Mikael Marttila

Concept for Machine Learning with Java MooC Course by Oracle Academy for Metropolia UAS

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and communications technology

Bachelor's Thesis

27 July 2025

Abstract

Author:	Mikael Marttila
Title:	Concept for Machine Learning with Java MooC Course by Oracle Academy for Metropolia UAS
Number of Pages:	14 pages
Date:	27 July 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Information and Communication Technology
Supervisors:	Janne Salonen, Head of Department (ICT)

This thesis presents the findings of a pre-study with the objective of assessing whether Oracle Corporation's Java with AI course could be implemented as a MOOC for Metropolia University of Applied Sciences. The course is available through Oracle Academy and introduces students to artificial intelligence concepts through Java programming, with a later focus on manually implementing a decision tree.

In addition to evaluating the content of the course and its potential implementation into Moodle as a MOOC, this thesis also explores Java in general as a programming language. Its history, current relevance, and future are examined to assess its value in both education and the software industry.

This study also examines four common Java-based machine learning libraries: Weka, Deeplearning4j, SMILE and Tribuo. While the Oracle Academy course focuses on manual implementation for educational purposes, these libraries reflect the practical tools commonly used in the industry.

The study highlights the strengths and limitations of the course content and challenges encountered during the Moodle-based implementation. The goal was to provide support for the integration of the course into Metropolia UAS' curriculum, which would be highly beneficial for the institution.

Keywords: Java, programming, machine learning, artificial intelligence, AI, ML, MOOC, Moodle, Oracle

Tiivistelmä

Tekijä:	Mikael Marttila
Otsikko:	Concept for Machine Learning with Java MooC Course by Oracle Academy for Metropolia UAS
Sivumäärä:	14 sivua
Aika:	27.7.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Tieto- ja viestintätekniikka
Ohjaajat:	Janne Salonen, Osaamisaluejohtaja (ICT)

Tässä opinnäytetyössä esitellään esitutkimuksen tulokset, jonka tavoite oli arvioida, voitaisiinko Oracle-yhtiön Java with AI-kurssi toteuttaa MOOC-muodossa Metropolia Ammattikorkeakoululle. Kurssi on saatavilla Oracle Academy-alustan kautta ja se johdattaa opiskelijat tekoälyn perusteisiin Java-ohjelmoinnin avulla, keskittyen myöhemmin päätöspuun manuaaliseen toteuttamiseen.

Kurssin sisällön ja sen mahdollisen MOOC-toteutuksen arvioinnin lisäksi työssä tarkastellaan Java-ohjelmointikieltä yleisesti: sen historiaa, nykymerkitystä ja tulevaisuudennäkymiä, jotta voidaan arvioida sen arvoa niin opetuksessa kuin ohjelmistoalalla.

Tutkimuksessa käsitellään myös neljää yleistä Java-pohjaista koneoppikirjastoa: Weka, Deeplearning4j, SMILE ja Tribuo. Vaikka Oracle Academy-kurssi keskittyy manuaaliseen toteutukseen opetustarkoituksessa, nämä kirjastot edustavat käytännön työkaluja, joita alalla yleisesti käytetään.

Tutkimus tuo esiin kurssisisällön vahvuudet ja rajoitukset sekä haasteet, jotka ilmeni Moodle-toteutuksen aikana. Tavoitteena oli tukea kurssin integroimista osaksi Metropolian opetussuunnitelmaa, mikä olisi erittäin hyödyllistä oppilaitokselle.

Avainsanat: Java, ohjelmointi, koneoppiminen, tekoäly, AI, ML, MOOC, Moodle, Oracle

Contents

List of Abbreviations

ICT: *Information and Communication Technology*. Technologies that include information processing and communications, such as computers and the internet.

AI: *Artificial Intelligence*. Technology that enables computers and machines to simulate human learning, comprehension, problem solving and creativity.

