NTNU | Norwegian University of Science and Technology

LightGBM

Group 3

Outline

- LightGBM
- Leaf growth
- Code example

What is LightGMB?

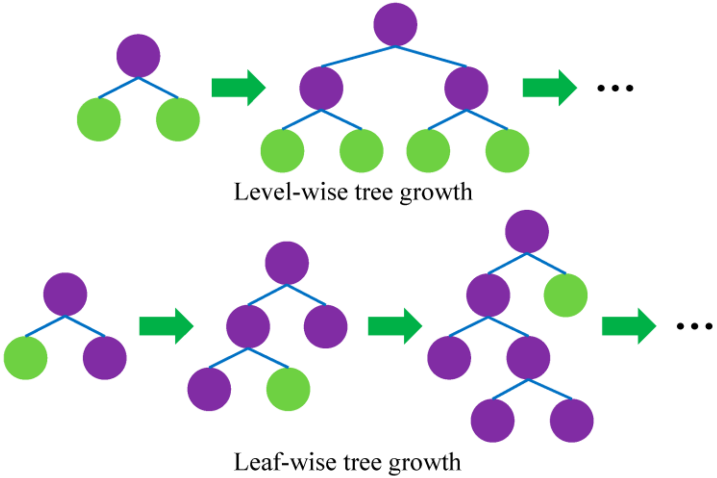
- Developed by Microsoft
- Gradient-boosting framework for machine learning
- Decision tree algorithm
- Used for:
  - Classification
  - Ranking
  - Regression

How it works

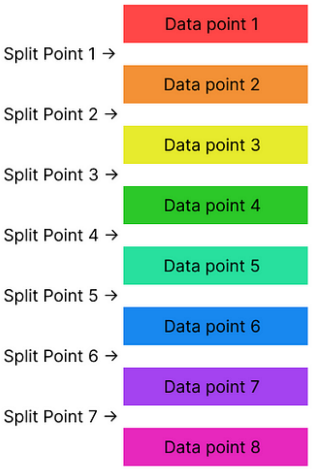
- Boosting
  - Each new tree corrects the errors of the previous one
- Gradient descent
  - Minimization of residual errors of earlier trees
- L1 & L2 regularization
  - Controls overfitting
- Histogram-based split finding

Leaves

- Leaf-wise
  - Grows by adding a leaf with maximum gain
  - Deeper but narrower trees
  - Much faster
- Level-wise
  - Grow all leaves at each level
- Full tree
  - Same tree for both methods
  - Early stopping etc. => order matters

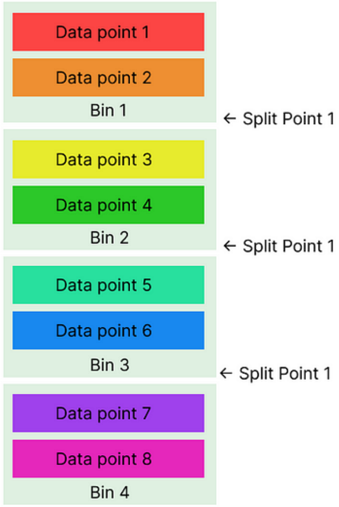


Histogram based split finding



Data point 1
Split Point 1 →
Data point 2
Split Point 2 →
Data point 3
Split Point 3 →
Data point 4
Split Point 4 →
Data point 5
Split Point 5 →
Data point 6
Split Point 6 →
Data point 7
Split Point 7 →
Data point 8

[Splitting based on every point]



Data point 1
Data point 2
Bin 1
← Split Point 1
Data point 3
Data point 4
Bin 2
← Split Point 1
Data point 5
Data point 6
Bin 3
← Split Point 1
Data point 7
Data point 8
Bin 4

[Histogram based split]

