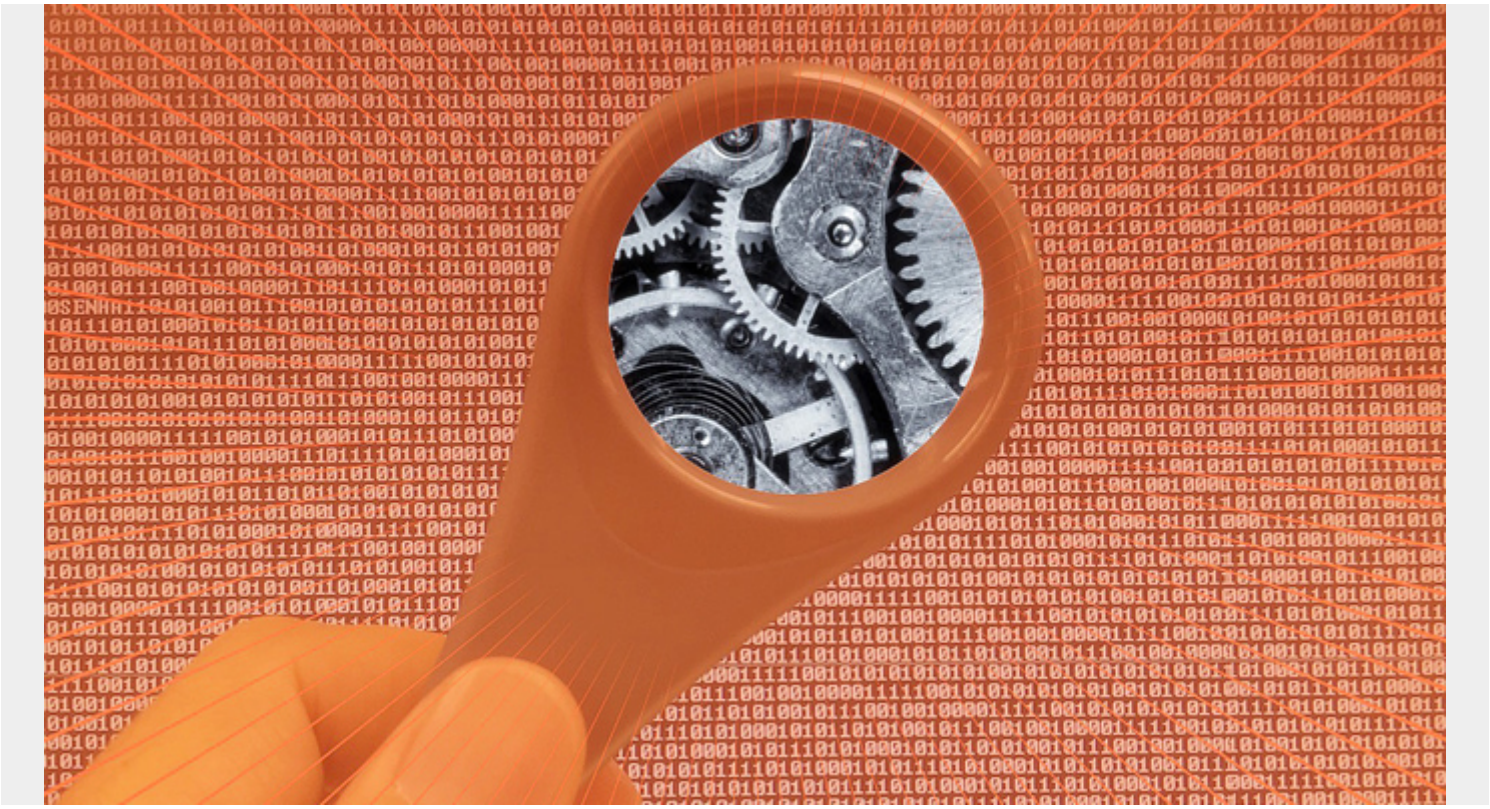
AWS LINEAR LEARNER: USING AMAZON SAGEMAKER FOR LOGISTIC REGRESSION



In the last blog post we showed you how to use Amazon SageMaker. So read that one before you read this one because there we show screen prints and explain how to use the graphical interface of the product, including its hosted Jupyter Notebooks feature. We also introduced the SageMaker API, which is a front end for Google TensorFlow and other opensource machine learning APIs. Here we focus more on the code than how to use the SageMaker interface.

In the last example we used k-means clustering. Here we will do **logistic regression**. Amazon calls their linear regression and logistic regression algorithms **Linear Learner**. The complete code for this blog post example is here.

We take the simplest possible example using data from Wikipedia. This is much easier than the examples provided by Amazon which use very large datasets and are geared toward handwriting recognition, etc. Most business problems are not handwriting recognition, but more everyday tasks, like preventive maintenance.

Here we only have 20 data records. That is too small to split the data into train, test, and validation data sets. So we will use the training data only and skip the validation step. Of course in a real world scenario you would want to validate how accurate your model is.

The data below shows what is the likelihood that a student will pass a certain test given how many hours they study.

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

Normally you would read the data from a .csv file. But there are so few records we can put this right into the code. So create a **condaPython3** notebook in SageMaker and paste in the following code.

