CLOUDYML

GRADIENT BOOTING EXPLAINED IN 5 SIMPLE STEPS

Akash Raj
Data Scientist

SWIPE <<<

GRADIENT BOOSTING

Gradient Boosting is a machine learning technique used to build predictive models by combining multiple weak models (usually decision trees) into a strong model that can make accurate predictions.

In Gradient Boosting, the algorithm starts by fitting a simple model to the data and then sequentially fits additional models, each one focused on correcting the errors of the previous model.

The new model is fit to the negative gradient of the loss function with respect to the previous model's predictions, which allows the new model to focus on the areas where the previous model made the most errors

Akash Raj
Data Scientist

SWIPE <<<

STEP 1

IMPORT LIBRARIES

First, you will need to import the necessary libraries for the project. For Gradient Boosting, we will need the following libraries: numpy, pandas, scikit-learn, and matplotlib

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

To get started with Gradient Boosting, you will need to have some knowledge of Python and machine learning concepts. Here is a step-by-step guide on how to use the Gradient Boosting algorithm in Python

Akash Raj
Data Scientist

SWIPE <<<

STEP 2

LOAD DATA

Next, you will need to load the data that you want to use for the algorithm. For this example, we will use the Boston Housing dataset from scikit-learn

```python
from sklearn.datasets import load_boston
boston = load_boston()
X = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.Series(boston.target)
```

Akash Raj
Data Scientist

SWIPE <<<

STEP 3

SPLIT DATA INTO TRAINING & TESTING DATA

Now that you have loaded the data, you will need to split it into training and testing sets. This will allow you to train the model on one set of data and evaluate its performance on another set of data. We will use the train_test_split function from scikit-learn to split the data

```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Akash Raj
Data Scientist

SWIPE

STEP 4

TRAIN & EVALUATE THE MODEL

With the data split into training and testing sets, you can now train the Gradient Boosting model. We will use the GradientBoostingRegressor class from scikit-learn to create the model. Now that the model has been trained, you can evaluate its performance on the testing set. We will use the R-squared value as a metric to evaluate the performance of the model

```python
model = GradientBoostingRegressor()
model.fit(X_train, y_train)

score = model.score(X_test, y_test)
print("R-squared:", score)
```

Akash Raj
Data Scientist

SWIPE <<<

STEP 5

PREDICT & VISUALIZE RESULTS

Finally, you can use the trained model to make predictions on new data. We will use the predict function of the model to make predictions on the testing set.
To visualize the results, we can plot the actual and predicted values of the target variable

```python
y_pred = model.predict(X_test)

plt.plot(y_test.values, label="actual")
plt.plot(y_pred, label="predicted")
plt.legend()
plt.show()
```

Akash Raj
Data Scientist

SWIPE <<<

CONCLUSION

Compared to other ensemble methods, gradient boosting focuses on reducing bias and is particularly effective for complex, high-dimensional data. By understanding these five simple steps, you can better appreciate the power and effectiveness of gradient boosting in solving a wide range of machine learning problems.

FOLLOW ME TODAY

98K+ 49K+ 48K+

Akash Raj
Data Scientist