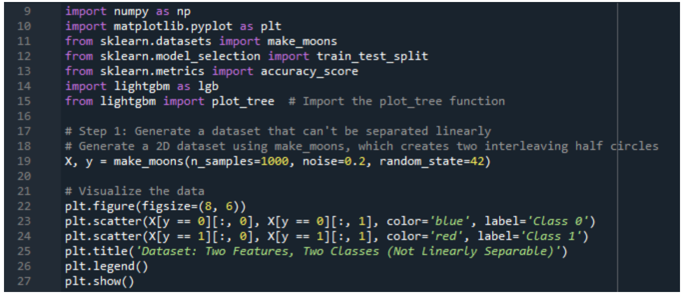
- Group like datapoints into bins
- Split based on bins

Advantages

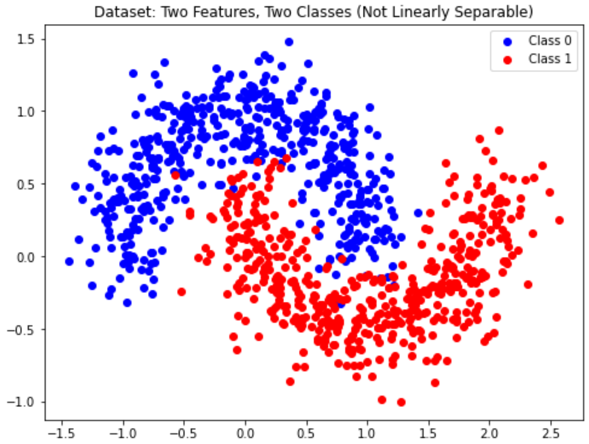
- Faster training speed
- Accuracy
- Parallel and Distributed Training
- Feature Importance
- Lower memory usage
- Effective with large scale datasets

Example

- Generated a dataset with two features and two classes which are not linearly separable

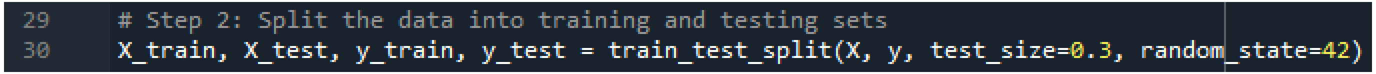


```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import lightgbm as lgb
from lightgbm import plot_tree  # Import the plot_tree function

# Step 1: Generate a dataset that can't be separated linearly
# Generate a 2D dataset using make_moons, which creates two interleaving half circles
X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)

# Visualize the data
plt.figure(figsize=(8, 6))
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='blue', label='Class 0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='red', label='Class 1')
plt.title('Dataset: Two Features, Two Classes (Not Linearly Separable)')
plt.legend()
plt.show()
```