ML: *Machine Learning*. A branch of artificial intelligence that focuses on enabling computers and machines to imitate human learning.

MOOC: *Massive Open Online Course*. An online course that can be enrolled on by students online, often used for self-paced studying. Managed by teachers.

HTTP: *Hypertext Transfer Protocol*. A foundational protocol for communication on the World Wide Web.

HTML: *Hypertext Markup Language*. The standard markup language for documents designed to be displayed in a web browser.

JVM: *Java Virtual Machine*. A virtual machine that enables a computer to run Java programs as well as programs written in other languages.

GUI: *Graphical User Interface*. Allows users to interact with electronic devices through a visual interface.

API: *Application Programming Interface*. A connection between computers or between computer programs.

I/O: *Input/Output*. Communication between a computer and the external world, e.g. keyboards, mice, monitors and networks.

SIMD: *Single Instruction, Multiple Data*. A type of parallel computing where a single instruction is used to operate on multiple data points simultaneously.

CPU: *Central Processing Unit*. The core component of a computer, which executes instructions and performs calculations.

GC: *Garbage Collection*. The automated process of deleting unused code from memory, freeing up space.

ONNX: *Open Neural Network Exchange*. An open-source framework designed to facilitate the exchange of neural network models among different deep learning frameworks.

CUDA: *Compute Unified Device Architecture*. NVIDIA's proprietary parallel computing platform and application programming interface.

GPU: *Graphics Processing Unit*. A specialized processor designed for digital image processing and to accelerate computer graphics.

BLAS: *Basic Linear Algebra Subprograms*. A specification that prescribes a set of low-level routines for performing common linear algebra operations.

IDE: *Integrated Development Environment*. A software application that provides facilities for software development.

1 Introduction

The aim of this thesis is to explore the suitability of a Java programming course provided by Oracle Academy, as part of a technical education curriculum. Java is one of the most versatile, general-purpose coding languages. It has a strong industry presence and is commonly found across multiple operating systems and platforms. Java also often represents the first exposure to real-world object-oriented programming for many learners. Everything considered, it was a natural choice for this thesis.

In particular, the objective is to evaluate whether Oracle's Java course could serve as a foundation for a MOOC course for Metropolia. Since Java is so widespread and with the advancement of AI and ML, it would be a great advantage to Metropolia to have resources and courses delving into coding with Java and more specifically, machine learning. Java may not be the most common language associated with AI, but its scalability, type safety and easy maintainability make it a solid language for all types of applications.

Implementing the course is expected to be relatively easy, as most of the material is standalone, and the tasks can be done using any Java IDE, although Eclipse is recommended. The course is well-suited for independent study, however, peer reviews or collaborative elements could be included as well.

From Metropolia's perspective, adopting such a course would be beneficial. It allows students to build a solid foundation in Java while at the same time opening a path towards more advanced, AI/ML-focused studies later on. And since Oracle already provides the course materials, implementing the course would not require significant new resources.

This thesis focuses on assessing whether Oracle's Java with AI course is suitable and beneficial to be implemented as a Massive Open Online Course (MOOC) for Metropolia University of Applied Sciences (UAS). The course would

be implemented using Moodle, which is a learning management system (LMS) used worldwide.

2 About Java Programming Language

2.1 Brief History

Work on the project originally began in June 1991, when three engineers, James Gosling, Mike Sheridan and Patrick Naughton, part of a Sun Microsystems team, wanted to create a language applicable to small electrical devices. [1]

Originally, Gosling attempted to use a modified and extended version of C++, but later decided to abandon the prospect in favor of designing a new platform called Green and a completely new language, which he called Oak, named after an oak tree outside his office. [2] This new language was used in the creation of a digital remote control, which had a graphic and animated touch screen. The remote was capable of managing all the devices in a living room. It was also one-of-a-kind, because unlike most devices of its time, the remote didn’t rely on any specific processor, meaning the remote could execute the same program on different hardware platforms. [1]

