Experiences With XGBoost And The Financial Markets

(Or How To Avoid An Embarrassing Call From The Bugatti Garage)

PyData Bristol

@PyDataBristol

PyData meetups in London for data-loving pythonistas. Powered by @john_sandall, Frank Kelly, Miquel Perello Nieto.

Bristol, England

meetup.com/PyData-Bristol/

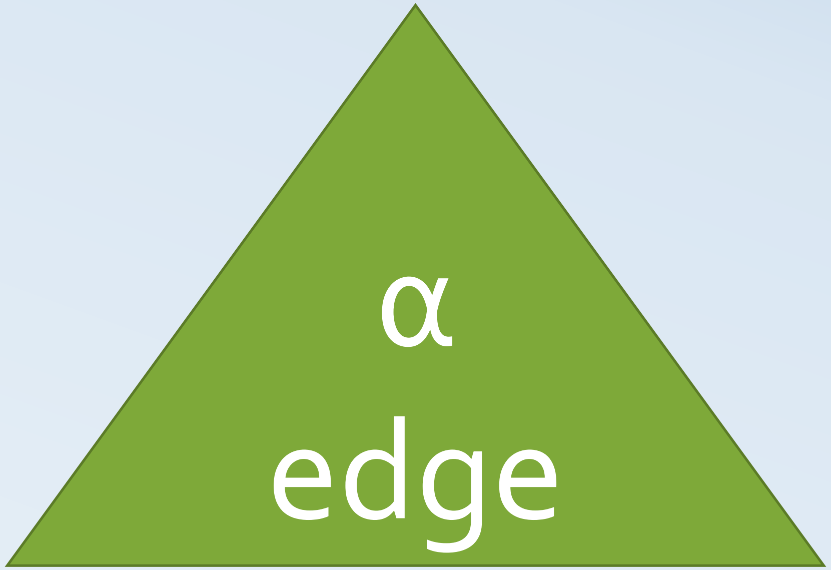
Sponsored by…

nano tick

on-demand high resolution historical market data including receive time at colocation with atomic clock precision

Strategy Paradigms

Fast: < 500 ns?



