Predictive Modeling XGBoost & CatBoost

DS Development Presentations

Peter Nicholas S. Onglao | MNL

Contents

1

XGB Hyperparameters

A guide to relevant parameters

List of Parameters

Learning Task Parameters		
Parameter	**Default**	**Description**
objective	“reg:squarederror”	The objective function ex. reg:logistic, binary:logistic, multi:softprob
base_score	0.5	Initial prediction score of all instances
eval_metric	Depends on obj.	Evaluation metric for validation data
seed	0	Random seed number (use set.seed() instead in R)

List of Parameters

Parameters for Tree Booster		
Parameter	**Default**	**Description**
eta	0.3	Step size shrinkage to prevent overfitting
gamma	0	Minimum loss reduction to make a further partition
lambda	1	L2 regularization term on weights
alpha	0	L1 regularization term on weights

$$Gain = \left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

$$\frac{G^2}{H+\lambda} = \frac{(\sum g)^2}{\sum h+\lambda} = \frac{\left(\sum(residuals)\right)^2}{\sum \hat{y}_i \times (1-\hat{y}_i)+\lambda}$$

List of Parameters

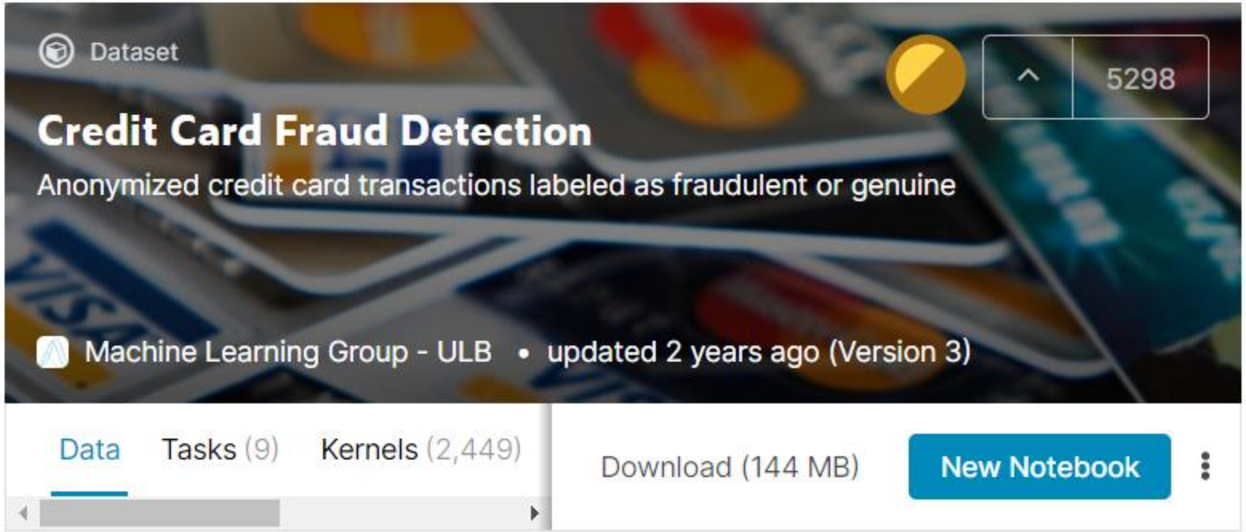
Parameters for Tree Booster		
Parameter	**Default**	**Description**
max_depth	6	Maximum tree depth
min_child_weight	1	Minimum ‘score’ needed in a child; further partitioning stops if a tree partition results in a node less than this value
max_delta_step	0	Usually not needed, but helps in logistic regression when class is extremely imbalanced. Set between 1-10

$$\frac{G^2}{H+\lambda} = \frac{(\sum g)^2}{\sum h + \lambda} = \frac{\left(\sum (residuals)\right)^2}{\sum \hat{y}_i \times (1-\hat{y}_i) + \lambda}$$

List of Parameters

Parameters for Tree Booster		
Parameter	**Default**	**Description**
subsample	1 range: (0,1]	Subsamples training data at every iteration
colsample_bytree	1 range: (0,1]	Subsamples columns for each tree
colsample_bylevel		Subsamples columns for each tree level, using the prev. columns
colsample_bynode		Subsamples columns for each split, using the prev. columns