In 1993, with the arrival of the HTTP protocol and the Mosaic browser, the team realized that the Internet would be the perfect network for their product. Later, in 1995, James Gosling revealed a browser called WebRunner, which was capable of rendering HTML content and running Applets (small Java-applications). Then, WebRunner turned into HotJava, followed by java.sun.com opening to the public. Eventually the name would be changed into Java (slang for coffee). [1]

Soon after, Sun and Netscape revealed their intent to embed the new technology into their browsers, marking the official launch of the language. Then, in 1996, the first public implementation was released as Java 1.0.

Java 1.0 promised “write once, run anywhere” functionality, and included many essential features: automatic garbage collection, strong type checking, multithreading support and the Abstract Window Toolkit (AWT) for creating

graphical user interfaces. Many complex and error-prone features common in C++ were intentionally left out, for example: pointer arithmetic and multiple inheritance. [3] Java was designed to be a simpler, more readable and more secure language, which is crucial for developing networked applications.

The promise of “write once, run anywhere” was not just a part of marketing. Instead of compiling directly to native machine code such as C or C++, the compiler (javac) compiles the source code into bytecode, which is then executed by the Java Virtual Machine (JVM), transforming it into machine-readable code. [1]

Java Applets were also a novel use case which made Java more popular at the time. They were small Java programs that could be run directly within a browser. The integration of Java Applet support into Netscape’s Navigator 2 browser further increased the adoption of Java in the mid-to-late 1990s. [4] Applets would eventually fall out of favor though, due to lack of security and performance limitations. [5]
Sun Microsystem’s strategy to offer Java as an open standard also played a big role in the rise of Java’s popularity. It was free to use, free to distribute, and free to port to different platforms. [6] Also, Java’s simple object-oriented model led to rapid adoption by educators, software engineers, and large-scale enterprises. [7]

Java kept evolving after the release of 1.0, with several key versions that followed. Version 1.1 included inner classes, a new event-handling model, and Remote Method Invocation (RMI). Later, with Java 2 came the Swing GUI toolkit, the Collections Framework, and enhanced security features. Version 1.4 added the New I/O (NIO) API, assertions and logging support. The release of Java 5.0 introduced generics, enums, annotations and enhanced concurrency tools. The final notable version was Java 8, which brought with it lambda expressions and the Stream API, that enabled functional-style programming and made data processing more efficient. [2]

2.2 Present

Since Java's debut in 1996, it remains as one of the world's most popular and trusted programming languages, embedded in enterprise systems, Android development, cloud applications and large-scale infrastructure. Its original guiding principles, namely platform independence, security and simplicity, still guide its development and the language keeps adapting through a steady stream of updates. It continues to rank highly in surveys and utilization metrics like TIOBE and Stack Overflow, despite the rise of newer languages.

The language has undergone a significant amount of modernization, especially with the release of Java 17 and newer versions. Record classes and sealed types help streamline data modelling and enhance type hierarchy safety. Pattern matching, switch expressions, and record patterns reduce boilerplate and improve code clarity. Also, virtual threads (via Project Loom) enable lightweight, high-throughput concurrency. [8]

Java is also highly backwards compatible, which has ensured its presence in older systems, all the while powering modern cloud-native applications using frameworks such as Spring Boot, Quarkus, and Micronaut. Also, tools like Gradle and Maven support large-scale builds, and the OpenJDK community helps by providing regular updates every six months. [9]

Another factor that helps keep Java relevant is its cross-platform Virtual Machine (JVM), a highly optimized runtime that does not just support Java, but also other languages like Kotlin, Scala and Groovy. Its flexibility, along with being widely adopted in DevOps and backend development, ensures Java's position as a key technology in modern software ecosystems. [9] [10]

2.3 Future

While Java has a long-standing and stable foundation, it is still improving in order to maintain relevance in the modern, rapidly evolving software development landscape. Also, guided by multiple ambitious OpenJDK projects,

Java is now focusing on adaptability, expanding the surrounding ecosystem, and refining the developer experience.