α

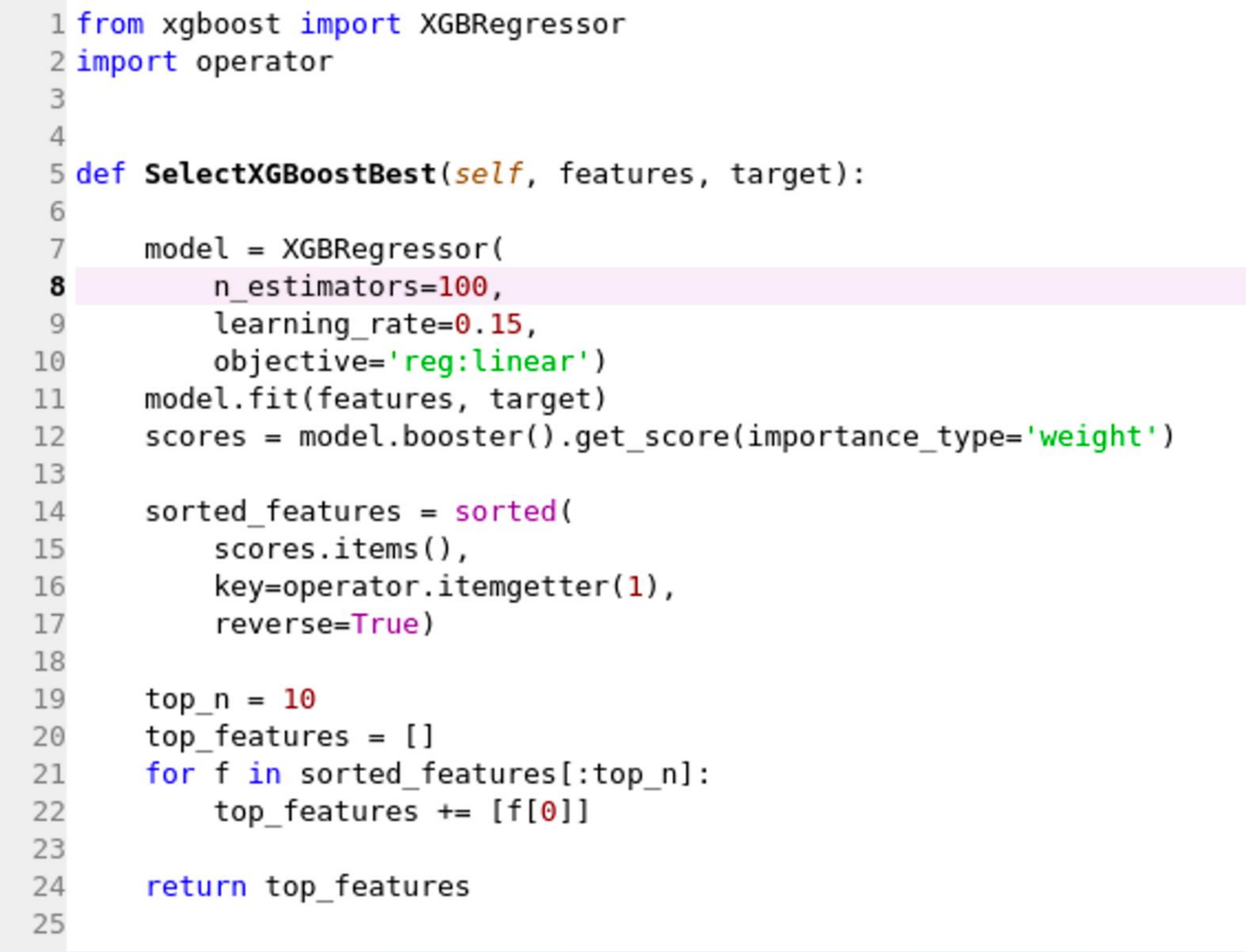
edge

Dirty / Naughty / Black edge

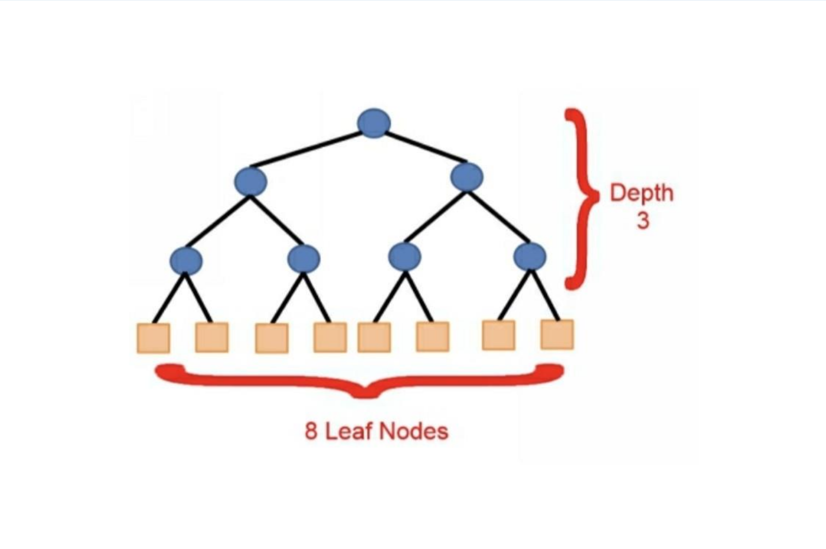
Smart

XGBoost & Decision Trees

- Non-linear
- Feature range insensitive
- Interpretable
- Feature selection & ranking



