National College of Ireland

MSc/PG Diploma in Artificial Intelligence (MSCAI_JAN24, PGDAI_JAN24)

Release Date: 12th June 2024
Submission Date: 10th July 2024 @ 23.55 hrs
Lecturers: Jaswinder Singh, Muslim Jameel Syed

Engineering and Evaluating Artificial Intelligence Systems (EEAI)

Continuous Assessment (CA) - 50%

Weight: The assignment will be marked out of 100. "[Click here and enter CA Weight]"

Instructions: Students will work in group of 2. These groups should be identified before the CA submission to the lecturer.

SUBMISSION DETAILS:
The document must be submitted as a Word or PDF document to Moodle before the deadline. The report should be concise, with the main part of the report (excluding references and appendix), limited to 7 pages in a typical 1-column format with a paragraph font size of 12 pt. Include student names, student IDs, and group number at the top of the first page. Late submissions will not be penalized if the student applied for an extension through NCI360 and it was approved.

TURNITIN: All report submissions will be electronically screened for evidence of academic misconduct (i.e., plagiarism and collusion). Ensure your work is original and properly cited where necessary.

PREFACE AND CONTEXT

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

Consider you have been hired by a multinational company as an AI engineer. The multinational company has adopted Agile Extreme Programming practices for designing and developing AI projects. As AI Engineers, you and your teammate have provided the source code implementing several functionalities related to 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. **Modular Preprocessing and Modeling:** Ensure that changes in preprocessing (including data loading) activities do not affect the code related to the training and testing of ML models and vice versa.
2. **Consistent Data Format:** Model the input data so that adding/removing new ML models allows developers to avoid changing the input data format as much as possible. The format of input data should be maximally consistent across all ML models employed.
3. **Unified Interface for Models:** Implement multiple ML models in such a way that model-specific behavioral differences (e.g., differences in training and testing codes) are hidden from the controller. All modeling-related functionalities should be accessible using a consistent interface (e.g., a set of methods), regardless of the internal differences of each model.

2 machine learning

multi class classification

CONTINUOUS ASSESSMENT

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 Cancelled

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 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):

if type 2 wrong, type 3, 4 does not matter
type 3,4 depend on type 2
Accuracy is 0

D.

Type 2	Type 3	Type 4
Other	Payment	Subscription Cancelled

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 though it classified two types correctly.

E.

Type 2	Type 3	Type 4
Suggestion	Refund	Subscription Cancelled

CONTINUOUS ASSESSMENT TASK DESCRIPTION

First, 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., Random Forest) 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:

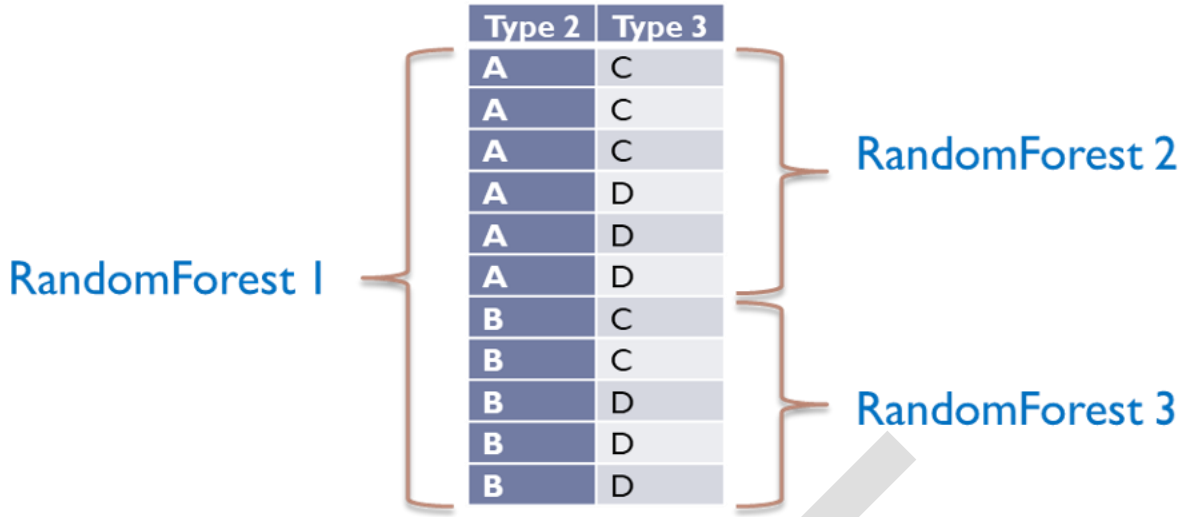
(1) in classifying Type2
(2) in classifying the combined version of Type2+Type3, and;
(3) In classifying the combined version of all dependents variables: that is Type2+Type3+Type4.

