Course Recommender System API using Multi-Objective Optimization

Author: Taha Jomehpour Arashlou

Consultant: MireLa Relijan-Delany

1. Abstract

Recommender systems are used in different fields of life including academia and e-commerce. These recommender systems use objective-based approach to suggest solutions. Traditional approach to develop these recommender systems is to use traditional algorithms such neural networks, support vector etc. These traditional algorithms use single objective to recommend solution. Optimization algorithms can use multiple objectives to recommend solution. The API developed recommends courses from Coursera dataset using multi-objective optimization algorithms. Three objectives used by API for recommendations are skills, rating, and difficulty level. The API is developed using python framework flask. The optimization algorithm is implemented using python framework Pymoo. Pymoo has the implementation of many single and multi-objective algorithms. The API is using multi-objective optimization algorithm NSGA2 for course recommendations. API can work in any environment with any interface. The project has developed the interface using flask jinja template, HTML and CSS. Currently the API is using single objective in recommending courses which is Skill. The API recommends five solutions to user when requests the API for recommendations.

Table of Contents

1. Abstract 2

2. Introduction 5

Problem to be solved 5

Project Objectives 5

Beneficiaries 6

3. Output Summary 7

API 7

4. Literature Review 8

Methodology 12

Implementation of recommender system 12

Dataset 12

Implementation 13

Data Cleaning 14

Implementation of recommender system using Pymoo python framework 14

Problem Class 15

_evaluate Function 16

Parameters of _evaluate function 17

Algorithm used for optimization problem of course Recommender system 18

Minimize Function 19

Getting user inputs through flask API 19

Flask API 20

Flask API to interact between the user and the recommender system 21

User Interface to interact with recommender system 21

Visualization of the performance of multi-objective optimization 23

Pareto of the multi-objective optimization algorithm 23

5. Results 24

Recommender System API Architecture: 24

Implementation 26

Recommendations of API with ElementWiseProblem Class 26

Output of the pymoo interface or minimize function 33

Visualization using Scatter 35

Evaluation 37

Evaluation of number of records used in testing 37

Evaluation of course recommendation on skills and course description columns of the dataset 37

6. Conclusions and Discussion 39

Future Work 40

7. References 41

8. Appendices *44*

Appendix A: Project Definition Document **44**

Problem to be solved 45
Project Objectives 46
Beneficiaries 47
Work Plan 47
Methodology 48
Timeline 49
Risks 49
References 49
ETHICS FORM 51

Appendix B: Reuse Summary **56**

Appendix C: Changes to original submission of code **57**

Appendix C Diagrams and Screeshots **65**

2. Introduction

This project has created a recommender system API using multi-Objective optimization with library pymoo to help students to make a better decision for choosing an online course. The API is developed using flask python framework. The API has a graphical user interface that the users can use to enter their objectives to learn a course. They will enter their objectives in graphical interface and the API will recommend five courses that best match with user's objectives. The API has used multi-objective optimization algorithm NSGA2 with ElementwiseProblem class that is used to define the optimization problem. This project goes through the research and developing the system with different methods and ways.

Problem to be solved

Many students struggle to find a course that can expand their skills without researching extensively to reach the right course within their range of ability. The student may become discouraged to look for more learning as the material that they have found might not be a good suit, the student might not understand the topic if the material is too advanced and would get bored of it if the material is too easy for them. Even if the course is a great suit for the student, it can lack the quality content and good teaching material included in the course. Recommender systems can help to narrow down the courses for the student help them decide without experiencing information overload. Recommended systems make recommendations to user based on set criteria. Recommender systems are quite common in retail and businesses but, there hasn't been much work in the field of academia.

Most of the recommender systems do not consider multiple objectives in their recommendations, this project will create an API using Multi-Objective-Optimization from Skills, Rating and difficulty to recommend the best possible courses to the user.

Project Objectives

1. Efficiency and usability of the system.
    a. The Recommender system shall be efficient and use a suitable algorithm to provide personalised and reasonable options to the user.
    b. The system shall help the user to choose a suitable course from the online learning platform coursera.
    c. The system must give 5 recommended courses that matches the most with the description and the skills provided by the user.

   d. The system must run without any problems and must not contain any empty or harmful materials.
   e. The system must consider multiple factors to recommend a suitable course to the user

2. User friendliness of the system.
   a. The system must implement an easy and accessible interface.
   b. The system needs to have a feedback and refresh option, so it does not show duplicated or identical courses.
   c. The system must use an API to connect to the user.
   d. The system must provide description, course rating, university of the course, difficulty level and the URL so the user can easily choose and access.
   e. The system must be able to run on most devices without any problem.

3. Resource optimization
   a. The system shall have a credible source such as the database and the research.
   b. The system must not be financially costly to run and operate.
   c. The system shall have large enough database to not limit the users' choice.
   d. The system must give results in a reasonable time.

4. Research
   a. Research about how to implement Multi Objective Optimization
   b. Research different way of Multi Objective Optimization

Beneficiaries

- **Students**: students who want to learn new skillsets or expand their pre-existing knowledge can use the system to get recommendation and compare courses relevant to them so they can more confidently choose a course.

- **Employers:** employers can use this system to get relevant courses recommendations and trainings relevant to their business to train employees.

- **Graduates**: graduated who want to expand their CVs or need an insight and learning before they secure a role, can get recommendation of what they need to learn before getting into their desired field and increases their employability.

3. Output Summary

API

The output of the project is course recommender system API. This API is developed using flask and pymoo frameworks of python. The API has 119 lines of code as part of the API's and optimization algorithm implementation. The user interface of the API has two file index.html(36 lines of code) and styles.css that has 45 lines of code. All the code is my own code apart from the style.css which is taken from (wwww.w3schools.com).

The API has report of words that gives brief description of the objectives, methodology, implementation, testing and evaluation.

The course recommender system API can be used by anyone who is interested in course recommendations from Coursera online platform for online courses. Initially I will give it to some friend in my circle and some professor in my university for course recommendation from Coursera.

4. Literature Review

Recommenders are software programs that are used to recommend user items based on their interest from the information overload (Jannach et al., 2021). Recommender systems are developed to solve the problem of information explosion and improve user experience by modelling the user preference (Guo et al., 2020). Verma and Aggarwal (2020) states the views of Resnick and Varian (1997) that the recommender systems are important to avoid information overload and provide some useful recommendations. Recommender systems are a type of information filtering that are used to predict user's preference to a particular topic (Khanal et al., 2020). Recommender systems make recommendations to its users of items from the service that they are using. This service can be video streaming, which will recommend a personalized list of videos or movies to its users (Hug, 2020). Other real-life use of recommender systems is to get recommendation of books on amazon, recommendation of movies on Netflix and recommendation of documents from newsgroups to users etc. These recommender systems use the domain and the key features of data in that domain for recommendation along with user-specific and item specific features for example the description of the product and its demographic (Meville and Sindhwani, 2010). Recommender system provide solution to information overload in e-commerce by providing services like healthcare, online learning, marketing, and food industry (Alamdari et al., 2020).

Recommender system in academia

Recommender systems provide decision support and knowledge discovery when people need to choose from lots of available choices for example, they can be used by students to select a university of their choice from a list of universities (Mokarrama et al., 2020). Machine learning algorithms can be used by educational institutions to predict and propose a suitable action for a course based on the course specification and some other features such as learning outcomes and academic records that will enhance the quality of the course and the educational outcome. By using this system educational institutions can only focus on those areas of the course which are important for the students (Yanes et al., 2020). E-learning has changed the way students learn; however, they often found the topic that they are taught in are not much relevant. Recommender systems can help in such situations by recommending topic and contents such as videos that are relevant to the topic (Kulkarni et al., 2020). Number of massive open online courses (MOOC) and the number of students are growing on online MOOC platforms such as EdX, Coursera, Udacity, NetEase etc. therefore there is need for

recommender system for recommending courses from the MOOC platforms (Khalid et al., 2022).

Different types of recommender systems:

Most of the recommender systems usually are made up of 3 types of recommendations,
The most popular and implemented in various scenarios is Collaborative Filtering which based on retrieving and analyzing data by observing the user's interaction with the system and predicting their preferences based of the interests and the content that similar users have liked to generate a new recommendation(Gazdar et al., 2020).

The second type of recommender systems are content based recommender systems the objects are linked to their values and features. The system reads the user preferences and the items that user has shown interest in before to generate a new recommendation based of the keywords that describes the items and the rating so the user will be recommended based of the current or past interests. The third type of common recommender systems is Hybrid Recommender System that is combining any of two recommender system together to strengthen the upsides of both recommendation and minimize the downsides(Engati, n.d.).

