

# Wine Quality

AWS Sagemaker - XG Boost

# Problem Being Solved & Dataset

# Guidelines

**Problem being solved:** Can machine learning be used to determine if a wine is good given its physico-chemical properties? If we get a dataset with the amount of chlorides, acidity of a wine, its residual sugars, citric acid, sulfur dioxide and other properties, will an AI find out if a particular wine is of grade 5, 6 or 8 for example? Let's find out.

**Dataset and features :** The project is based on a dataset donated to the UCI machine learning repository in 2009, <https://archive.ics.uci.edu/ml/datasets/wine+quality> and it was collected by: Paulo Cortez, University of Minho, Guimarães, Portugal, <http://www3.dsi.uminho.pt/pcortez> A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal @2009

**Relationship between the data and problem being solved:** The dataset collected seems perfect for the problem. It provides all the physicochemical characteristics of white wines and about 4898 wines analysed. This dataset also looks great for the fact that all the data is completed and there are no empty/blank data in the .csv file.

# Guidelines

**How an update to the dataset will impact the solution:** This dataset is a precious material for sommeliers all over the world. They can get information about a wine's acidity and grade instantly. The idea is that any person can insert all the characteristics of a new wine in the dataset and get an estimated grade about its quality. To update it; the user would need to insert: both fixed & volatile acidity, residual sugars, citric acid, chlorides, free & total sulfur dioxide, pH, sulphates, alcohol, and density to get an estimated quality grade between 1-9. The dataset doesn't regard region of wine production or year; only its chemical information.

**The pros and the cons of the available data:** The pros is that the data was publicly available. In the field of data science; I'm happy to refer the authors Paulo Cortez, University of Minho, Guimarães, Portugal, <http://www3.dsi.uminho.pt/pcortez> A. Cerdeira, F. Almeida, T. Matos and J. Reis as an appreciation gesture to their work. The dataset is supervised learning and all the features and labels were completed, with no missing information for the 4898 wines (labels) and its 12 features.

**The additional resources that will help to improve the solution and possible means to collect it:** Since this dataset is already 11 years old; it would be interesting to get additional recent data to compare and improve the dataset with more reliable data. The only way to collect this would be based on generosity of portuguese white wine producers.

# The Dataset

```
!wget -N https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv
--2020-12-03 14:10:49--  https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 304 Not Modified
File 'winequality-white.csv' not modified on server. Omitting download.

!head winequality-white.csv
"fixed acidity";"volatile acidity";"citric acid";"residual sugar";"chlorides";"free sulfur dioxide";"total sulfur dioxide";"density";"pH";"sulphates";"alcohol";"quality"
7;0.27;0.36;20.7;0.045;45;170;1.001;3;0.45;8.8;6
6.3;0.3;0.34;1.6;0.049;14;132;0.994;3.3;0.49;9.5;6
8.1;0.28;0.4;6.9;0.05;30;97;0.9951;3.26;0.44;10.1;6
7.2;0.23;0.32;8.5;0.058;47;186;0.9956;3.19;0.4;9.9;6
7.2;0.23;0.32;8.5;0.058;47;186;0.9956;3.19;0.4;9.9;6
8.1;0.28;0.4;6.9;0.05;30;97;0.9951;3.26;0.44;10.1;6
6.2;0.32;0.16;7;0.045;30;136;0.9949;3.18;0.47;9.6;6
7;0.27;0.36;20.7;0.045;45;170;1.001;3;0.45;8.8;6
6.3;0.3;0.34;1.6;0.049;14;132;0.994;3.3;0.49;9.5;6

import numpy as np # For matrix operations and numerical processing
import pandas as pd # For munging tabular data

# https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html
data = pd.read_csv('winequality-white.csv', sep=';')
pd.set_option('display.max_columns', 500)      # Make sure we can see all of the columns
pd.set_option('display.max_rows', 50)           # Keep the output on one page
data[:5]

   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  free sulfur dioxide  total sulfur dioxide  density  pH  sulphates  alcohol  quality
0            7.0            0.27            0.36          20.7      0.045             45.0                170.0     1.0010    3.00        0.45       8.8       6
1            6.3            0.30            0.34            1.6      0.049             14.0                132.0     0.9940    3.30        0.49       9.5       6
2            8.1            0.28            0.40            6.9      0.050             30.0                97.0     0.9951    3.26        0.44      10.1       6
3            7.2            0.23            0.32            8.5      0.058             47.0                186.0     0.9956    3.19        0.40       9.9       6
4            7.2            0.23            0.32            8.5      0.058             47.0                186.0     0.9956    3.19        0.40       9.9       6

data.shape # (number of lines, number of columns)
(4898, 12)
```

# Julien Simon - XGBoost Tutorial

<https://medium.com/@julsimon/aws-ai-machine-learning-podcast-episode-11-xgboost-special-492f64356a29>  
<https://www.youtube.com/watch?v=wOF4zodMdzl&t=206s>

His problem: Classification / Direct Marketing Dataset //using just 1 label (y\_yes):

XGBoost Classifier: { objective: binary\_logistic ; eval\_metrics: auc ; max\_depth: max\_depth}

**Bibliography** : <https://gitlab.com/juliensimon/ent321/-/blob/master/ENT321%20-%20short%20version.ipynb>

4 days stuck in one line of code due to an SDK version update and the clue went straight to my Junkmail folder::

