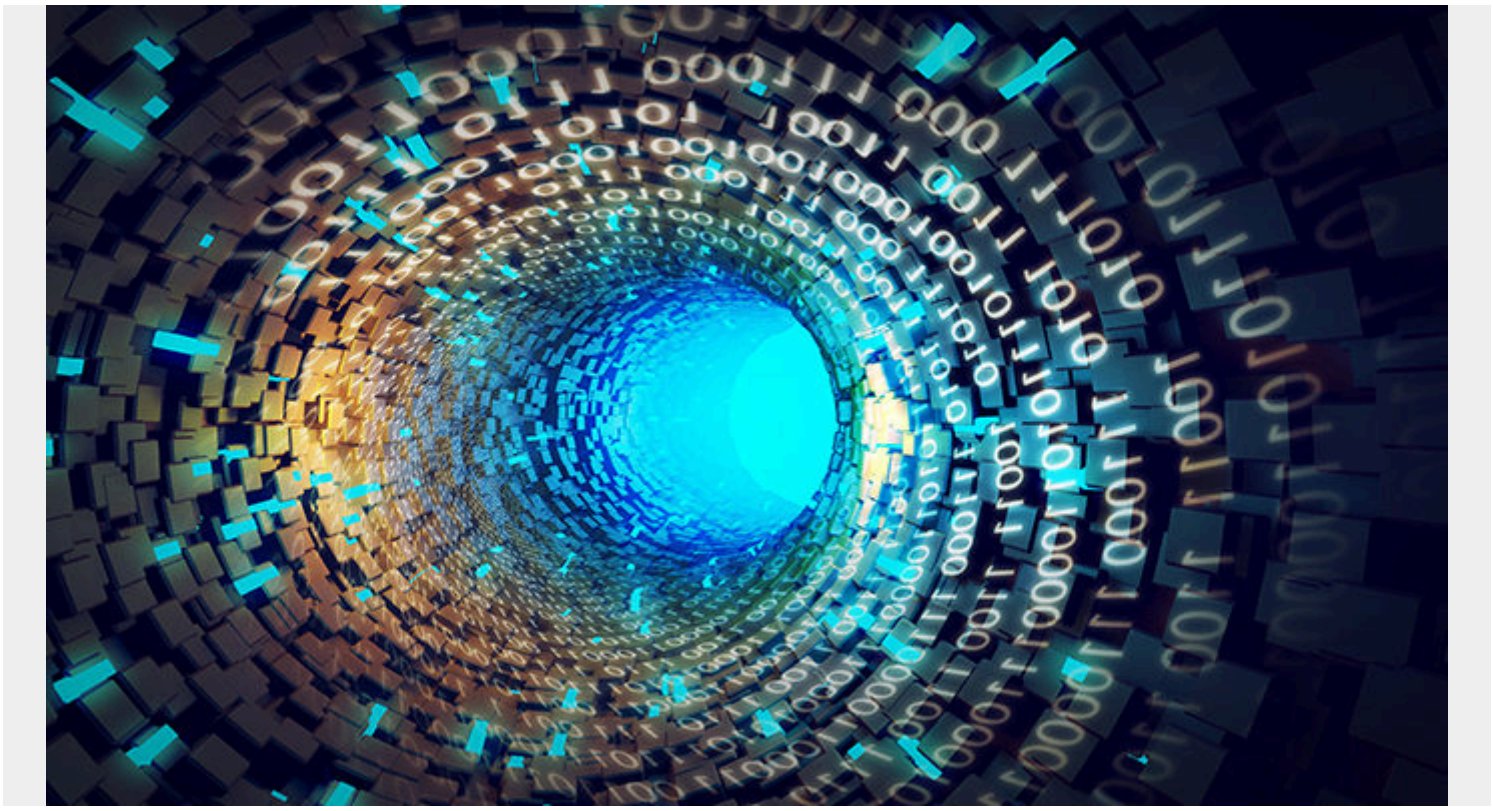
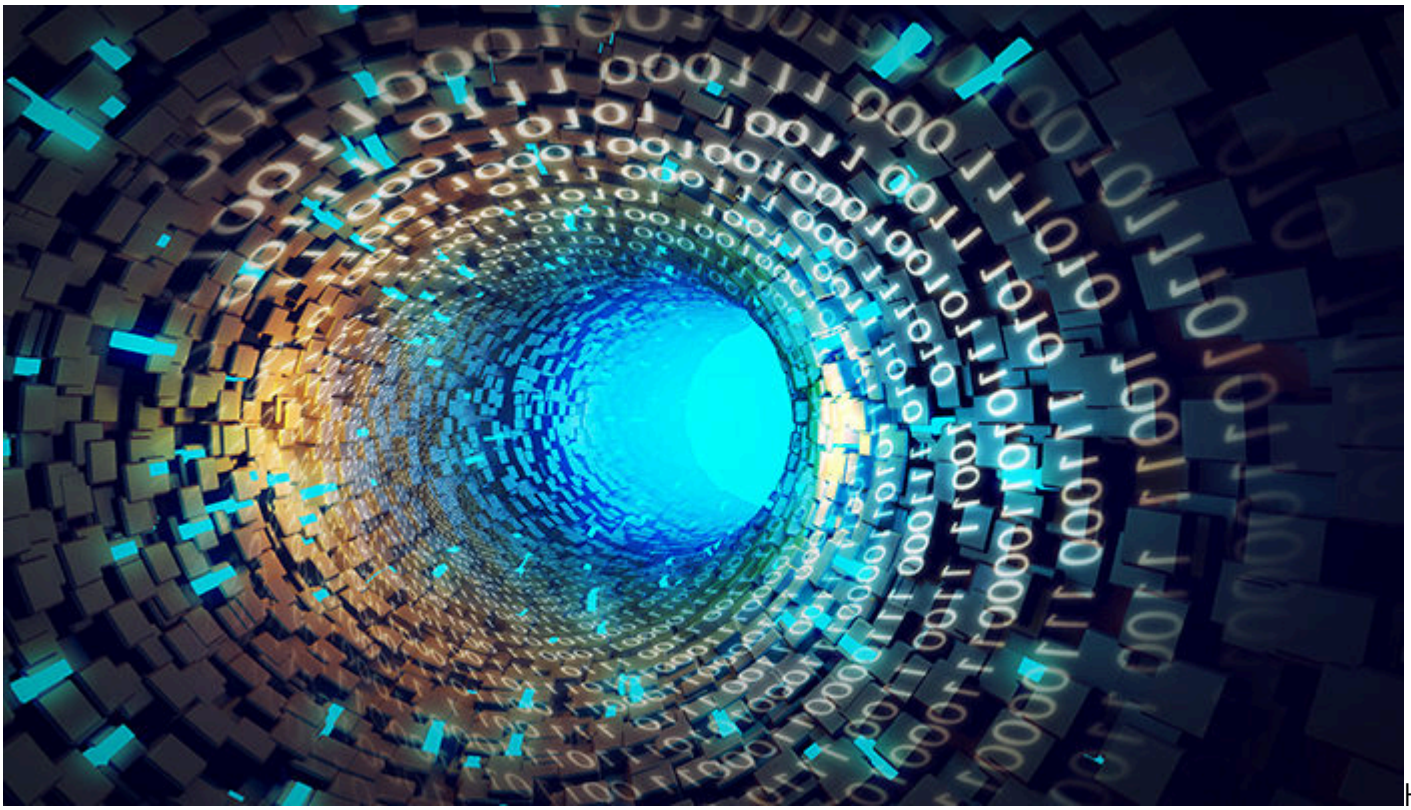
INTRO TO AMAZON MACHINE LEARNING WITH LOGISTIC REGRESSION





