LGBM

* Light Gradient Boosting Machine
* Microsoft (2017)
* Regression & Classification

General:

Already discussed (Ensemble learning)

Mathematical

Already discussed
(Same as QB)

features	Adaboost	GB	XgBoost	Catboost	LGBM
Boosting Type	Sequential	Sequential	Sequential parallelization	Sequential	Sequential (speed optimiza
Error Handling	Misclassified instances	Residuals	Residuals (regularization)	Residuals (ordered boosting)	Residuals (GOSS, EFB)
Memory usage	Low	Moderate	Better as compared to Adaboost & GB	Efficient with categorical data	Highly efficient
Use	Simple problem	Medium problem	Complex problem	Many categorical features	Large dataset high-D data

① Speed Optimization

How DT are made → level-wise
→ leaf-wise (LGBM)

Eg. Manger → Bookstore → Ad Budget

Scenario

fictional

Non-fictional

Goal → Target might customers.

Data

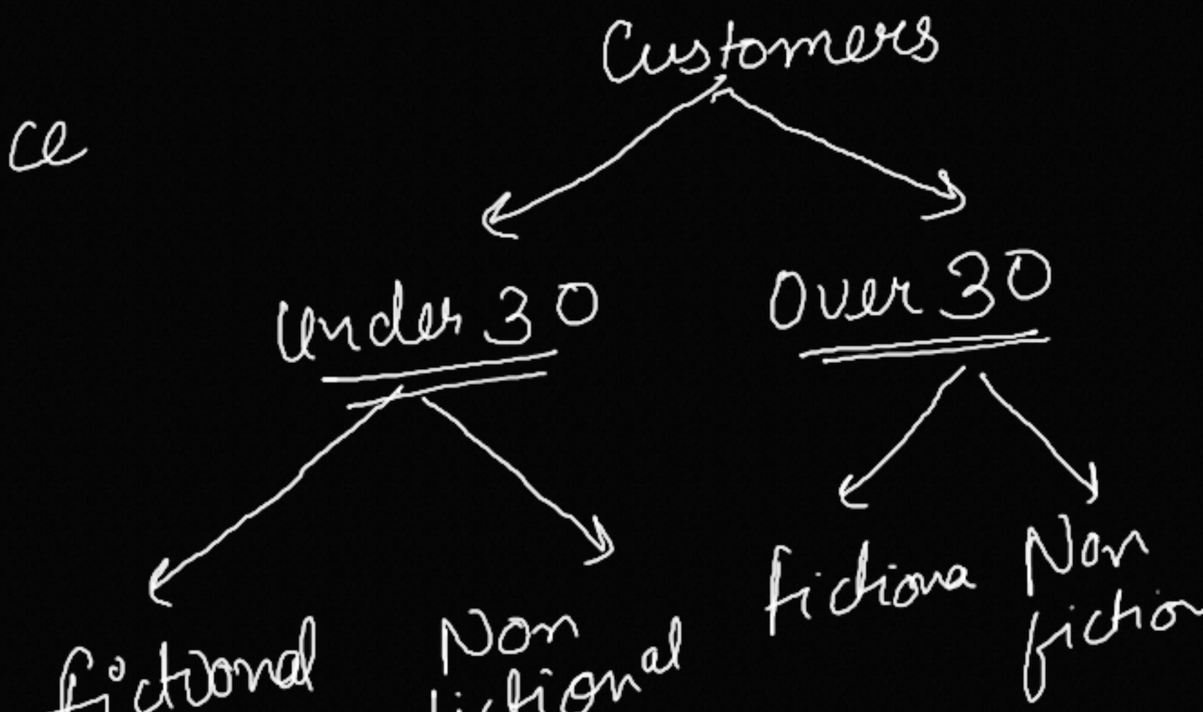
Age

Genre

⇒ Level-wise

all groups → equal importance

Splitting is useless

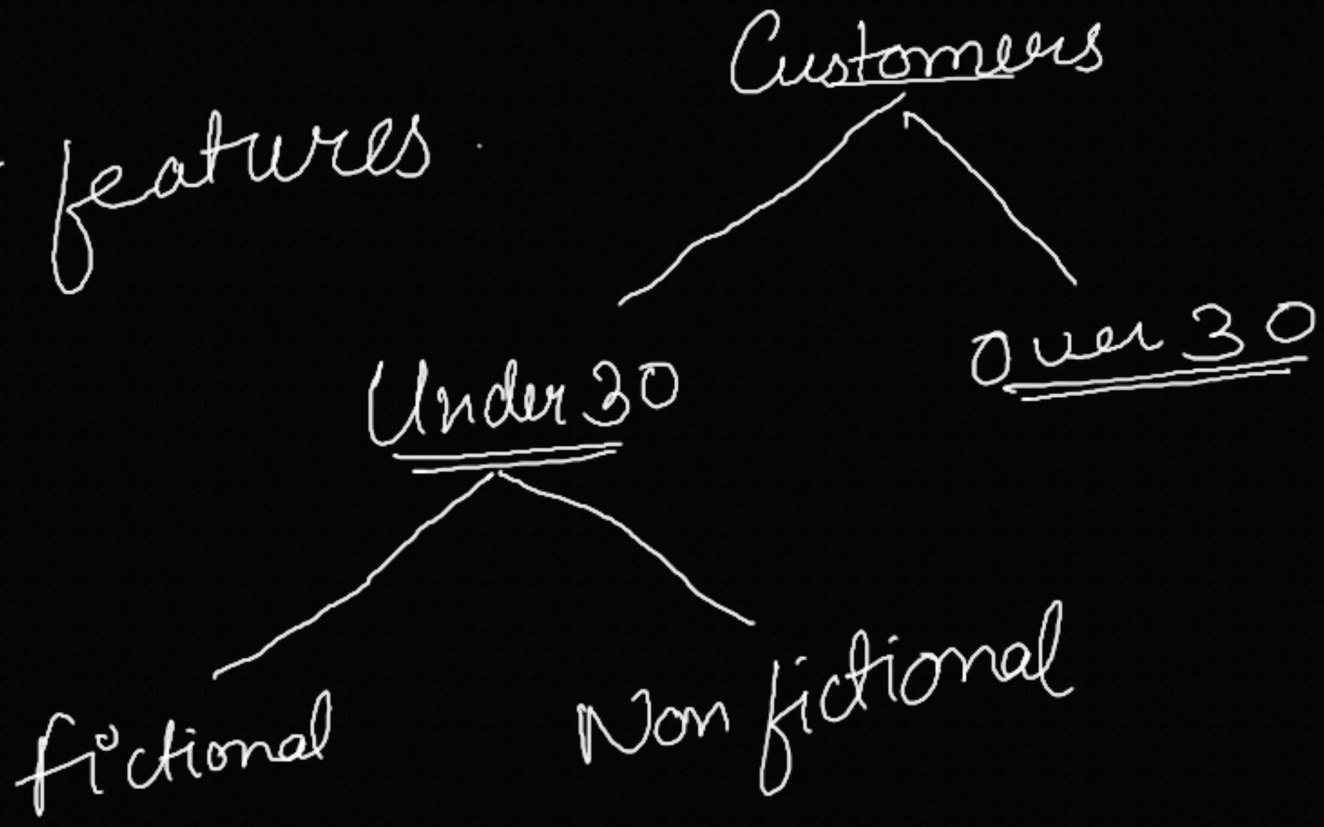


One more example → flyers for shop

all over city distribute

⇒ leaf wise

Importance to important features.



One more example

flyers

Neighbourhood

② GOSS (Gradient based One Side Sampling)

Working:

GB ⇒ Each row/sample ⟶ gradient (how error will change)

large gradient ⇒ sample hard to predict (difficult is the best) (important)

small gradient ⇒ model can easily predict

⇒ Keep only large gradient samples
(because they are important for model)

* Save time & resources
* Speed ↑
* Efficiency ↑
* Maintain accuracy
(weight assigning powerful)

} Benifits of GOSS.