Lauren Yu Inbox - julia...itsencrypted.com December 1, 2020 at 7:54 PM Details

Re: [aws/sagemaker-python-sdk] SDK 2.0: image\_uris.retrieve() fails for XGBoost 1.0-1 (#1748)

To: aws/sagemaker-python-sdk, Cc: itsencrypted, Mention,

Reply-To: aws/sagemaker-python-sdk

This message is from a mailing list. Unsubscribe

@itsencrypted you need to swap the order of your first two arguments for retrieve():

```
container = sagemaker.image_uris.retrieve('xgboost', region, 'latest')
```

docs: [https://sagemaker.readthedocs.io/en/stable/api/utility/image\\_uris.html#sagemaker.image\\_uris.retrieve](https://sagemaker.readthedocs.io/en/stable/api/utility/image_uris.html#sagemaker.image_uris.retrieve)

if you need more assistance, please open a new issue (since this one is closed).

You are receiving this because you were mentioned.  
Reply to this email directly, [view it on GitHub](#), or [unsubscribe](#).

```
from sagemaker.amazon.amazon_estimator import get_image_uri

container = get_image_uri(boto3.Session().region_name, 'xgboost')
role      = sagemaker.get_execution_role() # Role when working on a notebook instance
sess     = sagemaker.Session()

xgb = sagemaker.estimator.Estimator(container,
                                      role,
                                      train_instance_count=1,
                                      train_instance_type='ml.m4.2xlarge',
                                      input_mode="File",
                                      output_path='s3://{}//{}//output'.format(bucket, prefix),
                                      sagemaker_session=sess)
```

# Jupyter Notebook - Data Manipulation

```
29 7.2 0.32 0.36 2.00 0.033 37.0 114.0 0.9900 3.10 0.71 12.3 / /  
  
In [11]: data.shape # (number of lines, number of columns)  
Out[11]: (4898, 12)  
  
In [27]: one_class = data[data['quality']<=7]  
one_class_count = one_class.shape[0]  
print("Normal quality wines: %d" % one_class_count)  
  
zero_class = data[data['quality']>=8]  
zero_class_count = zero_class.shape[0]  
print("Excellent Quality wines: %d" % zero_class_count)  
  
zero_to_one_ratio = zero_class_count/one_class_count  
print("Ratio: %.2f" % zero_to_one_ratio)  
  
Normal quality wines: 4718  
Excellent Quality wines: 180  
Ratio: 0.04  
  
In [30]: [np.min(data['residual sugar']), np.max(data['residual sugar'])]  
Out[30]: [0.6, 65.8]  
  
In [31]: #wines with very high residual sugar are low quality. wines with a balanced residual sugar are better quality  
  
In [32]: # Indicator variable to capture when residual sugar takes a value higher than 2.5  
# https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.where.html  
data['very sweet'] = np.where(data['residual sugar'] >= 2.5, 1, 0)  
data = data.drop(['residual sugar'], axis=1)  
  
In [33]: data['fixed acidity'].value_counts()
```

**Initial idea:** do a classification model and separate the wine between “good quality” = 1 and “not good quality” = 0

 **Julien Simon**  
@julsimon 

Joined March 2009



itsencrypted/winequality  
Using an XG Boost Training Model to p...  
[github.com/itsencrypted/winequality](https://github.com/itsencrypted/winequality)

Salut Julien! It's nice to connect with you here. I'm finalizing a course about Sagemaker and for the final project, I plan to deliver an XG Boost model to predict wine quality. The whole model was based on your tutorial dated from February, about a telemarketing campaign from a bank. Is there any chance that you could take a look at my repo and point me if I'm in the right direction?  
I'm a horrible algorithm builder! :-)  
[github.com/itsencrypted/winequality](https://github.com/itsencrypted/winequality)

Let me know if the train/test/validation outputs make sense before I start playing with Sagemaker and the hyperparameters.

Tks so much for your work & tutorials !! All the best

Nov 21, 2020, 9:15 PM ✓

 Hi Juju, I took a quick look and I think you're on the right track. You can go with a binary classification model, and you can also try a regression model that would predict your 1-10 note. XGBoost can do that too :) Keep me posted!

Nov 23, 2020, 7:21 AM

# Feature Engineering

# Feature Engineering

The features and basic characteristics

The process of creating features from raw data

The statistics of the features after feature engineering & comment on possible variations

Create multiple datasets with different feature engineering techniques and describe the results

