Photo by NeONBRAND on Unsplash



Member-only story

# Gradient Boosting Decision Tree Algorithm Explained

Cory Maklin · Follow
Published in Towards Data Science
5 min read · May 18, 2019

Listen Share More

In the proceeding article, we'll take a look at how we can go about implementing Gradient Boost in Python. Gradient Boosting is similar to AdaBoost in that they both use an ensemble of decision trees to predict a target label. However, unlike AdaBoost, the Gradient Boost trees have a depth larger than 1. In practice, you'll typically see Gradient Boost being used with a maximum number of leaves of between 8 and 32.

## Algorithm

Before we dive into the code, it's important that we grasp how the Gradient Boost algorithm is implemented under the hood. Suppose, we were trying to predict the price of a house given their age, square footage and location.

| age | square footage | location | price |
|---|---|---|---|
| 5 | 1500 | 5 | 480 |
| 11 | 2030 | 12 | 1090 |
| 14 | 1442 | 6 | 350 |
| 8 | 2501 | 4 | 1310 |
| 12 | 1300 | 9 | 400 |
| 10 | 1789 | 11 | 500 |

### Step 1: Calculate the average of the target label

When tackling regression problems, we start with a leaf that is the average value of the variable we want to predict. This leaf will be used as a baseline to approach the correct solution in the proceeding steps.

$$\frac{490+1090+350+1310+400+500}{6} = 688$$

688

**Step 2: Calculate the residuals**

For every sample, we calculate the residual with the proceeding formula.

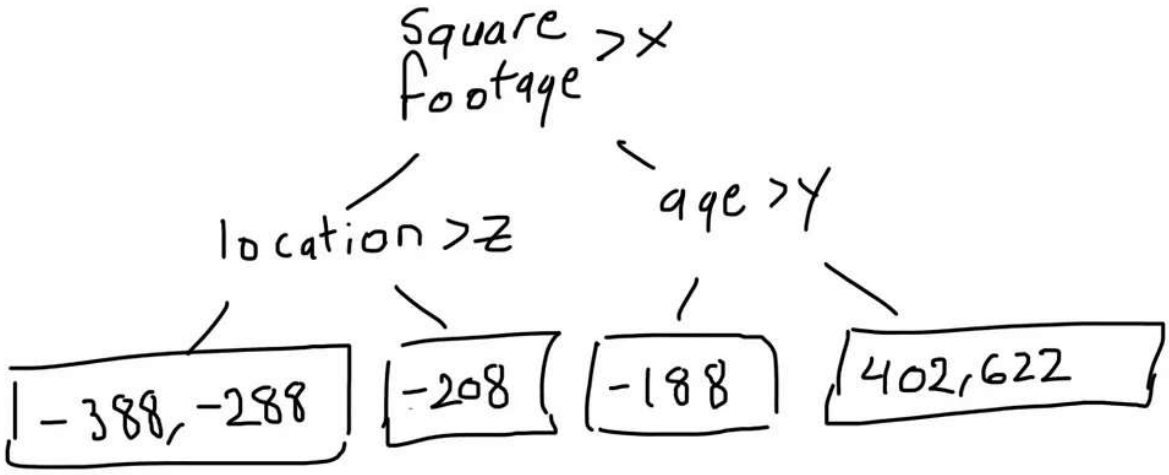
$$\text{residual} = \text{actual value} - \text{predicted value}$$

In our example, the predicted value is the equal to the mean calculated in the previous step and the actual value can be found in the price column of each sample. After computing the residuals, we get the following table.

| age | square footage | location | price | residuals |
|---|---|---|---|---|
| 5 | 1500 | 5 | 480 | -208 |
| 11 | 2030 | 12 | 1090 | 402 |
| 14 | 1442 | 6 | 350 | -338 |
| 8 | 2501 | 4 | 1310 | 622 |
| 12 | 1300 | 9 | 400 | -288 |
| 10 | 1789 | 11 | 500 | -188 |

**Step 3: Construct a decision tree**

Next, we build a tree with the goal of predicting the residuals. In other words, every leaf will contain a prediction as to the value of the residual (not the desired label).

