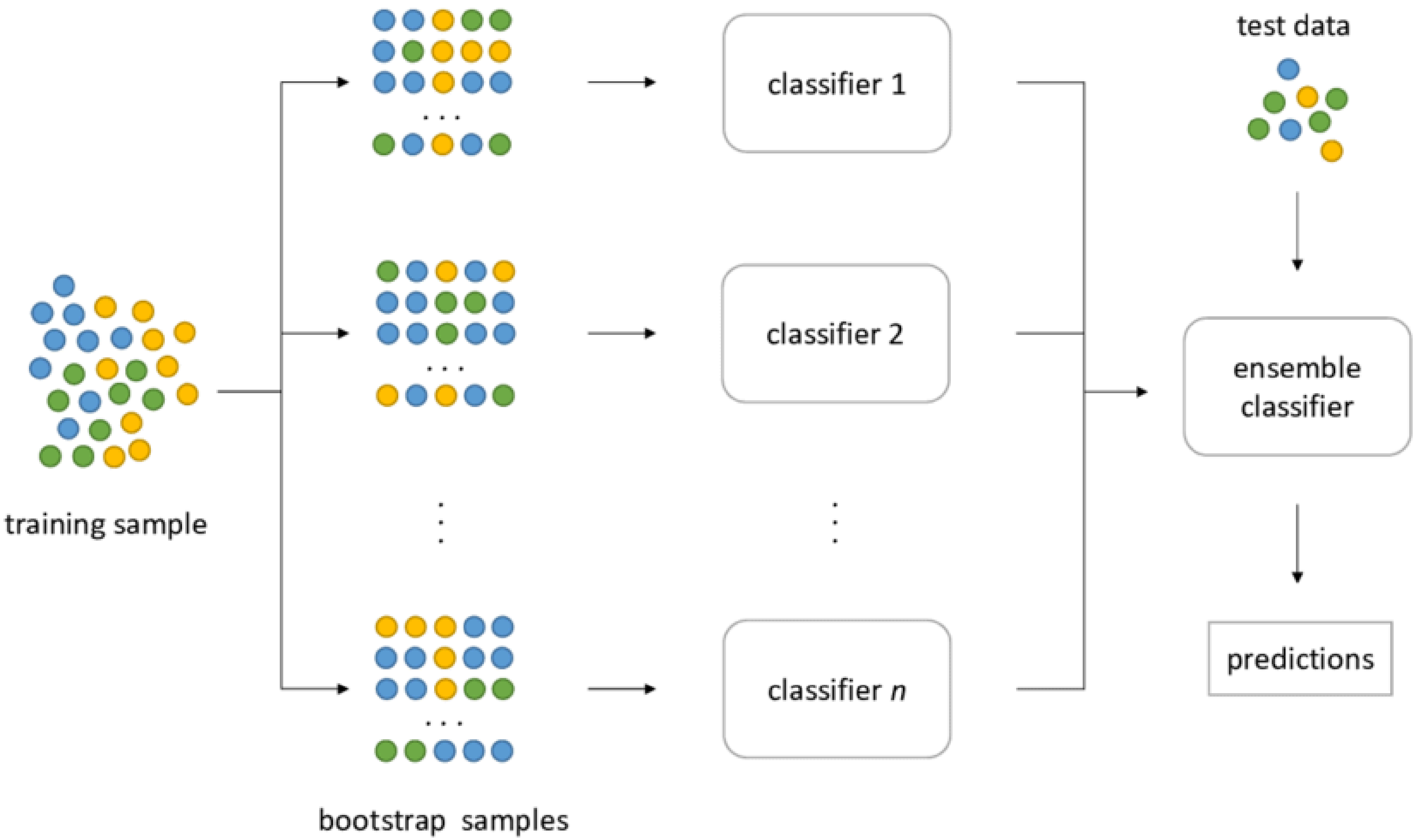
e.g. Gradient Boosting

Can overfit

Reduce bias & variance

Sequential classifiers

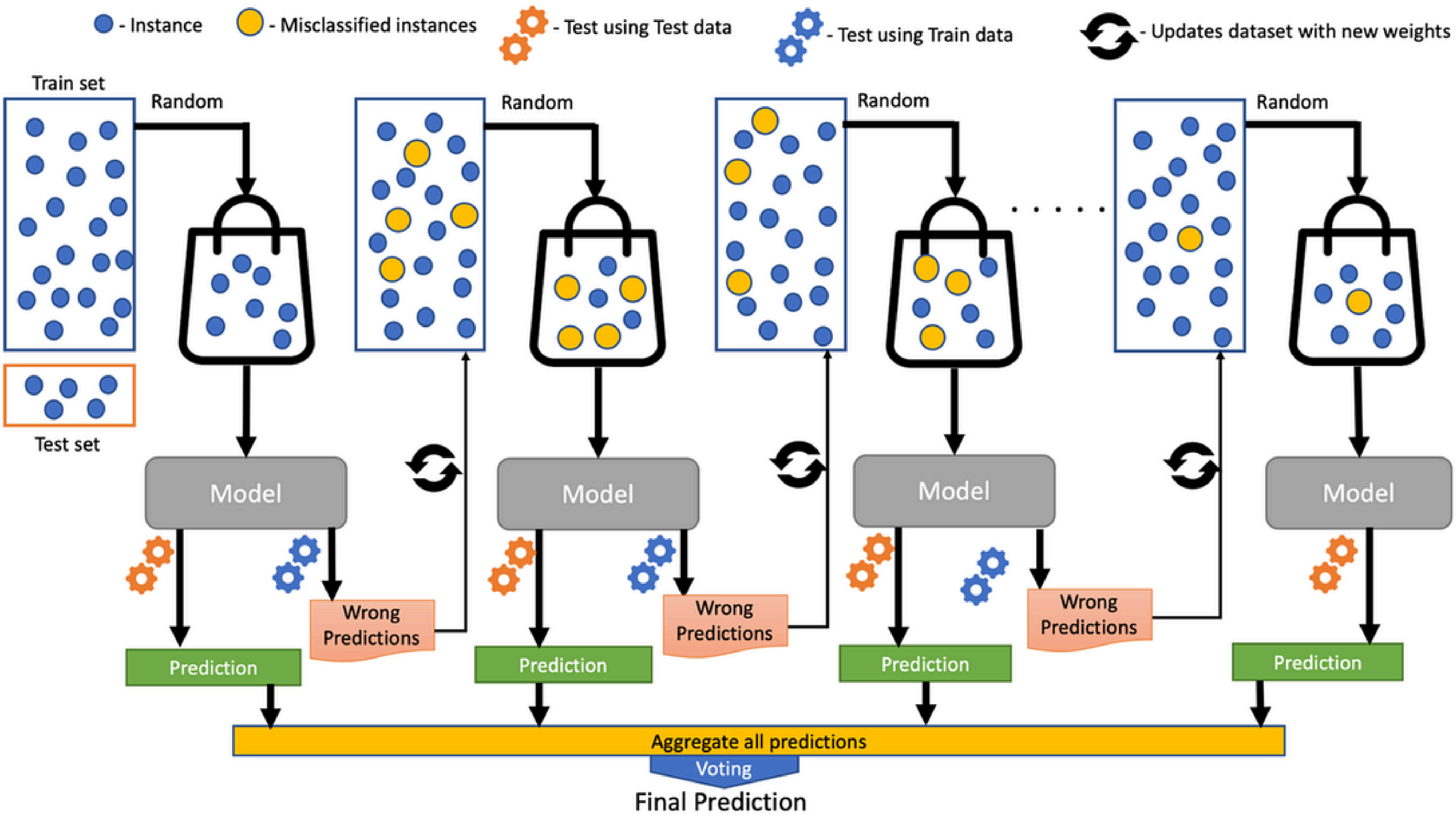
i) XG Boost


test data
training sample
classifier 1
classifier 2
classifier *n*
ensemble classifier
predictions
bootstrap samples