Sample Code


Dataset
5298
Credit Card Fraud Detection
Anonymized credit card transactions labeled as fraudulent or genuine
Machine Learning Group - ULB • updated 2 years ago (Version 3)
Data
Tasks (9)
Kernels (2,449)
Download (144 MB)
New Notebook

Sample Code

- Columns:
  - Time – time from first transaction in dataset
  - V1-V28 – de-identified numeric data
  - Amount – amount in transaction
  - Class – 1 fraudulent, 0 otherwise
- Imbalanced classes

	Class
0	99.83%
1	0.17%

Sample Code

- Preliminaries
  - Separate data into y (Class) and X (other cols)
  - Split data into train and test sets

Separate the data into X and y

In [32]:
```
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

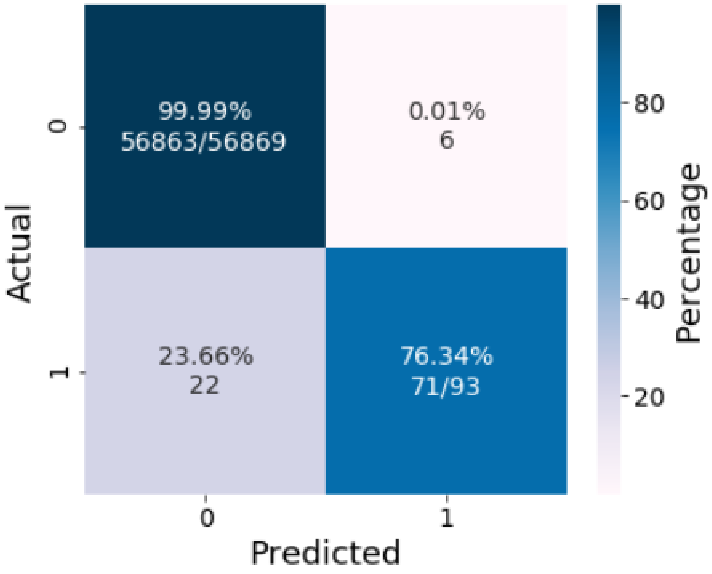
Split the data into train and test sets

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

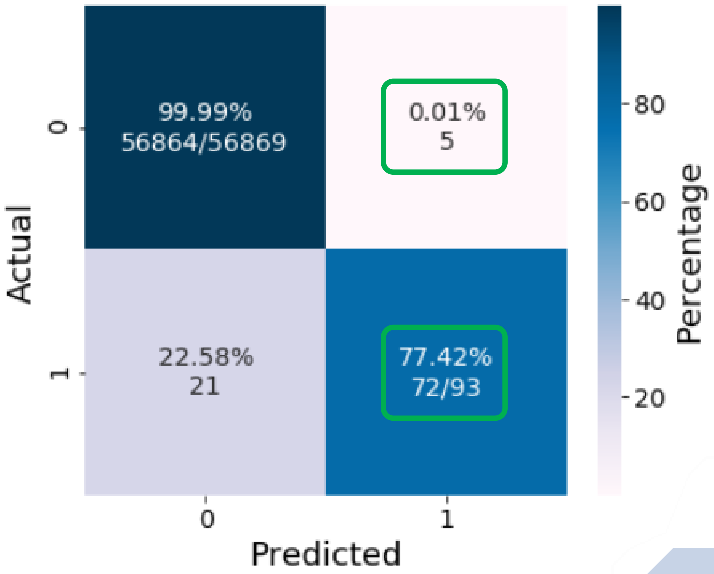
Sample Code