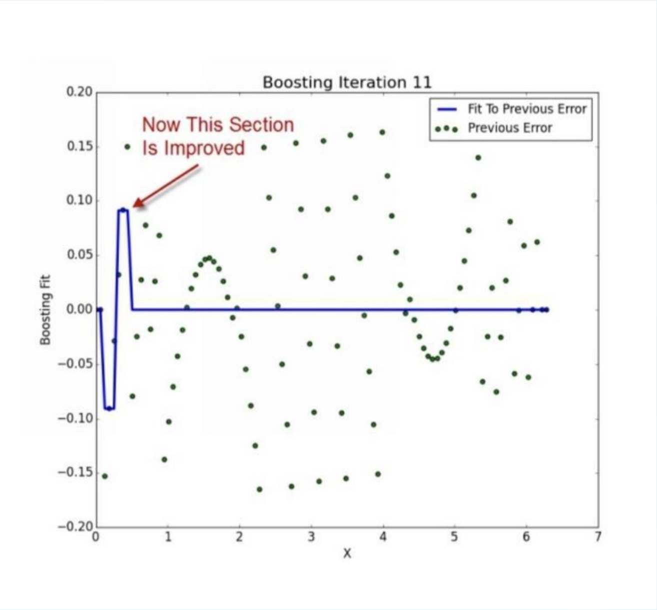
```python
from xgboost import XGBRegressor
import operator


def SelectXGBoostBest(self, features, target):

    model = XGBRegressor(
        n_estimators=100,
        learning_rate=0.15,
        objective='reg:linear')
    model.fit(features, target)
    scores = model.booster().get_score(importance_type='weight')

    sorted_features = sorted(
        scores.items(),
        key=operator.itemgetter(1),
        reverse=True)

    top_n = 10
    top_features = []
    for f in sorted_features[:top_n]:
        top_features += [f[0]]

    return top_features