Here

we look at Amazon's Machine Learning cloud service. In this first article we will look at logistic regression. In future blog posts we will see what other algorithms it offers.

Remember that **logistic regression** is similar to **linear regression**. It looks at a series of **independent variables** and calculates one **dependant variable**. If the probability of that outcome is > 50%, the that is classified as a 1 (true). Otherwise it is false (0). (Amazon lets you change that threshold, which is a little strange, as 50% is the standard value used by statisticians. But you could fiddle around with that nevertheless, such as when, for example, 30% means **true** in your situation.)

Here is related reading if you are just getting started:

- Using Logistic Regression, Scala, and Spark
- Introduction to TensorFlow and Logistic Regression
- SGD Linear Regression Example with Apache Spark
- Machine Learning and AI Frameworks: What's the Difference and How to Choose?
- Amazon Machine Learning and Analytics Tools

Explanation of the Process

The idea behind Amazon ML is that you can run predictive models with without any programming. That is true for logistic regression. But you still need to put your data into a .csv format. Then you upload it to Amazon S3, which is their file storage system.

Here we run logistic regression using the sample banking.csv data set provided by Amazon. The goal is to predict whether a customer is likely to buy the banking service given the attributes shown below:

```
{
  "version" : "1.0",
  "rowId" : null,
  "rowWeight" : null,
  "targetAttributeName" : "y",
  "dataFormat" : "CSV",
  "dataFileContainsHeader" : true,
  "attributes" : ,
  "excludedAttributeNames" :
}
```

When you load this data set into ML, Amazon walks you through each field. It looks at each and determines whether they are **numeric** (could be any number), **categorical** (a specific set of numbers or text values), or **binary** (y or n or 1 or 0). The binary answers the question of whether this customer has pushed the banking product. That is the value we want to predict.