- n_estimators = 20, random state = 2020

Default



max_delta_step = 1



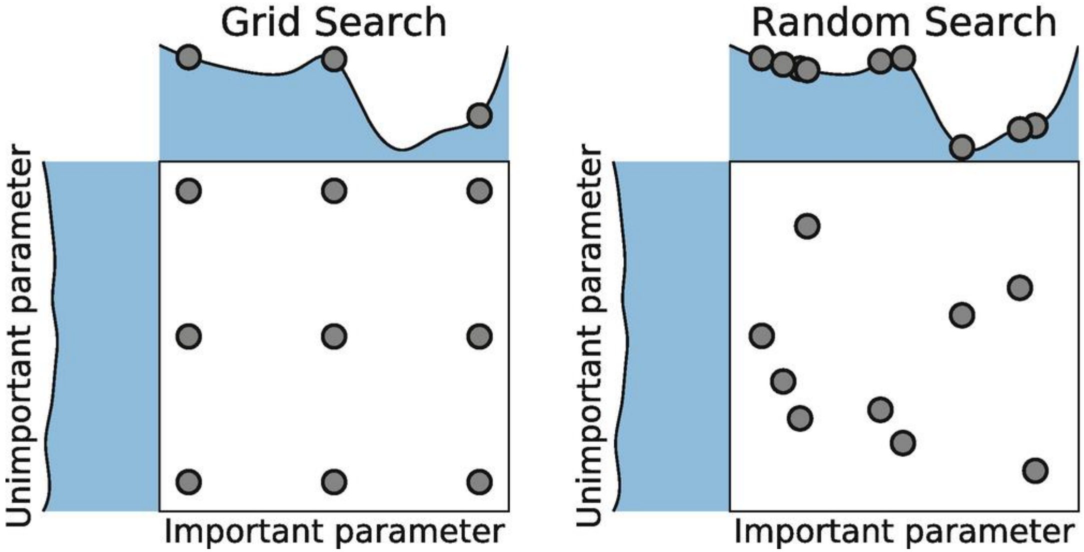
2

XGB Tuning Options

Tuning process & available methods

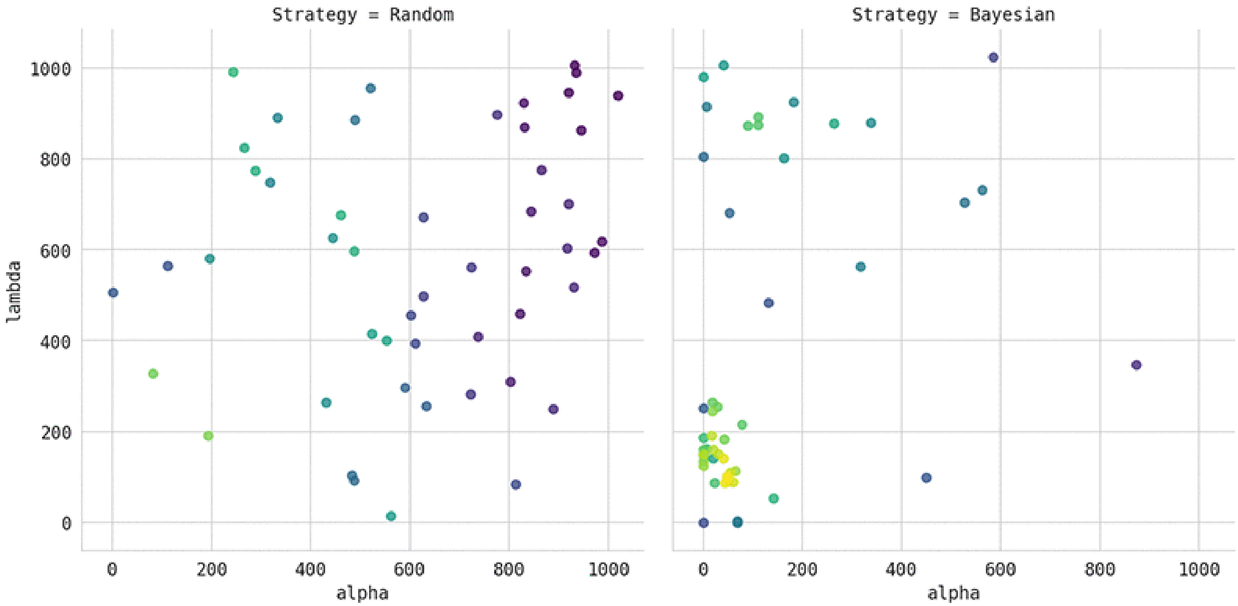
Methods for Tuning

- 2 common methods:



Methods for Tuning

- Becoming popular: Bayesian Optimization



- There are others!

Methods for Tuning

- Which one to use?

Method	Speed	No. params	Results
Grid Search	Poor	Poor	Great
Random Search	Great	Great	Good
Bayesian	Good	Great	Great

Tuning Packages in Python

Python	
Grid Search	sklearn.model_selection.GridSearchCV
Random Search	sklearn.model_selection.RandomizedSearchCV
Bayesian	bayes_opt.BayesianOptimization

Sample Code: Random Search

- Preliminaries
  - Import packages
  - Define parameters
  - Create model (metric = Precision-Recall AUC)

In [96]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from scipy import stats
```

