# HOW TO USE APACHE IGNITE FOR MACHINE LEARNING

In this article, we show how to do machine learning with Apache Ignite.

## What is Apache Ignite?

Apache Ignite is an in-memory database that includes a machine learning framework. (If you wonder why it has an ML framework, consider that Apache Spark has one too, probably for the same reason.)

Ignite is written for Java programmers. Of course, that means you can use it with Scala, too, since that sits on top of Java. They have connectors for different languages, including:

- Python
- C++
- C#

## Why does the Ignite database have an ML frame?

Apache Ignite says they believe that the ETL (extract, transform, and load) should all take place on the same system. They point out that machine learning pipelines often involve more than one person and more than one system. So, the data scientist might be waiting around for the data engineers to provide data. It would be better to put it all onto one system, they say. That would reduce the number of times you convert the data from one format to another in order to machine

learning.

They also make the very valid point that certain other ML frameworks, like scikit-Learn, don't scale. That is a good point because it means your data could be too big to fit into the memory of one machine. For example, Python Pandas and NumPy data structures cannot be run on a cluster.

So, what do you do when your data is too big? In that case you would probably have to use something that does run on a cluster, like Spark ML or TensorFlow. That said, Ignite works with Spark, TensorFlow, Hadoop, Kafka, and other systems too.

## Simple example: Linear Regression

Here, we show a simple example of how to use Apache Ignite to do linear regression. The idea is to show how Ignite runs in-place—not so much to explain linear regression. What is quite remarkable is you don't even need to install Apache Ignite to run this example. And you don't need to start a server. The Ignite ML framework does that.

Instead, you just download the framework using a **Maven pom.xml** file. Then, when you write Java code to do the regression analysis, Ignite launches an instance of itself.

You can download the **pom.xml** here and code here.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Insight</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <ignite.version>2.8.0</ignite.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.apache.ignite</groupId>
            <artifactId>ignite-core</artifactId>
            <version>2.8.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.ignite</groupId>
            <artifactId>ignite-spring</artifactId>
            <version>2.8.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.ignite</groupId>
            <artifactId>ignite-ml</artifactId>
            <version>2.8.0</version>
        </dependency>
    </dependencies>


    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>


</project>
```

Here is the Java code. It's pretty simple, as the data is mocked up.

However, I can say that the Ignite documentation is not very thorough, which makes things less than simple. For documentation, they provide only Javadocs and sample code. I would welcome a much more detailed user guide. For example, it's not clear what the third argument in this does:

```
trainer.fit(ignite, dataCache, vectorizer);
```

Here is full the code. Below, I explain certain sections.

```
package com.bmc.ml;

import java.io.IOException;
import java.util.List;
import java.util.UUID;

import org.apache.ignite.Ignite;
import org.apache.ignite.IgniteCache;
import org.apache.ignite.Ignition;


import
org.apache.ignite.cache.affinity.rendezvous.RendezvousAffinityFunction;
import org.apache.ignite.configuration.CacheConfiguration;
import org.apache.ignite.configuration.IgniteConfiguration;
import org.apache.ignite.ml.dataset.feature.extractor.Vectorizer;
import org.apache.ignite.ml.dataset.feature.extractor.impl.DummyVectorizer;
import org.apache.ignite.ml.math.primitives.vector.Vector;
import org.apache.ignite.ml.math.primitives.vector.VectorUtils;
import org.apache.ignite.ml.regressions.linear.LinearRegressionLSQRTrainer;
import org.apache.ignite.ml.regressions.linear.LinearRegressionModel;
import org.apache.ignite.ml.selection.scoring.evaluator.Evaluator;
import org.apache.ignite.ml.selection.scoring.metric.MetricName;
import org.apache.ignite.ml.math.primitives.vector.Vector;
import org.apache.ignite.ml.math.primitives.vector.VectorUtils;

public class LRExample {

    public static void main(String[] args) throws IOException {
        System.out.println();
        System.out.println(">>> Linear regression model over cache based
dataset usage example started.");
        // Start ignite grid.