Algorithms used for recommender system
Recommender systems are generally based on machine learning algorithms. Machine learning algorithms, dataset used, evaluation based on dataset and algorithms and prediction of the algorithms are key components of recommender systems (Khanal et al., 2020).
Genetic Algorithm is an AI technique which follows the principle of natural selection and evolution. By using Genetic Algorithms, the current best match will be replaced if a superior match is found continuously until there are no more option available. Genetic Algorithms are different from non-linear optimization procedures, instead of modifying a singular solution incrementally it creates diverse and enhanced solutions called population.A distinctive feature of genetic algorithms is the ability to manage a population in that each member possess a chromosome which allows a range of solutions to be explored (Min et al., 2005).

Multi-objective optimizationAccording to Konak et al., (2006) most of the real-world engineering problems have multiple objectives which are conflicting, and it is difficult to

achieve optimization of each objective such as increasing the performance and reliability but at a minimum cost

In the optimization of a single objective, it is feasible to identify the better solution within a given set. This typically results in a single optimal solution. However, in multi-objective optimization, there is no straightforward method for proving the superiority of one solution over another. This is due the fact that outcome involves a set of solutions that address multiple conflicting objectives at the same time (Pereira et al., 2022).

Other approaches in Recommender systems

Support Vector Machine (SVM):

One of the algorithms that we can use in a recommender system is Support Vector Machine (SVM), Support Vector Machines are a recent but powerful technique for regression and classification. The Support Vector Machine learns a separating hyperplane to expand and maximize the margin and to gain the ability to create an efficient generalization. In the recent time the SVM has been successful in applying in numerous areas and it has shown promising and impressive results. By use of pattern recognition for two class problem it determines by separating hyperplane with maximum distance to the nearest point of the training set. We call these points support vectors. SVM is a hyperplane that separates positive set from the negative set with the highest margin in it most simple linear form. (Min et al., 2005).

Random Forest:

Random Forest is recognized as one of the most accurate ensemble classifiers, especially adept at handling large datasets. It excels in predicting missing data with high accuracy, even when significant amounts of data are absent, and this is achieved without the need for pre-processing. This classifier integrates techniques of bagging and random feature selection, comprising multiple decision trees that serve as individual learners. Each tree is generated using a random subset of the training data. Once the forest is trained, test instances are processed through it, with each tree providing a classification outcome. The final output of the Random Forest is determined by taking the statistical mode of these classifications (AI Paraphrased) (Ajesh et al., 2016).

API:

Application Programming interfaces or for short APIs are mechanisms that allow communication between two separate software through its defined protocol. For instance, the

weather app that we use daily on our smartphones is connected by an API to the meteorological bureau's system which stores weather data throughout the day, which allows the weather application to showcase the updated daily weather at any given time(Amazon web services, 2023).

Flask is a Python based API web framework that falls into the category of a micro framework due to its design and implementation, which does not need specific tools or libraries and lacks an in built database. While it does not feature a database abstraction layer, Flask supports many extensions that widen the range of its functionality. These extensions include capabilities for form validation, object relational mapper, file upload handling, numerous open authentication technologies, and other tools commonly associated with frameworks. Also, these extensions receive updates more frequently than the core Flask program itself thus that means the extensions are an important part of the program. (Singh et al., 2019).

Methodology

In this section, I will outline the methods and the tools I have used to help me achieve my goals in this project which is to build a recommender system with the use of Multi Objective Optimization NSGA-II. The research is using prince2 methodology which is using an agile methodology. The agile methodology develops a prototype and improve it based on user's recommendations and continue doing that until the final product is not ready. In my project I developed a prototype model to recommend the courses. The protype model was developed using features from the dataset used. These features can be changed if required after the initial prototype. That's why the research is using the agile methodology. The user will interact with the model through an API. The API is developed in flask which is a python framework. A web interface is developed in flask that the user will use to interact with the API and the multi-object optimization model. The API can be refined based on the user's feedback that's another reason for using the agile methodology.

Implementation of recommender system

Dataset

The data is a csv file that is collected from the online platform Kaggle about the Coursera. Coursera is an online platform that offers variety of online courses. The data is originally collected from publicly available data of Coursera. The key features of the dataset are course name, university partner, course rating, difficulty of the course, description of the course, skills associated with the course and course URL. The dataset contains 3523 rows of data. The screen shot of the dataset used in shown in figure 1.

Coursera

Course Name	University	Difficulty Level	Course Rating	Course URL	Course Description	Skills
Write A Feature Length Screenplay For	Michigan State University	Beginner	4.8	https://www.coursera.org/learn/	Write a Full Length Feature Film	Drama Comedy peering
Business Strategy: Business Model Ca	Coursera Project Network	Beginner	4.8	https://www.coursera.org/learn/	By the end of this guided project	Finance business plan pe
Silicon Thin Film Solar Cells	cole Polytechnique	Advanced	4.1	https://www.coursera.org/learn/	This course consists of a genera	chemistry physics Solar E
Finance for Managers	IESE Business School	Intermediate	4.8	https://www.coursera.org/learn/	When it comes to numbers, ther	accounts receivable dupo
Retrieve Data using Single-Table SQL C	Coursera Project Network	Beginner	4.6	https://www.coursera.org/learn/	In this course youll learn how to	Data Analysis select (sql)
Building Test Automation Framework u	Coursera Project Network	Beginner	4.7	https://www.coursera.org/learn/	Selenium is one of the most wid	maintenance test case te
Doing Business in China Capstone	The Chinese University of Ho	Advanced	3.3	https://www.coursera.org/learn/	Doing Business in China Capsto	marketing plan Planning
Programming Languages, Part A	University of Washington	Intermediate	4.9	https://www.coursera.org/learn/	This course is an introduction to	inference ml (programmin
The Roles and Responsibilities of Nonp	The State University of New	Intermediate	4.3	https://www.coursera.org/learn/	This course provides a more in-c	Planning Peer Review fun
Business Russian Communication. Par	Saint Petersburg State Unive	Intermediate	Not Calibrated	https://www.coursera.org/learn/	Russian is considered to be one	Russian market (economic
Agile Projects: Developing Tasks with	Coursera Project Network	Beginner	4	https://www.coursera.org/learn/	By the end of this guided project	project modeling Project
Esports Management Capstone Projec	University of California, Irvine	Beginner	4.4	https://www.coursera.org/learn/	We will analyze the pros and cor	Average Planning Leader
Hacking and Patching	University of Colorado Syste	Advanced	3.4	https://www.coursera.org/learn/	In this MOOC, you will learn how	Security Design design pa
Business Statistics and Analysis Capst	Rice University	Beginner	4.7	https://www.coursera.org/learn/	The Business Statistics and Ana	Statistical Analysis Micros
Grab Data Fast with Vertical and Horiz	Coursera Project Network	Beginner	4.5	https://www.coursera.org/learn/	Data can come our way in multip	evaluation software prese
Global Health: An Interdisciplinary Ove	University of Geneva	Beginner	4.6	https://www.coursera.org/learn/	This course proposes an overvie	sustainability research and
Python Programming Essentials	Rice University	Beginner	4.8	https://www.coursera.org/learn/	This course will introduce you to	semantics Python Progran
Creating Dashboards and Storytelling	University of California, Davis	Advanced	4.6	https://www.coursera.org/learn/	Leveraging the visualizations you	neuroscience Data Visualiz
Parallel programming	cole Polytechnique Fdrale de	Beginner	4.4	https://www.coursera.org/learn/	With every smartphone and com	Data Structures parallel al
Recommendation Systems with Tensor	Google Cloud	Advanced	4.2	https://www.coursera.org/learn/	In this course, you'll apply your k	systems architecture Clou
The Changing Arctic	National Research Tomsk St	Not Calibrated	4.3	https://www.coursera.org/learn/	What will I learn? After taking thi	Human Learning Geology
COVID-19 - A clinical update	University of Florida	Beginner	4.7	https://www.coursera.org/learn/	As an expert in infectious diseas	disease mechanical ventil
How to Create Text Effects in GIMP	Coursera Project Network	Beginner	4.7	https://www.coursera.org/learn/	In this 1-hour long project-basec	pointing device gesture r&
Preparing for the Google Cloud Profes	Google Cloud	Beginner	4.5	https://www.coursera.org/learn/	From the course: "The best way	business requirements Clo
Multiple Regression Analysis in Public	Johns Hopkins University	Intermediate	4.6	https://www.coursera.org/learn/	Biostatistics is the application of	Regression Analysis Regre
General Pathophysiology	Saint Petersburg State Unive	Intermediate	4	https://www.coursera.org/learn/	Dear listeners! Warning: this cou	medicine biomedical scien
Population Health: Alternative Payment	Universiteit Leiden	Advanced	4.9	https://www.coursera.org/learn/	The way we currently pay our ca	incentive Behavioral Econ
AWS Elastic Beanstalk: Build & Deploy	Coursera Project Network	Advanced	5	https://www.coursera.org/learn/	In this 1-hour long project-basec	representational state trans
Philosophy, Science and Religion: Philo	The University of Edinburgh	Intermediate	4.6	https://www.coursera.org/learn/	Philosophy, Science and Religio	ordered pair arts and hum
Poverty & Population: How Demograph	Columbia University	Advanced	4.9	https://www.coursera.org/learn/	This course has four modules, o	education Billing political

Figure 1: Dataset used for recommender system

The recommender system is using the feature of skills, difficulty level and rating to recommend the courses to users. User will enter these attributes on front end provided by the API and the recommender system will use the user input to recommend courses to user.