# Training Model

WineQuality - XGBoost - Good | Amazon SageMaker | JupyterLab

xgboost-winequality.notebook.us-east-1.sagemaker.aws.lab

File Edit View Run Kernel Git Tab Settings Help

/ xgboost\_regression / wh

Name

- test.csv
- train.csv
- winequality-white.csv
- xgb\_reg\_whitewineq..**

Launcher x g x Code v git conda\_python3

```
[18]: from sagemaker.tuner import HyperparameterTuner
tuner = HyperparameterTuner(xgb,
    objective_metric_name='rmse',
    hyperparameter_ranges,
    objective_type='objective_type',
    max_jobs=10,
    max_parallel_jobs=1)

[19]: xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

2020-12-04 01:36:31 Starting - Starting the training job...

2020-12-04 01:36:34 Starting - Launching requested ML instances.....

2020-12-04 01:38:06 Starting - Preparing the instances for training.....

2020-12-04 01:39:53 Downloading - Downloading input data...

2020-12-04 01:40:15 Training - Downloading the training image.Arguments: train

[2020-12-04 01:40:36:INFO] Running standalone xgboost training.

[2020-12-04 01:40:36:INFO] File size to be processed in the node: 0.35mb. Available memory size in the node: 24413.25mb

[2020-12-04 01:40:36:INFO] Determined delimiter of CSV input is ','

[01:40:36] 3919x11 matrix with 43998 entries loaded from /opt/ml/input/data/train?format=csv&label\_column=0&delimiter=,

[01:40:36] 44x11 matrix with 43998 entries loaded from /opt/ml/input/data/validation?format=csv&label\_column=0&delimiter=,

[01:40:36] Determined delimiter of CSV input is ','

WineQuality - XGBoost - Good | Amazon SageMaker | JupyterLab

xgboost-winequality.notebook.us-east-1.sagemaker.aws.lab

File Edit View Run Kernel Git Tab Settings Help

/ xgboost\_regression / wh

Name

- test.csv
- train.csv
- winequality-white.csv
- xgb\_reg\_whitewineq..**

Launcher x g x Code v git conda\_python3

```
[91]: #!#1train-rmse@0.39780#01validation-rmse@0.651672
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 56 extra nodes, 6 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 60 extra nodes, 0 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 58 extra nodes, 0 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 56 extra nodes, 2 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 50 extra nodes, 12 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 60 extra nodes, 2 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46 extra nodes, 4 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46 extra nodes, 4 pruned nodes, max_depth=5
[01:40:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46 extra nodes, 4 pruned nodes, max_depth=5
[01:40:36] Stopping. Best iteration:
```

2020-12-04 01:41:00 Uploading - Uploading generated training model

2020-12-04 01:41:00 Completed - Training job completed

Training seconds: 67

Billable seconds: 67

```
[*]: tuner.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

Amazon SageMaker Studio

Amazon SageMaker > Training jobs

Training jobs

Name	Creation time	Duration	Status
xgboost-201204-0141-002-4e5754f1	Dec 04, 2020 01:45 UTC	4 minutes	Failed
xgboost-201204-0141-001-f79d1da	Dec 04, 2020 01:41 UTC	3 minutes	Failed
xgboost-2020-12-04-01-36-31-508	Dec 04, 2020 01:36 UTC	4 minutes	Completed
sagemaker-xgboost-2020-12-03-14-17-15-860	Dec 03, 2020 14:17 UTC	4 minutes	Failed
sagemaker-xgboost-2020-12-03-14-11-18-441	Dec 03, 2020 14:11 UTC	4 minutes	Failed
sagemaker-xgboost-2020-12-03-14-00-40-778	Dec 03, 2020 14:00 UTC	3 minutes	Failed
sagemaker-xgboost-2020-12-02-15-57-17-817	Dec 02, 2020 15:57 UTC	3 minutes	Completed
sagemaker-xgboost-201202-1516-004-7fe5f5b	Dec 02, 2020 15:29 UTC	3 minutes	Failed
sagemaker-xgboost-201202-1516-003-7bc3a5c	Dec 02, 2020 15:24 UTC	4 minutes	Failed

aws Services

CloudWatch Dashboards Alarms

CloudWatch Metrics

Metrics

Untitled graph

validation:rmse

validation:rmse

2020-12-04 01:40 UTC

0.645

0.528

0.405

23:00 23:15 23:30 23:45 00:00 00:15 00:30 00:45 01:00 01:15 01:30 01:45 01:59 12:04 01:39

All metrics Graphed metrics (2) Graph options Source

N. Virginia All /aws/sagemaker/TrainingJobs > TrainingJobName xgboost-2020-12-04-01-36-31-508

TrainingJobName (2)

xgboost-2020-12-04-01-36-31-508 train:rmse

xgboost-2020-12-04-01-36-31-508 validation:rmse

Graph search

# About the Algorithm chosen

## XGBoost Algorithm

[PDF](#) | [Kindle](#) | [RSS](#)

The [XGBoost](#) (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. The XGBoost algorithm performs well in machine learning competitions because of its robust handling of a variety of data types, relationships, distributions, and the variety of hyperparameters that you can fine-tune. You can use XGBoost for regression, classification (binary and multiclass), and ranking problems.