In [117]:
```
param = {
    'objective':'binary:logistic',
    'random_state':2020
}
```

In [118]:
```
xgb_cv_model = XGBClassifier(**param, nthread = -1, eval_metric = 'aucpr')
```

Sample Code: Random Search

- Define distributions to sample from

In [119]:
```
cv_params = {
    'eta':stats.uniform(0.01,0.59),
    'min_child_weight': stats.randint(1,10),
    'gamma': stats.uniform(0,10),
    'subsample': stats.uniform(0.5,0.5),
    'colsample_bytree': stats.uniform(0.5,0.5),
    'colsample_bylevel': stats.uniform(0.5,0.5),
    'colsample_bynode': stats.uniform(0.5,0.5),
    'max_depth': [3,5,7,9],
    'max_delta_step': stats.randint(1,10)
}
```

→ [0.5 , 1]

Sample Code: Random Search

Run the random search

In [120]:
```
cv_model = RandomizedSearchCV(estimator = xgb_cv_model,
                              param_distributions = cv_params,
                              random_state = 2020,
                              verbose = 2,
                              n_jobs = 1,
                              )
cv_model.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:  3.9min finished
```

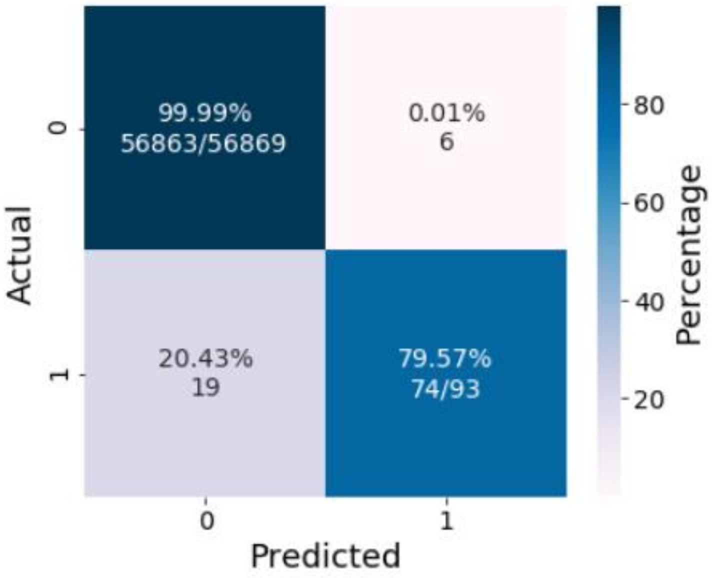
In [121]:
```
print(cv_model.best_params_)
```

```
{'colsample_bylevel': 0.6496611229613486, 'colsample_bynode': 0.834270438868537
8, 'colsample_bytree': 0.9762762178047335, 'eta': 0.49156213718080255, 'gamma':
0.7786223582619556, 'max_delta_step': 3, 'max_depth': 5, 'min_child_weight': 7,
'subsample': 0.9972293498411935}
```

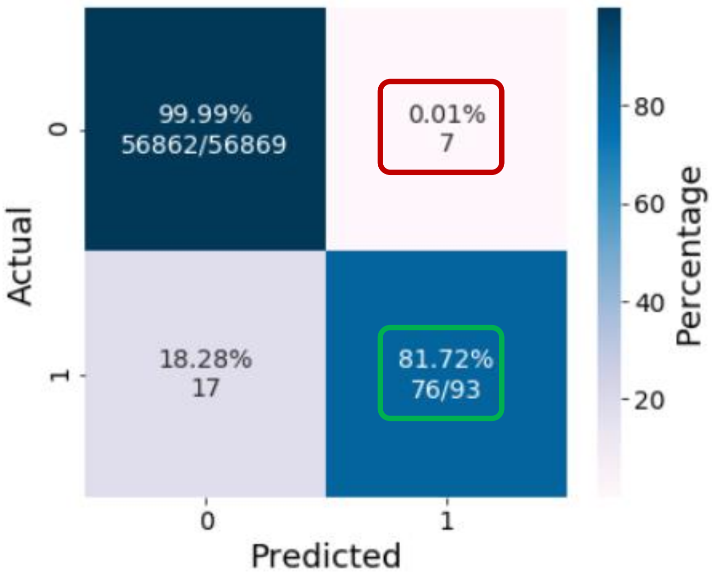
Sample Code: Random Search

- n_estimators = 100, random state = 2020

No Tuning



With Tuning



Sample Code: Bayesian Optimization

- Preliminaries
  - Import packages
  - Create D-matrices (default API easier than sklearn API)