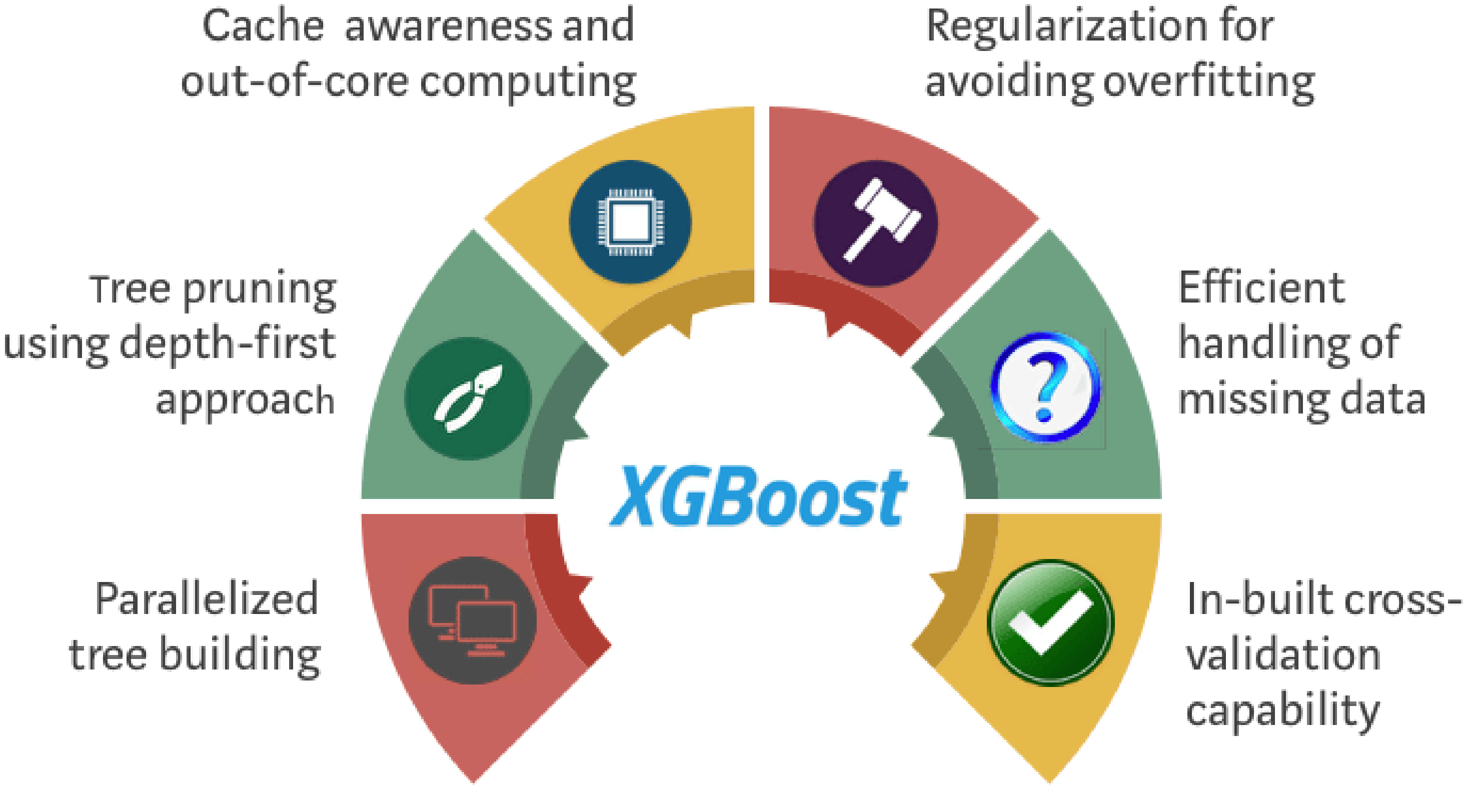
XG Boost(Extreme Gradiant) working:

- It is an implementation of gradient boosting to provide better speed and performance. It is decision tree based ensemble machine learning algorithm.
- It works same like gradiant boosting but it has additional functionality.
- XGBoost predicts with greater accuracy and with less time complexity as compared to other machine learning algortihms.
- It is a distributed Machine Learning Process.
- It can handle large datasets.