Below we take the grades and pass-fail and make a tuple:

```
study=((0.5,0),(0.75,0),(1.0,0),(1.25,0),(1.50,0),(1.75,0),(2.0,0),(2.25,1),(
2.5,0),(2.75,1),(3.0,0),(3.25,1),(3.5,0),(4.0,1),(4.25,1),(4.5,1),(4.75,1),(5
.0,1),(5.5,1))
```

Then we convert this to a numpy array. It has to be of type **float32**, as that is what the SageMaker Linear Learner algorithm expects.

We then take a slice and put the labels (i.e., pass-fail) into the field **labels**. The Linear Learner algorithms expects a features matrix and labels vector.

```
import numpy as np
a = np.array(study).astype('float32')

labels = a
```

In the last example we used the **record_set()** method to upload the data to S3. Here we use the algorithms provided by Amazon to upload the training model and the output data set to S3.

Create a bucket in S3 that begins with the letters **sagemaker**. Then Amazon will create the subfolders, which in needs, which in this case are **sagemaker/grades** and others. **It is important** that you create the S3 buckets in the same Amazon region as your notebook. Otherwise Amazon will throw an error saying it cannot find your data. See the note below on that.

Copy this text into a notebook cell and then run it.

```
import boto3

sess = sagemaker.Session()
bucket = "sagemakerwalkerlr"
prefix = "sagemaker/grades"

buf = io.BytesIO()
smac.write_numpy_to_dense_tensor(buf, a, labels)
buf.seek(0)

key = 'linearlearner'
boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train',
key)).upload_fileobj(buf)
s3_train_data = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
print('uploaded training data location: {}'.format(s3_train_data))

output_location = 's3://{}/{}/output'.format(bucket, prefix)
print('training artifacts will be uploaded to: {}'.format(output_location)
```

Amazon will respond:

```
uploaded training data location:
s3://sagemakerwalkerml/sagemaker/grades/train/linearlearner
training artifacts will be uploaded to:
s3://sagemakerwalkerml/sagemaker/grades/output
```

Below we copy the code from Amazon that tells it which Docker container to use and which version of the algorithm. Here we use version **latest**. Below I put Amazon zone **us-east-1** because this is where I created my notebook. You can look at other examples of Amazon code to get the name of the 4 containers and which Docker containers to use.)

```
 containers = {
              'us-east-1': '382416733822.dkr.ecr.us-
east-1.amazonaws.com/linear-learner:latest'
              }
```

You can check to see from which region Amazon will pull the Docker image by putting this line into the notebook and look at the output. So your Amazon S3 buckets should be there.

```
containers
```

Here is the output for my notebook.

```
'382416733822.dkr.ecr.us-east-1.amazonaws.com/linear-learner:latest'
```

Now we begin to set up the **Estimator**. Amazon will not let you use any of their smaller (i.e. less expensive) images, so here we use a virtual machine of size **ml.p2.xlarge**.

```
linear = sagemaker.estimator.Estimator(containers,
                                       role=role,
                                       train_instance_count=1,
                                       train_instance_type='ml.p2.xlarge',
                                       output_path=output_location,
                                       sagemaker_session=sess)
```

Now we provide **hyperparameters**. There are many, like which loss function to use. Here we put only the most important ones:

feature_dim—is the number of columns in our feature array. In this case it is 2: hours of study and pass-fail.
mini_batch_size—is the number of batches into which to split the data. This number should be smaller than the number of records in our training set. We only have 20 records, so 4 will work.
predictor_type—we use **binary_classifier**, which means logistic regression.

When you run the **fit()** method Amazon will kick off this job. This will take several minutes to run.

```
%%time
linear.set_hyperparameters(feature_dim=2,
                           mini_batch_size=4,
                           predictor_type='binary_classifier')
```

```
linear.fit({'train': s3_train_data})
```

Amazon responds like this. Wait several minutes for the job to complete.

```
INFO:sagemaker:Creating training-job with name: linear-
learner-2018-04-07-14-33-25-761

Docker entrypoint called with argument(s): train
…

===== Job Complete =====
Billable seconds: 173
CPU times: user 344 ms, sys: 32 ms, total: 376 ms
Wall time: 6min 8s
```

When the training model is done, deploy it to an **endpoint**. Remember that Amazon is charging you money now. So when you get done delete your endpoints unless you want to be charged.

```
linear_predictor = linear.deploy(initial_instance_count=1,
                                 instance_type='ml.p2.xlarge')
```

Amazon responds:

```
INFO:sagemaker:Creating model with name: linear-
learner-2018-04-07-14-40-41-204
INFO:sagemaker:Creating endpoint with name linear-
learner-2018-04-07-14-33-25-761
```

Now copy this code. We will put just 1 record **a** into the linear_predictor. The value is 0.5 hours, so obviously we expect this student to fail.

```
from sagemaker.predictor import csv_serializer, json_deserializer

linear_predictor.content_type = 'text/csv'
linear_predictor.serializer = csv_serializer
linear_predictor.deserializer = json_deserializer

a

array(, dtype=float32)
```

Now we run the prediction.

```
result = linear_predictor.predict(train_set)
print(result)
```

Amazon shows us what we would expect, which is that the student is most likely to fail having studied only ½ an hour.

```
{'predictions': }
```