In [141]:
```
from bayes_opt import BayesianOptimization
```

In [148]:
```
D_train = DMatrix(X_train, label = y_train)
```

In [149]:
```
D_test = DMatrix(X_test, label = y_test)
```

Sample Code: Bayesian Optimization

- Define upper and lower bounds to sample from
  - pbounds = dictionary of parameter bounds

In [404]:
```
pbounds = {
    'eta':(0.01, 0.6),
    'min_child_weight': (1,10),
    'gamma': (0,10),
    'subsample': (0.5, 1),
    'colsample_bytree': (0.5,1),
    'colsample_bylevel': (0.5,1),
    'colsample_bynode': (0.5,1),
    'max_depth': (3,10),
    'max_delta_step': (1,10)
}
```

Sample Code: Bayesian Optimization

- How to run BayesianOptimization
  - f = function that takes in the parameters being optimized, and outputs a real number (metric: PR AUC of val. set)
  - pbounds – earlier slide

```
xgbc = xgb.cv(
                paramt,
                D_train,
                num_boost_round = 100,
                nfold = folds,
                metrics = 'aucpr',
                stratified = True,
           )
```

Sample Code: Bayesian Optimization

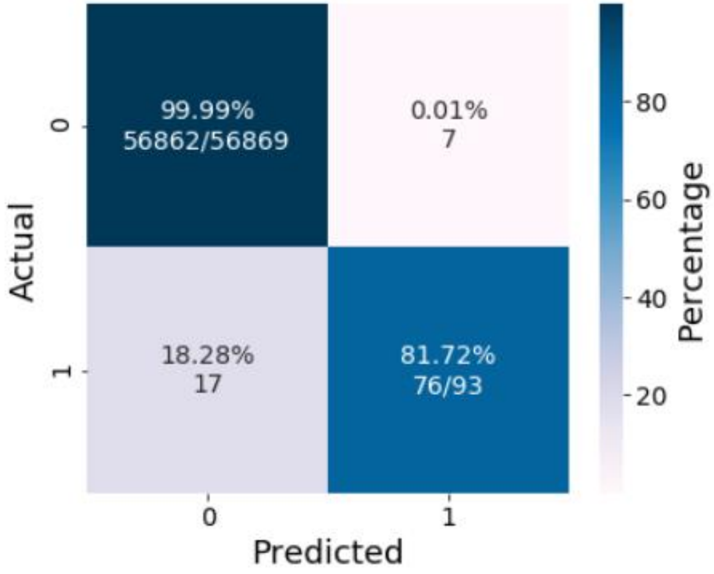
- Maximize the AUC PR! (took ~1 hour)

```
In [171]: XGB_BO = BayesianOptimization(
              f=XGB_CV,
              pbounds=pbounds,
              random_state=2020,
              verbose = 10
          )

In [172]: XGB_BO.maximize(init_points = 15, n_iter = 30)

In [429]: XGB_BO.max['params']

Out[429]: {'colsample_bylevel': 0.8429091762081982,
           'colsample_bynode': 0.5186575195775127,
           'colsample_bytree': 0.816311624054564,
           'eta': 0.17053487229722672,
           'gamma': 1.7892788652395508,
           'max_delta_step': 9.947823071229202,
           'max_depth': 9.794647541185961,
           'min_child_weight': 1.3421690213440391,
           'subsample': 0.8351187167964789}
```