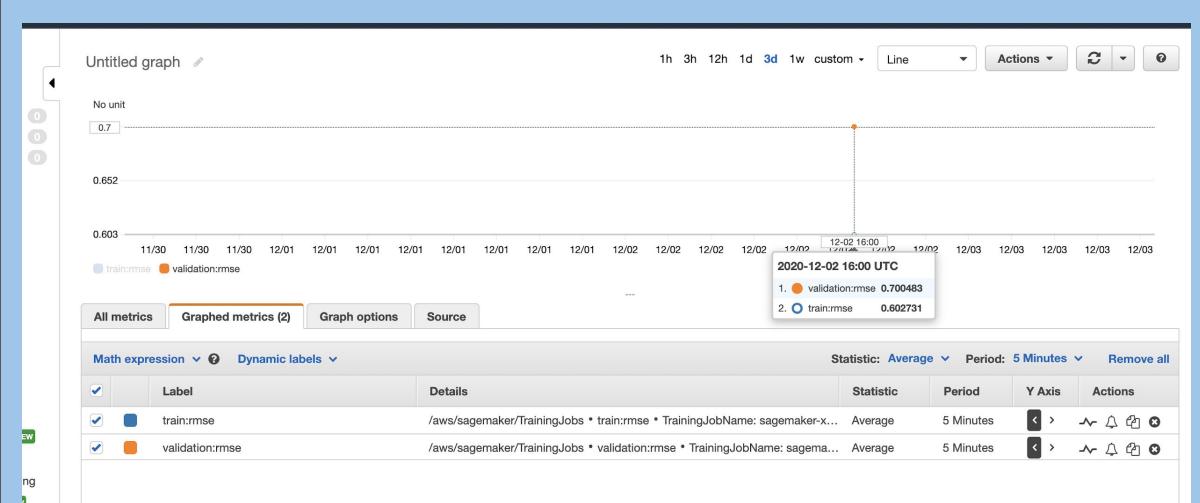
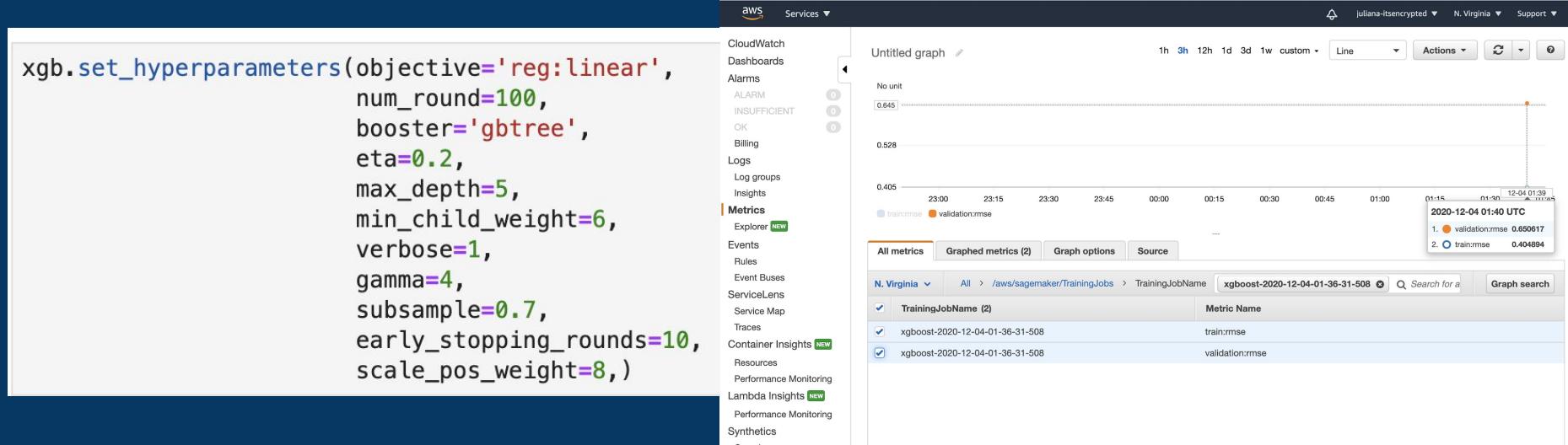
You can use the new release of the XGBoost algorithm either as a Amazon SageMaker built-in algorithm or as a framework to run training scripts in your local environments. This implementation has a smaller memory footprint, better logging, improved hyperparameter validation, and an expanded set of metrics than the original versions. It provides an [XGBoost estimator](#) that executes a training script in a managed XGBoost environment. The current release of SageMaker XGBoost is based on the original XGBoost versions 0.90, 1.0, and 1.2.

# Algorithm Tuning

**Choose and algorithm and run it with default values** - see slides 12, 14, 15 and 16 for screenshots of the hyperparameters chosen with XG Boost Algorithm

**Hyperparameter tune different algorithms in a systematic way and choose the best one** : this job is an automated feature provided by AWS. The Python SDK (from v. 0.90-2 until its newest stable version 2.0) allowing Data Scientist Developers to run many training jobs in parallel. With only a few lines of code; the SDK finds the best combination of hyperparameters to maximize the training job results. The **tuner.fit()** function takes a few minutes (20' sometimes) to run many jobs and bring the best training job back to become the model to go to deployment/production. (see slides

Hyperparameter tune different algorithms in combination with different feature engineering



## Hyperparameters

Key	Value
num_round	10
objective	reg:linear
scale_pos_weight	1

# Uploading train/test data into an S3 bucket

The screenshot shows the AWS S3 console interface. At the top, a green success banner displays the message "Successfully created folder 'output' Operation successfully completed." Below the banner, the navigation path is shown as "Amazon S3 > xgboost-winequality > whitewine/". The main area is titled "whitewine/" and contains a "Folder overview" section. This section includes details about the Region (US East (N. Virginia) us-east-1), the S3 URI (<s3://xgboost-winequality/whitewine/>), and the Amazon resource name (ARN) ([arn:aws:s3:::xgboost-winequality/whitewine/](#)). A large dashed box allows for dragging and dropping files or folders to upload them. Below this is a "Objects (4)" section. It features a toolbar with "Delete", "Actions", "Create folder", and "Upload" buttons, along with a search bar. The table lists four objects:

Name	Type	Last modified	Size	Storage class
output/	Folder	-	-	-
test.csv	csv	November 26, 2020, 18:17 (UTC-02:00)	72.4 KB	Standard
train.csv	csv	November 26, 2020, 18:17 (UTC-02:00)	289.4 KB	Standard
winequality-white.csv	csv	November 26, 2020, 18:17 (UTC-02:00)	258.2 KB	Standard

# SageMaker - Training Model

## Setting the Hyperparameters

```
xgb.set_hyperparameters(objective='reg:linear', num_round=100, booster=gbtree, eta=0.3, max_depth=6,  
min_child_weight=1.0, scale_pos_weight=1, eval_metric=rmse)
```

Hyperparameters	
Key	Value
eta	0.2
gamma	4
max_depth	5
min_child_weight	6
num_round	50
objective	reg:linear
silent	0
subsample	0.7

```
xgb.set_hyperparameters(objective='reg:linear',  
                      num_round=100,  
                      booster='gbtree',  
                      eta=0.2,  
                      max_depth=5,  
                      min_child_weight=6,  
                      verbose=1,  
                      gamma=4,  
                      subsample=0.7,  
                      early_stopping_rounds=10,  
                      scale_pos_weight=8, )
```

# Hyperparameters

- **eta**: Step size shrinkage used in updates to prevent overfitting. After each boosting step, you can directly get the weights of new features. The eta parameter actually shrinks the feature weights to make the boosting process more conservative.
- **alpha**: L1 regularization term on weights. Increasing this value makes models more conservative.
- **min\_child\_weight**: Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min\_child\_weight, the building process gives up further partitioning. In linear regression models, this simply corresponds to a minimum number of instances needed in each node. The larger the algorithm, the more conservative it is.
- **max\_depth**: Maximum depth of a tree. Increasing this value makes the model more complex and likely to be overfitted.

XGBoost algorithm has dozens of hyperparameters and we need to pick the right values for those hyperparameters in order to achieve the desired model training results. Since each hyperparameter setting can lead to the best result depends on the dataset as well, it is almost impossible to pick the best hyperparameter setting without searching for it, and a good search algorithm can search for the **best hyperparameter setting in an automated and effective way**.

I used SageMaker hyperparameter tuning to automate the searching process effectively. Specifically, we specify a range, or a list of possible values in the case of categorical hyperparameters, **for each of the hyperparameter that we plan to tune**. **SageMaker hyperparameter tuning will automatically launch multiple training jobs with different hyperparameter settings**, evaluate results of those training jobs based on a predefined "objective metric", and select the hyperparameter settings for future attempts based on previous results. For each hyperparameter tuning job, we gave it a budget (max number of training jobs) and it completed once that many training jobs have been executed. (**max jobs =10**)

All this can be done using SageMaker Python SDK to set up and manage the hyperparameter tuning job. First, I configured the training jobs the hyperparameter tuning job will launch by initiating an estimator, which includes:

The container image for the algorithm (XGBoost)

```
container = get_image_uri(region, 'xgboost', repo_version='latest')
```

Configuration for the output of the training jobs

The values of static algorithm hyperparameters, those that are not specified will be given default values

The type and number of instances to use for the training jobs

# ML Algorithm Tuning

## Testing the AI Service

# Testing the AI Service

Report the test accuracy / RMSE on the test dataset

Report several other performance metrics that might be relevant

Add noise to the features of the test dataset and report the sensitivity of the algorithm to noise on different features

# Model Tuning

Tuning resource configurations such as: Number of training jobs to run in total and how many training jobs can be run in parallel.

Next we'll specify the objective metric that we'd like to tune and its definition, which includes the regular expression (Regex) needed to extract that metric from the CloudWatch logs of the training job. Since we are using built-in **XGBoost algorithm** here, it **emits two predefined metrics: validation:auc and train:auc**, and I initially elected to monitor validation:auc but got error messages mentioning that it only worked for datasets with a mix of positive and negative numbers. Hence, my struggle to find another good objective metric.

In this case, we only need to specify the metric name and do not need to provide regex.

Attention If you bring your own algorithm, your algorithm emits metrics by itself. In that case, you'll need to add a MetricDefinition object here to define the format of those metrics through regex, so that SageMaker knows how to extract those metrics from your CloudWatch logs.

In [27]:

```
from sagemaker.tuner import IntegerParameter, ContinuousParameter

hyperparameter_ranges = {'eta': ContinuousParameter(0, 1),
                         'min_child_weight': ContinuousParameter(1, 10),
                         'alpha': ContinuousParameter(0, 2),
                         'max_depth': IntegerParameter(2, 8)}
```

In [28]:

```
objective_metric_name = 'validation:auc'
objective_type = 'Maximize'
```