Project Valhalla is changing Java's type system by introducing value classes, which are objects without identity that can avoid heap allocation and eliminate pointer indirection. This helps the JVM stop wasting time tracking pointless references.

Project Panama is also another initiative that aims to simplify interoperability between Java and native C libraries, which could improve performance by a significant margin and expand Java's reach into systems programming, embedded devices and high-performance computing. Also, through the upcoming Vector API and SIMD programming, Java can finally fully utilize the power of modern CPUs.

Meanwhile, Project Lilliput intends to slim down object headers within the JVM. It will start with shrinking the headers down to 64 bits, and then later on to 32 bits. This will lead to lower memory consumption, less GC pressure, and quicker execution.

Project CRaC and Project Leyden on the other hand, aim to reduce start-up times. CRaC allows Java applications to be restored almost instantly by taking snapshots of the entire JVM state, while Leyden aims to introduce ahead-of-time (AOT) compilation and closed-world constraints.

And lastly, there is Project Babylon. Its goal is to evolve reflection, which is a feature in Java that allows a program to examine and manipulate the internal properties of classes, methods, fields, and constructors at runtime. Babylon seeks to turn reflection into code reflection, which would allow Java to analyze and manipulate its own code at a higher level. [11]

3 Machine Learning with Java

While it is possible to implement machine learning algorithms manually in Java, and this is the focus of the Java with AI course by Oracle, in practice, most machine learning projects typically rely on existing libraries. Manual implementation is mainly useful in an educational setting, helping students develop a deeper understanding of the logic behind the algorithms.

There are multiple approaches to implementing Machine Learning with Java. While the Java with AI course by Oracle emphasizes a low-level understanding by guiding students to implement decision tree algorithms manually, there is also a diverse collection of Machine Learning toolkits available. This chapter will focus on 4 of these toolkits: Weka, Deeplearning4j, SMILE, and Tribuo.

First released in 1999, Weka remains a common reference point for classic ML algorithms in Java. Distributed under the GNU General Public License, Weka emphasizes accessibility and clarity over raw performance or deep learning capabilities. It integrates data preprocessing, modeling techniques, evaluation, and visualization into a single desktop application, supported by a well-documented API and command-line-interface (CLI). Input to Weka is formatted using ARFF (Attribute-Relation File Format), which embeds metadata alongside data values, enhancing reproducibility in experimental workflows. [13] Though Weka scales poorly to large datasets and multicore systems due to its single-threaded training loop, it remains suitable for exploratory analysis, algorithm comparison, and education. [12]

Deeplearning4j (DL4J) is an open-source deep-learning library for the JVM, developed by Konduit AI. It includes implementations of standard neural network architectures, and supports both forward and backpropagation for supervised learning tasks. It is designed to run natively on the JVM, allowing integration into existing Java applications, without dependencies on external language bindings (e.g. Python) or native code interfaces. DL4J is implemented on top of ND4J, a numerical computation library for the JVM, which provides multi-dimensional arrays and linear algebra operations similar to those found in

NumPy for Python. DL4J also supports hardware-acceleration with CUDA, allowing for GPU-based training using NVIDIA GPUs. Its 2024 release added ONNX support as well, enabling the importing of models trained in Python. Despite its smaller ecosystem compared to Python libraries such as TensorFlow or PyTorch, DL4J is a practical choice for environments already based on Java. [14]