To use this, you need to do is to put your data into a spreadsheet format, with the first row as column headers. Unlike writing code yourself, where you have to convert all values to number, the algorithm here lets you use text or numeric values. Amazon will then take a guess as to which is the dependant variable and ask you to confirm that.

Then Amazon does what any ML programmer would do. It splits the input data set into a **training** data set and a **test** data set. It uses a 70/30 split, meaning 70% for one data set and 30% for another. Then it **evaluates** the model, meaning shows how accurately the independent variables predict the

dependant ones.

It could be that there is not much relationship at all between these variables. That would mean your assumption that this data is correlated is wrong. Of course, Amazon picked this banking data because it is correlated.

Having done the model correlation and evaluation, you can now use the trained model to run a **prediction**. In other words you go get some new data and run your prediction on whether this batch of persons might buy your banking product. Here Amazon charges you. They charged me $2.90 to do this.

Getting Started

Now we show how to use the service.

First you sign into the service by clicking on the Amazon AWS Console and click on **Machine Learning** to add that service to your account. Note that this service is not free. So set up a billing alert so that you do not get changed more than you have budgeted for.

AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).

Recently visited services

- S3
- Machine Learning
- Billing

All services

Compute
- EC2
- Lightsail
- Elastic Container Service
- Lambda
- Batch
- Elastic Beanstalk

Storage
- S3
- EFS
- Glacier
- Storage Gateway

Database
- Relational Database Service
- DynamoDB
- ElastiCache
- Amazon Redshift

Migration
- AWS Migration Hub
- Application Discovery Service
- Database Migration Service
- Server Migration Service
- Snowball

Networking & Content Delivery
- VPC
- CloudFront
- Route 53
- API Gateway
- Direct Connect

Developer Tools
- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- Cloud9
- X-Ray

Management Tools
- CloudWatch
- AWS Auto Scaling
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Systems Manager
- Trusted Advisor
- Managed Services

Media Services
- Elastic Transcoder
- Kinesis Video Streams
- MediaConvert
- MediaLive
- MediaPackage
- MediaStore
- MediaTailor

Machine Learning
- Amazon SageMaker
- Amazon Comprehend
- AWS DeepLens
- Amazon Lex
- Machine Learning
- Amazon Polly
- Rekognition
- Amazon Transcribe
- Amazon Translate

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- QuickSight
- Data Pipeline
- AWS Glue

Security, Identity & Compliance

Mobile Services
- Mobile Hub
- AWS AppSync
- Device Farm
- Mobile Analytics

AR & VR
- Amazon Sumerian

Application Integration
- Step Functions
- Amazon MQ
- Simple Notification Service
- Simple Queue Service
- SWF

Customer Engagement
- Amazon Connect
- Pinpoint
- Simple Email Service

Business Productivity
- Alexa for Business
- Amazon Chime
- WorkDocs
- WorkMail

Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

Internet of Things
- AWS IoT
- IoT Analytics
- IoT Device Management
- Amazon FreeRTOS
- AWS Greengrass

Game Development
- Amazon GameLift

Helpful tips

Manage your
Get real-time billi
usage budgets. S

Create an orga
Use AWS Organiz
management of
now

Explore AWS

Amazon Relational Da
RDS manages and scales y
supports Aurora, MySQL, Po
SQL Server. Learn more.

Real-Time Analytics w
Stream and analyze real-tim
insights and react quickly. L

Get Started with Conta
Amazon ECS helps you build
application. Learn more.

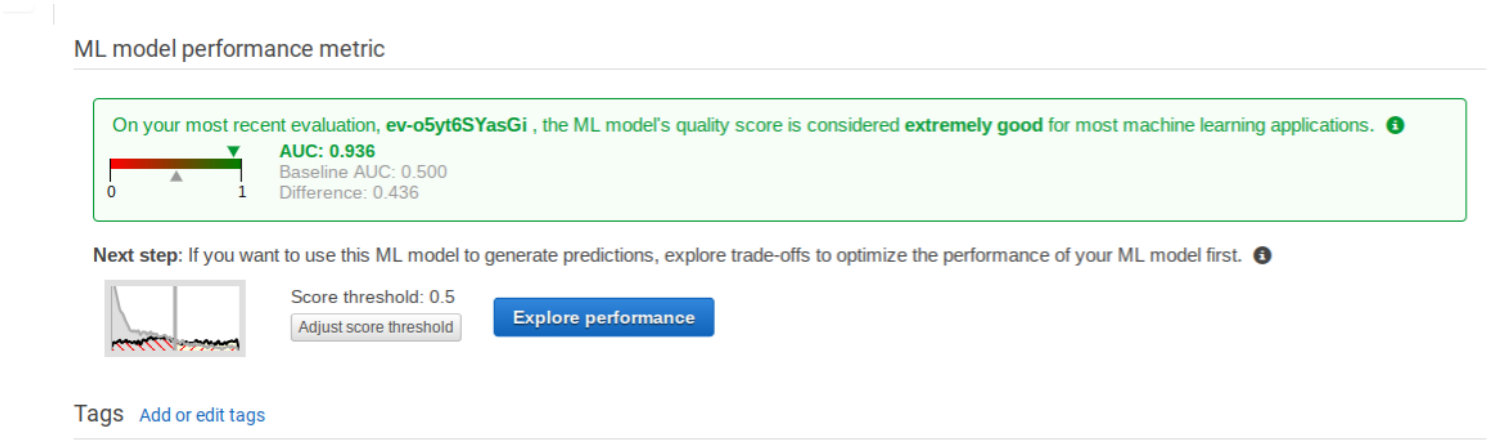
AWS Marketplace
Discover, procure, and deplo
run on AWS. Learn more.