```

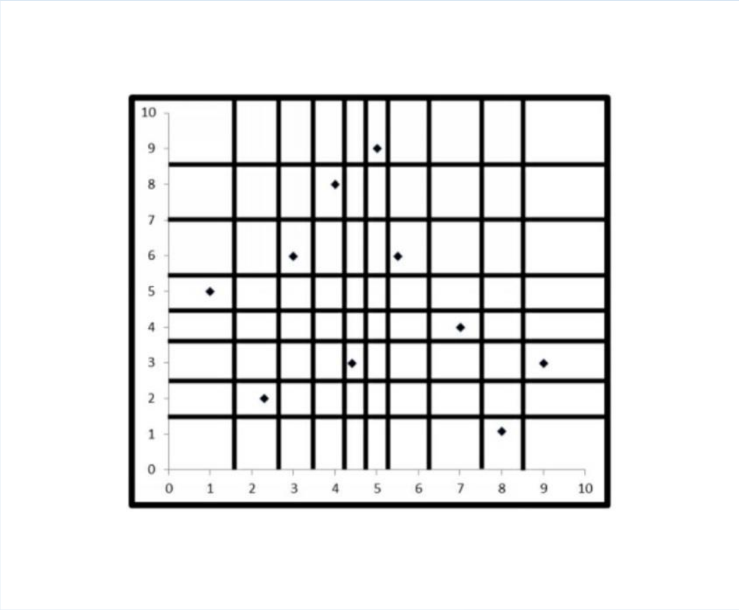


“Machine learning with boosting, a beginner’s guide”, Scott Hartshorn

Gradient Boosted Decision Tree Algorithm

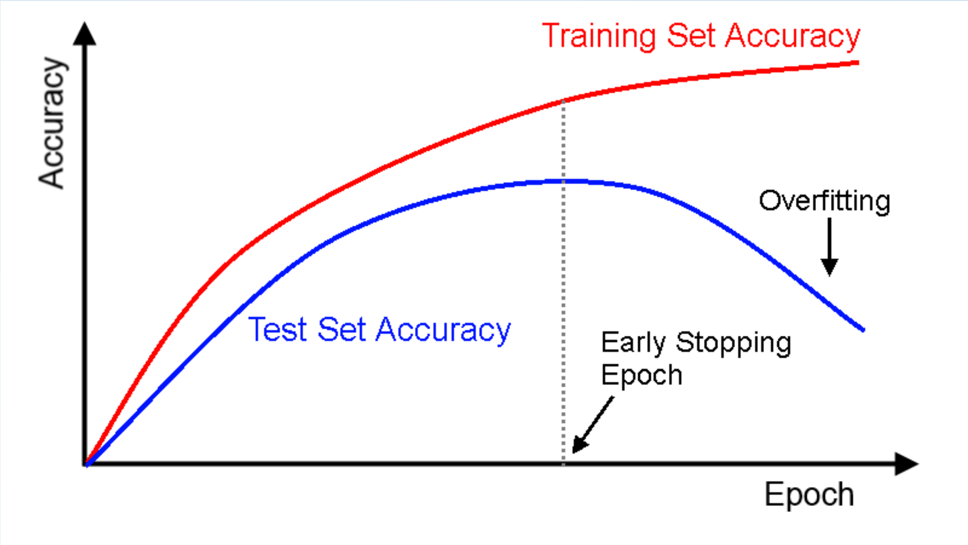
- Iterative algorithm, incremental addition of weak learners focused on areas of poor prediction
- XGBoost has both native and scikit learn style python APIs, very quick to get going
- Multi-platform, for instance linux for simulation and windows for live trading