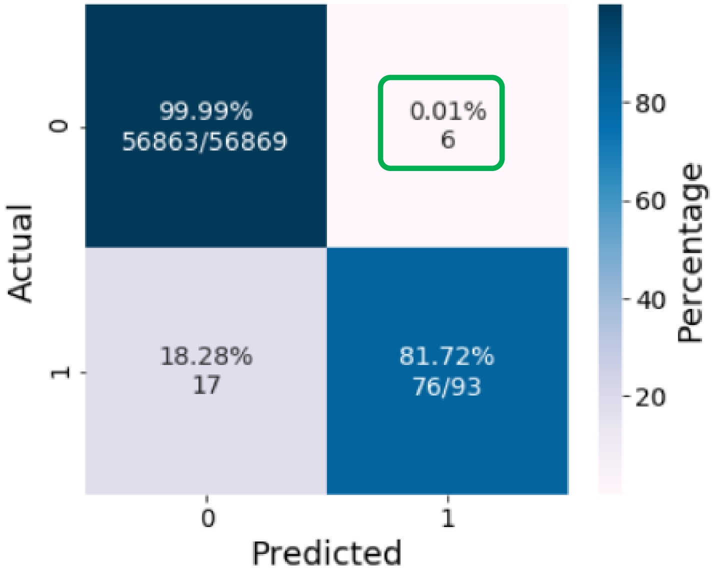
Sample Code: Bayesian Optimization

- n_estimators = 100, random state = 2020

RandomSearch



BayesianOptimization



Sample Code: Bayesian Optimization

- Comparison of parameters

Parameter	RandomSearch	BayesianOptimization
eta	0.49	0.17
gamma	0.78	1.79
max_delta_step	3	9
max_depth	5	9
min_child_weight	7	1.34
subsample	1.00	0.84
colsample_bylevel	0.65	0.84
colsample_bynode	0.83	0.52
colsample_bytree	0.98	0.82

3

Intro to CatBoost

Categorical Boosting

History of CatBoost

March, 2014

XGBoost initially started as research project by Tianqi Chen
but it actually became famous in 2016

Jan, 2017

Microsoft released first stable version of LightGBM

April, 2017

Yandex, one of Russia's leading tech companies open sources CatBoost

Comparison of Boosting Algorithms

- XGBoost
  - Biggest community
  - Most resources
- LightGBM
  - Fastest
  - The new “meta” for Kaggle competitions
  - You have to convert categorical data to integers
- CatBoost
  - Newest
  - Smallest community (but has Telegram w/ developers)
  - Best with categorical data, can be used w/o conversion

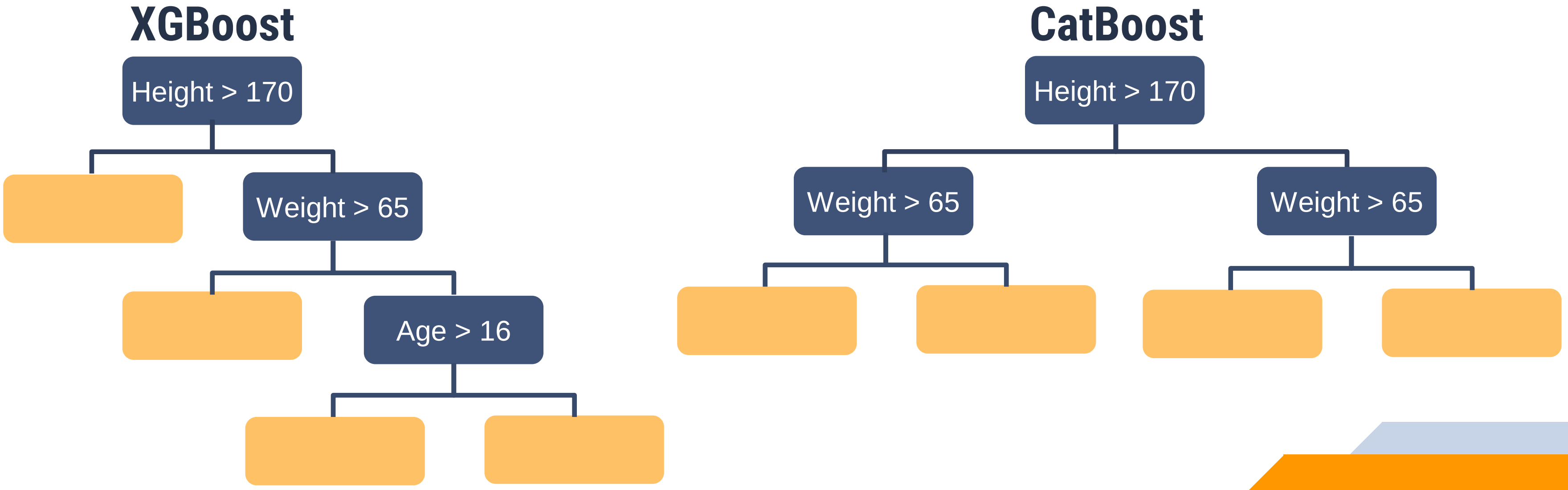
Features of CatBoost

- Great performance right out of the box
  - “CatBoost with default parameters beats all other algorithms with tuned parameters except for one case” - Anna Dorogush (Yandex)