<https://xgboost-winequality.notebook.us-east-1.sagemaker.aws/lab>

5/1

12/3/2020

xgb\_reg\_whitewinequality

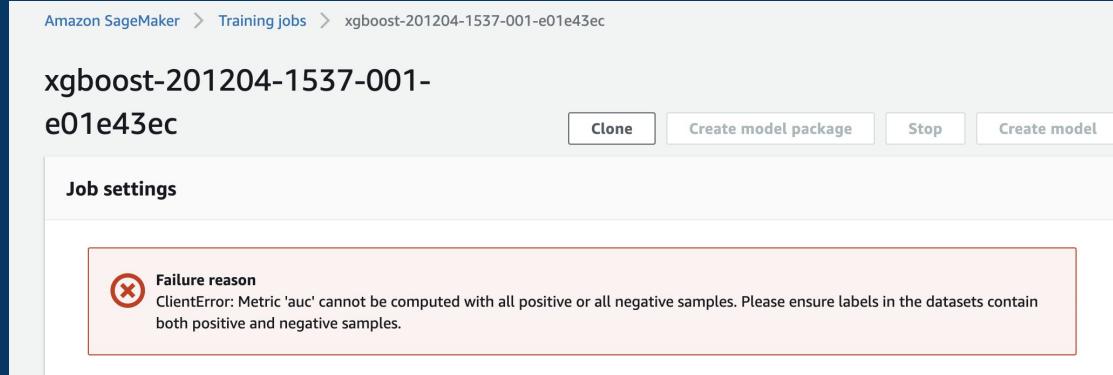
In [29]:

```
from sagemaker.tuner import HyperparameterTuner

tuner = HyperparameterTuner(xgb,
                             objective_metric_name,
                             hyperparameter_ranges,
                             objective_type=objective_type,
                             max_jobs=10,
                             max_parallel_jobs=1)
```

# AWS Documentation for Model Tuning

The following guide: <https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-tuning.html> allowed me to remember that **linear regression** requires **Root Mean Square Error (rmse)**; so, after many trial and errors I finally found the following error message:



which was causing the majority of the training jobs to fail. When I finally changed the model to tune itself automatically optimizing **validation:rmse** to the lowest amount (**objective\_type: 'Minimize'**); I got successful training jobs marked as **completed**

**The Hyperparameter Tuner actually looks to the model with the highest precision; in other words, where the square root mean error is the lowest.**

# Some light in the end of the tunnel

Training jobs				
	Name	Creation time	Duration	Status
<input type="text"/> Search training jobs				
<input type="radio"/>	xgboost-201204-1541-006-fbb18817	Dec 04, 2020 16:00 UTC	4 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1541-005-8c54292e	Dec 04, 2020 15:57 UTC	3 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1537-005-f631c015	Dec 04, 2020 15:54 UTC	3 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1541-004-35b3176f	Dec 04, 2020 15:53 UTC	4 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1537-004-3f1be815	Dec 04, 2020 15:50 UTC	4 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1541-003-549fe008	Dec 04, 2020 15:49 UTC	4 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1537-003-619d87d3	Dec 04, 2020 15:46 UTC	3 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1541-002-9dfa3ace	Dec 04, 2020 15:45 UTC	3 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1537-002-a2a6f51d	Dec 04, 2020 15:42 UTC	3 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1541-001-fae30744	Dec 04, 2020 15:41 UTC	3 minutes	✓ Completed
<input type="radio"/>	xgboost-201204-1537-001-e01e43ec	Dec 04, 2020 15:37 UTC	5 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1456-005-2305da19	Dec 04, 2020 15:12 UTC	3 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1456-004-0c7fdb06	Dec 04, 2020 15:08 UTC	4 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1456-003-13b7bd75	Dec 04, 2020 15:04 UTC	4 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1456-002-d570563a	Dec 04, 2020 15:00 UTC	3 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-1456-001-c3c39703	Dec 04, 2020 14:56 UTC	4 minutes	✗ Failed
<input type="radio"/>	xgboost-201204-0245-005-97371eae	Dec 04, 2020 02:59 UTC	4 minutes	✗ Failed