Implementation

The table below is a list of the softwares implemented into this project.

Tools	Description
Visual Studio Code	Visual Studio code is a free and open sourced editor that supports many programming languages including python and it is a very good option for lightweight programs, Visual studio code also supports range of libraries in the environment which allows the programmer to explore and implement them.

Jupyter Notebook	Jupyter notebook is a python editor which has been used for visualization in this project and it is a great tool for data analysis
Programming Languages	Python, HTML, CSS
Flask	API
Post man	Post man is a tool that was used to test the API of the system in the previous versions.

Figure 2: table

Data Cleaning

Data cleaning is the finding and correcting inaccurate data inside the dataset.
Data cleaning is a crucial step of any project working with data as it ensures the data is accurate and reliable to use within the project and it also helps with eliminating duplicates and null data to make a better analysis.

Implementation of recommender system using Pymoo python framework

To implement the multi-objective optimization algorithm, we have used python library Pymoo which is a python framework that can implement the single and Multi-objective Optimization problems.
The research has used multi-objective optimization algorithm for course recommender system. The traditional methods don't consider multiple objectives which are crucial to improve user experience. Multi-objective optimization can balance the multiple objectives (Zaizi et al., 2023). The research has developed a machine learning model for course recommender system in python that is trained and tested on data collected from Kaggle. Pymoo offers visualization and decision making on the data. Pymoo framework offers single objective, multi-objective, many objective and dynamic algorithm implementations. Pymoo offers two multi-objective optimization algorithms NSGA-II and R-NSGA-II. The research is using NSGA-II algorithm. One of the facets of the multi-objective optimization is the defining the optimization problem. In pymoo an optimization problem can be defined using different classes that includes Problem, ElementWiseProblem and FunctionalProblem. It has an

interface to interact with optimization problem in case of minimize function. The _evaluate function of problem class is used to recommend multiple solutions based on the criteria entered.

The following section will discuss the problem class, algorithm, evaluate and minimized function used to solve the course recommender system problem using multi-object optimization algorithm NSGA-II.

Problem Class

To solve an optimization in pymoo a problem is defined that problem class contains some metadata about the prob. This meta includes the number of variables which is the number of records in our coursera dataset. Metadata also states the number of objectives which in course recommender system API were three originally but later we changed it to just 1. Metadata also contains number of equality constraints and number of inequality constrains along with upper bound and lower bound of our design space. The figure shows the list of metadata value that can be used to define the problem.

Argument	Description
`n_var`	Integer value representing the number of design variables.
`n_obj`	Integer value representing the number of objectives.
`n_constr`	Integer value representing the number of constraints.
`xl`	Float or `np.ndarray` of length `n_var` representing the lower bounds of the design variables.
`xu`	Float or `np.ndarray` of length `n_var` representing the upper bounds of the design variables.
`vtype`	(optional) A type hint for the user what variable should be optimized.

Types of Output

Figure 3: metadata of problem class in pymoo (pymoo, 2020)

This project has used ElementWiseProblem class to define the optimization problem.

The optimization problem of course recommendation defined using the ElementWiseProblem is given below:

```
class CourseRecommendationProblem(ElementwiseProblem):
```

```
    def __init__(self, skills, rating, difficulty):
        self.skills = skills
        self.rating = rating
        self.difficulty = difficulty
        xl = np.full(len(courses_data), 0.0)
        xu = np.full(len(courses_data), 1.0)

        super().__init__(
            n_var=len(courses_data), n_obj=1, n_constr=0, xl=xl, xu =xu)
    def _evaluate(self, x, out, *args, **kwargs):
        # distances = np.array([[np.linalg.norm(np.array([x[i][0], x[i][1]]) -
np.array([courses_data[course]["rating"], courses_data[course]["difficulty"]])) for i in
range(len(x))] for course in courses_data])
        # out["F"] = np.column_stack([np.min(distances, axis=0), np.mean(distances,
axis=0), np.max(distances, axis=0), cosine_similarity_function(x, keyword)])

        out["F"] = cosine_similarity_function(x, self.skills)# Define the course
recommendation problem instance
```

Figure 4: code snippet of the CourseRecommendationProblem Class

The xl and xu are the lower and upper bounds NumPy array respectively which are the range of different variables. The class CourseRecommendation problem is inheriting from the ElementWiseProblem class therefore the parent constructor is called to pass the Coursera data and the number of objectives, constraints along with upper bound and lower bound.

_evaluate Function

Recommender system will use the _evaluate method for recommendations. _evaluate is an overridden function and it used to optimize the problem define in problem class using the user objectives which in our case are the skills, rating, and difficulty level of the course that the user is looking for. Initially we tried use all the objectives of the user for recommendation but because code issues we must just consider one objective at the end and that was the skills that user selected. The _evaluate function is using the cosine similarity to find the similarity between all the solutions one by one with skills that the user selected. The similarity is then passed to the _evaluate function.

```
#_evaluate is a overridden funciton
    def _evaluate(self, x, out, *args, **kwargs):
        #distances = np.array([[np.linalg.norm(np.array([x[i][0], x[i][1]]) -
np.array([courses_data[course]["rating"], courses_data[course]["difficulty"]]))
for i in range(len(x))] for course in courses_data])
        #out["F"] = np.column_stack([np.min(distances, axis=0),
np.mean(distances, axis=0), np.max(distances, axis=0),
cosine_similarity_function(x, keyword)])

        out["F"] = cosine_similarity_function(x, self.skills)# Define the course
recommendation problem instance
```

Figure 5: _evaluate function of problem class

As you can see the x parameter of the _evalutae function is taking the similarity of value generated by cosine similarity and generate the out in the parameter out and return to output function F.

_evaluate function will go through each solution and use the similarity of skill and _evaluate will always return a vector of length 10. This _evaluate function is called for each recommended solution. The output F is obtained using the cosine similarity between x and skills. Although initially I used liner algebra function to find the similarity between all the selected objectives of the user. It was showing some errors and I had leave two of the objectives (Rating and Difficulty level). Eventually I just used the one objective which is the Skills. The lines commented in the code snippet above are trying to find the similarity between multiple objectives.

Parameters of _evaluate function

_evaluate overridden function takes x which is the numpy array of two dimensions that contains the solutions and n_variables which are the number of objectives. _evaluate calculates the evaluations and result is then in out parameter. All the optimization algorithms store the output in a dictionary called out. In figure above the out["F"] is a dictionary where the F is the key that stores the final objective value after all the evaluation are performed.

Cosine similarity

Cosine similarity is used to find the similarity between two vectors by measuring the angle between the vectors and then taking cosine of that angle. That gives a value that then can be used to find the similarity between the vectors.

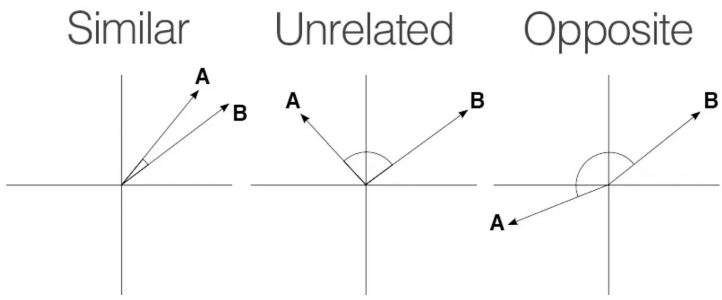


Figure 5: Cosine similarity figure(Manu, 2024).

We can also use this cosine similarity to find the similarity between the other values including the two strings. In course recommender system cosine similarity use the skills and the dataset to find the similarity between the skills objective and the Coursera dataset. The higher the value the higher the similarity. The value returned is then used for recommending the courses to the user. _evaluate function of problem class is called on each solution hundred times and each time it used cosine similarity to find the similarity between skills selected by the user and the Coursera dataset.

