

# Ass. #1 :Integrating Java with Python for Machine Learning

---

## Objective and preamble

Learn how to invoke a specific Python script named 'final.py' for machine learning from a Java application. The Java code is provided on D2L. This activity aims to understand Python's integration process and benefits for machine learning, explore different classification methods, and grasp key performance metrics. So, in general, in this lab, we will study the integration of the Java and Python code in one code base or project: the credit card fraud detection data from <https://www.kaggle.com/code/anmol111pal/credit-card-fraud-detection> has been used to exhibit such as mixed-language programming. The Python code is taken mainly from. This Jupyter Notebook file was converted into Python, and debugging and data plotting have been removed. The data and code have been available on D2L. You must have your Kaggle account to download the data (CSV file that contains the financial transactions)

## Rationale

### 1. Introduction to Mixed-Language Programming

- Familiarize students with the concept and practices of utilizing multiple programming languages within a single project.
- Highlight the strategic importance of selecting the optimal language for specific tasks to enhance functionality and performance.

### 2. Data Transfer Between Languages

- Provide hands-on experience with methods for transferring data between programs written in different programming languages.
- Demonstrate the practicality and efficiency of integrating diverse programming environments to achieve seamless interoperability.

### 3. Exposure to Machine Learning

- Introduce the fundamentals of machine learning (ML) to illustrate its applicability and effectiveness in solving real-world problems.
- Offer insights into the burgeoning field of ML, emphasizing its significance in the technological landscape.

#### 4. **Application in Financial Fraud Detection**

- Explore the application of machine learning techniques in detecting financial fraud, showcasing a practical example of ML's utility.
- Highlight the role of computational methods in enhancing security and integrity within the financial sector.

Through this lab, students will not only grasp the technical aspects of mixed-language programming and data exchange but also appreciate the broader implications of machine learning in addressing and mitigating contemporary challenges, such as financial fraud.

## **Introduction**

Due to its extensive libraries and community support, this lab focuses on using Python for machine-learning tasks. While Java is a powerful language, it lacks the ecosystem that Python boasts for data science and machine learning. In this lab, we will use a provided Java code from D2L to call a Python script named 'final.py.'

## **Why Python for Machine Learning?**

**Extensive Libraries:** Python's ecosystem includes libraries such as NumPy, pandas, scikit-learn, and TensorFlow, which facilitate various machine learning tasks.

**Simplicity:** Python offers a straightforward syntax, making coding more accessible and efficient.

**Community Support:** A robust and active community continuously contributes to improving Python's machine-learning capabilities.