Have feedback?
Submit feedback to tell us a
AWS Management Console.

Building the Model

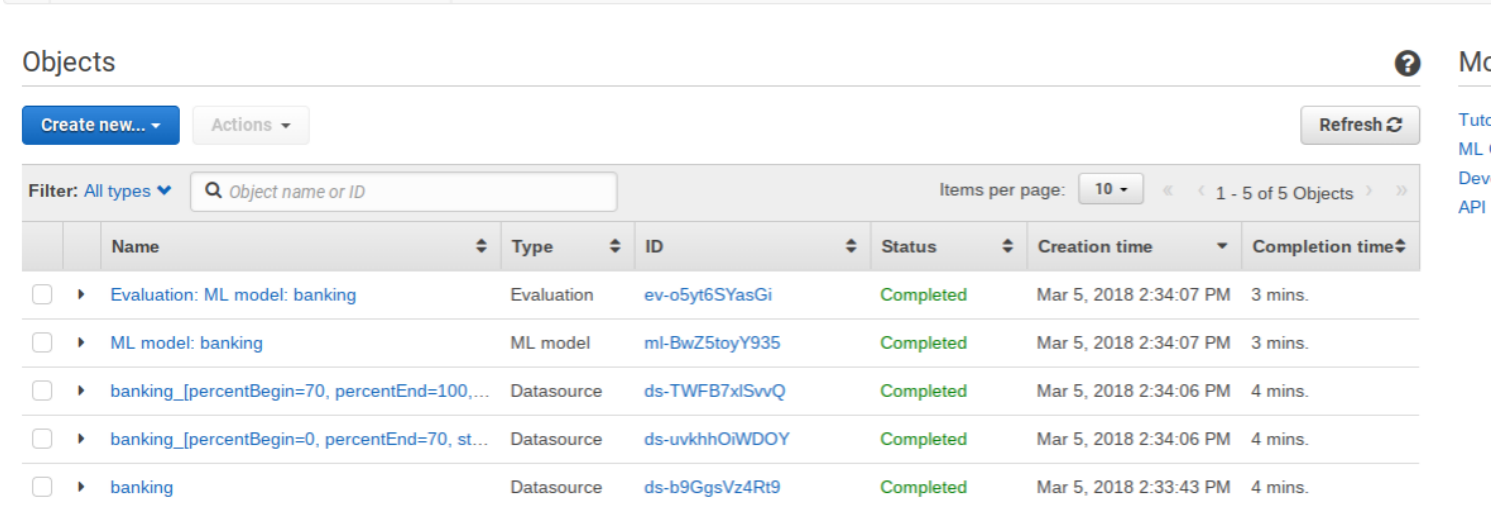
Here are the steps to build and use the model. We do not go in any particular order. Do not worry as Amazon has wizards to guide you through the process.

You can see how accurate the model is by the AUC (area under the curve). Don't worry about the exact definition. Unless you are a mathematician or statistician you will not understand it. Just understand that it is the difference between the observed values and predicted values. If they value was 1 then he model is perfect. 0.936 is a very high level of correlation. Anything below 0.5 is deemed to indicate that the data is not sufficiently correlated. In other words, that would mean your assumption of whether a customer might buy this banking product has nothing to do with those input values.