	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
Adult	**0.26974**	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
Amazon	**0.13772**	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%

Features of CatBoost

- Tree building is symmetric
  - Each level has the same feature & number
  - You can build trees faster



Features of CatBoost

- Feature interpretations:
  - Feature importance
  - Feature interactions
  - Per object feature importance (SHAP)
- CatBoost Viewer:
  - Can plot loss function during fitting

Sample Code

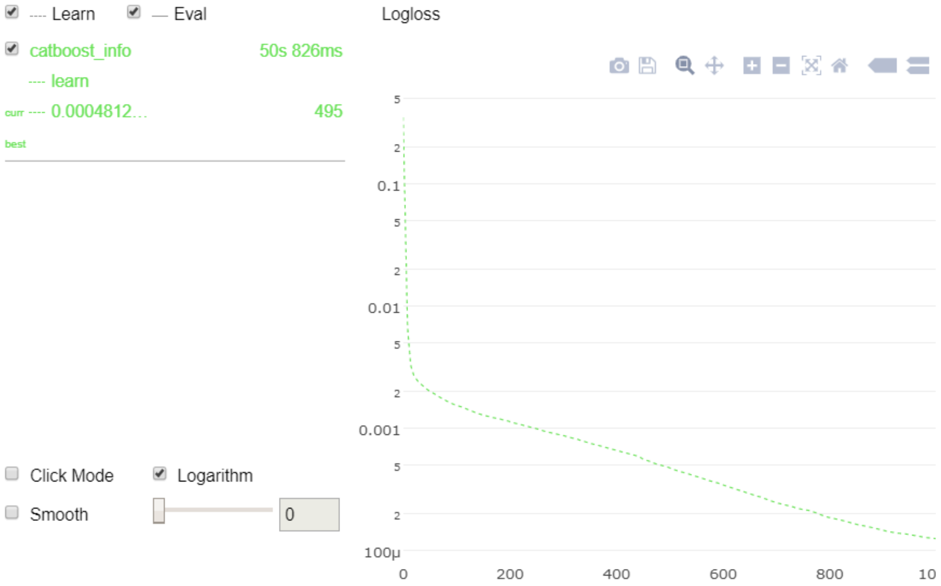
- Preliminaries
  - Import packages
  - Split data into y (labels) and X (other columns)
  - Train-test split

In [212]:
```
from catboost import CatBoostClassifier, Pool, cv
```

Sample Code

Model and plot the evaluation metric

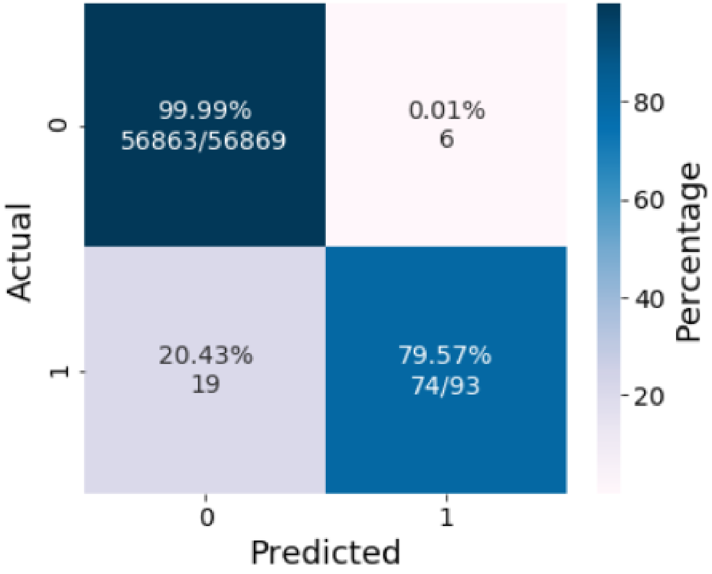
```
In [233]: model = CatBoostClassifier(random_state=2020)

In [234]: model.fit(X_train, y_train, plot = True)
          preds_class = model.predict(X_test)
          preds_proba = model.predict_proba(X_test)
```