“Machine learning with boosting, a beginner’s guide”, Scott Hartshorn

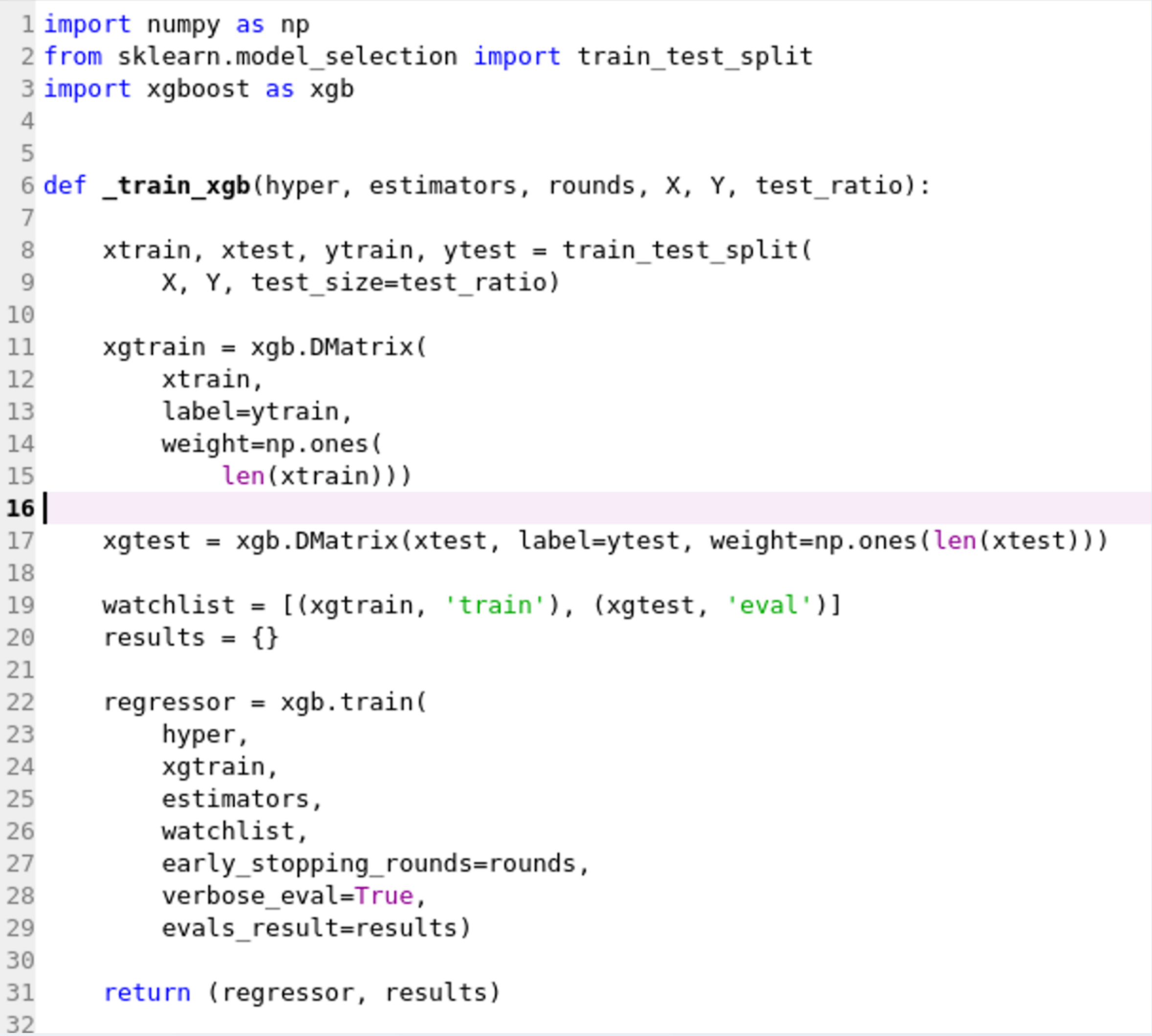
Robustness & Overfitting



https://deeplearning4j.org/earlystopping