Flow Architecture:

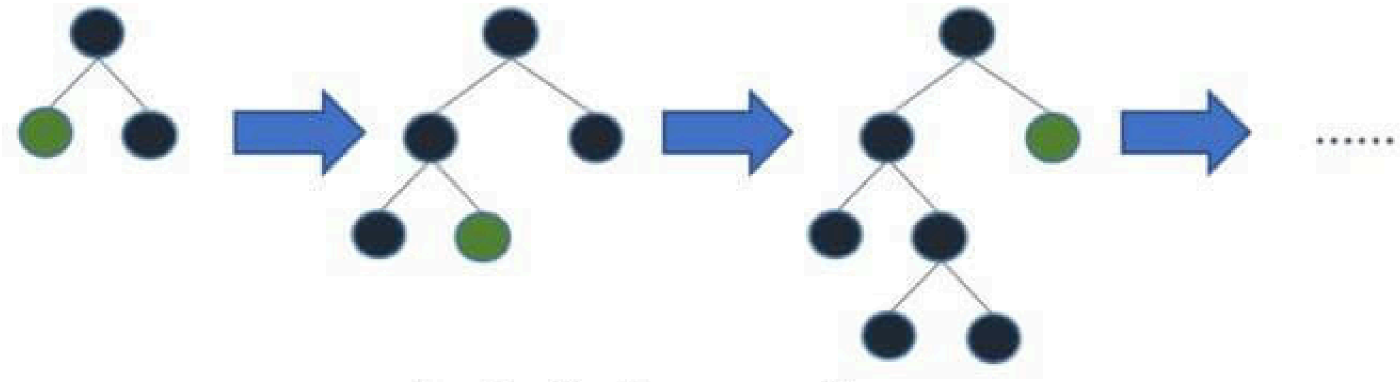


Advantages of XG Boosting:



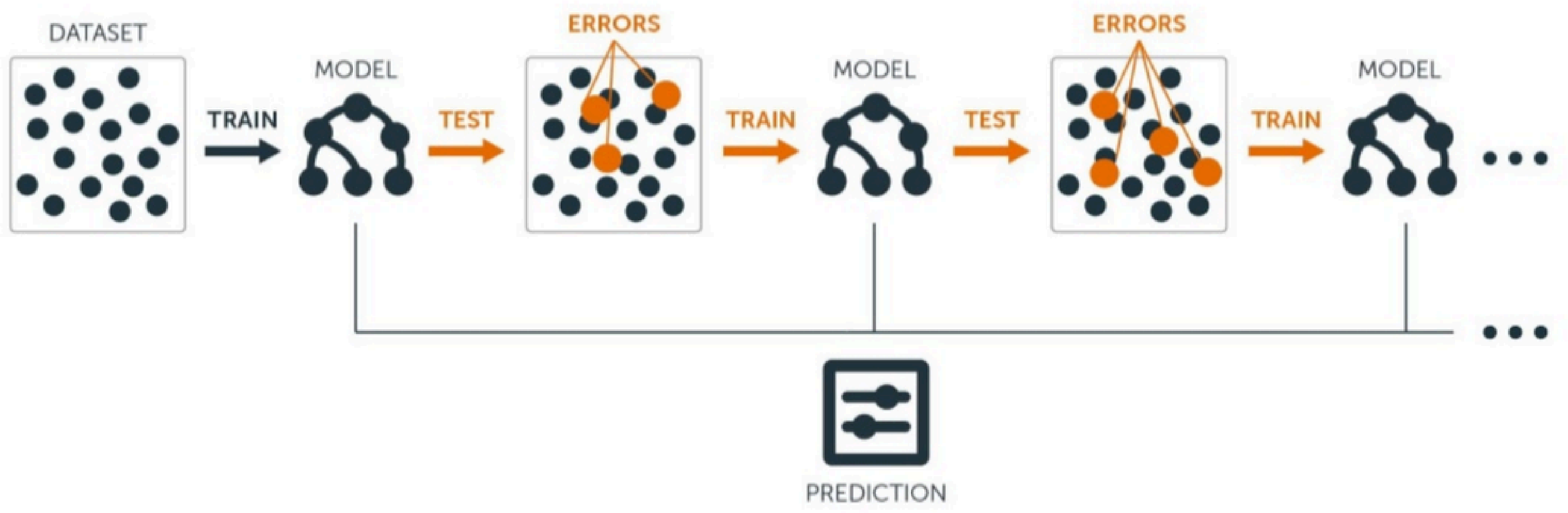
Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

