Task 3: Advanced (Peter).
Machine Learning

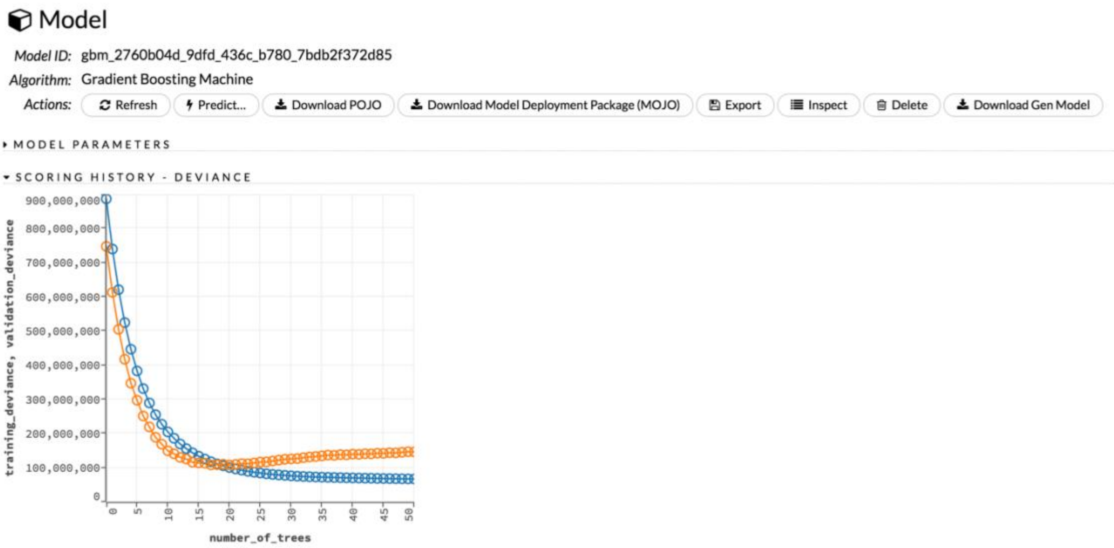
Looking at a use case
To illustrate this article, let’s take one of the most common use cases of Machine Learning: estimating prices of real estate. You have a dataset of past observations, with the characteristics and the selling price of some houses:

Row	sqft	lot_size_acres	sale_price
1	2231.0	0.3900	251000.0
2	3112.0	0.4400	321000.0
3	2121.0	0.4000	245000.0
4	2532.0	0.5000	294000.0
5	2632.0	0.4500	285400.0
6	2743.0	0.3700	299000.0
7	2823.0	0.3900	308000.0
8	2918.0	0.4900	321000.0
9	2021.0	0.3700	232000.0

You can build a regression model so that, when there is a new house to sell, you can estimate what the selling price will be.

The training process should be done in batch, from time to time, with a fixed dataset, and will produce a model.
A streaming application can use the model (without updating it), and we may need to update the application when a new model is produced.

Creating a ML model with H2O



Go ahead and click on “Download POJO” to get a Java file that looks like this:

```
@ModelPojo(name="gbm_2760b04d_9dfd_436c_b780_7bdb2f372d85", algorithm="gbm")
public class gbm_2760b04d_9dfd_436c_b780_7bdb2f372d85 extends GenModel {
  public hex.ModelCategory getModelCategory() { return hex.ModelCategory.Regression; }

  public boolean isSupervised() { return true; }
  public int nfeatures() { return 10; }
  public int nclasses() { return 1; }

  // Names of columns used by model.
  public static final String[] NAMES = NamesHolder_gbm_2760b04d_9dfd_436c_b780_7bdb2f372d
```

Using the model in a Java application

We are going to use H2O’s Gen Model library to interact with the model, so let’s add a dependency in our Gradle build:

```
dependencies {
    compile 'ai.h2o:h2o-genmodel:3.20.0.6'
```

```
    // other dependencies...
}
```

You must now load the model, prepare a record, then make a prediction:

```
fun main(args: Array<String>) {
    val rawModel = gbm_2760b04d_9dfd_436c_b780_7bdb2f372d85()
    /** TO DO Model **/
    val row = RowData().apply {
        put("sqft", 2342.0)
        put("lot_size_acres", 0.4)
        put("stories", 1.0)
        put("number_bedrooms", 3.0)
        put("number_bathrooms", 3.0)
        put("attached_garage", "yes")
        put("has_pool", "no")
        put("has_kitchen_island", "yes")
        put("main_flooring_type", "hardwood")
        put("has_granite_counters", "yes")
        put("house_age_years", 4.0)
    }
    /** TO DO prediction **/
    println("Prediction: \$${prediction.value}")
}
```

If we run this code, we get something like:

```
Prediction: $290591.70578645257
```

Great, we just used our Machine Learning model in a standalone application!

Using the model in a Kafka Streams application

I put together a basic producer that pushes *unlabeled records* to a Kafka topic called `housing`. The data is in JSON:

```
$ kafka-console-consumer --bootstrap-server localhost:9092 --topic housing

{"sqft":2572,"lot_size_acres":0.2318716,"stories":2,"number_bedrooms":1,"number_bathrooms":0,"attached_garage":false,"has_pool":true,"has_kitchen_island":true,"main_flooring_type":"hardwood","has_granite_counters":false,"house_age_years":18}

{"sqft":1256,"lot_size_acres":0.774486,"stories":2,"number_bedrooms":1,"number_bathrooms":2,"attached_garage":false,"has_pool":false,"has_kitchen_island":true,"main_flooring_type":"hardwood","has_granite_counters":false,"house_age_years":3}

{"sqft":2375,"lot_size_acres":0.7467226,"stories":2,"number_bedrooms":4,"number_bathrooms":0,"attached_garage":false,"has_pool":true,"has_kitchen_island":false,"main_flooring_type":"hardwood","has_granite_counters":true,"house_age_years":3}
...
```

You must build a [Kafka Streams](https://kafka.apache.org/documentation/streams/) application to process this stream and make predictions as new records arrive.

Now, if we look at the output topic, we can see the input data enhanced with predicted selling prices:

```
$ kafka-console-consumer --bootstrap-server localhost:9092 --topic predictions

{"sqft":2572,"lot_size_acres":0.2318716,...,"selling_price":297625}

{"sqft":1256,"lot_size_acres":0.774486,...,"selling_price":254118}
```

```
{"sqft":2375,"lot_size_acres":0.7467226,...,"selling_price":303408}
...
```

Conclusion

You must embed a Machine Learning model in a Kafka Streams application. The model was built outside of the streaming pipeline, and the generated POJO was completely independent from the platform where the model was trained.

Let us know if it helped!