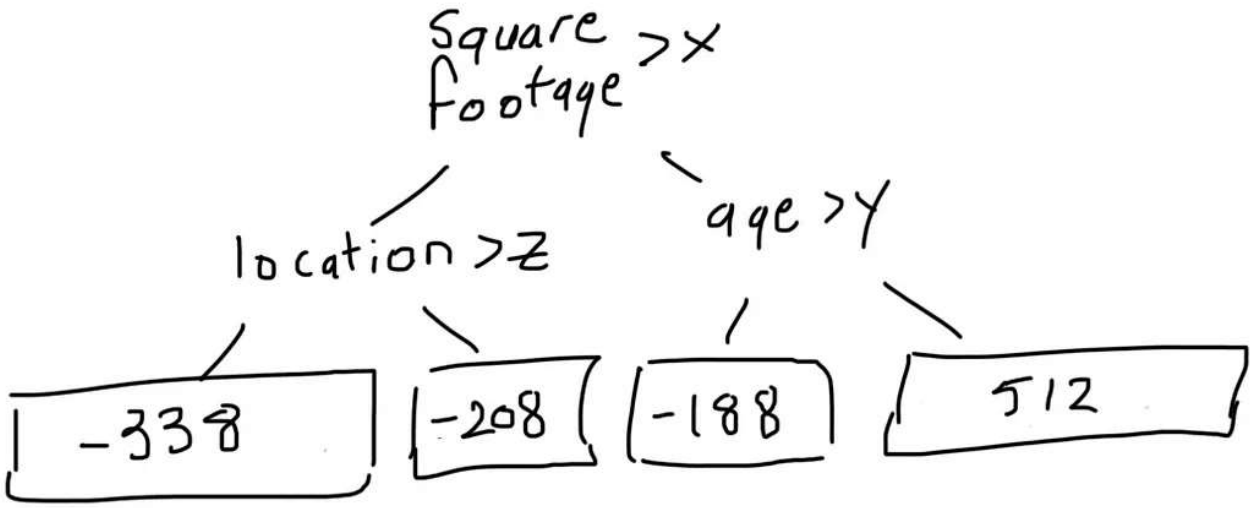



In the event there are more residuals than leaves, some residuals will end up inside the same leaf. When this happens, we compute their average and place that inside the leaf.

$$\frac{-388 + -288}{2} = -338$$

$$\frac{402 + 622}{2} = 512$$

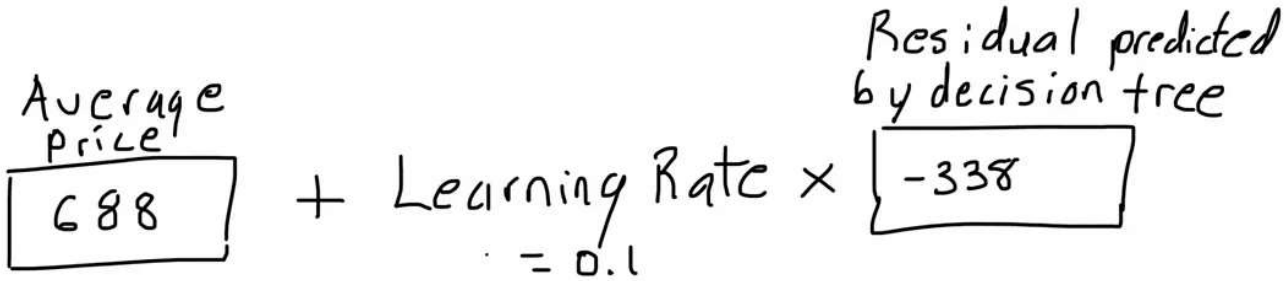
Thus, the tree becomes:

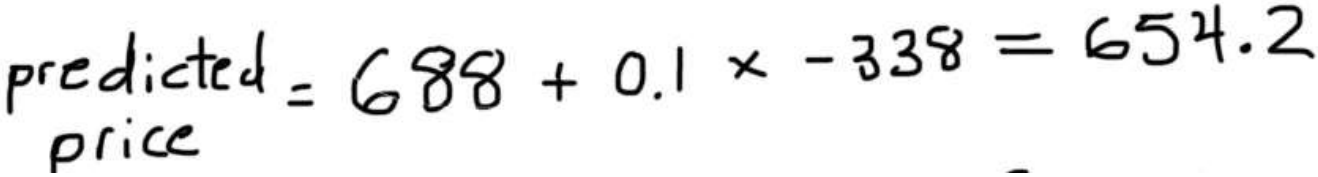



**Step 4: Predict the target label using all of the trees within the ensemble**

Each sample passes through the decision nodes of the newly formed tree until it reaches a given lead. The residual in said leaf is used to predict the house price.