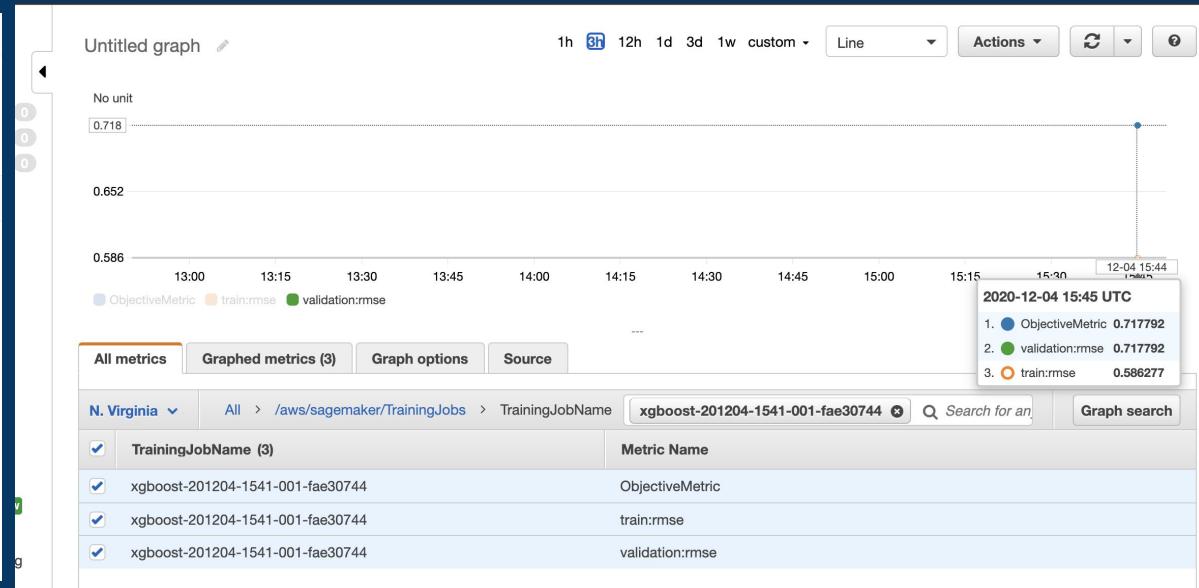
# Automatic Model Tuning

Output data configuration

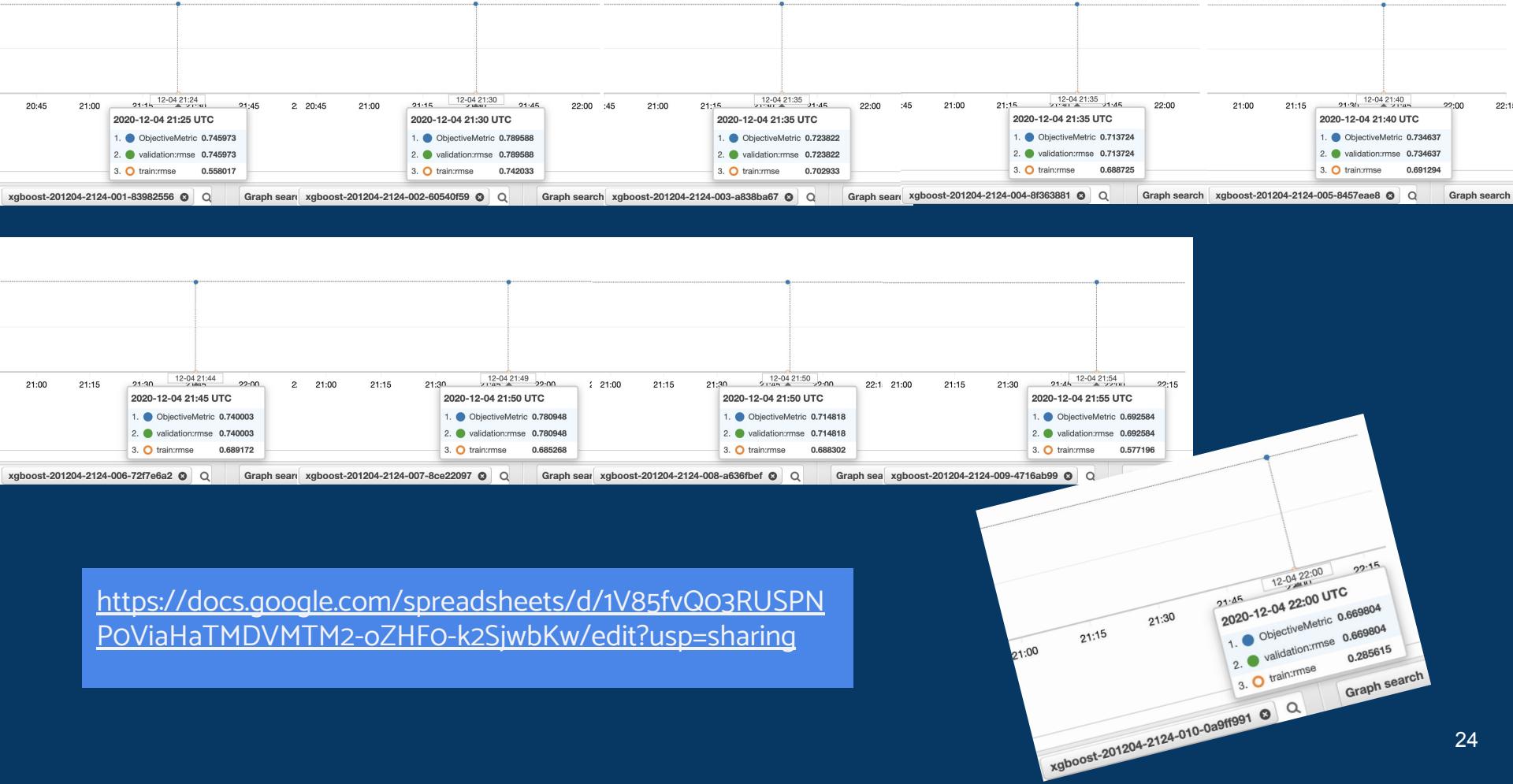
S3 output path: s3://sagemaker-us-east-1-029754830265/sagemaker/xgboost-whitewinequality/output  
Output encryption key: -

Hyperparameters

Key	Value
_tuning_objective_metric	validation:rmse
alpha	0.09892440128424873
booster	gbtree
early_stopping_rounds	10
eta	0.7077653378927984
gamma	4
max_depth	5
min_child_weight	7.105857974946885
num_round	100
objective	reg:linear
scale_pos_weight	8
subsample	0.7
verbose	1



Completed training jobs started to produce some good results back from hyperparameter tuning