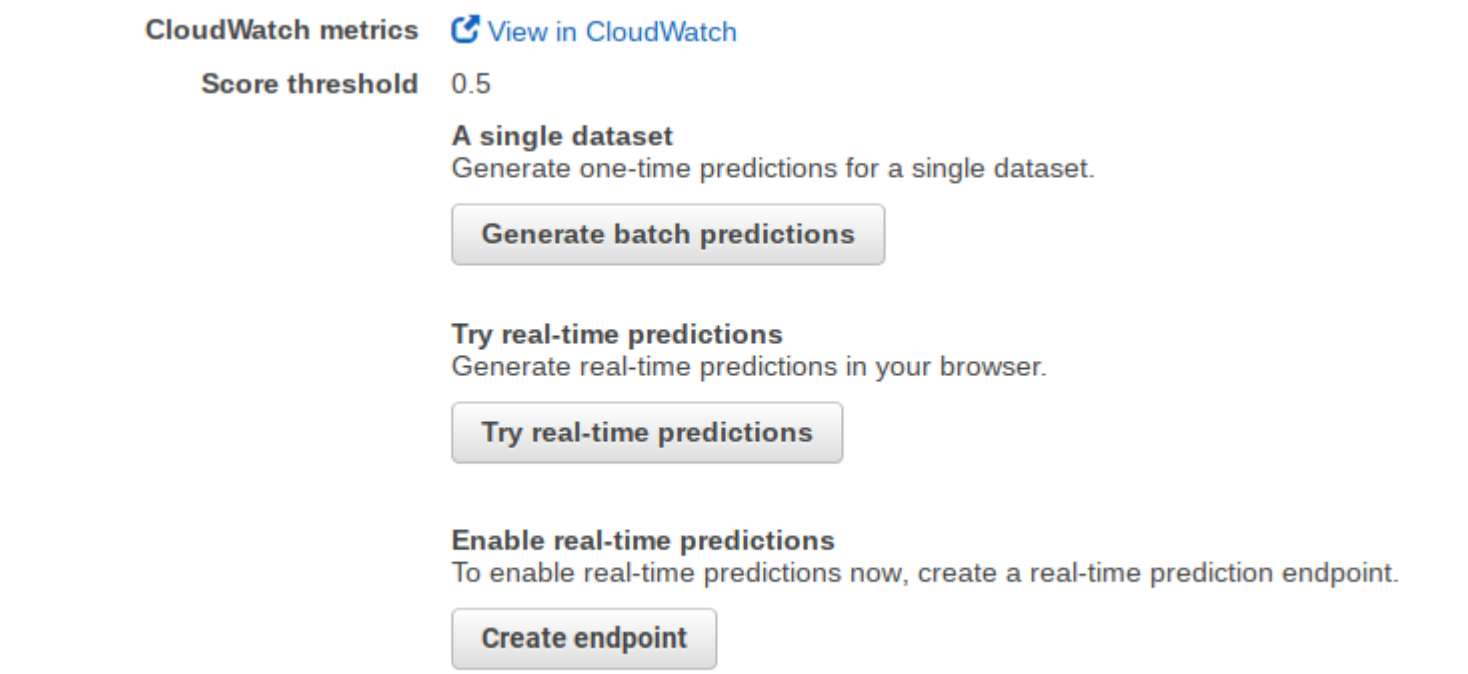


The ML Dashboard

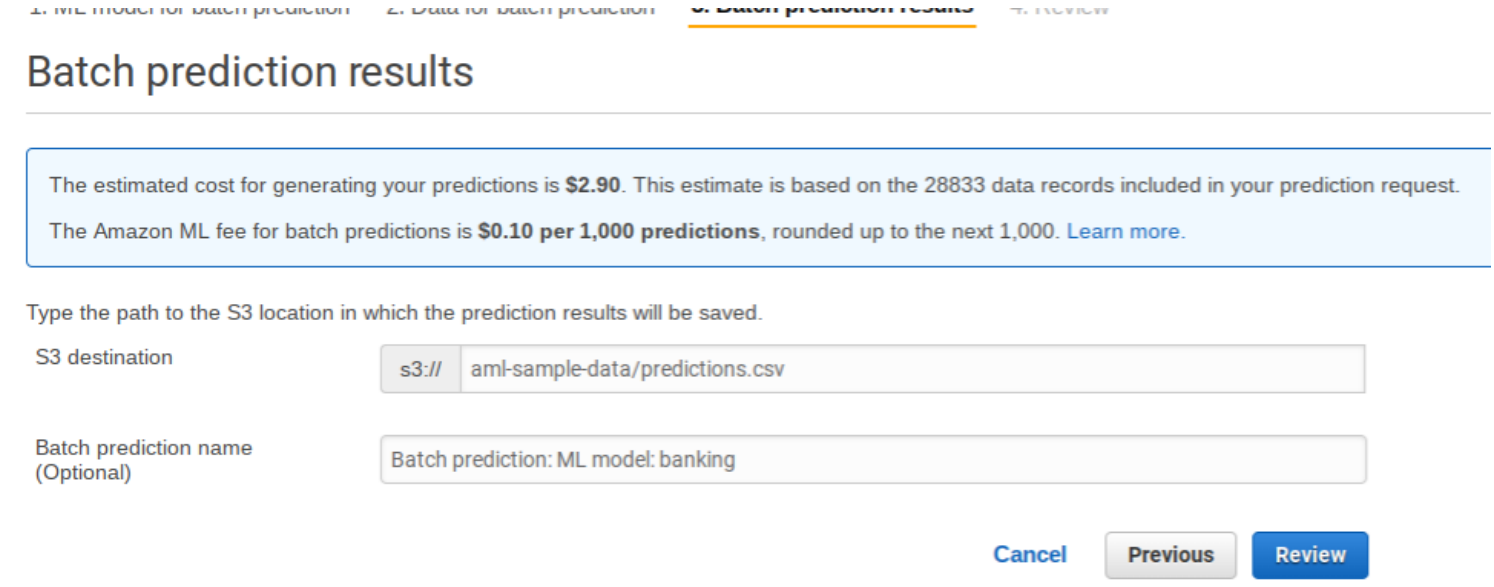
Below is my dashboard showing what I have run. It's all the same model, but each time I used different datasets. One is prediction and the others Amazon generated automatically when it did the training and evaluation steps.

