Northeastern University

Gradient Boosting Libraries

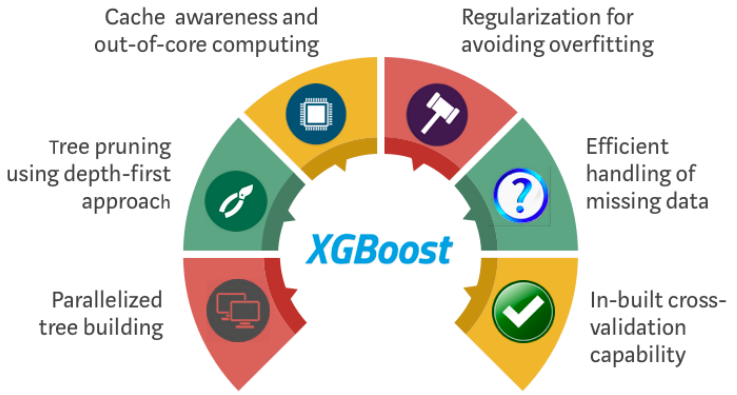
Roi Yehoshua

Agenda

- XGBoost

XGBoost (eXtreme Gradient Boosting)

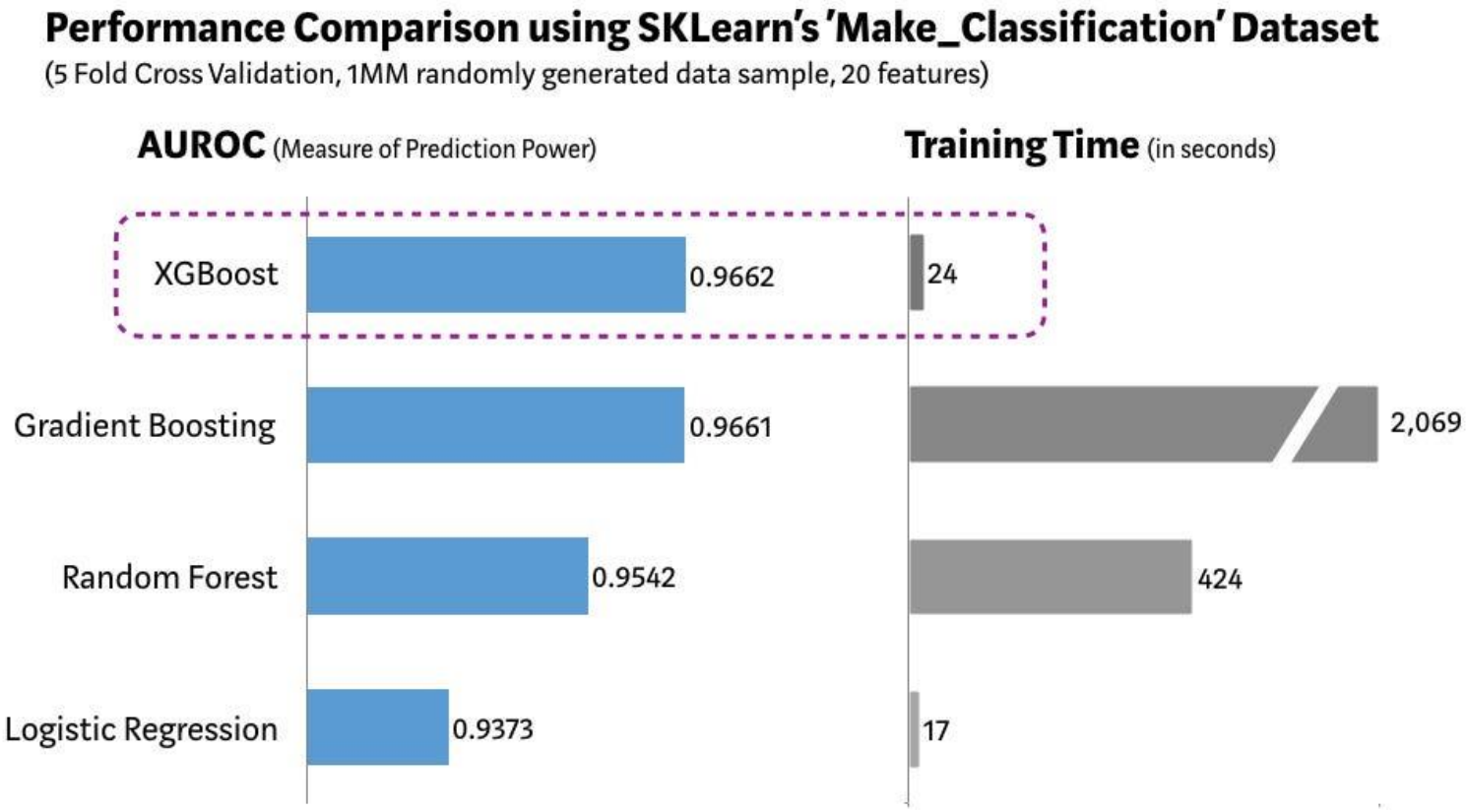
- An optimized distributed gradient boosting library
- Provides state-of-the-art results on many real-world data sets with structured data
- Can solve problems with billions of examples
- Can run on various distributed environments (e.g., Hadoop, AWS, Azure)
- Provides Scikit-Learn wrapper interfaces for its native classes
- Documentation of the library can be found at https://xgboost.readthedocs.io/
- Installation: pip install xgboost



XGBoost Features

- Use Newton's method in the function space instead of gradient descent
- Incorporates clever regularization techniques to penalize complex trees
- Uses a novel tree learning algorithm for handling sparse data
- Supports feature subsampling for each tree / tree level / tree node
- Uses theoretically justified quantile sketching for efficient computation
- Can handle missing values without imputation
- Supports parallel processing by distributing the workload across multiple threads
- Supports out-of-core computation using an efficient cacheable block structure
- Supports distributed computation
- Supports cross-validation
- Can handle wide range of problems, including classification, regression and ranking

XGBoost Features



XGBClassifier

```
class xgboost.XGBClassifier(*, objective='binary:logistic', use_label_encoder=None, **kwargs)
```

Parameter	Description
n_estimators	Number of boosting rounds (defaults to 100)
max_depth	Maximum tree depth for base learners (defaults to 6)
learning_rate	Boosting learning rate (defaults to 0.3)
objective	Specify the learning task and the corresponding learning objective
n_jobs	Number of parallel threads used to run xgboost
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree
subsample	Subsample ratio of the training set
reg_alpha	L1 regularization penalty term on weights
reg_lambda	L2 regularization penalty term on weights
scale_pos_weight	Balancing of positive and negative weights
early_stopping_rounds	Activates early stopping

XGBRegressor

```
class xgboost.XGBRegressor(*, objective='reg:squarederror', **kwargs)
```

- Has similar parameters to XGBClassifier

Classification Example

- Training an XGBoostClassifier on the Iris data set:

```
from xgboost import XGBClassifier

clf = XGBClassifier(random_state=42, max_depth=3, learning_rate=0.1)
clf.fit(X_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)
```

```
print(f'Train accuracy: {clf.score(X_train, y_train):.4f}')
print(f'Test accuracy: {clf.score(X_test, y_test):.4f}')
```

```
Train accuracy: 0.8839
Test accuracy: 0.7895
```