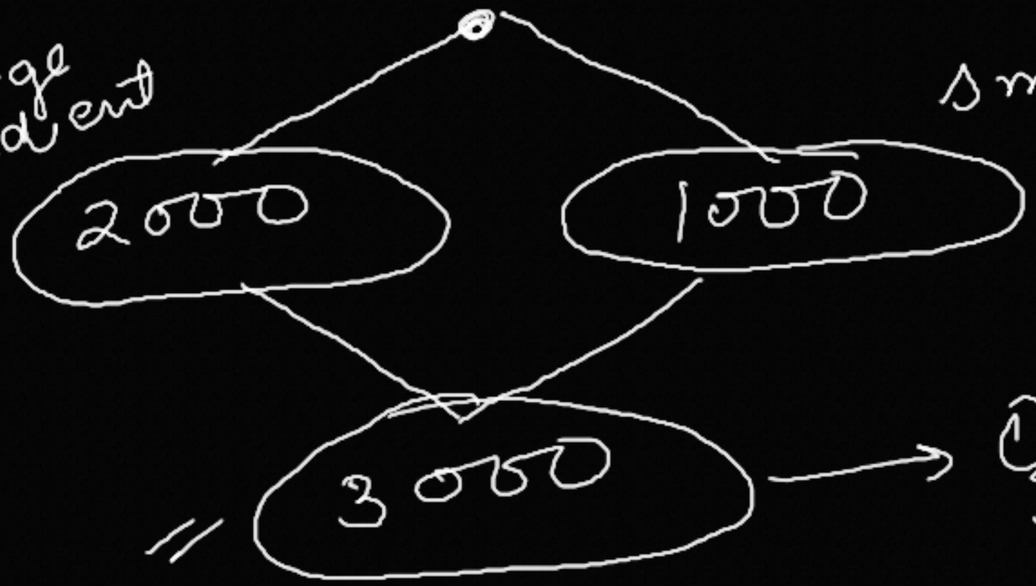
eg. Total → 10000
Large gradient → 2000 , Small grad → 8000

Without GOSS:

Iteration 1 → 10000
Iteration 2 → 10000
:: ::
:: ::

{ It uses all samples for each iteration }

GOSS:



→ Overall Samples

Speed ↑

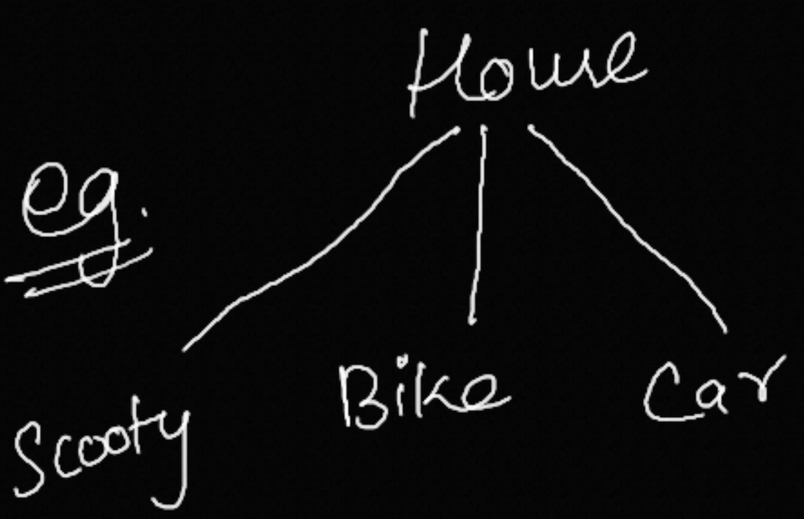
less computations

③ EFB (Exclusive Feature Bundling)

Working

⇒ Identify all mutually exclusive features (features where one thing can be true at a time)

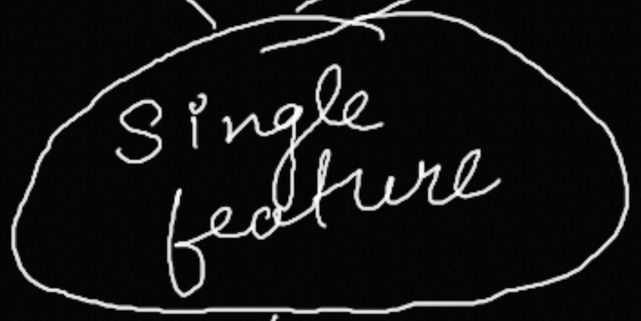
eg.



→ Want to go Office → You select "Bike" → [0, 1, 0]

(Sparse data)

⇒ f1 f2 f3 f4 (Mutualy exclusive features)



Training

Unbundle

Seperately treat

Why unbundle?

⇒ So that it dosnot affect our output.

Benifits

Save memory (efficient)

Speed ↑

large & sparse data

When to Use

* Large / complex / Sparse data
* High dimensional data
* Can handle categorical features (one-hot encoding)
* High speed
  Good accuracy
* Support parallelization

When Not to Use

* Small dataset (overfitting)
* Imbalanced dataset