National College of Ireland

Engineering and Evaluating Artificial Intelligence

Release Date: 24th February 2025
Due Date: 26th March 2025
Dr Abdul Razzaq, Dr Abdul Shahid, Sallar Khan

Continuous Assessment 1

Continuous Assessment (CA) Type: Project

Weight: The assignment will be marked out of 100. 50%

Instructions: Using Extreme Programming Method 2 Students in a group will work on this Assessment

PREFACE AND CONTEXT

Several concepts related to software architecture design can be implemented into Artificial Intelligence (AI) projects. One of the goals of implementing those concepts in AI is to develop modular and more maintainable AI projects. Toward this end, we are adopting the possible architecture for one AI project.

Consider you have been hired by a multinational company as an AI engineer. The multinational company has adopted Agile Xtreme Programming practices for designing and developing AI projects. As AI Engineers you and your (sole) teammate have provided the source code implementing several functionalities (activities) related to the multi-label email classification. You and your teammate are asked to design and develop an architecture for an email classifier that meets the following features:

1. Architecture should allow the modification in preprocessing (including data loading) activities without affecting the code related to the training and testing of ML models and vice versa.
2. Architecture should allow modeling the input data in such a way that adding/removing new ML models in code, allows developers to maximally avoid changing the format of input data. That is, the format of input data should be maximally consistent across all ML models we will employ.
3. The third and final feature that architecture should allow is to implement multiple ML models in such a way that the model-level behavioral differences (e.g., differences in training and testing codes related to each model) should be hidden from the controller. In other words, all the modeling-related functionalities should be able to access using a consistent interface (e.g., a set of methods) no matter how different each model is in terms of their coding for those functionalities.

A. You and your colleague start working on this together. For feature 1, you decided to implement separation of concerns (SoC) and for that, you initially chose the "Main Program and Subroutine Architecture" where you converted each functionality (provided to you) into functions. Hence, you separated the preprocessing and modeling (training/testing) into two different components (files) and design a main controller that can access the functions related to preprocessing and modeling. While accessing those functions you also realize that you may also separate some other functionality into different codes that might help in creating modular architecture. Those functionalities are converting textual data into numeric representation and modeling the input data. At this stage, you also realized that some of the variables need to be accessed in multiple components. To resolve this issue, you define such variables in the config.py file and import that file where required.

B. Then, to model the input data you and your teammate decided to implement an encapsulation strategy where you can encapsulate all your required input data in one object and can pass that object as data elements to all ML models. You know that you will need at-least training x, training y, testing x, and testing y data. Hence, you encapsulated these four data into an object. While performing that step, you also realize that while creating those data you may also remove the records for the classes having very few instances. Therefore, you removed such classes from the data.

C. To allow hiding the differences in modeling and to provide consistent access to different training/testing codes of ML models, you and your teammate decided to implement the concept of abstraction. In this implementation, you keep the model-specific codes in the same name methods (e.g., in addition to the constructor you have train(), predict(), and print_results() methods). You not only want to access all model-specific codes using same name methods but also you want to make sure that all of the new models should have to implement those same name methods. To accomplish this, you define a class base having the same name methods as abstract methods and inherit your ML model classes (e.g., RandomForest) from the base class. In this way, you ensure that every new model will have to implement those methods.

You have already done these tasks in your Lab, and thus you are supposed to employ your gained experience and the code you developed during labs as a starting point to solve the following continuous assessment.

CONTINUOUS ASSESSMENT

KEY OBJECTIVES

1. **Ensure modularity** by structuring code such that preprocessing and model training/testing remain independent.
2. **Maintain input data consistency** to allow seamless integration of multiple ML models.
3. **Implement abstraction** so that different ML models can be accessed via a uniform interface.
4. **Extend the current system** to support **multi-label classification** using two distinct architectural approaches:
    - **Chained Multi-Output Classification**
    - **Hierarchical Modeling**

Building on the previously designed architecture, your team needs to do the following task:
So far, your designed architecture is only classifying one dependent variable (i.e., Type 2). That is, it is only working as a multi-class classification problem and not a multi-label classification problem, whereas we know that each of the emails will be assigned one label from each dependent variable.
In such a problem, we can't just measure the accuracy of each dependent variable (That is, Type 1, Type 2, and Type 4) exclusively because the accuracy of the latter variable (e.g., Type 3) will depend upon the accuracy of former variable (e.g., Type 2). For example, if the accuracy of an ML model on the former variable is 80% then the accuracy of the latter variable will definitely be less or equal to 80% and can't be more than that. This can be illustrated as follows:
Consider that we only have one email instance in our data set and the true classification for that email instance is the following:

A.