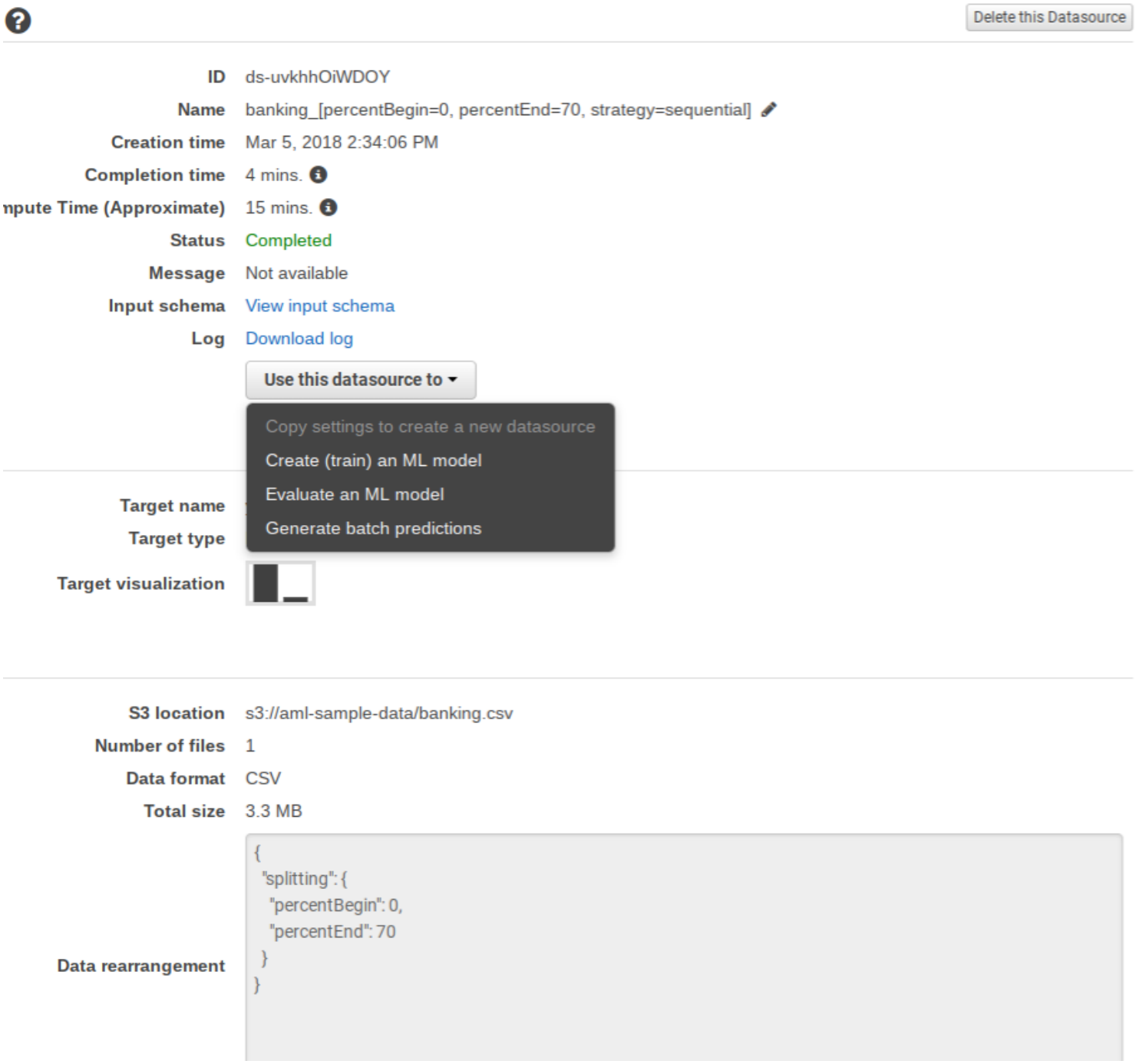


Here is the screen to kick off the prediction step. Most people would do **Generate Batch Predictions.** That runs the model against data you have loaded into S3. **Real-Time Predictions** lets you type one record into a screen and it will run a prediction against that.