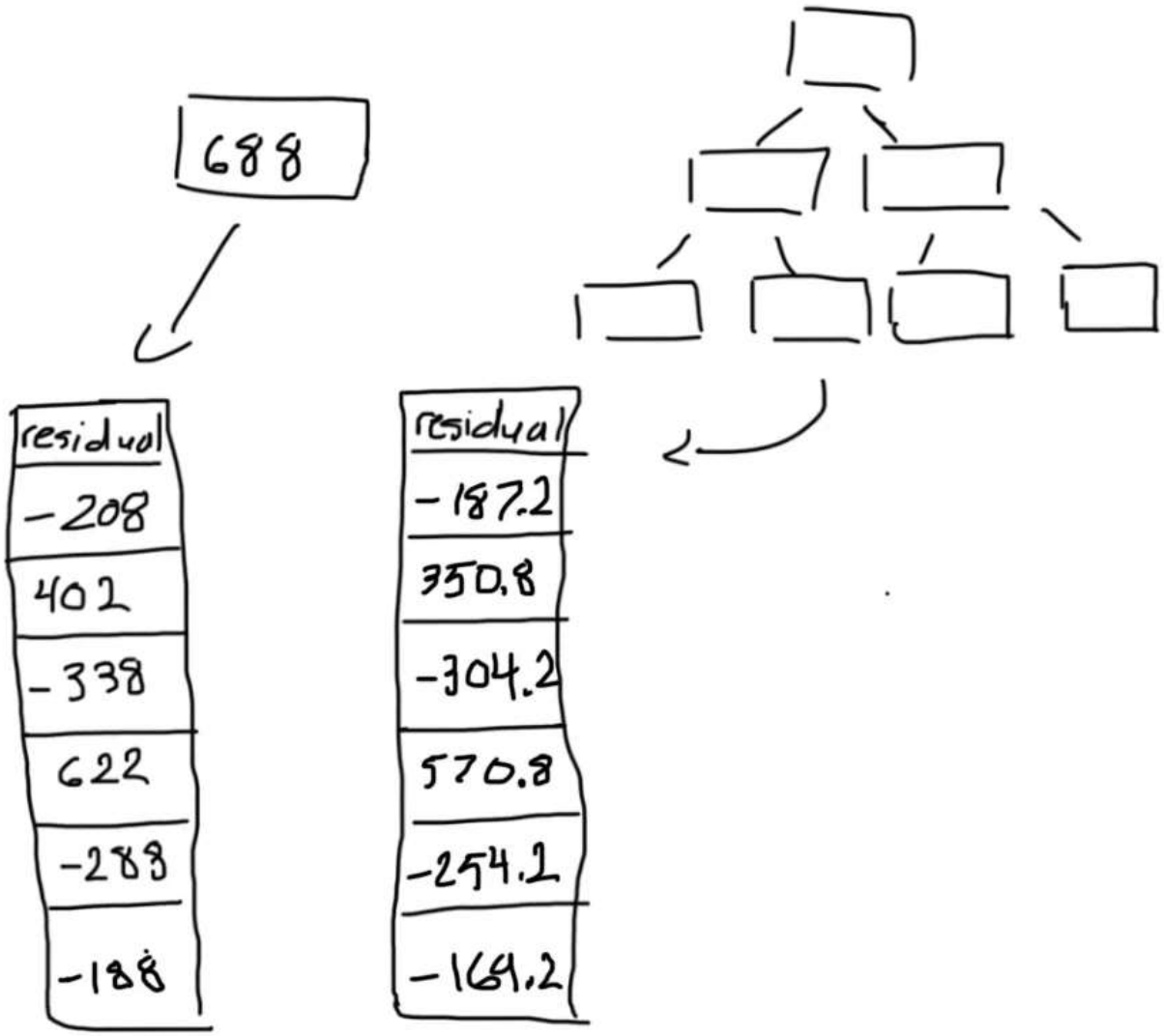
It's been shown through experimentation that taking small incremental steps towards the solution achieves a comparable bias with a lower overall vatiance (a lower variance leads to better accuracy on samples outside of the training data). Thus, to prevent overfitting, we introduce a hyperparameter called learning rate. When we make a prediction, each residual is multiplied by the learning rate. This forces us to use more decision trees, each taking a small step towards the final solution.