Type 2	Type 3	Type 4
Suggestion	Payment	Subscription cancellation

Now if a model (A) will label each type as above then the accuracy of that algorithm will be 100%, that's obvious. Now if the other two models (B), (C) classify it in the following way then the accuracy is of that 67% and 33%, respectively.

B.

Type 2	Type 3	Type 4
Suggestion	Payment	Subscription retained

C.

Type 2	Type 3	Type 4
Suggestion	Refund	Subscription retained

Now consider the accuracy of the following model (D):

D.

Type 2	Type 3	Type 4
Other	Payment	Subscription cancellation

The accuracy of the D model is 0% although it classified type 3 and type 4 correctly. This is because the accuracy of Type 3 depends upon Type 2 and the accuracy of Type 4 depends upon the accuracy of Type 2 and Type 3. Likewise, the accuracy of the following model is 33%, even it classified two types correctly.

E.

Type 2	Type 3	Type 4
Suggestion	Refund	Subscription cancellation

CONTINUES ASSESSMENT TASK DESCRIPTION

First of all observe your input data (csv files) and consider that Type 1 always has only one class in each file. This suggests that we may ignore that Type because no classification is required for that column. Your task for this Continuous Assessment is to design, develop and compare the following two architectural design decisions to enable our architecture to support multi-label classification.

Design Decision 1: Chained Multi-outputs

Consider a design decision for architecture that allows the chaining of dependent variables: That is, Type 2, Type 3, and Type 4. By chaining, we mean that **ONE Instance of** a **model-A** (e..g., RandomForest) will assess each of the following three (singular or combined) types for each email instance:

(1) Type 2
(2) Type 2 + Type 3
(3) Type 2 + Type 3 + Type 4

Example:

(1) Suggestion
(2) Suggestion + Refund

(3) Suggestion + Refund + Subscription cancellation

Now, the effectiveness of Model-A will be assessed:

(1) in classifying Type2
(2) in classifying the combined version of Type2+Type3, and;
(3) in classifying the combined version of all dependents variables: that is Type2+Type3+Type4.

Design Decision 2: Hierarchical Modelling

Now consider a second design decision. Here, we are not chaining the outcome variables rather we are chaining the instances of a model in a way that the output of the previous model instance will decide the input data for the next model instance by filtering the data. For example, a model instance (of e.g., random forest) will classify Type 2, then data from Type 3 will be filtered for each class in Type 2 and a new model instance will perform classification on that filtered data. Consider such filtered data as a filter set A. Note that we are filtering Type 3 data for each class in Type 2, this means multiple filter sets A will be created whose count will be equal to the number of classes in Type 2 (And of course, for each filter set A a new model instance will be created). Then, at the next level, we consider each filter set A as a data set for Type 3 and will filter Type 4 under each filter similarly set A as we did for Type 2-Type 3 combination. This architecture will result in multiple models based on the number of classes in Type 2 and Type 3.