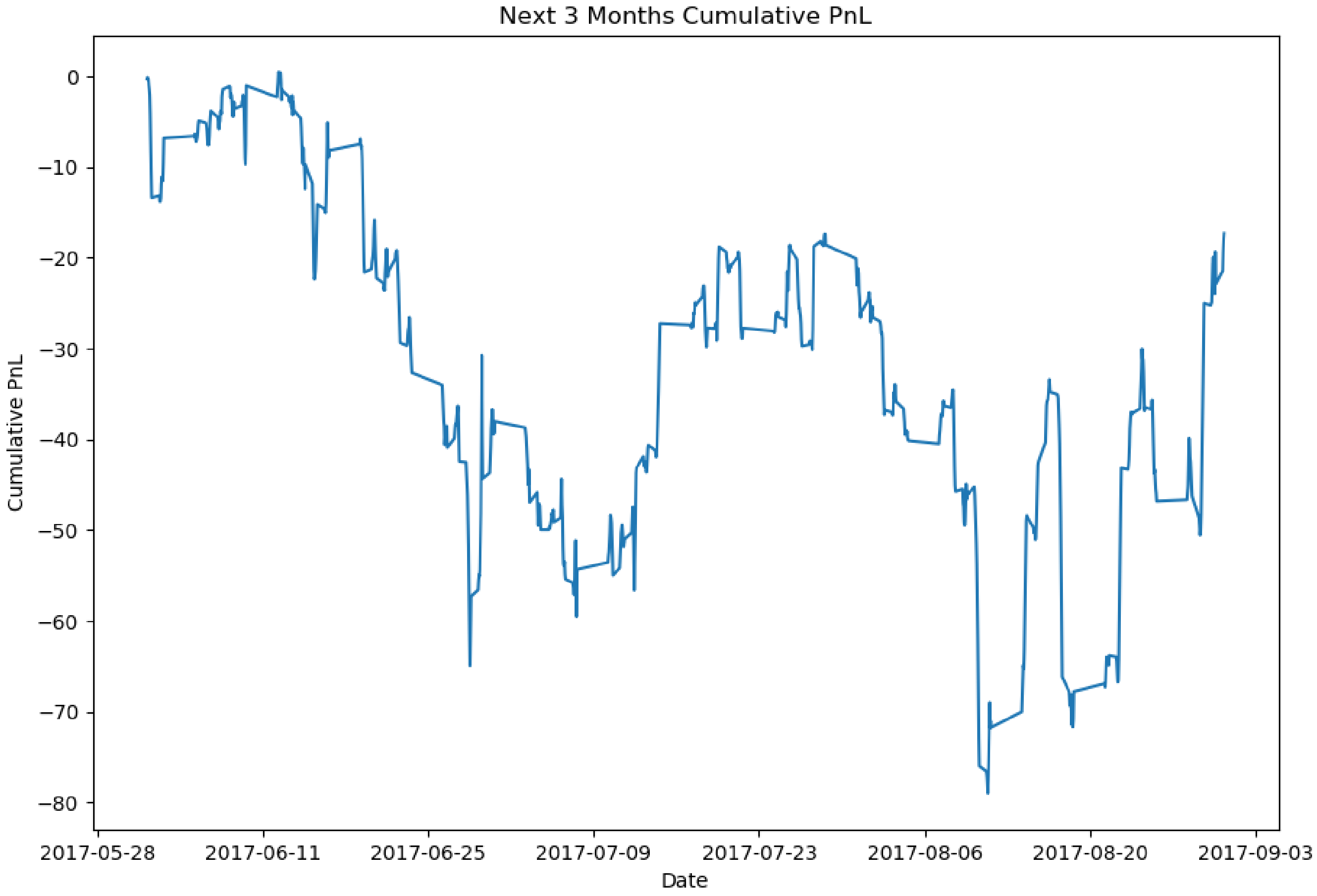
- Loss of robustness when training set improves but test does not
- XGBoost has a whole host of robustness settings e.g. early stopping