SMILE (Statistical Machine Intelligence and Learning Engine) is another machine learning library written in Java and Scala. Developed by Haifeng Li and released as open-source under a BSD 2-Clause license, it was designed for performance and low-overhead integration in JVM-based applications. SMILE comes with a broad range of machine learning methods, including classification, regression, clustering, manifold learning, and graph analysis. [15] It operates without external dependencies like ND4J or TensorFlow, and instead relies on native Java or BLAS-based linear algebra, thus favoring low-latency and resource-constrained environments, such as embedded platforms or high-throughput applications. The SMILE API provides direct control over data preparation, feature construction, and model selection, requiring a high level of familiarity with the underlying statistical principles. In exchange, it offers precise control over algorithm behaviour, including hyperparameter tuning and support for non-standard workflows. Though because of this, and the fact that it does not include a GUI, SMILE is not intended for educational or exploratory use by non-programmers. [16]

Then there is Tribuo, which is a machine learning library licensed under Apache 2.0. Developed by Oracle Labs, it provides tools for classification, regression, clustering, model development, anomaly detection, and multi-label classification. It also provides a strongly typed modular API, with built-in interfaces to XGBoost, TensorFlow, and ONNX runtime. One of Tribuo’s core principles is provenance, meaning every Model, Dataset, and Evaluation object automatically captures metadata, such as data sources, transformation steps, trainer parameters, timestamps, and hash values. This allows deterministic model recreation and supports transparent model benchmarking. Tribuo also supports the export of a variety of its models in the ONNX format, including

linear models, sparse linear models, and ensemble models based on them, and also embeds the model's provenance metadata as part of the exported file. Its modular architecture allows projects to include only what is needed, avoiding unnecessary dependencies, unless chosen. [17] [18]

In conclusion, Weka's emphasis is accessibility, DL4J focuses on deep-learning within Java-first environments, SMILE's priority is performance and minimalism, and Tribuo centers on reproducibility and deployment traceability.

4 Concept for Machine Learning with Java MOOC Course

The goal of this pre-study was to explore whether the Oracle Academy course, “Java with AI” could be implemented in Metropolia’s Moodle learning management platform. It would be beneficial to Metropolia if such a course and others could be accessed from Moodle directly, instead of Oracle Academy if possible. This should be possible, since the materials are already basically standalone, they would just have to be licensed for use by Metropolia. Also, Moodle could be used to monitor the students’ progress, which is not possible through Oracle Academy.

4.1 Java with AI by Oracle

Java with AI is one of the many courses offered by Oracle Corporation, accessible through their Oracle Academy learning site. In order to enroll on the course, students need to be granted access. Students can engage with the course content without direct supervision, meaning they can progress through the course at their own pace.

The course’s learning materials consist of .pdf-files, one for each subchapter. Also, code files for the tasks introduced in the course are provided, including a demo version of a yes/no game. They are accessible through the Oracle Academy website. They can also be downloaded directly to the user’s computer, which means the course can also be done offline.

The course contains 4 chapters, with about 4 subchapters in each chapter. It begins with the basics of Artificial Intelligence and Machine Learning, teaching the student about common terminology and concepts related to the topic, such as supervised and unsupervised learning, classification, regression etc. The course gradually becomes more difficult, when more complicated topics are introduced, like the CRISP-DM model, binary trees, and recursion among others. Although it becomes more difficult, the progression feels intuitive, as the chapters and subchapters build on each other. Most of the subchapters include a small task at the end of the slides, such as researching Moore’s law. These

tasks also become more difficult later on, such as coding recursive methods, and their own yes/no game. Solutions to most tasks are also provided, in case students struggle with the tasks. At the end of the course, a final exam is included, with typical multi-choice questions.

4.2 Moodle Implementation

As mentioned earlier, the goal of this pre-study was to implement the Java with AI course by Oracle as a MOOC course for Metropolia's Moodle. The logical starting point was to enroll in the course and begin progressing through the material and the tasks, in order to find out whether the course could be implemented as is or if changes were necessary. Originally, access to Oracle Academy and the course itself was granted by Janne Salonen, the teacher supervising this pre-study, but the website was down for renovation while progressing through the course, so Salonen provided the course materials directly.