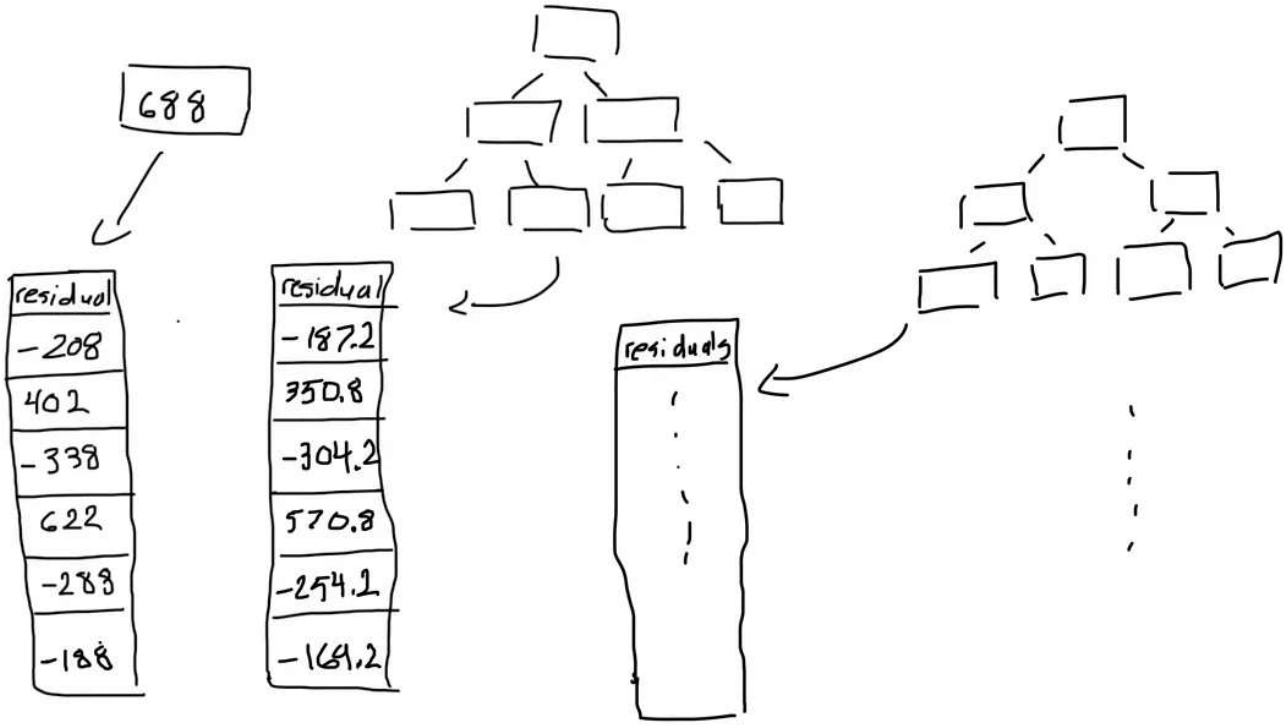




**Step 5: Compute the new residuals**

We calculate a new set of residuals by subtracting the actual house prices from the predictions made in the previous step. The residuals will then be used for the leaves of the next decision tree as described in step 3.

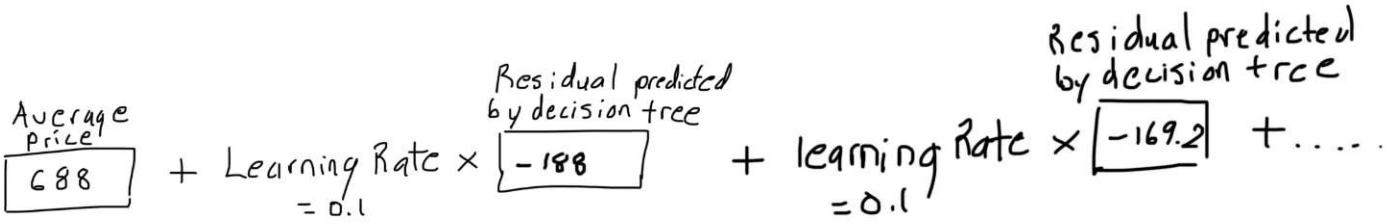



**Step 6: Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyperparameter (i.e. number of estimators)**




**Step 7: Once trained, use all of the trees in the ensemble to make a final prediction as to the value of the target variable**

The final prediction will be equal to the mean we computed in the first step, plus all of the residuals predicted by the trees that make up the forest multiplied by the learning rate.




## Code

In this tutorial, we'll make use of the **GradientBoostingRegressor** class from the **scikit-learn** library.

```
from sklearn.ensemble import GradientBoostingRegressor
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston
from sklearn.metrics import mean_absolute_error
```

For the proceeding example, we'll be using the Boston house prices dataset.

```
boston = load_boston()
X = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.Series(boston.target)
```

In order to evaluate the performance of our model, we split the data into training and test sets.

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Next, we construct and fit our model. `max_depth` refers to the number of leaves of each tree (i.e. 4) whereas `n_estimators` refers to the total number of trees in the ensemble. As mentioned previously, the `learning_rate` hyperparameter scales the contribution of each tree. If you set it to a low value, you will need more trees in the ensemble to fit the training set, but the overall variance will be lower.

```
regressor = GradientBoostingRegressor(
    max_depth=2,
    n_estimators=3,
    learning_rate=1.0
)
regressor.fit(X_train, y_train)
```

The `staged_predict()` method measures the validation error at each stage of training (i.e. with one tree, with two trees...) to find the optimal number of trees.