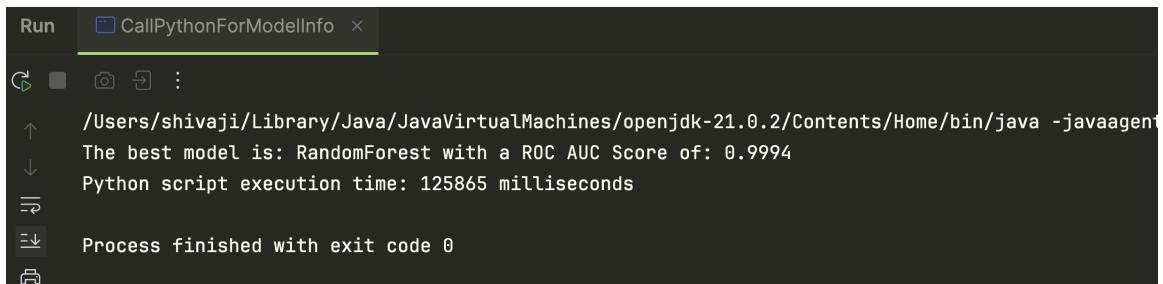
## Procedure

1. Retrieve the provided Java code from D2L.
2. Ensure Python is installed and the 'final.py' script is ready.
3. Use Java's ProcessBuilder class within the provided Java code to invoke the 'final.py' Python script. Both Python and Java code files are available on the D2L. The CSV file containing the financial data must be in the same folder as the Python script. Depending upon your machine's operating system, you need to put either "python" or "python3"; in most cases, the "python" will work.
4. In the Java application, handle the Python script's input/output streams to capture the machine learning model's results. Both fully functional codes should work without compile/runtime errors. If you don't have any library installed, you can install it on the terminal or command prompt if you have admin rights. For instance, the library xgboost can be installed, as shown below. Please notify your instructor if you are required to install any Python library on college MACs and cannot do so. You might be required to do some online searches to get some information,

```
Last login: Sun Mar 31 22:09:10 on ttys000
(base) shivaji@Shivajis-MBP ~ % pip install xgboost

Collecting xgboost
  Obtaining dependency information for xgboost from https://files.pythonhosted.org/packages/03/e6/4aef6799badc2693548559bad5b56d56cfe89eada337c815fdfe92175250/xgboost-2.0.3-py3-none-macosx_12_0_arm64.whl.metadata
  Downloading xgboost-2.0.3-py3-none-macosx_12_0_arm64.whl.metadata (2.0 kB)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in ./anaconda3/lib/python3.11/site-packages (from xgboost) (1.11.4)
Downloading xgboost-2.0.3-py3-none-macosx_12_0_arm64.whl (1.9 MB)
1.9/1.9 MB 7.3 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-2.0.3
```

5. The code will take a lot of time to execute; for instance, it has taken around 125 seconds on my MAC. Be patient and intervene only when you see any error.



```

Run CallPythonForModelInfo x
/Users/shivaji/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -javaagent
The best model is: RandomForest with a ROC AUC Score of: 0.9994
Python script execution time: 125865 milliseconds
Process finished with exit code 0

```

6. Modify the Python and Java codes so you send the name of the best model with the best F1 score.

### What needs to be submitted to D2L as a part of this lab submission?

- 1) The output the java code which shows F1 score rather than ROC AUC. it should show the name of the model and value. You need to modify the code for that.  
Write short answers to the following questions.
  - a) From the credit card data provided in the csv file, how many records and features are there? Why PCA (Principal Component analysis) can be used for dimension reduction?
  - b) Explain Accuracy, F1 score and ROC in your own words.
  - c) What is metric will you choose if the classify more than two classes?

### Performance Metrics

Accuracy: Correctly predicted observations ratio.

F1 Score: Harmonic mean of precision and recall.

ROC AUC: Model's ability to distinguish between classes metric.

### Accuracy

Accuracy is the proportion of accurate results among the total number of cases examined. It is a measure of how many observations, both positive and negative, were correctly identified. Accuracy is best used when the class distributions are

balanced. However, it can be misleading when dealing with imbalanced classes, as it might reflect the underlying class distribution rather than the model's ability to distinguish between classes.

A **Confusion Matrix** is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. It is especially useful in statistical classification, where it displays the number of correct and incorrect predictions made by the model compared to the actual outcomes (target values) in the data.

Here's how it is structured using ASCII art for a binary classification problem:

		Predicted	
		+ve	-ve
Actual	+ve	TP	FN
	-ve	FP	TN

Where:

- **TP (True Positives)**: The cases in which the model correctly predicted positive.
- **TN (True Negatives)**: The cases in which the model correctly predicted negative.
- **FP (False Positives)**: The cases in which the model incorrectly predicted positive (Type I error).
- **FN (False Negatives)**: The cases in which the model incorrectly predicted negative (Type II error).

### **Relation with Accuracy**

**Accuracy** measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions (or the total number of instances being classified). The formula for accuracy is:

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

### Example

Imagine we have a dataset of 100 fruit images, and our task is to classify them as either "Apple" (positive class) or "Not Apple" (negative class). Let's say our model has made predictions as follows:

Correctly identified 30 images as "Apple" (TP).

Correctly identified 50 images as "Not Apple" (TN).

10 images were incorrectly identified as "Apple" (FP).

10 images were incorrectly identified as "Not Apple" (FN).

The confusion matrix for this would look like:

	Predicted	
	Apple	Not Apple
Actual Apple	30	10
Actual Not Apple	10	50

$$\text{Accuracy} = \frac{(30+50)}{(30+50+10+10)} = \frac{80}{100} = 0.80 \text{ or } 80\%$$

Our model's accuracy in this scenario is 80%. This means that out of 100 images, our model correctly identified 80 of them.

This example highlights how the confusion matrix is a useful tool for understanding how well a classification model is performing, not just in terms of how many instances were correctly classified (accuracy), but also how it mistakes one class for another (through FP and FN).

## **F1 Score**

The F1 Score is the harmonic mean of precision and recall, balancing the two when their importance is roughly equivalent. It is beneficial when classes are imbalanced or false negatives and positives have different implications. A higher F1 Score suggests a balance between precision and recall, emphasizing minimizing false positives and negatives.