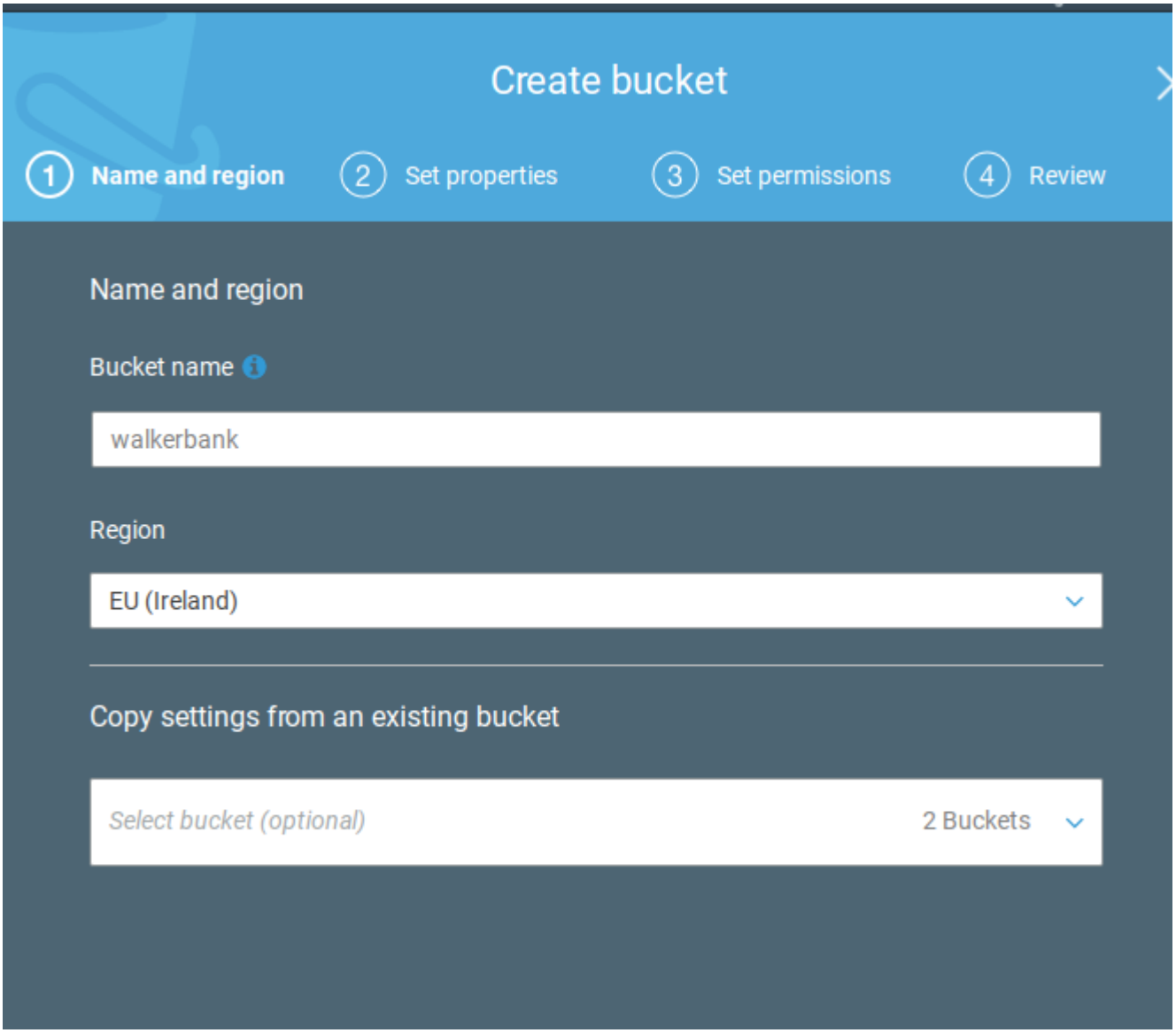
Here are the prediction results. As you can see it charged me $2.90, which is $0.10 per 1,000 predictions. It saves the results in S3, which we show below.