Now, we can build and fit our model using the optimal number of trees.

```
best_regressor = GradientBoostingRegressor(
    max_depth=2,
```

```
        n_estimators=best_n_estimators,
        learning_rate=1.0
    )
    best_regressor.fit(X_train, y_train)
```

Sklearn provides numerous metrics to evaluate the performance of our machine learning models. What I found particularly useful, it that they categorize the each metric according to the problem domain which they're applicable. For example, precision only makes sense in the context of classification.

| Scoring | Function |
|---|---|
| **Classification** | |
| 'accuracy' | metrics.accuracy_score |
| 'balanced_accuracy' | metrics.balanced_accuracy_score |
| 'average_precision' | metrics.average_precision_score |
| 'brier_score_loss' | metrics.brier_score_loss |
| 'f1' | metrics.f1_score |
| 'f1_micro' | metrics.f1_score |
| 'f1_macro' | metrics.f1_score |
| 'f1_weighted' | metrics.f1_score |
| 'f1_samples' | metrics.f1_score |
| 'neg_log_loss' | metrics.log_loss |
| 'precision' etc. | metrics.precision_score |
| 'recall' etc. | metrics.recall_score |
| 'jaccard' etc. | metrics.jaccard_score |
| 'roc_auc' | metrics.roc_auc_score |
| **Clustering** | |
| 'adjusted_mutual_info_score' | metrics.adjusted_mutual_info_score |
| 'adjusted_rand_score' | metrics.adjusted_rand_score |
| 'completeness_score' | metrics.completeness_score |
| 'fowlkes_mallows_score' | metrics.fowlkes_mallows_score |
| 'homogeneity_score' | metrics.homogeneity_score |
| 'mutual_info_score' | metrics.mutual_info_score |
| 'normalized_mutual_info_score' | metrics.normalized_mutual_info_score |
| 'v_measure_score' | metrics.v_measure_score |
| **Regression** | |
| 'explained_variance' | metrics.explained_variance_score |
| 'max_error' | metrics.max_error |
| 'neg_mean_absolute_error' | metrics.mean_absolute_error |
| 'neg_mean_squared_error' | metrics.mean_squared_error |
| 'neg_mean_squared_log_error' | metrics.mean_squared_log_error |
| 'neg_median_absolute_error' | metrics.median_absolute_error |
| 'r2' | metrics.r2_score |

https://scikit-learn.org/stable/modules/model_evaluation.html

We use the mean absolute error which can be interpreted as the average distance from our predictions and the actual values.

```
y_pred = best_regressor.predict(X_test)
mean_absolute_error(y_test, y_pred)
```

```
3.6452601648381675
```

Machine Learning | Data Science | Towards Data Science | Programming | Artificial Intelligence

Follow

## Written by Cory Maklin

3.9K Followers · Writer for Towards Data Science

Problem Solver • QuantumBlack • Read more content for free: https://corymaklin.substack.com