Formula:

$$F1 = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

## **ROC AUC**

The Receiver Operating Characteristic (ROC) curve illustrates the performance of a binary classifier system as its threshold is varied. The Area Under the ROC Curve (AUC) compares classifiers and chooses the one that optimally discriminates between the classes. A higher AUC indicates a better performance, with 1 representing a perfect model and 0.5 a model equivalent to random chance.

The ROC AUC is calculated by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across different thresholds. The area under the curve is the area under the curve.

Metric	Definition	When to Use	Considerations
Accuracy	The ratio of correctly predicted observations (true positives and true negatives) to the total number of observations.	Best used when class distributions are nearly balanced, and the costs of false positives and false negatives are similar.	Can be misleading in the presence of imbalanced class distributions; does not reflect the performance on minority class well.
F1 Score	The harmonic mean of precision (the ratio of true positives to the sum of true and false positives) and recall (the ratio of true positives to the sum of true positives and false negatives).	Most useful when dealing with imbalanced datasets or when the cost of false positives and false negatives is high and different.	Provides a balance between precision and recall. It can be more informative than accuracy when classes are imbalanced.
ROC AUC	The area under the receiver operating characteristic (ROC) curve, a plot of the true positive rate against the false positive rate at various threshold settings.	Useful for evaluating the overall performance of a model, especially when comparing different models and when the classification threshold is not fixed.	Unaffected by class imbalance; however, it does not reflect the impact of different classification thresholds on the actual number of false positives and false negatives.

---

**Note:** Accuracy can fail to capture the effectiveness of a model when the classes are imbalanced (e.g., one class represents 95% of the data). In such cases, even a naive model that predicts the majority class for all instances would have high accuracy, hence the preference for the F1 Score or ROC AUC.



The F1 Score is more informative than accuracy for imbalanced datasets because it considers both false positives and false negatives, which is important when these misclassifications have a significant impact.

ROC AUC measures a model's ability to distinguish between classes and is particularly useful when comparing different models. It is threshold-independent and recommended when ranking or prioritizing different classes rather than hard-cut classifications.

## **Appendix**

### **Mixed Language Programming: Java and Python**

Mixed language programming involves using two or more programming languages within a single program or system. This approach can take advantage of the strengths and capabilities of each language to optimize performance, leverage specific libraries, or handle different tasks within an application.

Here's an explanation of how mixed language programming might work with Java and Python:

#### **Reasons for Mixed Language Programming**

**Leverage Strengths of Each Language:** Different languages have different strengths. For instance, Java might be used for its robustness and performance in handling backend operations. Python could also be employed for its rich set of data analysis and machine learning libraries.

**Existing Codebase:** In some cases, a project might need to incorporate a legacy system written in one language (like Java) with new development in another (like Python).

**Community and Resources:** Some languages have more robust support in specific domains. Python, for example, has a strong community and resource base in data science and machine learning.

**Developer Preference and Expertise:** Teams may use a language they are more comfortable with for specific project parts, resorting to another language for different components where they might have less expertise.

### How It Works

Mixed language programming can be achieved through several methods:

**Native Interface:** Languages like Java provide a native interface (JNI - Java Native Interface) that allows for calling C and C++ code. While JNI does not directly support Python, wrappers like Jython allow Python code to interact with Java directly.

**Interprocess Communication (IPC):** Two processes, one running Java and the other Python, can communicate using IPC mechanisms like sockets, shared memory, or files. Java might typically invoke a Python script by running it as a separate process and communicating with it through streams.

**Foreign Function Interface (FFI):** Some languages offer an FFI that enables them to call functions written in another language directly. This can be seen in Python with libraries like ctypes or cffi that allow calling C functions.

**Embedded Scripting:** Some applications written in a language like Java might embed a scripting engine, such as the Python interpreter, to allow for running Python code within the Java application context.

**Command Line Invocation:** A Java application can invoke a Python script using the command line, capturing the output for use within the Java program, as demonstrated with ProcessBuilder in Java.

### Challenges

Mixed language programming comes with its own set of challenges:

**Complexity:** Debugging, testing, and maintenance become more complex with the addition of another language.

**Performance Overhead:** Calling code between languages can introduce performance overhead, particularly if done frequently or with large data transfers.

**Error Handling:** Handling exceptions and errors across language boundaries can be tricky, as different languages have different mechanisms for dealing with errors.

**Development Environment:** Developers must manage multiple environments and toolchains, which can complicate the build and deployment processes.

In practice, while mixed language programming can be powerful, it requires careful consideration and design to ensure that the benefits outweigh the complexities. Using the right tool for the job is important, and sometimes, that means integrating multiple languages to build the best system possible.