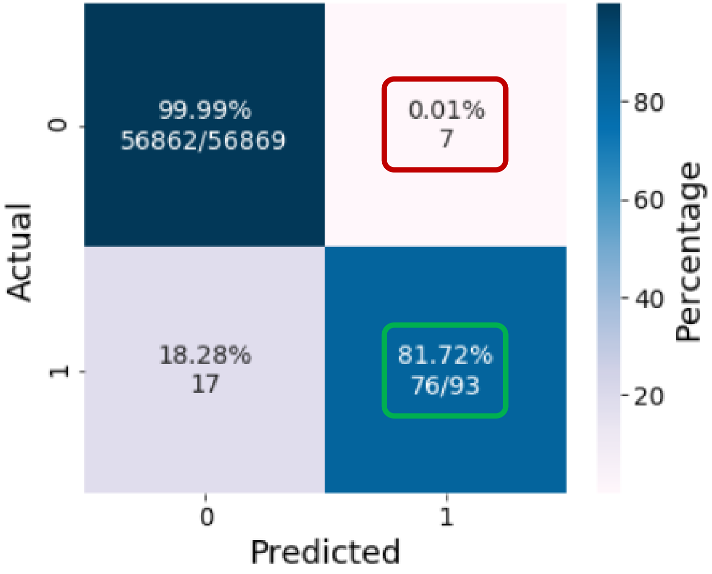
Sample Code

- Comparison of results

XGBoost Default



CatBoost Default



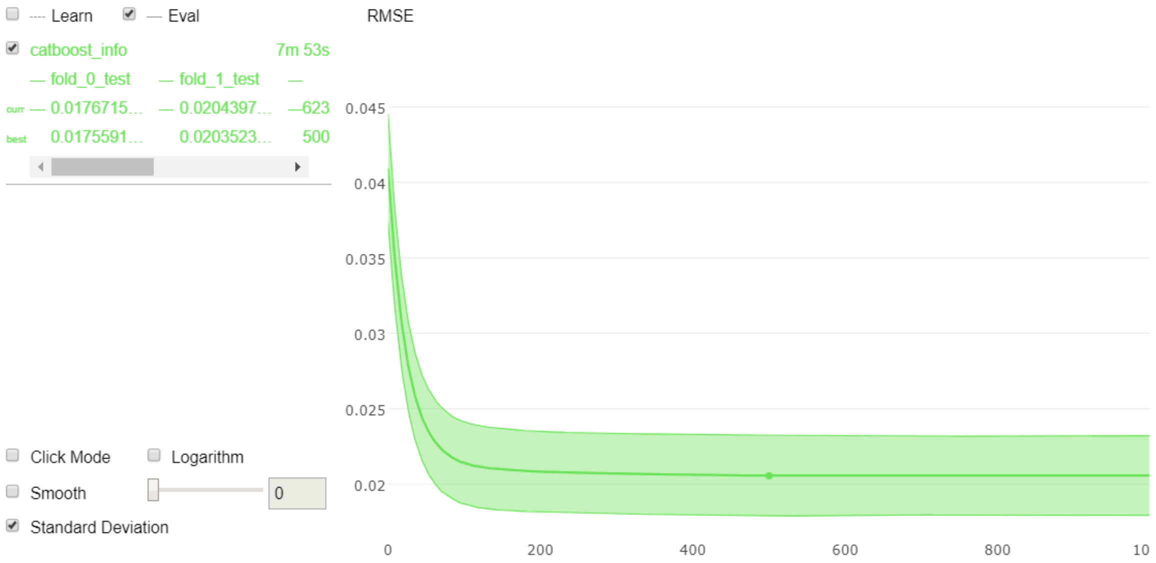
Sample Code

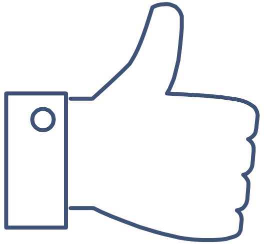
Cross Validation with CatBoost

In [368]:

```
cv_dataset = Pool(data = X_train,
                  label = y_train)

params = {'loss_function':'RMSE'}

scores = cv(cv_dataset,
            params,
            fold_count = 5,
            plot = 'True',
            seed = 2020)
```

```
bestTest = 0.002279584376
bestIteration = 285

Shrink model to first 286 iterations.
```





Thank you!

References

- BayesianOptimization github
- Bayesian Optimization of XGBoost Parameters
- CatBoost – Anna Veronika Dorogush (Yandex) @ PyData
- CatBoost GPU Performance
- CatBoost Docs
- Details on CatBoost Feature Importances
- CatBoost: unbiased boosting with categorical features (paper)

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Startup Stock Photos