The task began with checking the course requirements. Since there are coding tasks included, the course requires the use of a Java Integrated Development (IDE). Oracle recommended using NetBeans or Eclipse. In this evaluation, the Eclipse IDE was chosen, partly due to recommendations and also to gain experience with a different IDE, having previously used NetBeans in the earlier studies. Both IDEs are open-source and readily accessible. Other modern IDEs could be used as well, such as IntelliJ, Visual Studio Code, or JetBrains for example.

By going through the course, it is clear that implementing said course would be easy, since all the materials are readily available, and easy to go through. The main focus therefore is how to implement the course for Metropolia's Moodle.

Initially, access to Oracle Academy and the course needs to be granted by the supervising teacher. The teacher would provide instructions on how to create an Oracle Academy account, and after its creation, the teacher can enroll the student on the course. These instructions could be on Moodle for example.

Once the student has access, they can begin studying at their own pace, provided they finish the course in the time required by the Metropolia course. Going through the course would be the same as studying directly through Oracle Academy, by reading and learning through the provided .pdf-files. By also including Moodle, the student could contact their supervising teacher, should any questions pop up.

Oracle Academy doesn’t require that the students actually turn in their answers or solutions to the tasks, therefore, at the end of each chapter, the student would find the solution to the given task, and turn it in through Moodle. If it’s a coding task, they could just send the output of their code for example, and the code used to generate said output.

Multi-option quizzes could also be implemented in between chapters, like on Oracle Academy presumably, since access to the site was lost during the renovation as mentioned. The provided course materials did not include them, only the final exam. Parts of the final exam could be turned into quizzes, or entirely new questions could be used.

After passing everything that is required, the student would undertake the final exam, which by default only includes multiple-choice questions. The final exam could be changed so it also includes written tasks as well. Then, after passing the final exam, the student would contact their teacher and have them mark the course as completed. This could be done automatically through Moodle.

In summary, Oracle Academy’s Java with AI course can be integrated into Moodle with ease. Students would access and study the core learning materials via the Oracle Academy platform, while course activities such as quizzes, task submissions, and progress tracking would be managed within Moodle.

5 Conclusion

This thesis set out to evaluate the suitability and possibility of implementing Oracle Academy's Java with AI course as a Massive Open Online Course (MOOC) for Metropolia UAS, using the Moodle platform. The objective was to assess whether the course's structure, content, and delivery methods could be integrated into Metropolia's existing educational platform.

To find out whether integration into Moodle was possible, access was provided by a teacher supervising this work, after which the course was initiated. After finishing the course, it became clear that integration is indeed feasible and would be relatively easy.

By default, the course already aligns well with the goals of a MOOC course. It is self-paced and already structured in a clear way, with well defined learning objectives. Provided Metropolia has a license to use the course materials, there would be no legal or financial barriers to implementing said course.

In terms of technical implementation, the process is straightforward. All of the materials are available directly from Oracle Academy, therefore Moodle would only have to contain instructions on how to complete the course, quizzes, task submissions, and the final exam. Students would only need to be granted access to the Oracle Academy platform, which can be granted by the teacher in charge of the Moodle course. Also, the use of open-source Integrated Development Environments (IDEs) ensures that any student can access and progress through the course.

Implementing the course on Moodle would be beneficial, since Moodle would make tracking the students' progress easier. Most students should also be familiar with the Moodle platform, making enrollment easier while offering a more familiar interface.
Looking forward, the course could be improved upon through the integration of adaptive learning technologies, including artificial intelligence. Diagnostic tests could be done at the beginning of the course, to assess each students' skill

level, and each student could be assigned tasks based on their individual abilities. Also, gamification or automatic feedback systems could be implemented to enhance independent study.

In conclusion, the integration of the Java with AI course by Oracle Academy is not only possible, but would enhance and build upon the strong foundation that Oracle Academy already provides.