ii) LG Boost Algorithm



Leaf-wise tree growth

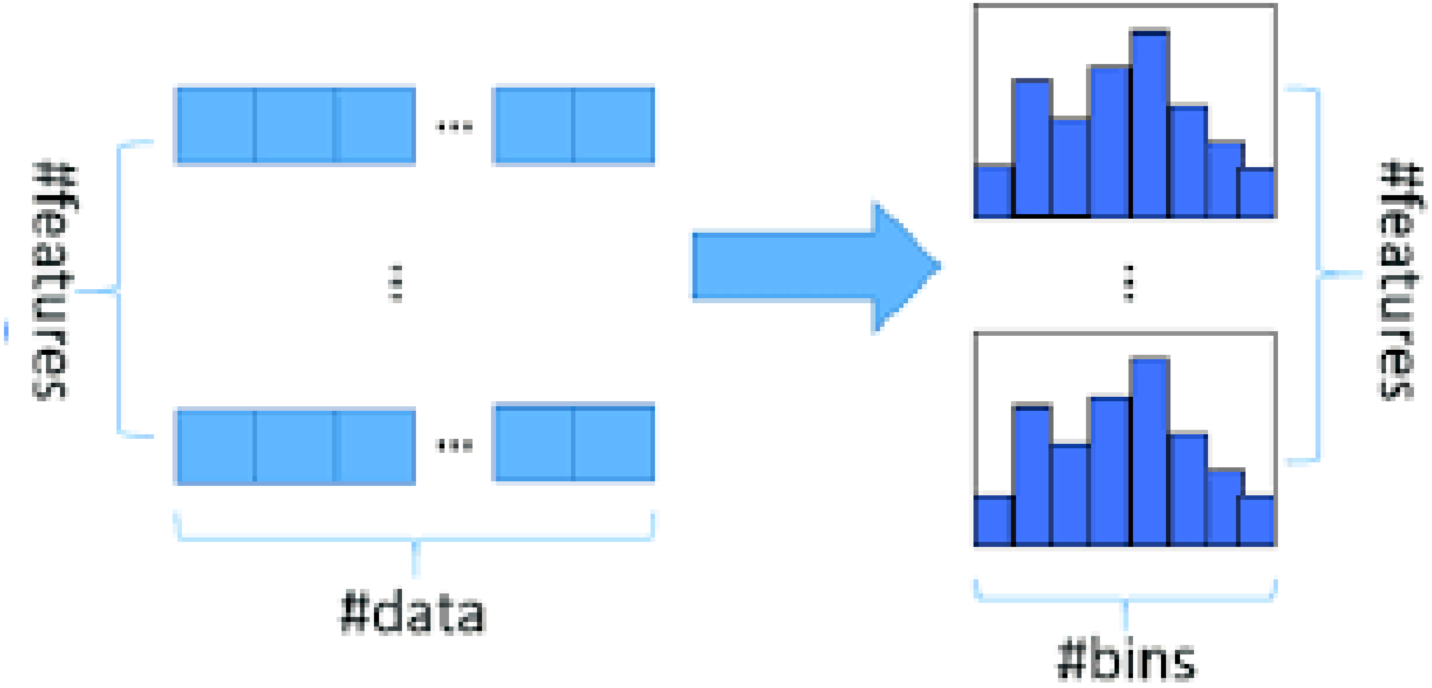
Flow Architecture for LG Boosting:



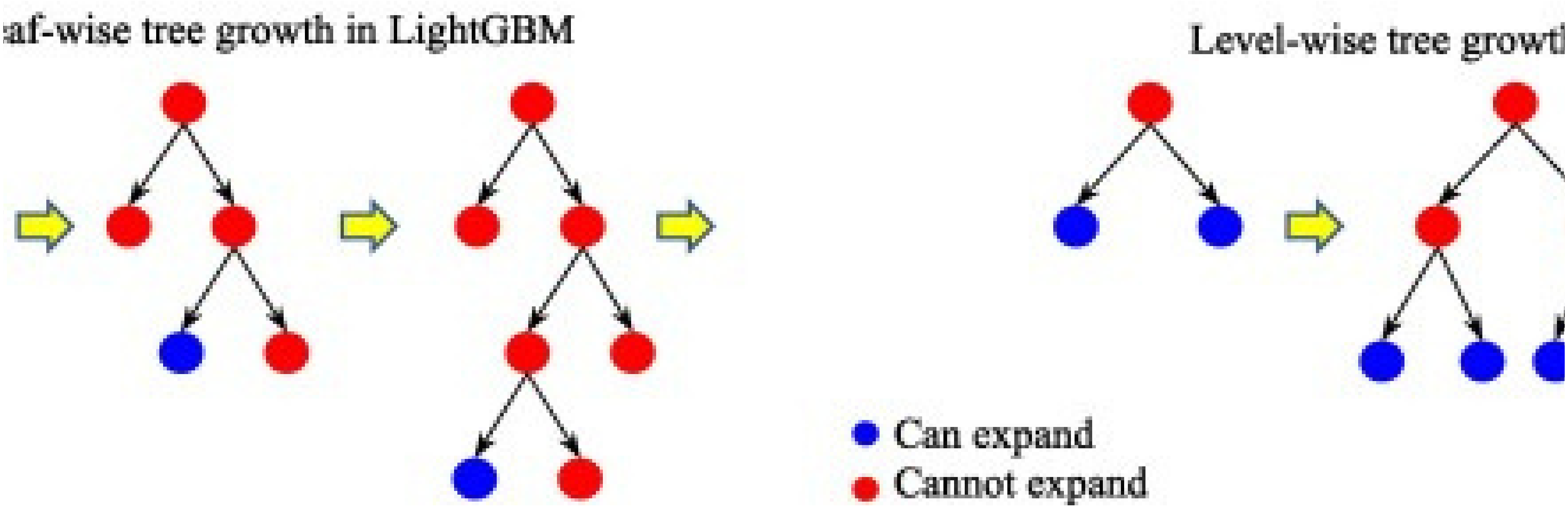
LG Boost(Light Gradiant Boosting)

- LightGBM is an open-source high-performance framework developed by Microsoft.
- It is an ensemble learning framework that uses gradient boosting method which constructs a strong learner by sequentially adding weak learners in a gradient descent manner.
- It uses histogram method for selecting the best fit.
- It can handle huge amount of data and poor to handle small dataset.

LG Boosting uses histogram based method for continuous split into bins or buckets