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 (Random Forest 1) classifies the instances for Type 2, and then Random Forest 2 and Random Forest 3 classify instances in Type 3. But the difference here is that the number of Random Forest 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.



STEPS TO COMPLETE

Step	Task	Details	Marks
1	Design Sketches	Create visual sketches for both design choices. The sketches should clearly outline the architecture for both the Chained Multi-outputs and Hierarchical Modeling approaches. Include all components and connectors in the sketches.	25
2	Component Identification	List all components (parts) for each design choice. Identify and describe all the essential components for both design choices. Explain their roles and interactions within the system.	10
3	Connector Identification lab: identify connectors	List all connectors (how parts connect) for each design choice. Identify and describe all the connectors for both design choices. Explain how these connectors facilitate communication between different components.	10
4	Data Elements Identification	Identify all data elements (types of data used) for each design choice. Describe all the data elements required for the system, including their types, sources, and how they are processed. Ensure that the data format remains consistent across different models.	10
5	Implementation of Design Choice	**Develop a full working example using Design Choice 1 (Chained Multi-outputs). Implement the Chained Multi-outputs approach, ensuring that the code is clean,	25

		well-documented, and functional. Use a version control system (like Git) to track your progress. Add the instructor as a collaborator to your repository.	
6	Report Presentation	Write a detailed report. The report should be well-written, clear, and concise, adhering to the specified format and length limits. It should include the design sketches, descriptions of components, connectors, and data elements. References should be complete and correctly used.	20

**You must implement task 5 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.

SUBMISSION

- **Document Submission:**
  - Submit a .doc or .pdf file with:
    - Design sketches for both choices.
    - Tables describing components, connectors, and data elements.
- **Code Submission:** collaborators add him as
  - Add your lecturer as a collaborator to your version control repository for code review.

Each member has contributed in the repository

Grade Criterion	H1 (> 70%)	H2.1 (> 60%)	H2.2 (> 50%)	Pass (> 40%)	Fail (< 40%)
Design Sketches: 25%	Exceptionally clear, detailed, and innovative sketches for both design choices. All components and connectors are accurately identified and well-integrated.	Clear and detailed sketches for both design choices. Most components and connectors are accurately identified and well-integrated.	Reasonably clear sketches with most components and connectors identified. Some minor integration issues.	Basic sketches with several components and connectors identified. Some integration issues.	Poor or incomplete sketches with few components and connectors identified. Significant integration issues.
Component Identification: 10%	All components for both design choices are thoroughly identified and described with excellent clarity.	Most components for both design choices are identified and described with good clarity.	Majority of components for both design choices are identified and described with reasonable clarity.	Some components for both design choices are identified and described with basic clarity.	Few or no components for both design choices are identified or described.
Connector Identification: 10%	All connectors for both design choices are thoroughly identified and described with excellent clarity.	Most connectors for both design choices are identified and described with good clarity.	Majority of connectors for both design choices are identified and described with reasonable clarity.	Some connectors for both design choices are identified and described with basic clarity.	Few or no connectors for both design choices are identified or described.
Data Elements Identification: 10%	All data elements for both design choices are thoroughly identified and described with excellent clarity.	Most data elements for both design choices are identified and described with good clarity.	Majority of data elements for both design choices are identified and described with reasonable clarity.	Some data elements for both design choices are identified and described with basic clarity.	Few or no data elements for both design choices are identified or described.
Implementation of Design Choice 1: 25%	Full working example is exceptionally well-implemented. Code is clean, well-documented, and demonstrates thorough functionality.	Full working example is well-implemented. Code is clean and well-documented with good functionality.	Working example is reasonably well-implemented. Code is mostly clean with adequate functionality.	Basic working example implemented with some functionality. Code may have issues and limited functionality.	Poor or incomplete implementation. Code is unclear, poorly documented, and lacks functionality.
Report Presentation: 20%	Report is exceptionally well-written with no language errors. All figures are clear, well-conceived, and easy to read. Format and length limits are adhered to, and references are complete and correctly used.	Report is well-written with few language errors. Figures are clear and well-presented. Format and length limits are adhered to, and references are mostly complete and correctly used.	Report is readable with some language errors. Figures are mostly clear. Format and length limits are mostly adhered to, and references are adequate.	Report is readable with many language errors. Figures are unclear. Format and length limits are not fully adhered to, and references are incomplete.	Report is poorly written with significant language errors. Figures are poor and hard to read. Format and length limits are not adhered to, and references are missing or poorly used.