Algorithm used for optimization problem of course Recommender system

The research is using NSGA-II algorithm (non-dominated sorting genetic algorithm). To use this algorithm in optimization problem an instance of it is passed to minimize function. As you can in figure 3 an instance of it is created with pop_size 100 that will be passed to minimize function. To initialize it we will use the following code.

```
from pymoo.algorithms.moo.nsga2 import NSGA2
algorithm = NSGA2(pop_size=100)
```

Figure 6: code snippet of the NSGA-II library import

Minimize Function

The minimize function is the interface to perform any optimization on a problem. So, for course recommendation the user will interact with this function for recommendation of the courses. The minimize function's signature is:

```
Def
minimize(problem,algorithm,termination=None,seed=None,verbose=False,display=None,callback=None,

return_least_infeasible=False,save_history=False)
```

Figure 7: code snippet of minimize function from pymoo website

It takes the problem defined for the optimization algorithm and some other values which are optional. The Coursera recommender system is passing the problem which is the object of the CourseRecommedenationProblem Class and NSGA-II algorithm, seed, save history and verbose.

```
    algorithm = NSGA2(pop_size=100)
    problem = CourseRecommendationProblem(skills, rating, difficulty)
    res = minimize(problem,
                   algorithm,
                   ('n_gen', 100),
                   seed=1,
                   verbose=True,
                   save_history=True,)
```

Figure 8: code snippet of minimize function inside the project

Getting user inputs through flask API

The get_recommender_course function gets the user request through the flask API and call the recommender system function for course recommendations. It then sends the response of the recommender system to flask API.

```
def get_recommended_course(skills, rating, difficulty):
    algorithm = NSGA2(pop_size=100)
    problem = CourseRecommendationProblem(skills, rating, difficulty)
    res = minimize(problem,
```

```
                algorithm,
                ('n_gen', 100),
                seed=1,
                verbose=True,
                save_history=True,)
    subject_requirements = [skills, rating, difficulty]  # Example requirements
(skills, rating, difficulty level)
    if res.X.ndim>1:
        a = res.X[0]
    else:
        a = res.X
    print(res.X)
    ans = sorted(range(len(a)), key=lambda i: a[i],reverse=True)[:5]
    recommend_courses = []
    for i,x in enumerate(new_dict):
        if i in ans:
            recommend_courses.append(x)
    return recommend_courses
```

Figure 9:

Flask API

APIs are used to create an interaction between two end points. In our case the user should interact with the recommender system for course recommendations. Flask is used to develop an API for the recommender system because it is easy to use and extensible. the API is a modern, high performant web framework for building APIs with python 3.8+. Flask is one of the most used frameworks to develop REST and RESTFUL APIs that can request HTTP requests. Flask is also one of the fastest python frameworks available for developing APIs which is quite easy to use. Flask has many built-in functions and templates that can be useful for developing the API's. An interface is developed using HTML, CSS and Jinja template of flask for user to interact with the recommender system by sending requests and get responses. The request is sent by user through the user interface will be passed to flask API. Flask API will then use this input and interact with recommender system (multi-object optimization model) for course recommendation. The multi-object optimization model will use the input to recommend the courses and send it back to flask which will then send it to user inter interface in browser, where it will be displayed using the jinja template of flask. The Key features of Flask API include enabling creation of endpoints of handling recommendation requests and

response. These features make it a great tool to choose in creating a recommender system API.

At the beginning I wanted to use Fast API butt after more research I have found Flask to be easier to implement and more resources available about it.

Flask API to interact between the user and the recommender system

The recommend_courses(I) function will be called when the user will click on the send button. This fucniton is working as an intermediatory between user and the recommender system. It receives user requests and forward them to recommender system by calling the function get_recommended_courses(). After receiving the response from the recommender system, it send it to the user which is then displayer in browser.

```
#Define the route for the landing page of flask app@app.route('/',
methods=['POST', 'GET'])

def index():

    return render_template('index.html')

# Define the Flask route for recommending courses based on keywords
@app.route('/recommend_courses/',  methods=['POST'])
def recommend_courses():
    skills = request.form['skills']
    rating = float(request.form['rating'])
    difficulty = float(request.form['difficulty'])
    recommended_courses = get_recommended_course(skills, rating, difficulty)
    return render_template('index.html',taskslist=recommended_courses)
```

Figure 10: Flask function receive the request and send the response of the recommender system to user

User Interface to interact with recommender system

The user interface is developed using HTML, CSS and jinja template of flask. HTML will create the contents such as labels, input boxes, headers and buttons etc. CSS will give styles to the input boxes and Jinja template will display the response from the API to the use.

```html
<html>
  <head>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{url_for('static',
filename='style.css' )}}"
    />
  </head>
  <body>
    <div id="mainDiv">
      <h1>Course Recommender System</h1>
      <form
action="{{url_for('recommend_courses')}}"
method="post">
        <label for=""> Skills </label>
        <input type="text" name="skills"
/> <br />
        <label for=""> Rating </label>
        <input type="text" name="rating"
/> <br />
        <label for=""> Difficulty Level
</label>
        <input type="text"
name="difficulty" /> <br />

        <button class="btnn" value="Send"
type="submit">Send</button>
      </form>
    </div>

  </body>
</html>
#this class been inspired from
freeCodeCamp.org. (2022). How to Center a
Div
with CSS. [online] Available at:
https://www.freecodecamp.org/news/how-to-
center-a-div-with-css/ [Accessed 21st
Apr. 2024].
```

```css
#mainDiv {
  width: 500px;
  height: 50px;
  margin: 100px;
  outline: solid 1px
black;
  text-align: center;
}

#resultDiv {
  width: 500px;
  margin: 650px;
  height: 500px;
  margin-top: 0px;
  outline: solid 1px
black;
  text-align: center;
}

form {
  border: 3px solid
#f1f1f1;
}

/* Full-width inputs */
input[type="text"],
input[type="password"] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid
#ccc;
  box-sizing: border-
box;
}


button {
  background-color:
#04aa6d;
  color: white;
  padding: 14px 20px;
```

```html
<div id="resultDiv">
      <h1>the
recommended courses
are:</h1>
      <p>{% if
taskslist %}
{{taskslist}} {%
endif %}</p>

</div>
```

```
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}
```

Figure 11: Code snippet of user interface

Visualization of the performance of multi-objective optimization

The performance of the algorithm is stored in out["F"] which is all values in the set of defined data and out["X"] which is the ideal values after the optimization. We can display the performance by print the outputs of F and X.

```
# Get the results
best_solution = res.X
best_fitness = res.F

# Print the best solution and fitness
print("Best Solution:")
print(best_solution)
print("Best Fitness:")
print(best_fitness)
```

Figure 12: code snippet of performance visualisation

Pareto of the multi-objective optimization algorithm

Pymoo has a built-in function to draw a scatter graph which is mainly used for low dimension objectives. The Scatter 3D representation of the three objectives/features is drawn as part of the research.

```
# Visualize the Pareto front
plot = Scatter(title="Pareto Front", labels=["Skills", "Rating", "Difficulty Level"])
plot.add(res.F)
plot.show()
```

Figure 13: creating and visualising the 3d scatter plot (pareto)

Recommender system API Architecture

5. Results

Recommender System API Architecture:

The course recommender system is developed and producing results. The user interface of the API is index.html that have form where the user will enter the course requirements and send

the the request to the user. This index.html file is the landing page of the recommender system which means it is loaded when the API runs. When the user sends a request for the recommendations from the index.html, the request goes to API function recommend_course(), that then interact with optimization algorithm for recommendations and sends the recommendations back to the user on index. Html webpage. The recommender system API's architecture is shown in figure below.