References

1 Baeldung. The history of Java [Internet]. Baeldung; [cited 2025 Jul 24]. Available from: https://www.baeldung.com/java-history

2 GeeksforGeeks. The complete history of Java programming language [Internet]. GeeksforGeeks; [cited 2025 Jul 24]. Available from: https://www.geeksforgeeks.org/java/the-complete-history-of-java-programming-language/

3 Newmarch J. The Java programming language [Internet]. jan.newmarch.name; [cited 2025 Jul 24]. Available from: https://jan.newmarch.name/java/auugjava/paper.html

4 Hill M. Web components before web components [Internet]. The History of the Web; 2020 Sep 21 [cited 2025 Jul 24]. Available from: https://thehistoryoftheweb.com/web-components-before-web-components/

5 Chemaxon. Java applet technology: the past and future and the end [Internet]. Chemaxon; 2021 Nov 30 [cited 2025 Jul 24]. Available from: https://chemaxon.com/blog/news/java-applet-technology-the-past-and-future-and-the-end

6 Loukides M. Java at the crossroads [Internet]. O'Reilly Media; 1999 Mar [cited 2025 Jul 24]. Available from: https://web.archive.org/web/19990508212734/http://java.oreilly.com/news/loukides_0399.html

7 GeeksforGeeks. Object-oriented programming (OOPs) concept in Java [Internet]. GeeksforGeeks; [cited 2025 Jul 24]. Available from: https://www.geeksforgeeks.org/java/object-oriented-programming-oops-concept-in-java/

8 Patil V. Modern Java mastery: best practices and performance tuning in 2024 [Internet]. Medium; 2024 Jan 15 [cited 2025 Jul 24]. Available from: https://medium.com/@vishwajitpatil1224/modern-java-mastery-best-practices-and-performance-tuning-in-202-69f66801f5e9

9 McCann A. Why Java remains a powerhouse for enterprise application development in 2025 [Internet]. Medium; 2025 Mar 3 [cited 2025 Jul 24]. Available from: https://medium.com/@anthony_mccann/why-java-remains-a-powerhouse-for-enterprise-application-development-in-2025-8f9040daa9a8

10 Better Projects Faster. Java full stack report 2023 [Internet]. Better Projects Faster; [cited 2025 Jul 24]. Available from: https://betterprojectsfaster.com/guide/java-full-stack-report-2023-01/lang/

11 Simálek D. Java’s future: the epic of Project Valhalla, Panama, and beyond [Internet]. Medium; 2023 Oct 12 [cited 2025 Jul 24]. Available

from: https://medium.com/@dalansimalek/javas-future-the-epic-of-project-valhalla-panama-and-beyond-b6f06d81412e

12 University of Waikato. Weka: machine learning software [Internet]. Hamilton (NZ): University of Waikato; [cited 2025 Jul 23]. Available from: https://ml.cms.waikato.ac.nz/weka/

13 University of Waikato. ARFF formats and processing [Internet]. Hamilton (NZ): University of Waikato; [cited 2025 Jul 23]. Available from: https://waikato.github.io/weka-wiki/formats_and_processing/arff_stable/

14 Konduit. Deeplearning4j: open-source deep learning library for Java [Internet]. Konduit AI; [cited 2025 Jul 24]. Available from: https://deeplearning4j.konduit.ai/

15 Haifeng L. SMILE: Statistical Machine Intelligence and Learning Engine [Internet]. smile.ai; [cited 2025 Jul 24]. Available from: https://haifengl.github.io/smile/

16 Haifeng L. Homepage of Haifeng Liu [Internet]. smile.ai; [cited 2025 Jul 24]. Available from: https://haifengl.github.io/

17 Oracle. Tribuo: machine learning in Java [Internet]. Tribuo.org; [cited 2025 Jul 25]. Available from: https://tribuo.org/

18 Oracle. Tribuo source code repository [Internet]. GitHub; [cited 2025 Jul 25]. Available from: https://github.com/oracle/tribuo