        IgniteConfiguration igniteCfg = new IgniteConfiguration();
        igniteCfg.setWorkDirectory("/Users/walkerrowe/Downloads");
        Ignite ignite = Ignition.start(igniteCfg);


        System.out.println(">>> Ignite grid started.");


        IgniteCache<Integer, Vector> dataCache = getCache(ignite);
```

```
        try {
           // dataCache = new SandboxMLCache(ignite).fillCacheWith();


            System.out.println(">>> Create new linear regression trainer
object.");
            LinearRegressionLSQRTrainer trainer = new
LinearRegressionLSQRTrainer();

            System.out.println(">>> Perform the training to get the model.");


            dataCache.put(1,VectorUtils.of(1,1.8));
            dataCache.put(2,VectorUtils.of(2,4.3));
            dataCache.put(3,VectorUtils.of(3,6.2));
            dataCache.put(4,VectorUtils.of(4,5));
            dataCache.put(5,VectorUtils.of( 5,11));
            dataCache.put(6,VectorUtils.of(6,11));
            dataCache.put(7,VectorUtils.of(7,15));



            Vectorizer<Integer, Vector, Integer, Double> vectorizer = new
DummyVectorizer()
                    .labeled(Vectorizer.LabelCoordinate.FIRST);


            LinearRegressionModel mdl = trainer.fit(ignite, dataCache,
vectorizer);

            double rmse = Evaluator.evaluate(
                    dataCache, mdl,
                    new
DummyVectorizer().labeled(Vectorizer.LabelCoordinate.FIRST),
                    MetricName.RMSE
            );

            System.out.println("rmse = " + rmse);

            System.out.println("intercept = " + mdl.getIntercept());


            System.out.println("Weights = " );

            Vector weights = mdl.getWeights();
            double[] w = weights.asArray();
            for (double v : w) {
```

```
                System.out.println(v);
            }

            System.out.println("==================");


        } finally {
            if (dataCache != null)
                dataCache.destroy();
        }
    }

    static private IgniteCache<Integer, Vector> getCache(Ignite ignite) {
        CacheConfiguration<Integer, Vector> cacheConfiguration = new
CacheConfiguration<>();
        cacheConfiguration.setName("ML_EXAMPLE_" + UUID.randomUUID());
        cacheConfiguration.setAffinity(new RendezvousAffinityFunction(false,
10));

        return ignite.createCache(cacheConfiguration);
    }

}
```

Here are the results:

```
rmse = 0.5963051208050183
intercept = 0.5762626813570405
Weights = 0.4413657685174977
```

When you start the code, it starts the server, as you can see:

```
    ___________  _________________
   /  _/ ___/ |/ /  _/_  __/ __/
  _/ // (7 7    // /  / / / _/
 /___/\___/_/|_/___/ /_/ /___/

 ver. 2.8.0#20200226-sha1:341b01df
 2020 Copyright(C) Apache Software Foundation

 Ignite documentation: http://ignite.apache.org

 Quiet mode.
   ^-- Logging by 'JavaLogger '
   ^-- To see **FULL** console log here add -DIGNITE_QUIET=false or "-v" to
```

```
ignite.{sh|bat}

 OS: Mac OS X 10.15.5 x86_64
```

# The code, explained

Give it a working directory and leave the constructor to the **IgniteConfiguration() constructor empty.**

```
IgniteConfiguration igniteCfg = new IgniteConfiguration();
igniteCfg.setWorkDirectory("/Users/walkerrowe/Downloads");
Ignite ignite = Ignition.start(igniteCfg);
```

This create the array and writes it to the Ignite database (which they call cache):

```
dataCache.put(1,VectorUtils.of(1,1.8));
```

Here you pass the **datacache** to the linear regression fit() method, to calculate the weights and coefficient.

```
LinearRegressionModel mdl = trainer.fit(ignite, dataCache, vectorizer);
```

# Additional resources

For more on this topic, check out our BMC Machine Learning & Big Data Blog or browse these articles:

- Apache Spark Guide, with 15+ articles
- Hadoop Guide, with 20+ articles
- Machine Learning with TensorFlow and Keras
- Enabling the Citizen Data Scientists