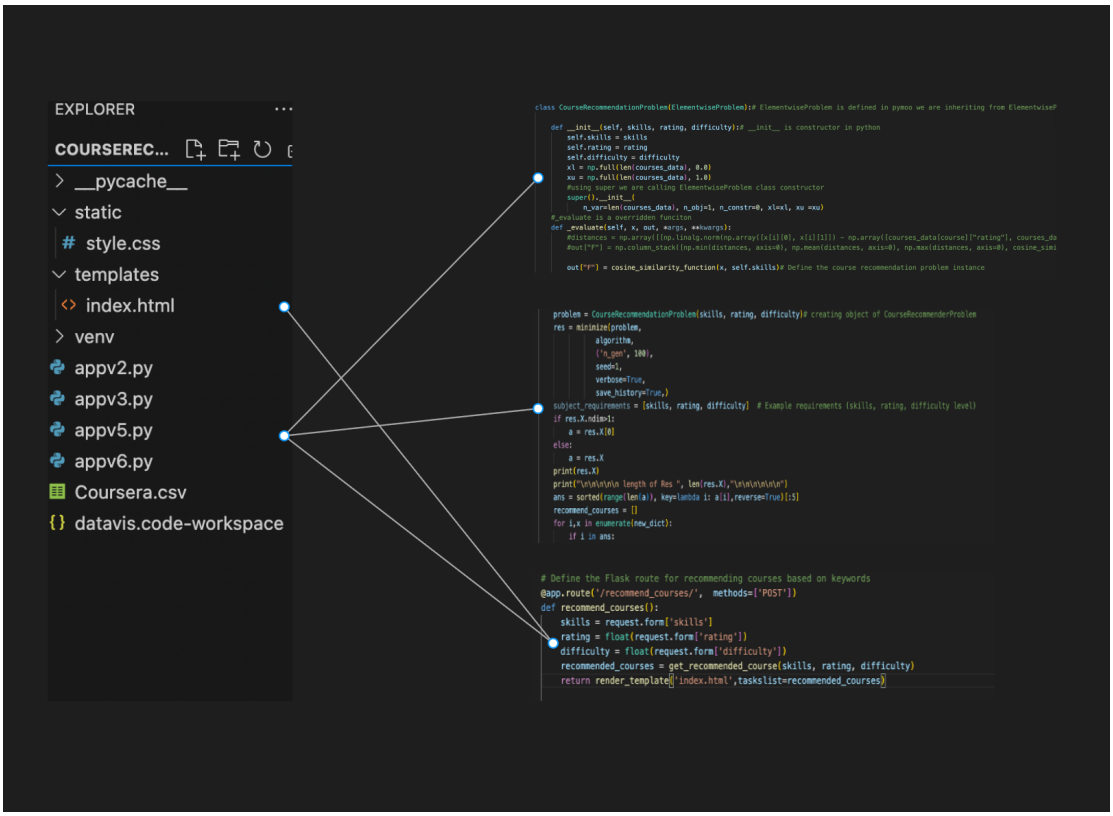


Figure 14: Recommender System API architecture

The API is developed is implemented in VS Code IDE. The user will enter the command in terminal of VS Code inside virtual environment with all the libraries installed:
flask -–app appv5 run

The API will run on localhost 127.0.0.1 on port 8080. When the API runs, the user will click on that localhost link to open it browser. The API works fine on all major web browser.

Original implementations and changes
The original system that was implemented would show a result like the figure provided below:

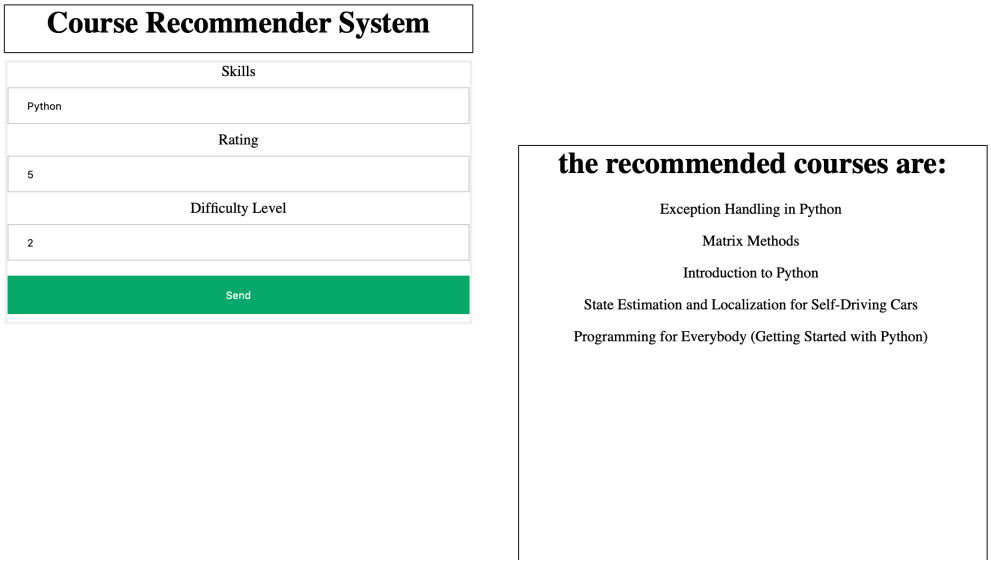


Figure 15 : Screenshot of user interface in the original submission

In the original submission the output we were getting was just the course names and the user was not able to access more details about the courses

Implementation

Recommendations of API with ElementWiseProblem Class

The course recommender system API has graphical user interface developed using HTML, CSS, and flask template jinja. The index.html file is the developed using these web technologies. The GUI of recommender system API looks like this:

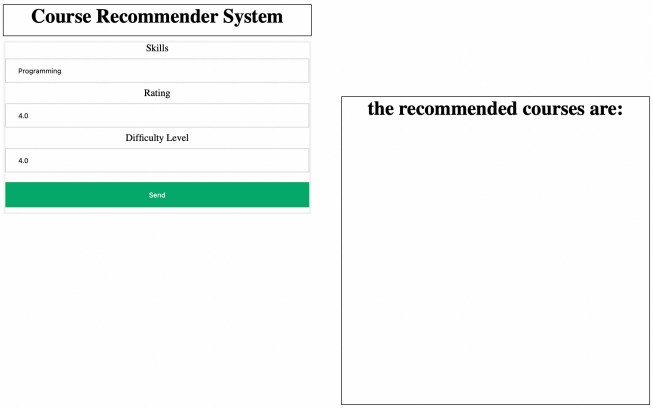


Figure 16: screenshot of user interface (send button not pressed)

The user will enter the criteria for the course based on skills, rating and difficulty and click on the send button. The request goes to flask API that connects to the pymoo model and get the recommendation which are then returned and displayed to user in GUI of recommender system. The precision of the result is calculated. The precision is calculated using number of correct recommendation and the total number or recommendations. It will be calculated by dividing the correct recommendation by total recommendations. Let K be the total number of recommendations and it is five as the API recommends maximum five recommendation.
The API is tested for different skills, rating and difficulty level and Test results are given below:

```
course_skills = [data["Skills"] for
data in courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_skills =
tfidf_vectorizer.fit_transform(course
_skills)
```

```
course_descriptions = [data["Course
Description"] for data in
courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix =
tfidf_vectorizer.fit_transform(course_desc
riptions)
```

Figure 17: code snippet showing the transformation of skills and course description columns of dataset into vectors using the label encoder

In figure17 both of this code is inside the application and by default the code is set to the left hand side which is vectorizing for the Skills and looks for keywords inside the skills column, by commenting this part and uncommenting the left hand side we can look for the similar keyword inputted by the user in course description (Keep in mind that in appv5 which is the previous version and it has been submitted on 23rd, it is only using the Course description(the right side of the table) and course skill has been added in the code in version appv6.

Case 1:

skill(Python), rating(5.0) and difficulty level(2) is given below:

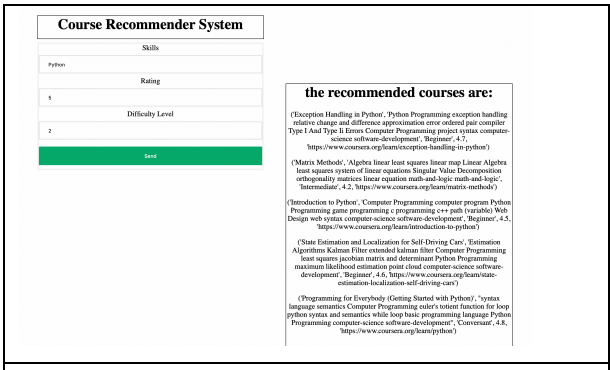


Figure18 : Screenshot of user Interface using Course Description

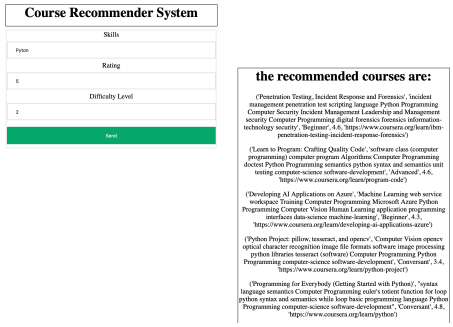


Figure19 : Screenshot of user Interface using Skills

Figure 18:

precision = 4/5 = 0.8

For recommendation of Python we can see 4 out of 5 results are related, the second result printed is not relevant to python, as it is 'showing a Matrix Methods' Course related to Mathematics.