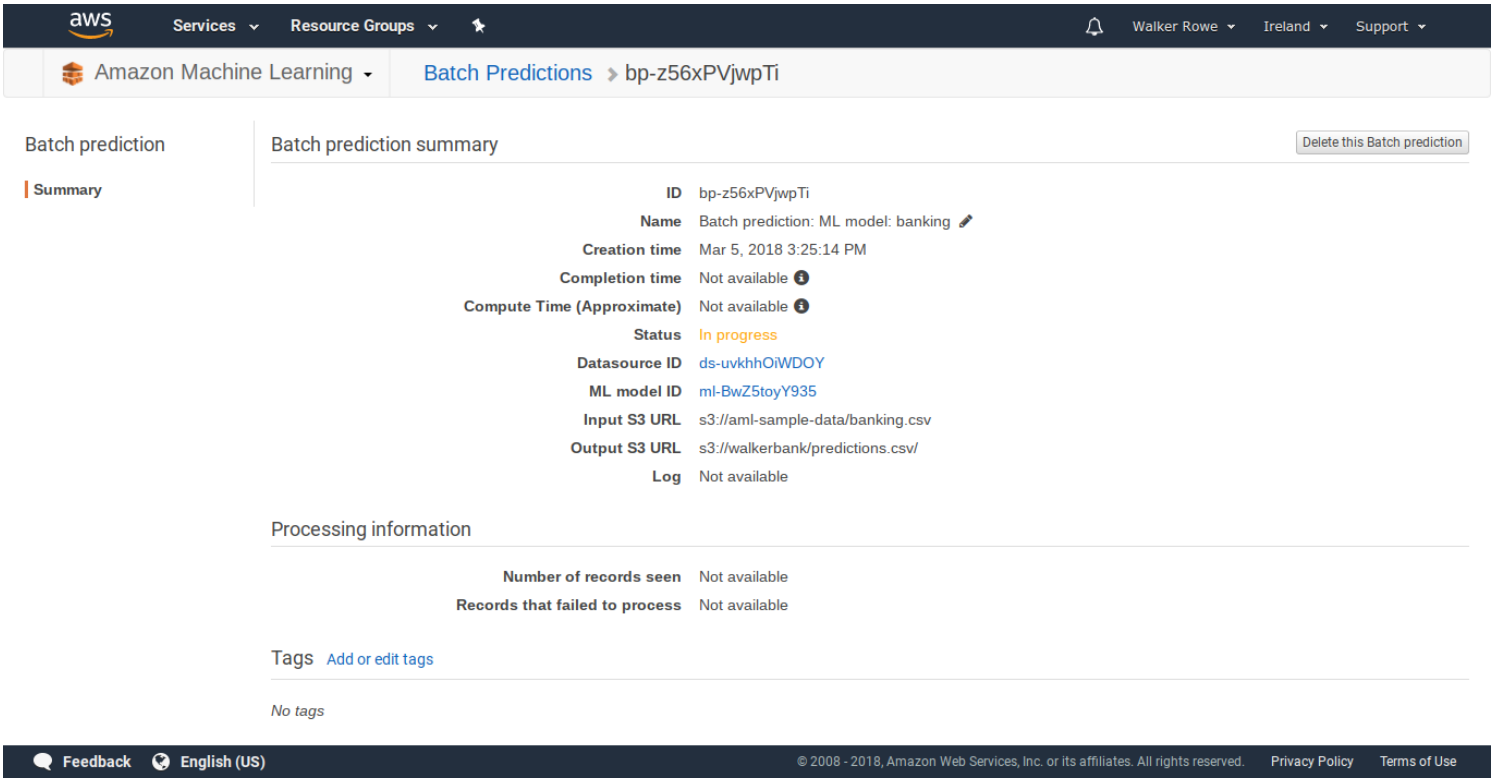
Load Data in S3

Amazon's banking data is already at a URL where you can use it. In other to use Amazon's data to run a prediction against it, which in real life you would do by gathering more data about your customers, you need to create a bucket in S3. That is like a folder. Below I create the bucket **walkerbank**.



Wait and Wait some More

It will take some time for your model to run as it gets in a queue behind other customers. Below you can see that this one is in a **pending** state.



Get the Results

Amazon saves the results in S3. You cannot really browse the results online. Instead you can download the file, unzip it, and then look at it. That is what I have done here.

Here is what Amazon has calculated. Too bad it put the results in a new file instead of appending the prediction as a new column in the input file. Below what we see is the actual value (**trueLabel**) from the input data and the predicted value (**bestAnswer**) based upon the model that Amazon built.

```
trueLabel,bestAnswer,score
0,0,1.437033E-2
0,0,1.139906E-2
1,1,8.305257E-1
0,0,8.966137E-2
1,0,4.096018E-1
0,0,3.634616E-3
0,0,2.641097E-2
0,0,3.487612E-2
1,1,5.777377E-1
0,0,4.469287E-2
0,0,2.456573E-3
0,0,4.300581E-1
1,0,8.399929E-2
0,0,1.024602E-2
```

Next Steps

In the next blog post we will see whether Amazon can do k-mean clustering, linear regression, or other types of analysis.