Early Stopping & Evaluation Data Selection

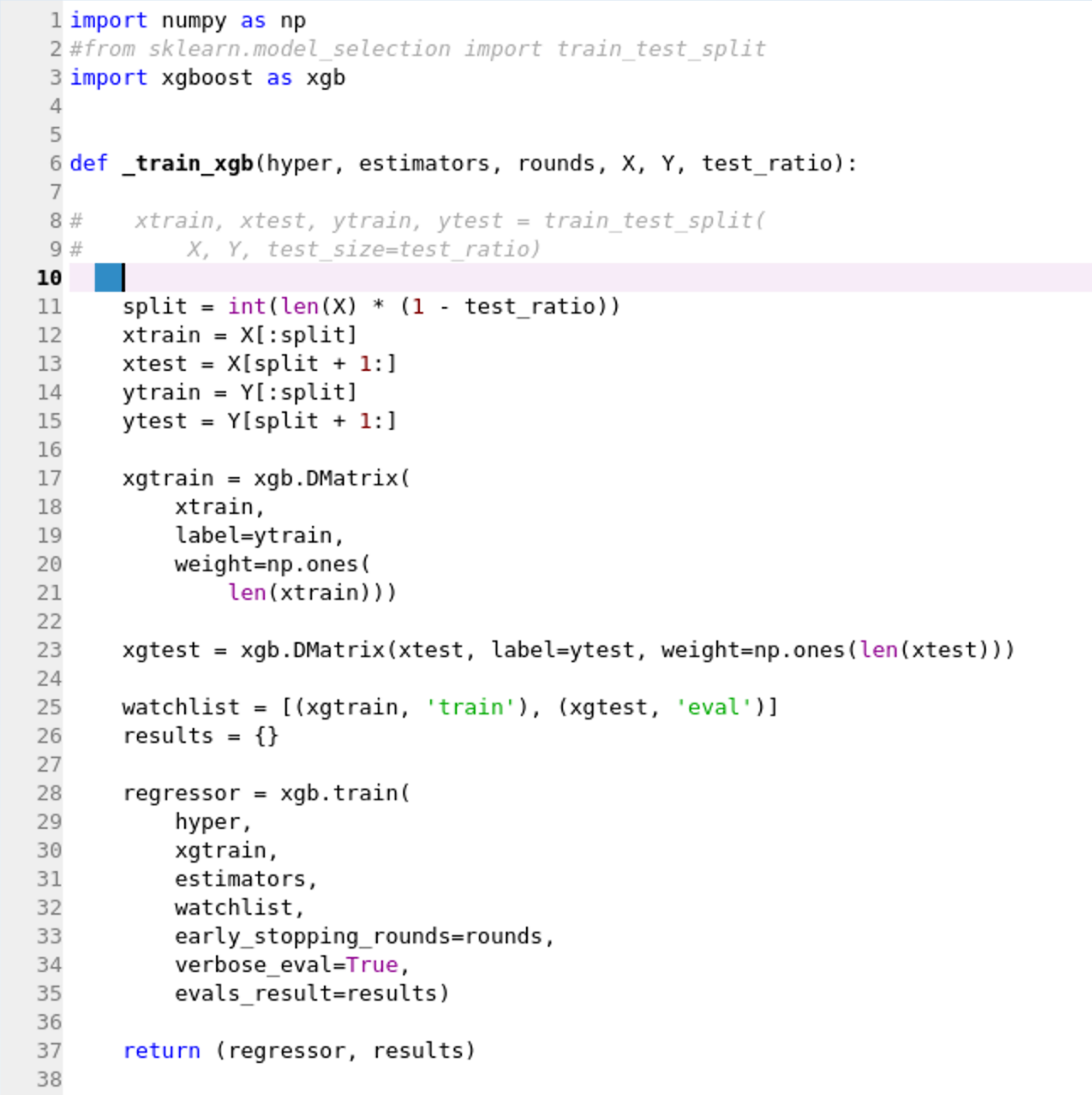


```python
import numpy as np
from sklearn.model_selection import train_test_split
import xgboost as xgb


def _train_xgb(hyper, estimators, rounds, X, Y, test_ratio):

    xtrain, xtest, ytrain, ytest = train_test_split(
        X, Y, test_size=test_ratio)

    xgtrain = xgb.DMatrix(
        xtrain,
        label=ytrain,
        weight=np.ones(
            len(xtrain)))

    xgtest = xgb.DMatrix(xtest, label=ytest, weight=np.ones(len(xtest)))

    watchlist = [(xgtrain, 'train'), (xgtest, 'eval')]
    results = {}

    regressor = xgb.train(
        hyper,
        xgtrain,
        estimators,
        watchlist,
        early_stopping_rounds=rounds,
        verbose_eval=True,
        evals_result=results)

    return (regressor, results)

```

The Next 3 Months…



Beware Random Cross Validation!



```python
import numpy as np
#from sklearn.model_selection import train_test_split
import xgboost as xgb


def _train_xgb(hyper, estimators, rounds, X, Y, test_ratio):

#    xtrain, xtest, ytrain, ytest = train_test_split(
#        X, Y, test_size=test_ratio)
    
    split = int(len(X) * (1 - test_ratio))
    xtrain = X[:split]
    xtest = X[split + 1:]
    ytrain = Y[:split]
    ytest = Y[split + 1:]

    xgtrain = xgb.DMatrix(
        xtrain,
        label=ytrain,
        weight=np.ones(
            len(xtrain)))

    xgtest = xgb.DMatrix(xtest, label=ytest, weight=np.ones(len(xtest)))

    watchlist = [(xgtrain, 'train'), (xgtest, 'eval')]
    results = {}

    regressor = xgb.train(
        hyper,
        xgtrain,
        estimators,
        watchlist,
        early_stopping_rounds=rounds,
        verbose_eval=True,
        evals_result=results)

    return (regressor, results)

```