Figure 19:

precision = 5/5 = 1.0

We can observe a higher rate of precision as all of the courses recommended are relevant in this example of a single word search.

Case 2:

recommendation of skill (Python, Neural Networks, Machine Learning), rating(5.0) and difficulty level(2) is given below:

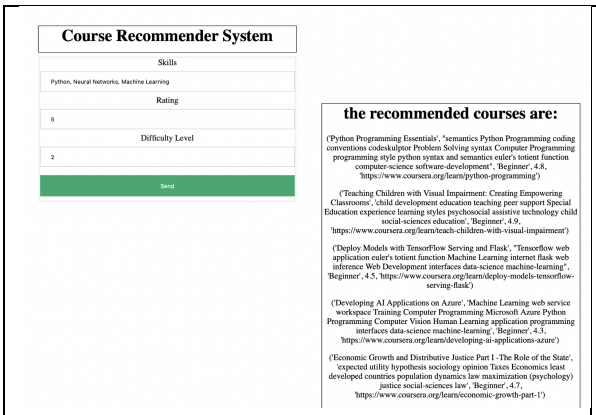


Figure20: Screenshot of user Interface using Course Description

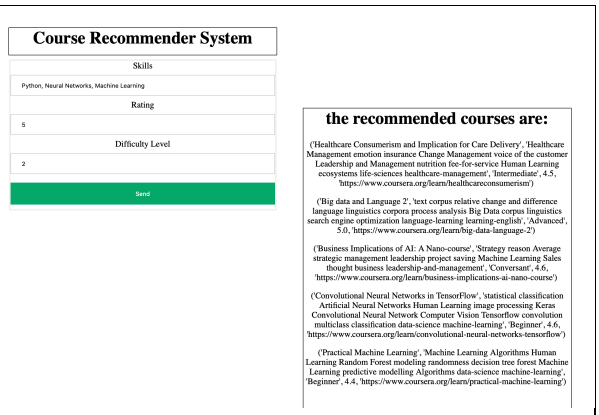


Figure21 : Screenshot of user Interface using Skills

Figure 20 :

precision = 3/5 = 0.6

In this case we are using multiple keywords to search instead of one skill
For recommendation of Python, Neural Networks, Machine Learning we can see 3 out of 5 results are related, the second and fifth results printed are not relevant to the search.
The system is recommending "Teaching Children with Visual Impairment: creating empowering classrooms" as we assume it contains Learning in its Description which can explain why it was recommended by the system as it is mistaking it for the search of Machine Learning.

Figure 21 :

precision = 4/5 = 0.8

in this case we can see the first result that is given to us by the system is '(Healthcare Consumerism and implications for Care Delivery)' we can understand why the system has given us a wrong predication as it is reading the 'Human Learning' keyword from skills section.
Overall, the recommendations case, shows a higher accuracy in searching the keywords in Skills.

Case 3:

In this case we will observe another example of multi-inputs recommendation of skill (Python, Neural Networks, Programming), rating(5.0) and difficulty level(2) is given below:

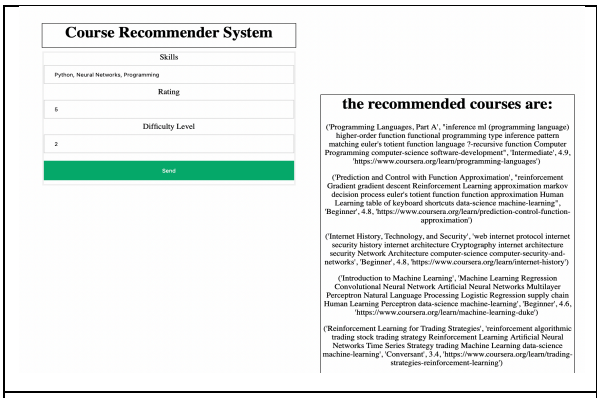


Figure22: Screenshot of user Interface using Course Description

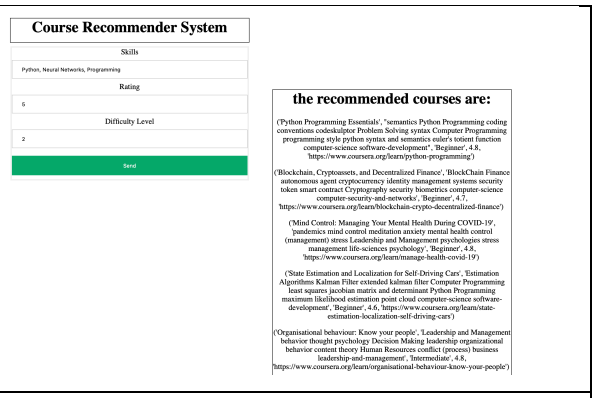


Figure23 : Screenshot of user Interface using Skills

Figure 22:

precision = 3/5 = 0.6

In case 3 we are testing the system with inputs Python, Neural Networks and Programming. In courses recommended by the system we can notice that the system has taken Keyword 'Networks' from Neural Networks and has returned a course related to security networks ('Internet History, Technology and Security') but, we can see the system correctly has identified Neural Networks in the last recommended course ('Reinforcement Learning for

Trading Strategies'). We can also Identify an incorrect course recommended In field of Machine learning which it was not in our Input.

Figure 23:

Precision = 3/5 = 0.6

In this case using Skills we can see that the precision has stayed the same but as we expected courses are slightly different, we have been recommended 2 Psychology Courses by the system ('Mind Control: Managing your Mental Health During Covid 1') and ('Organisational behaviour :Know your people') which are irrelevant. by further analysing the recommended courses we can see the system has recommended Python Programming Essential, which I believe it is a very good match for this search as it contains keywords python and programming.

Case 4:

recommendation of skill (Neural Networks), rating(5.0) and difficulty level(2) is given below:

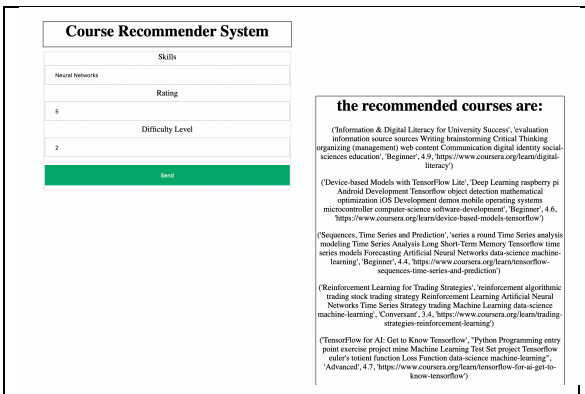


Figure24 : Screenshot of user Interface using Course Description

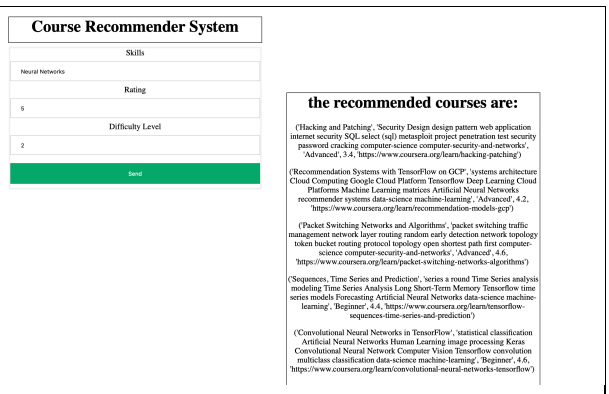


Figure25 : Screenshot of user Interface using Skills

Figure 24:

Precision = 4/5 = 0.8

In this result we can observe 4/5 recommended courses are related to computing and Neural Networks but, in the first result ('Information & Digital literacy for University Success') is an irrelevant course which is related to digital literacy.

Figure 25:

Precision = 3/5 = 0.6

We can see that the system is also picking up on courses containing the single keyword Networks but the last three recommendations are valid and relevant.