### More from Cory Maklin and Towards Data Science

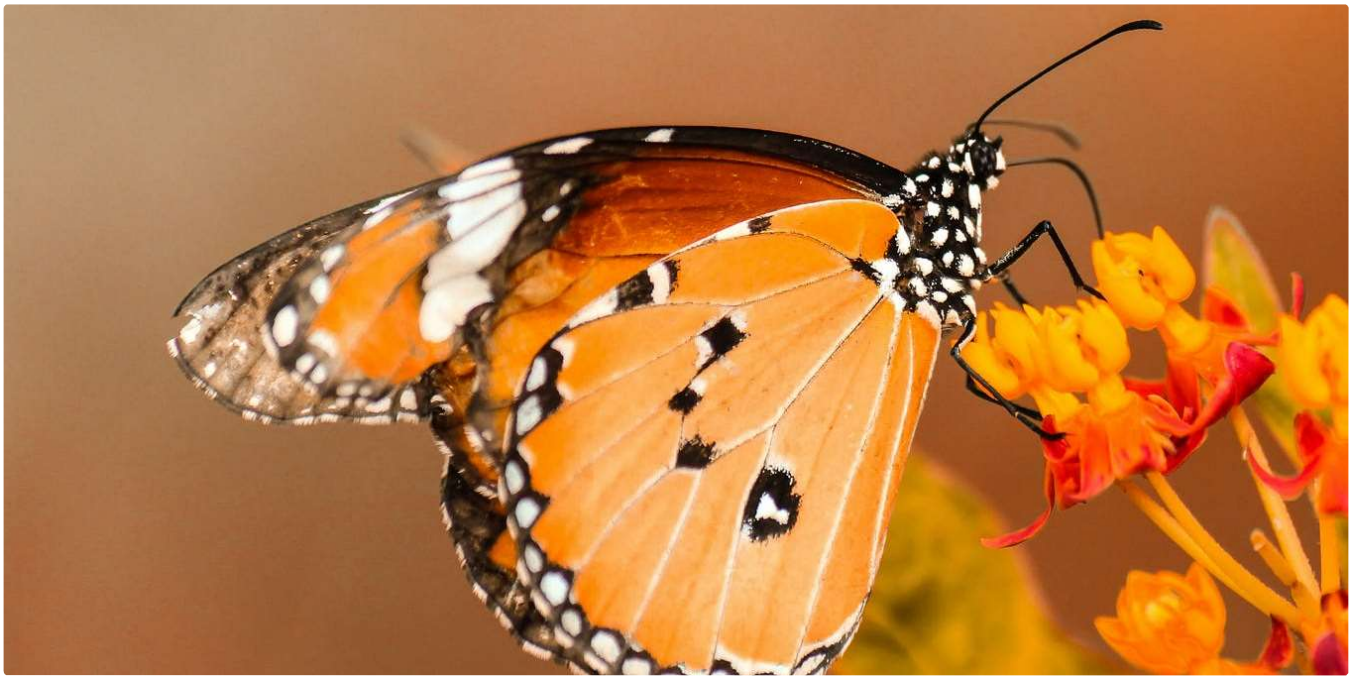
Cory Maklin

## Data Engineer Interview: Right To Be Forgotten

Interviewer: Let's say you were responsible for designing the data warehouse and associated pipelines. How would you support the right to...

✦ · 3 min read · Jun 7

80 2



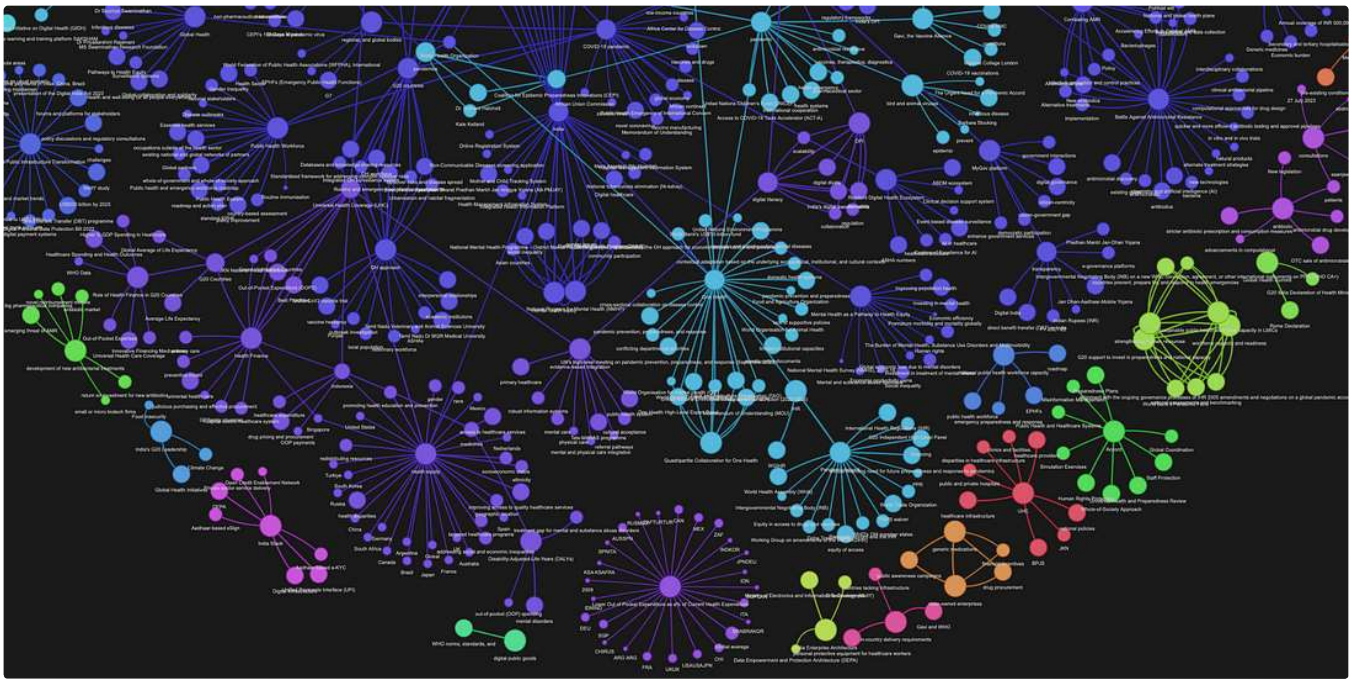
Marco Peixeiro in Towards Data Science

## TimeGPT: The First Foundation Model for Time Series Forecasting

Explore the first generative pre-trained forecasting model and apply it in a project with Python

✦ · 12 min read · Oct 24

2.3K 22



Rahul Nayak in Towards Data Science

## How to Convert Any Text Into a Graph of Concepts

A method to convert any text corpus into a Knowledge Graph using Mistral 7B.

12 min read · Nov 10

1.6K 28

Cory Maklin

## Data Engineer Interview: Backfill Data

Let's say you were responsible for maintaining the data warehouse pipelines. There was a bug in the code and now you need to re-ingest the...