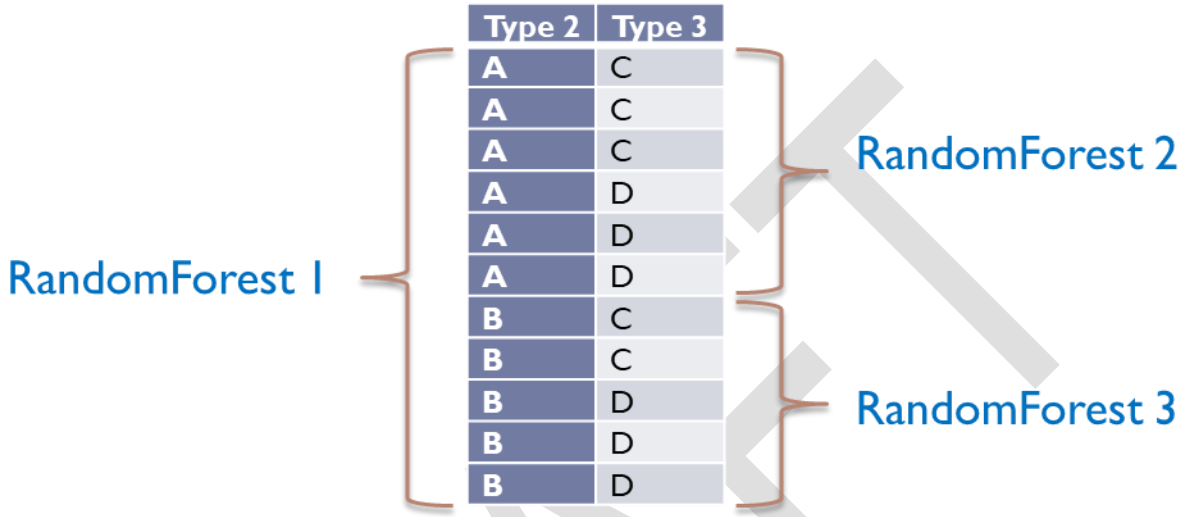
For a better understanding of how such an architecture will work, see the following figure:

In the figure, the Table represents Type 2 and Type 3 as dependent variables whereas A and B are the classes in Type 2, and C and D are the classes in Type 3. Three different models of random forest are in action here, where the first one (RandomForest 1) classifies the instances for Type 2, and then RandomForest 2 and RandomForest 3 classify instances in Type 3. But the difference here is that the number of RandomForest models implemented here is equal to the number of classes in the preceding dependent variable (i.e., Type 2). The benefit of such an architecture is that we can assess the effectiveness of the model for each class in the preceding dependent variable.

* You must need to implement task 9 using any version control system of your choice. At the time of submission, you only need to add me as a member of your repository so that I can access the commits and other metrics to assess the contribution of each team member.

How to Start:

To perform task 9, you may start by adding the solution of exercise 3 as your initial project and then can modify that.



STEPS TO PERFORM AND RUBRIC

	Steps	Marks	Teammate 1	Teammate 2
1	Design a Sketch for Design Choice 1	8	Major, Minor	Major, Minor
2	Design a Sketch for Design Choice 2	10	Major, Minor	Major, Minor
3	Identify **All** Components for Overall Architecture for Design Choice 1	3	Major, Minor, No	Major, Minor, No
4	Identify **All** Components for Overall Architecture for Design Choice 2	3	Major, Minor, No	Major, Minor, No
5	Identify **All** Connectors for Overall Architecture for Design Choice 1	3	Major, Minor, No	Major, Minor, No

6	Identify **All** Connectors for Overall Architecture for Design Choice 2	3	Major, Minor, No	Major, Minor, No
7	Identify Data Element(s) for **Overall Architecture for Design Choice 1**	2	Major, Minor, No	Major, Minor, No
8	Identify Data Element(s) for **Overall Architecture for Design Choice 2**	2	Major, Minor, No	Major, Minor, No
9	*Building on the Previous Exercise (as discussed at the start of this document) **develop a full working example for Design Choice 1**	16	Major, Minor	Major, Minor

SUBMISSION DETAILS:
An upload link will be provided to you at the start of Week 6 to submit your CA solutions. You need to submit a .doc file only where you need to present the sketch of each design choice and fill a similar table as above describing the components, connectors, and data elements. For the implementation part, you only need to add me as a group mate on your repository NO need to submit the code.

TURNITIN: All report submissions will be electronically screened for evidence of academic misconduct (i.e., plagiarism and collusion)

WHY IS THIS IMPORTANT?
This project simulates **real-world AI engineering challenges**, teaching you:

- **How to structure AI projects** for scalability.
- **How to compare architectural trade-offs** for performance and maintainability.
- **How to use software engineering principles** to create robust AI systems.

By the end of this assessment, you will have **practical experience** designing and implementing modular AI architectures—**a crucial skill for industry roles!**