Case 5:

We will now see if writing style of the input will affect our results

recommendation of skill (Neural Network), rating(5.0) and difficulty level(2) is given below:

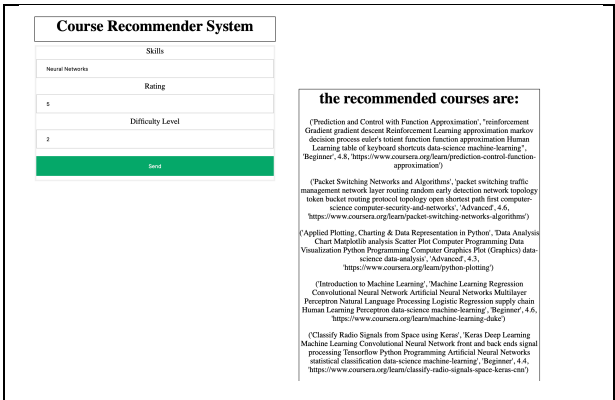


Figure 26: Screenshot of user Interface using Course Description

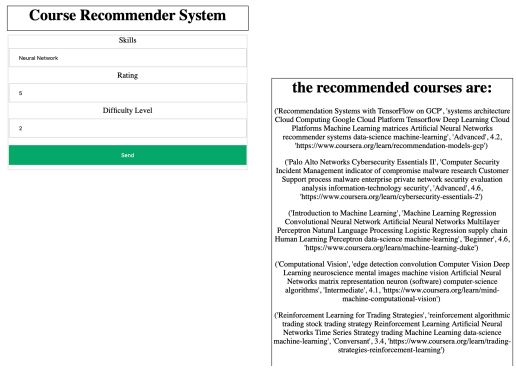


Figure 27: Screenshot of user Interface using Skills

Figure 26:

Precision = 4/5 = 0.8

In figure 18 we can see

All of the recommended courses are changed compared to our previous input Computer Networks but have the same accuracy as figure 16 with the same problem of detecting computer network instead neural network.

Figure 27:

Precision = 4/5 = 0.8

We are getting different recommendations compared to the previous input and we are getting a slightly better result of 4/5 relevant courses recommended. If we look closely in the skills displayed in the figure, we can see it is still picking the results with neural networks (with an extra s at the end).

Case 6:

recommendation of skill (Software Engineering), rating(5.0) and difficulty level(2) is given below:

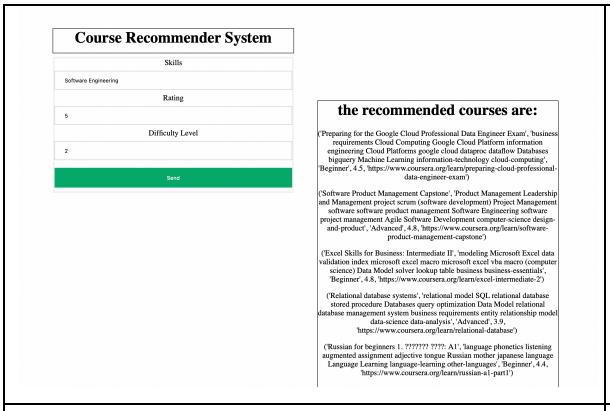


Figure 28: Screenshot of user Interface using Course Description

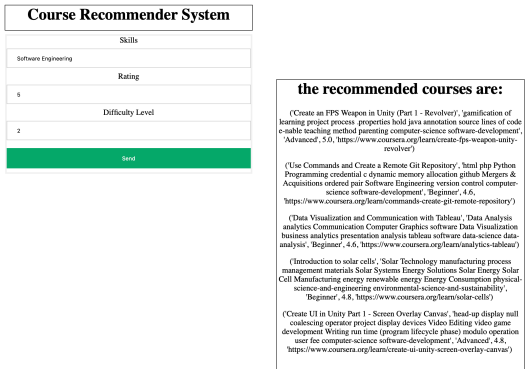


Figure 29: Screenshot of user Interface using Skills

Figure 28: screenshot of user interface

Precision = 3/5 = 0.6

This shows us that the system is keeping 3 of the results accurate as it is recommending the last 2 courses are irrelevant to software development, we can argue that (‘relational database system’) can be in the category of software engineering but, (‘Russian for Beginner’) is defiantly not a software engineering course, although there might be two reasons behind this recommendation as one of them can be the keyword of Language Learning that the system might have incorrectly related it to Software engineering .

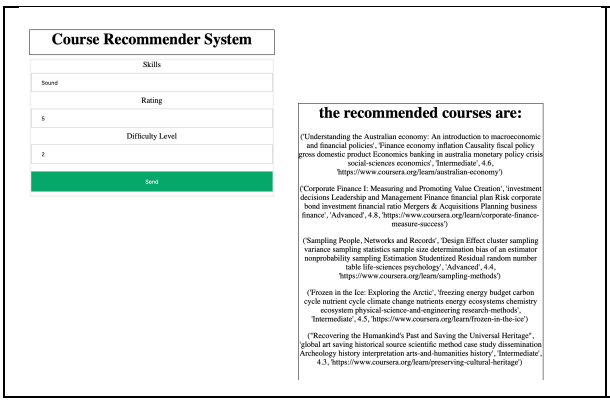
Figure 29: screenshot of user interface

Precision = 4/5 = 0.8

We are seeing a better result in this figure the only irrelevant course that we can see is (‘introduction to solar cells’) which can be chosen by the system because of the keyword engineering.

Case 7:

recommendation of skill (Sound Design), rating(5.0) and difficulty level(2) is given below:



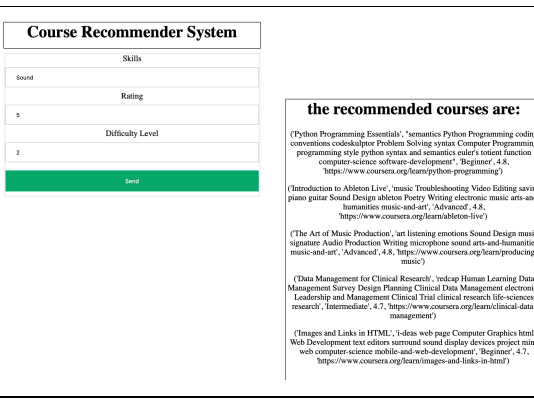


Figure 30: Screenshot of user Interface using Course Description	Figure 31: Screenshot of user Interface using Skills

Figure 30:

Precision = 0/5 = 0.0

If we wanted to look for a course related to either sound engineer or sound design, by inputting Sound in the skills we can observe that it does not recommend any relevant courses as I it shows the courses which have used sound in their description for example in ('Understanding the Australian economy :An introduction to macroeconomic and financial policies') it mentions 'Discussions will be centered around sound economic theories and reasoning' and in ('Corporate Finance I: Measuring and Promoting Value Creation') by looking inside its description we can see that it is stating 'You will also learn how to use valuation techniques to make sound business investment and acquisition decisions.' Which again has sound inside it but has no relevance to the course we are looking for.

Figure 31:

Precision = 2/5 = 0.4

With the same concept in mind, we can see the result using Skills is more accurate and it is recommending two relevant courses ('Introduction to Ableton live') and ('The art of music production'). Although in the last example we can see the word sound is used but it is not in a music context, so it is not suitable for this recommendation.

Keep in mind that the actual output screen does not contain values inside the input boxes, the values above in the screenshots are manually entered again for the purpose of analysis.

Output of the pymoo interface or minimize function

The minimize function works as an interface between flask API and optimization problem. The minimize function use the problem class object and the algorithm which is NSGA2. The problem class function _evaluate goes through each solution 100 times to get recommendation. The verbose parameter of the minimize function is set to True so that we can visualize the evolutionally process of the multi-object optimization algorithm of course