LG Boost Vs XG Boost

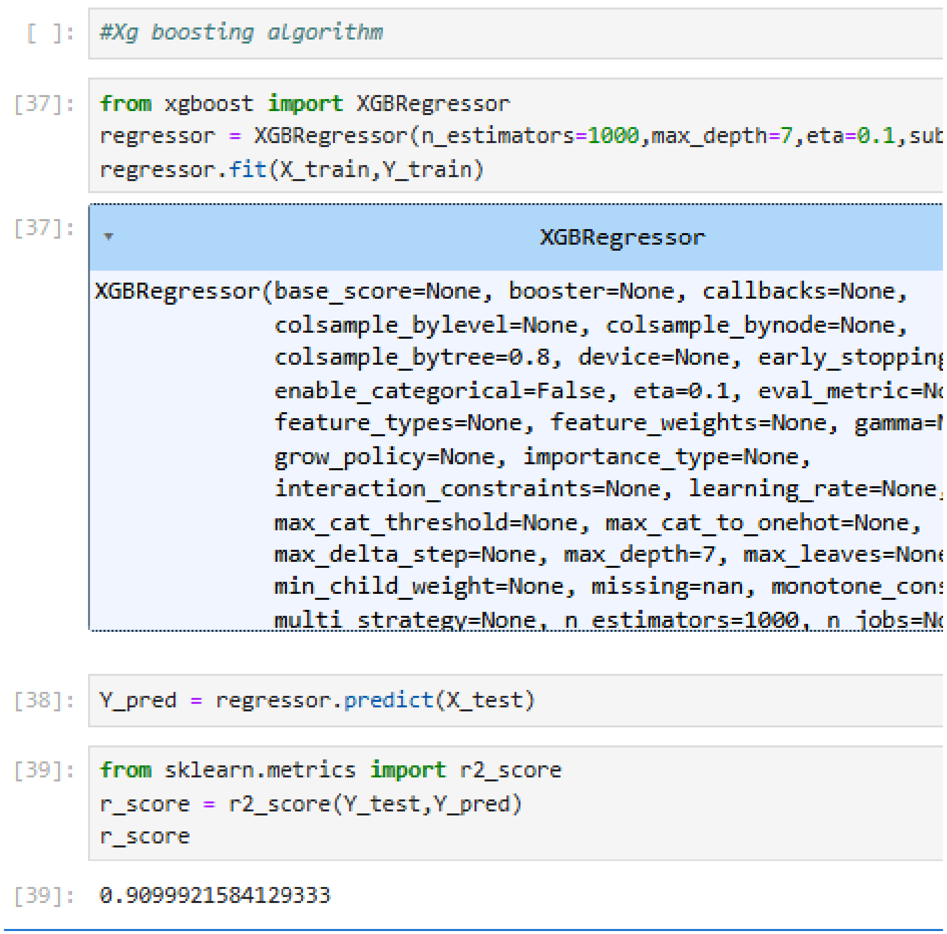


Difference between XG and LGBM

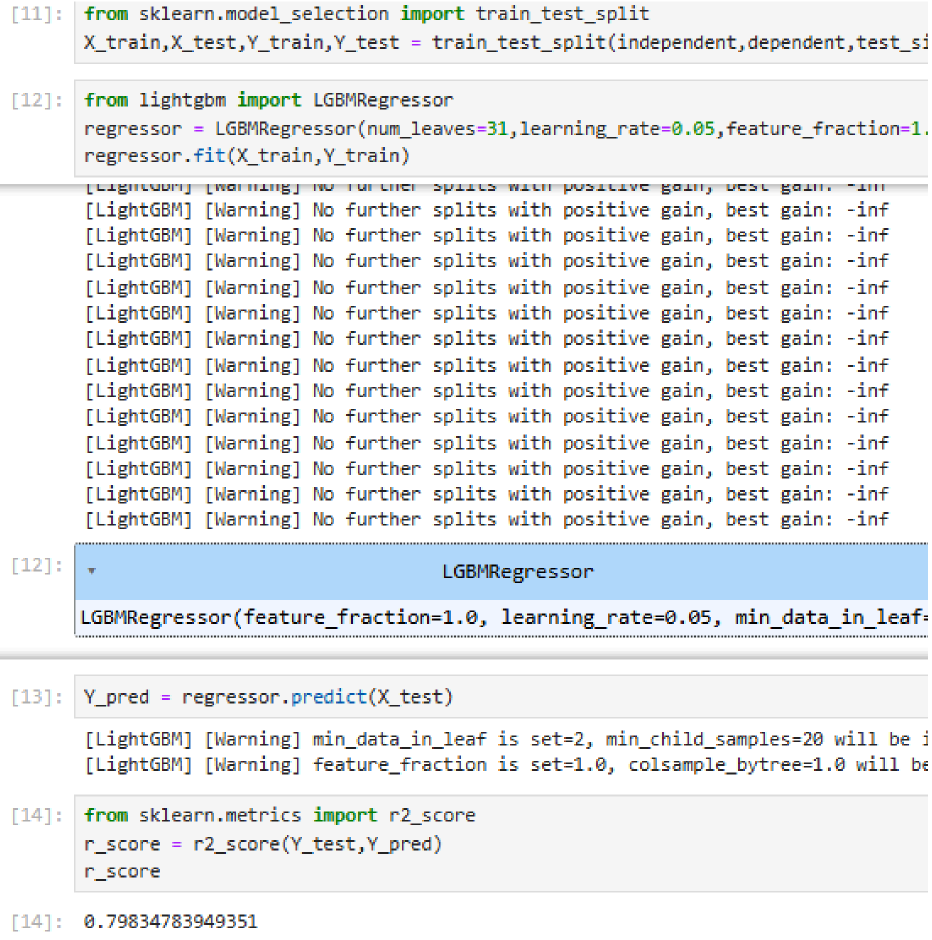
Aspect	XG Boost	Light GBM
Tree Growth	Level-wise	Leaf wise
Training Speed	Slower	Faster
Risk of Over fitting	Lower	Higher
CPU Performance	Strong	Moderate
Categorical Support Feature	Limited	Native

*Before boosting R2_score = -0.05

XGBoost LGBM



```
[ ]: #Xg boosting algorithm

[37]: from xgboost import XGBRegressor
      regressor = XGBRegressor(n_estimators=1000,max_depth=7,eta=0.1,sub
      regressor.fit(X_train,Y_train)

[37]: ▾                    XGBRegressor
      XGBRegressor(base_score=None, booster=None, callbacks=None,
                   colsample_bylevel=None, colsample_bynode=None,
                   colsample_bytree=0.8, device=None, early_stopping
                   enable_categorical=False, eta=0.1, eval_metric=No
                   feature_types=None, feature_weights=None, gamma=N
                   grow_policy=None, importance_type=None,
                   interaction_constraints=None, learning_rate=None,
                   max_cat_threshold=None, max_cat_to_onehot=None,
                   max_delta_step=None, max_depth=7, max_leaves=None
                   min_child_weight=None, missing=nan, monotone_cons