# Endpoint creation

```
xgb_reg_whitewinequality.i
Code git
[1]: tuner.fit({'train': s3_input_train, 'validation': s3_input_validation})
.....
[21]: sagemaker = boto3.Session().client(service_name='sagemaker')

# Get tuning job name
job_name = tuner.latest_tuning_job.job_name
print(job_name)

sagemaker.describe_hyper_parameter_tuning_job(
    HyperParameterTuningJobName=job_name)[['HyperParameterTuningJobStatus']]
xgboost-201204-2346
[21]: 'Completed'

[22]: # Deploying the best model
tuning_job_result = sagemaker.describe_hyper_parameter_tuning_job(HyperParameterTuningJobName=job_name)
best_model_name = tuning_job_result['BestTrainingJob']['TrainingJobName']
print(best_model_name)

import time
timestamp = time.strftime('%Y-%m-%d-%H-%M-%S', time.gmtime())
endpoint_name = best_model_name + '-ep-' + timestamp
print(endpoint_name)
xgboost-201204-2346-001-2df21373
xgboost-201204-2346-001-2df21373-ep-2020-12-05-13-17-34

[*]: tuner.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge', endpoint_name=endpoint_name)

2020-12-04 23:50:08 Starting - Preparing the instances for training
2020-12-04 23:50:08 Downloading - Downloading input data
2020-12-04 23:50:08 Training - Training image download completed. Training in progress.
2020-12-04 23:50:08 Uploading - Uploading generated training model
2020-12-04 23:50:08 Completed - Training job completed
-----
```

# Testing 10 samples

```
[29]: #Predicting the first 10 samples with the best model
smrt = boto3.client('sagemaker-runtime')

# Predict samples from the validation set
payload = validation_data[:10].drop(['quality'], axis=1)
payload = payload.to_csv(header=False, index=False).rstrip()

print(payload)
6.4,0.22,0.56,14.5,0.055,27.0,159.0,0.998,2.98,0.4,9.1
6.4,0.23,0.33,1.15,0.044000000000000004,15.5,217.5,0.992,3.33,0.44,11.0
7.8,0.13,0.3,1.8,0.04,43.0,179.0,0.9955,3.43,0.41,9.0
5.8,0.17,0.34,1.8,0.045,96.0,170.0,0.99035,3.38,0.9,11.8
7.3,0.17,0.23,6.3,0.051,35.0,240.0,0.9963,3.36,0.54,10.0
6.3,0.28,0.24,8.45,0.031,32.0,172.0,0.9958,3.39,0.57,9.7
7.3,0.21,0.49,1.8,0.038,44.0,152.0,0.9912,3.32,0.44,12.6
7.0,0.15,0.28,14.7,0.051,29.0,149.0,0.99792,2.96,0.39,9.0
7.5,0.14,0.34,1.3,0.055,50.0,153.0,0.9945,3.29,0.8,9.6
5.8,0.25,0.24,13.3,0.04400000000000004,41.0,137.0,0.9972,3.34,0.42,9.5

[31]: response = smrt.invoke_endpoint(
    EndpointName=endpoint_name,
    Body=payload.encode('utf8'),
    ContentType='text/csv')

print(response['Body'].read())
b'5.73413085938,5.72361278534,6.14390325546,7.92762136459,5.81308412552,5.9978966713,7.02605772018,6.8381
0567856,5.96017646795,5.40178489685'

[ ]: #And last but not least, let's not run unnecessary costs in the Cloud; so here's a handy script to delete
sagemaker.delete_endpoint(EndpointName=endpoint_name)

[ ]:
```

# Publishing the Model

# Publishing the Model

Publish the model on Github or AI Club Projects / Share with the class:

[https://github.com/itsencrypted/xgboost\\_regression](https://github.com/itsencrypted/xgboost_regression)

Notebook details and execution:

[https://drive.google.com/file/d/1hPCsFk9t\\_Gbu6K1XbrVIpThUxoG7G5wh/view?usp=sharing](https://drive.google.com/file/d/1hPCsFk9t_Gbu6K1XbrVIpThUxoG7G5wh/view?usp=sharing)

Spreadsheet with hyperparameters/Training Models metrics:

<https://docs.google.com/spreadsheets/d/1V85fvQo3RUSPNPoViaHaTMDVMTM2-oZHFo-k2SjwbKw/edit?usp=sharing>

Publish code parts on Github and link to projects in AI Club

Write a Medium blog describing your projects with links to AI Club and Github

# Special Thanks

My immense gratitude to the help and assistance of Sindhu Ghanta; our teacher at the AI Camp Full Stack AWS Sagemaker course for the knowledge shared and support with feature engineering.

Special thanks to Yamuna Dulanjani for the help with important details of this project such as container image configs and instance size and overall assistance.

For all the material digged under: <https://github.com/aws/amazon-sagemaker-examples> and the new stable version of SageMaker Python SDK: <https://sagemaker.readthedocs.io/en/stable/> and Julien Simon's Gitlab repo: <https://gitlab.com/juliensimon/dlnotebooks>

Without this support this project wouldn't be completed on time.

And a shoutout to all my colleagues and people interested in AI Camp courses: <https://learn.xnextcon.com/> there are some great hands on courses there if you're interested in machine learning frameworks. I highly recommend!

And if you want to see more projects like this; and engage in Full Stack ML with AWS please visit:  
<https://aiclub.world/workshops>      <https://aiclub.world/projects>      <https://aiclub.world/news>