| Machine learning
XGBoost

Mussa Akbota

Overview

Given document contains thoughts and some comments during writing the code. I've decided to separate implementation part and thinking part because two sides in one document make document overloaded.

Tools

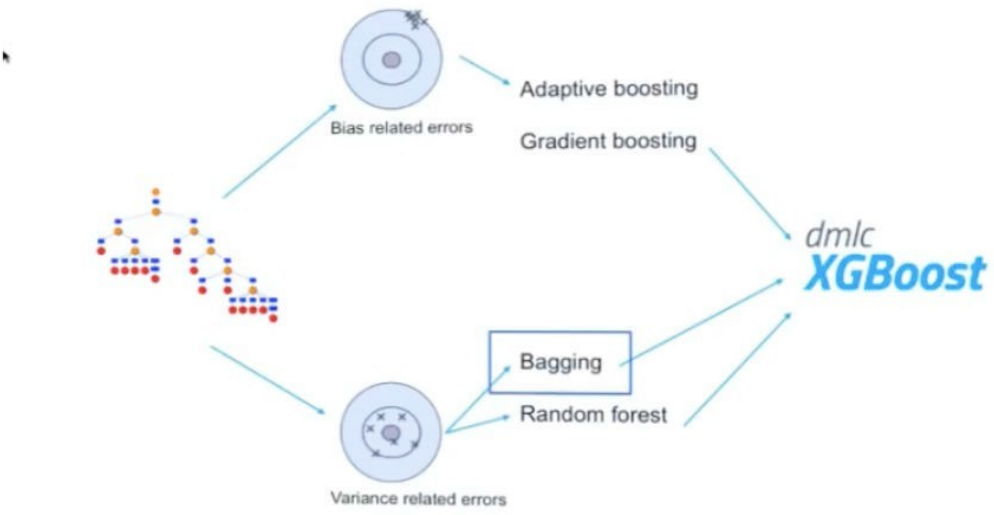
Numpy - for working with array
Pandas - for working with datasets
XGBoost - for training a model
sklearn.metrics.accuracy_score - for estimating accuracy rate
sklearn.preprocessing.LabelEncoder - for encoding label to categorical number
sklearn.preprocessing.OneHotEncoder - for encoding number to Dummy coding

Reasons of choosing XGBoost

Earlier i've asked myself can i put into one model multiple kernel. Was made a research and was obtained that exists some approaches to do it. And one from them was ensemble learning. It is't actually forming one kernel from two different, but it is a manipulation on output values by adding them weights.

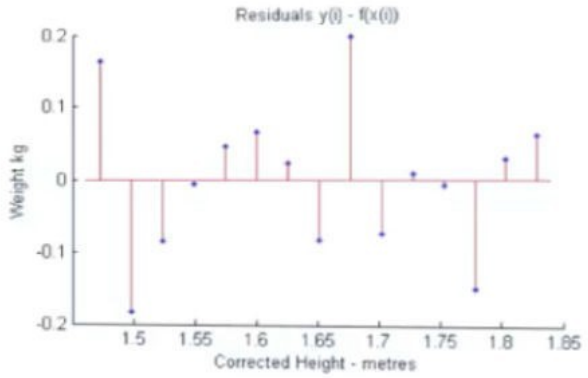
Theory

What actual is XGBOOST?



GRADIENT BOOSTING

What if we, instead of reweighting examples, made some corrections to prediction errors directly?



We add new model, to the one we already have, so:

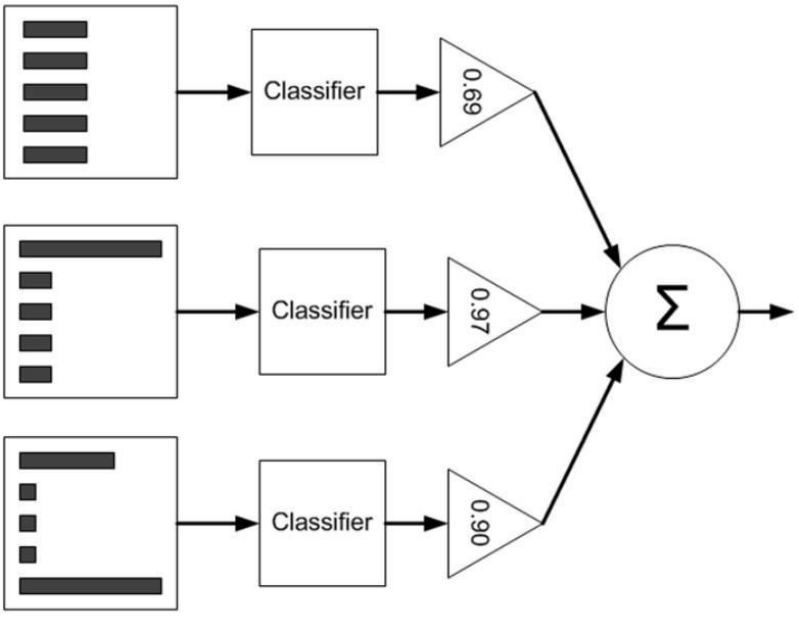
$Y_{pred} = X1(Y) + NEW(Y)$

$NEW(Y) = X1(Y) - Y_{pred}$

our residual

Note:

Residual is a gradient of single observation error contribution in one of the most common evaluation measure for regression: root mean squared error (RMSE)



Life cycle

1. Importing

2. Preprocessing
    1. Reading from .csv | Train & Test datasets
    2. Splitting into dependent & independent
    3. Analyzing on consistence of anomalies and dealing with them:
        - Existence missing value
        - Existence of categorical feature
        - Existence dummy-trap variable
    4. Prebuilding

3. Model processing
    1. Data structure preparation
    2. Building & training
    3. Predicting

    4. Benchmarking
    1. Preparing benchmark
    2. Calculating accuracy score

    1. Printing results

Comments & views

2.2 | Splitting into dependent & independent

Reasons of using columns:
Pclass - Could be important because different class may be placed in different blocks of ship.
Sex - Boys can be more faster in critical situation.
Age - Young guys more agile.
SibSP, Parch - Presence of any relatives could decrease the chances of surviving.

life cycle: 2.3 | Analyzing on consistence of anomalies and dealing with them:

Missing value

i've had 2 approaches to find out consistence of missing values:

1) By numpy method .isnan()
2) By pandas method .isnull()

I've chosed the second because it more robust (.isnan() requires to enter non object data type)
And more flexible (I've printed columns containting missing values.)

Then imputring because Multiple weak treas with a small depth more robust to noises. it's a feature of bagging approach.

Categorical feature

In our case we have variable column 'Sex' which has to be transformed:
from string representation to number representation and number to binary.
And converting to binary representation lets to dummy-trap vars which has to be eliminated by slicing.

And second one is Pclass which already is representing category
This column need transformation to binary and also removing dummy-trap