                   multi_strategy=None, n_estimators=1000, n_jobs=No

[38]: Y_pred = regressor.predict(X_test)

[39]: from sklearn.metrics import r2_score
      r_score = r2_score(Y_test,Y_pred)
      r_score

[39]: 0.9099921584129333
```



```
[11]: from sklearn.model_selection import train_test_split
      X_train,X_test,Y_train,Y_test = train_test_split(independent,dependent,test_si

[12]: from lightgbm import LGBMRegressor
      regressor = LGBMRegressor(num_leaves=31,learning_rate=0.05,feature_fraction=1.
      regressor.fit(X_train,Y_train)

      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
      [LightGBM] [Warning] No further splits with positive gain, best gain: -inf

[12]: ▾                    LGBMRegressor
      LGBMRegressor(feature_fraction=1.0, learning_rate=0.05, min_data_in_leaf=

[13]: Y_pred = regressor.predict(X_test)

      [LightGBM] [Warning] min_data_in_leaf is set=2, min_child_samples=20 will be i
      [LightGBM] [Warning] feature_fraction is set=1.0, colsample_bytree=1.0 will be

[14]: from sklearn.metrics import r2_score
      r_score = r2_score(Y_test,Y_pred)
      r_score

[14]: 0.79834783949351
```

https://github.com/Geetharani-CodeAI/HopeAI-Assignments/blob/main/Boosting.ipynb