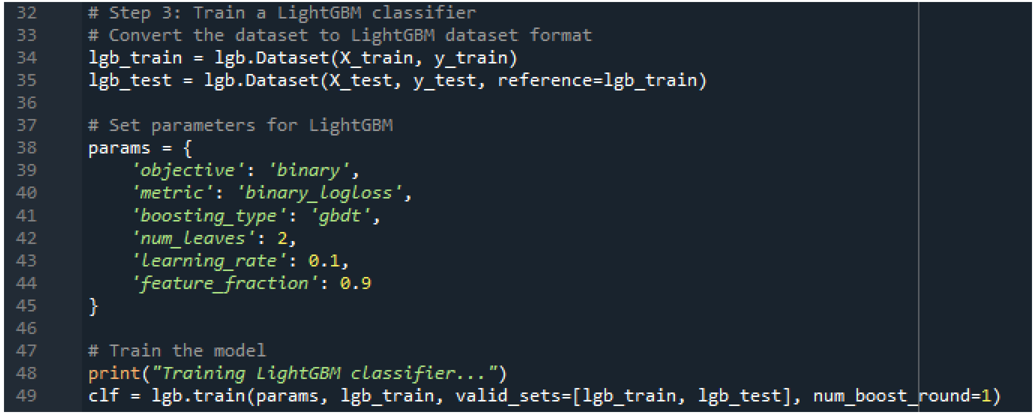
Example

- Separated the dataset into a training set and a test set



```
# Step 2: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

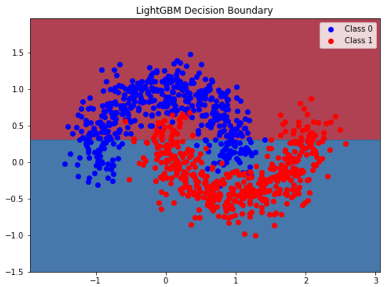
- Selected parameters for our LightGBM classifier and trained it using our training set

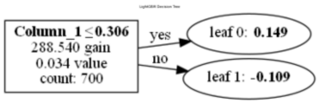


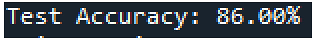
```
# Step 3: Train a LightGBM classifier
# Convert the dataset to LightGBM dataset format
lgb_train = lgb.Dataset(X_train, y_train)
lgb_test = lgb.Dataset(X_test, y_test, reference=lgb_train)

# Set parameters for LightGBM
params = {
    'objective': 'binary',
    'metric': 'binary_logloss',
    'boosting_type': 'gbdt',
    'num_leaves': 2,
    'learning_rate': 0.1,
    'feature_fraction': 0.9
}

# Train the model
print("Training LightGBM classifier...")
clf = lgb.train(params, lgb_train, valid_sets=[lgb_train, lgb_test], num_boost_round=1)
```

Example

2 leaves

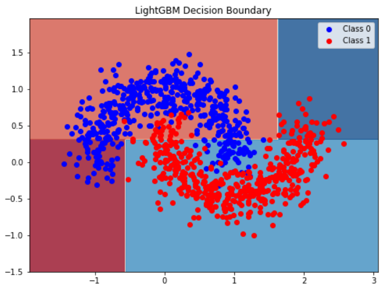


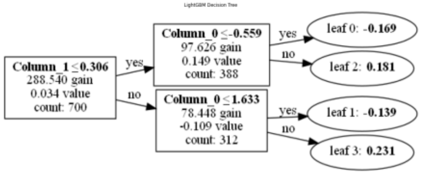


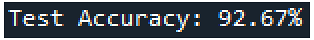


```
Test Accuracy: 86.00%
```

4 leaves

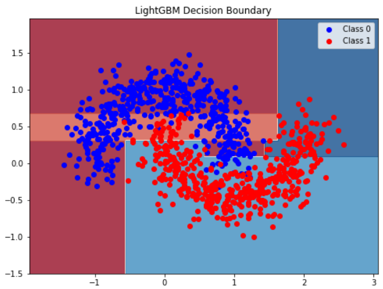


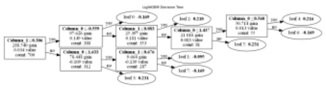


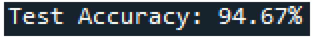


```
Test Accuracy: 92.67%
```

8 leaves

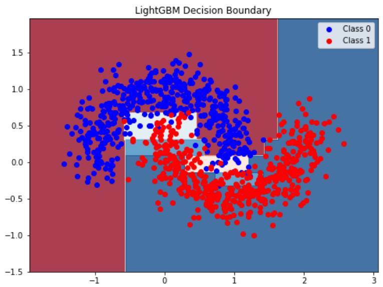


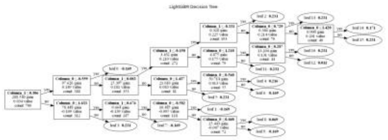


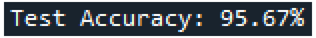


```
Test Accuracy: 94.67%
```

16 leaves







```
Test Accuracy: 95.67%
```

Pros and cons

- Pros
  - Faster training
  - Higher accuracy
  - Suitable for large dataset
  - Boosting reduce variance
- Cons
  - Prone to overfitting on small dataset
  - Bad for sparse data