3 min read · Jun 1

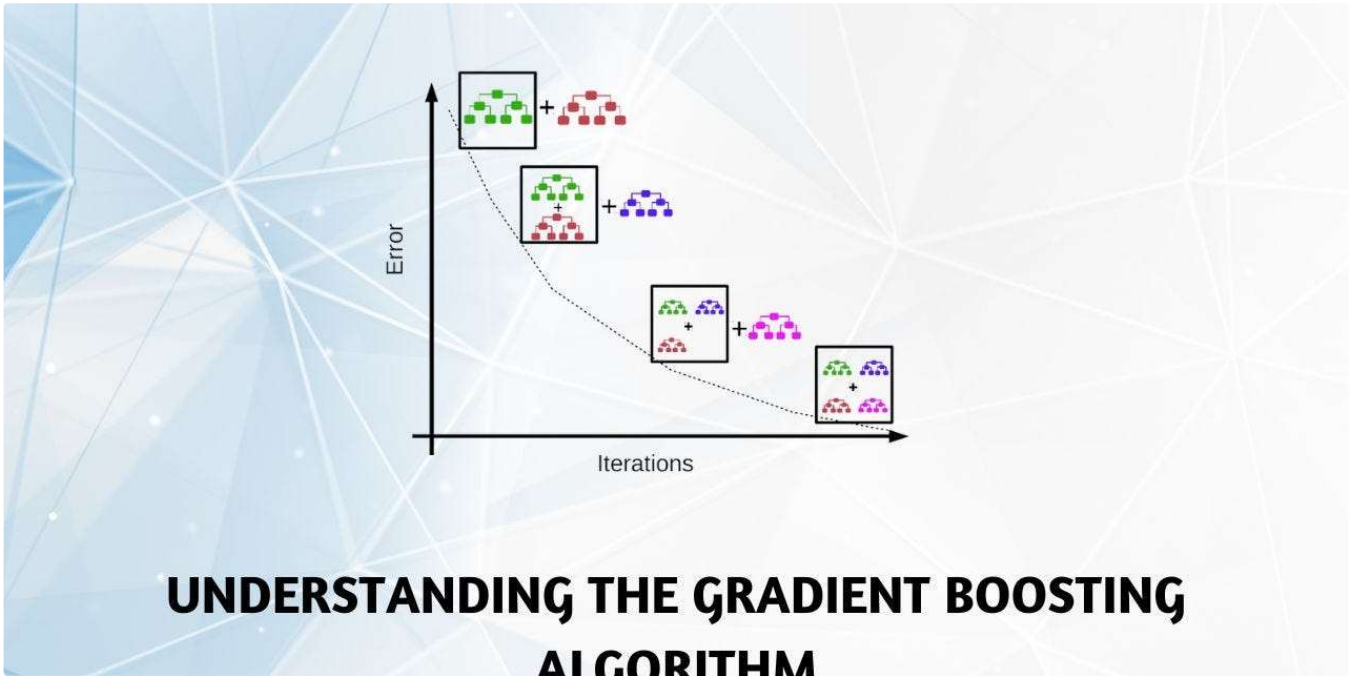

131



See all from Cory Maklin

See all from Towards Data Science

## Recommended from Medium



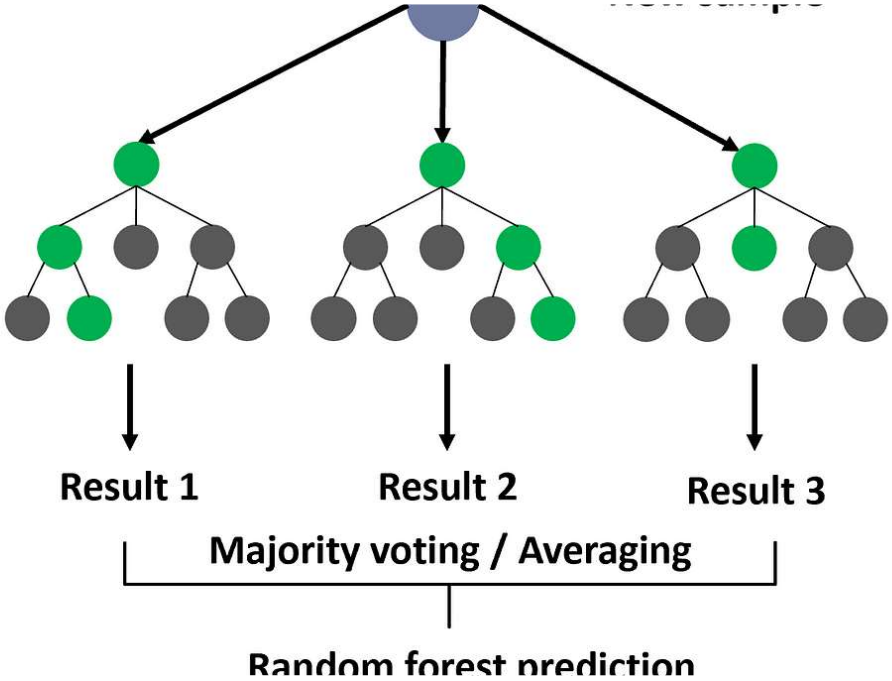

Data Science Wizards

## Understanding the Gradient Boosting Algorithm

Take a look in more depth at the boosting algorithms and see how the gradient descent optimization algorithm takes part and improve...

7 min read · Jul 13

7




Dr. Roi Yehoshua

## Random Forests

Random forests is a powerful machine learning model based on an ensemble of decision trees, where each tree is grown using a random subset...

✦ · 9 min read · Mar 25

158

### Lists

**Predictive Modeling w/ Python**

20 stories · 613 saves

**Practical Guides to Machine Learning**

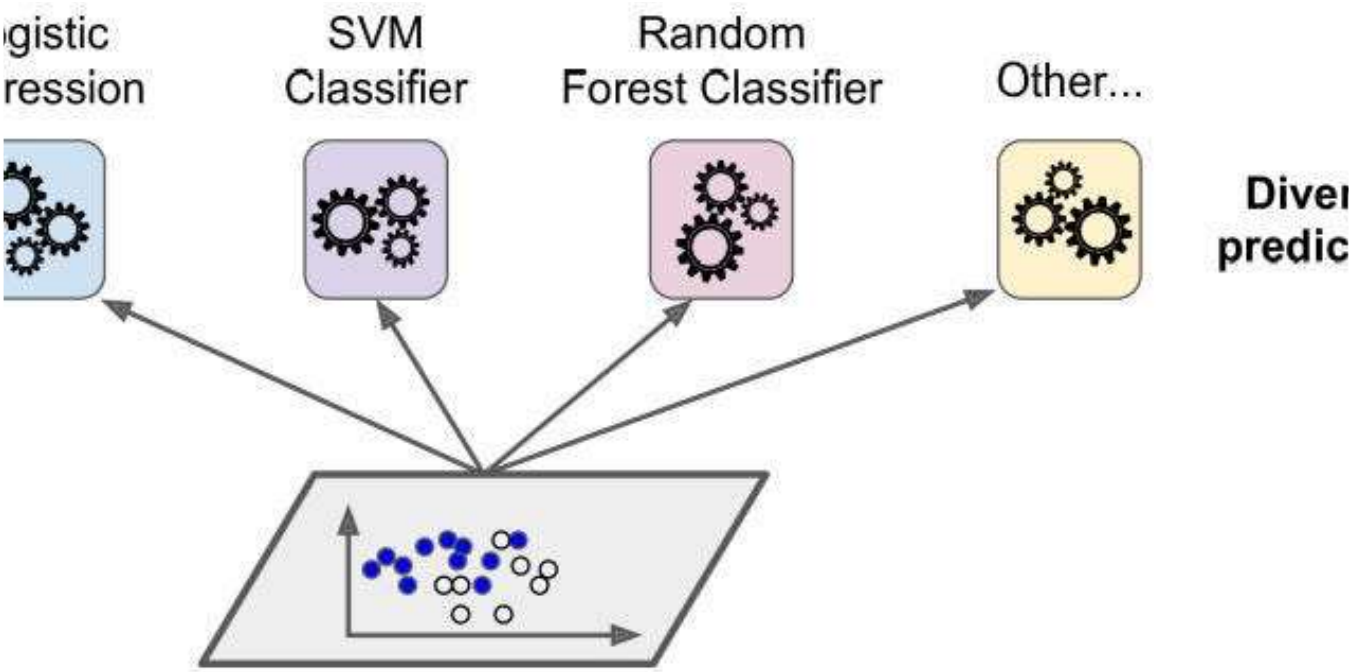
10 stories · 699 saves

**Natural Language Processing**

871 stories · 409 saves

**ChatGPT prompts**

30 stories · 690 saves




Amit Singh Rajawat

## Voting Classifiers in Machine Learning

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their...

4 min read · Jul 7

4


Anjali Ramesh

## Forecasting Walmart Sales with Machine Learning

In this machine learning project, we utilize historical Walmart sales data to predict store sales. The dataset can be found here.

7 min read · Jun 22

302 2

Bex T. in Towards Data Science

## 10 Confusing XGBoost Hyperparameters and How to Tune Them Like a Pro in 2023

✨ · 9 min read · Jun 11

694 3

Ignacio de Gregorio

## OpenAI Just Killed an Entire Market in 45 Minutes

The Story Everyone Should Have Seen Coming

✨ · 6 min read · Nov 9

13.5K 188