recommender system. The convergence process of the minimize and _evaluate function is shown below in figure. Each column of the table represents some aspects of the evolutionalry process of the optimization of course recommender system.

```
127.0.0.1 - - [26/Apr/2024 12:14:55] "GET /static/style.css HTTP/1.1" 304 -
=================================================================
n_gen  |  n_eval  | n_nds  |      eps      |  indicator
=================================================================
    1 |      100 |      1 |             - |          -
    2 |      200 |      1 |  0.0222524238 |      ideal
    3 |      300 |      1 |  0.0557163186 |      ideal
    4 |      400 |      1 |  0.1381953204 |      ideal
    5 |      500 |      1 |  0.1849396837 |      ideal
    6 |      600 |      1 |  0.0391339637 |      ideal
    7 |      700 |      1 |  0.1040198422 |      ideal
    8 |      800 |      1 |  0.0331744392 |      ideal
    9 |      900 |      1 |  0.1186878721 |      ideal
   10 |     1000 |      1 |  0.0933786015 |      ideal
   11 |     1100 |      1 |  0.0143444019 |      ideal
   12 |     1200 |      1 |  0.1041908633 |      ideal
   13 |     1300 |      1 |  0.0216649061 |      ideal
   14 |     1400 |      1 |  0.0295001786 |      ideal
   15 |     1500 |      1 |  0.0732510427 |      ideal
   16 |     1600 |      1 |  0.0229263132 |      ideal
   17 |     1700 |      1 |  0.0773561326 |      ideal
   18 |     1800 |      1 |  0.0277430185 |      ideal
   19 |     1900 |      1 |  0.0401789350 |      ideal
   20 |     2000 |      1 |  0.0304149773 |      ideal
   21 |     2100 |      1 |  0.0286961020 |      ideal
   22 |     2200 |      1 |  0.0111315829 |      ideal
   23 |     2300 |      1 |  0.0332419719 |      ideal
   24 |     2400 |      1 |  0.0353652540 |      ideal
   25 |     2500 |      1 |  0.0150088420 |      ideal
   26 |     2600 |      1 |  0.0337014765 |      ideal
   27 |     2700 |      2 |  0.000000E+00 |          f
   28 |     2800 |      1 |  0.0071155932 |      ideal
   29 |     2900 |      1 |  0.0110597330 |      ideal
   30 |     3000 |      1 |  0.0139567890 |      ideal
   31 |     3100 |      1 |  0.000000E+00 |          f
   32 |     3200 |      1 |  0.0080267351 |      ideal
   33 |     3300 |      1 |  0.0124382837 |      ideal
   34 |     3400 |      1 |  0.0024967648 |          f
   35 |     3500 |      1 |  0.0083199651 |      ideal
   36 |     3600 |      1 |  0.0123001940 |      ideal
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

   61 |     6100 |      1 |  0.0000687453 |          f
   62 |     6200 |      1 |  0.0077000556 |      ideal
   63 |     6300 |      1 |  0.0026151361 |      ideal
   64 |     6400 |      1 |  0.0011369694 |          f
   65 |     6500 |      1 |  0.0078177698 |      ideal
   66 |     6600 |      1 |  0.0037463823 |      ideal
   67 |     6700 |      1 |  0.0030591665 |      ideal
   68 |     6800 |      1 |  0.0004323594 |          f
   69 |     6900 |      1 |  0.0018116085 |          f
   70 |     7000 |      1 |  0.0066086087 |      ideal
   71 |     7100 |      1 |  0.0014874905 |          f
   72 |     7200 |      1 |  0.0023486325 |          f
   73 |     7300 |      1 |  0.0100526938 |      ideal
   74 |     7400 |      1 |  0.000000E+00 |          f
   75 |     7500 |      1 |  0.0008840850 |          f
   76 |     7600 |      1 |  0.0117188713 |      ideal
   77 |     7700 |      1 |  0.0015252681 |          f
   78 |     7800 |      1 |  0.0093156946 |      ideal
   79 |     7900 |      1 |  0.0003355859 |          f
   80 |     8000 |      1 |  0.0029480596 |      ideal
   81 |     8100 |      1 |  0.0055087825 |      ideal
   82 |     8200 |      1 |  0.000000E+00 |          f
   83 |     8300 |      1 |  0.0008628360 |          f
   84 |     8400 |      1 |  0.0058843734 |      ideal
   85 |     8500 |      1 |  0.000000E+00 |          f
   86 |     8600 |      1 |  0.0000526374 |          f
   87 |     8700 |      1 |  0.0015790799 |          f
   88 |     8800 |      2 |  0.0015790799 |          f
   89 |     8900 |      1 |  0.0029859142 |      ideal
   90 |     9000 |      1 |  0.0020557729 |          f
   91 |     9100 |      1 |  0.0033183136 |      ideal
   92 |     9200 |      1 |  0.0006484717 |          f
   93 |     9300 |      1 |  0.0028483649 |      ideal
   94 |     9400 |      1 |  0.000000E+00 |          f
   95 |     9500 |      1 |  0.0019909498 |          f
   96 |     9600 |      1 |  0.0038426863 |      ideal
   97 |     9700 |      1 |  0.0029313531 |      ideal
   98 |     9800 |      1 |  0.0012599533 |          f
   99 |     9900 |      1 |  0.0036887206 |      ideal
  100 |    10000 |      1 |  0.0015215770 |          f
```

Figure 32: screenshot of the terminal after user clicks on send button

In NSGA algorithm the third parameter is (‘n_gen’, 100) tupple is the number of iterations of multi-objective course recommender system algorithms. It is used to find new solutions from the current population by using techniques such as mutation, crossover, and selection. we set n-gen to 100 because that is the standard value of this variable. It calculates the total nuber of evaluations from all the generations. In this case there 100 evaluations for each generation so the sum of all the evaluations is 100 X 100 which 10000. The column n_nds (non-dominated solutions) shows the number of solutions for each population. In eps that is the epsilon metrics which is measuring each solutions proximity compared to the pareto front, word ideal inside indicator means it is a pass in the optimization and it is considered a candidate, on the other hand F means Fail and not suitable for this search.

Inputting a skill that does not match

by inputting

```
127.0.0.1 - - [30/Apr/2024 02:52:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2024 02:52:39] "GET /static/style.css HTTP/1.1" 304 -
=========================================================
n_gen  |  n_eval  | n_nds  |      eps       |  indicator
=========================================================
     1 |      100 |    100 |              - |          -
     2 |      200 |    100 |  0.000000E+00  |          f
     3 |      300 |    100 |  0.000000E+00  |          f
     4 |      400 |    100 |  0.000000E+00  |          f
     5 |      500 |    100 |  0.000000E+00  |          f
     6 |      600 |    100 |  0.000000E+00  |          f
     7 |      700 |    100 |  0.000000E+00  |          f
     8 |      800 |    100 |  0.000000E+00  |          f
     9 |      900 |    100 |  0.000000E+00  |          f
    10 |     1000 |    100 |  0.000000E+00  |          f
    11 |     1100 |    100 |  0.000000E+00  |          f
    12 |     1200 |    100 |  0.000000E+00  |          f
    13 |     1300 |    100 |  0.000000E+00  |          f
    14 |     1400 |    100 |  0.000000E+00  |          f
    15 |     1500 |    100 |  0.000000E+00  |          f
    16 |     1600 |    100 |  0.000000E+00  |          f
    17 |     1700 |    100 |  0.000000E+00  |          f
    18 |     1800 |    100 |  0.000000E+00  |          f
    19 |     1900 |    100 |  0.000000E+00  |          f
    20 |     2000 |    100 |  0.000000E+00  |          f
    21 |     2100 |    100 |  0.000000E+00  |          f
    22 |     2200 |    100 |  0.000000E+00  |          f
    23 |     2300 |    100 |  0.000000E+00  |          f
    24 |     2400 |    100 |  0.000000E+00  |          f
    25 |     2500 |    100 |  0.000000E+00  |          f
    26 |     2600 |    100 |  0.000000E+00  |          f
    27 |     2700 |    100 |  0.000000E+00  |          f
    28 |     2800 |    100 |  0.000000E+00  |          f
    29 |     2900 |    100 |  0.000000E+00  |          f
    30 |     3000 |    100 |  0.000000E+00  |          f
    31 |     3100 |    100 |  0.000000E+00  |          f
    32 |     3200 |    100 |  0.000000E+00  |          f
    33 |     3300 |    100 |  0.000000E+00  |          f
    34 |     3400 |    100 |  0.000000E+00  |          f
    35 |     3500 |    100 |  0.000000E+00  |          f
    36 |     3600 |    100 |  0.000000E+00  |          f
    37 |     3700 |    100 |  0.000000E+00  |          f
    38 |     3800 |    100 |  0.000000E+00  |          f
    39 |     3900 |    100 |  0.000000E+00  |          f
    40 |     4000 |    100 |  0.000000E+00  |          f
    41 |     4100 |    100 |  0.000000E+00  |          f
    42 |     4200 |    100 |  0.000000E+00  |          f
    43 |     4300 |    100 |  0.000000E+00  |          f
    44 |     4400 |    100 |  0.000000E+00  |          f
    45 |     4500 |    100 |  0.000000E+00  |          f
    46 |     4600 |    100 |  0.000000E+00  |          f
    47 |     4700 |    100 |  0.000000E+00  |          f
    48 |     4800 |    100 |  0.000000E+00  |          f
    49 |     4900 |    100 |  0.000000E+00  |          f
```

Figure 33: screenshot of output terminal

Figure 33 highlights the result of user inputting something that is not in the dataset for example ‘kajnc’ which is a meaningless integer.

We will still get recommendations but all of them are irrelevant

Visualization using Scatter

The API was initially developed using the problem class of the pymoo. But because when used with real data from the coursera, it was not giving the accurate results. Using Problem class therefore we have used another class from pymoo which was ElementwiseProblem class for optimization of course recommender system. The visualization of the optimization problem is done using implantation of Problem class. The visualization is Preto front which is using the built-in function of pymoo. It shows the objectives that the user selected and their relevance in optimization of the course recommendation problem. In Pareto front the most important factors are displayed at front using a bar graph and then a line graph is drawn from the features or the most important factor to the course recommendation.

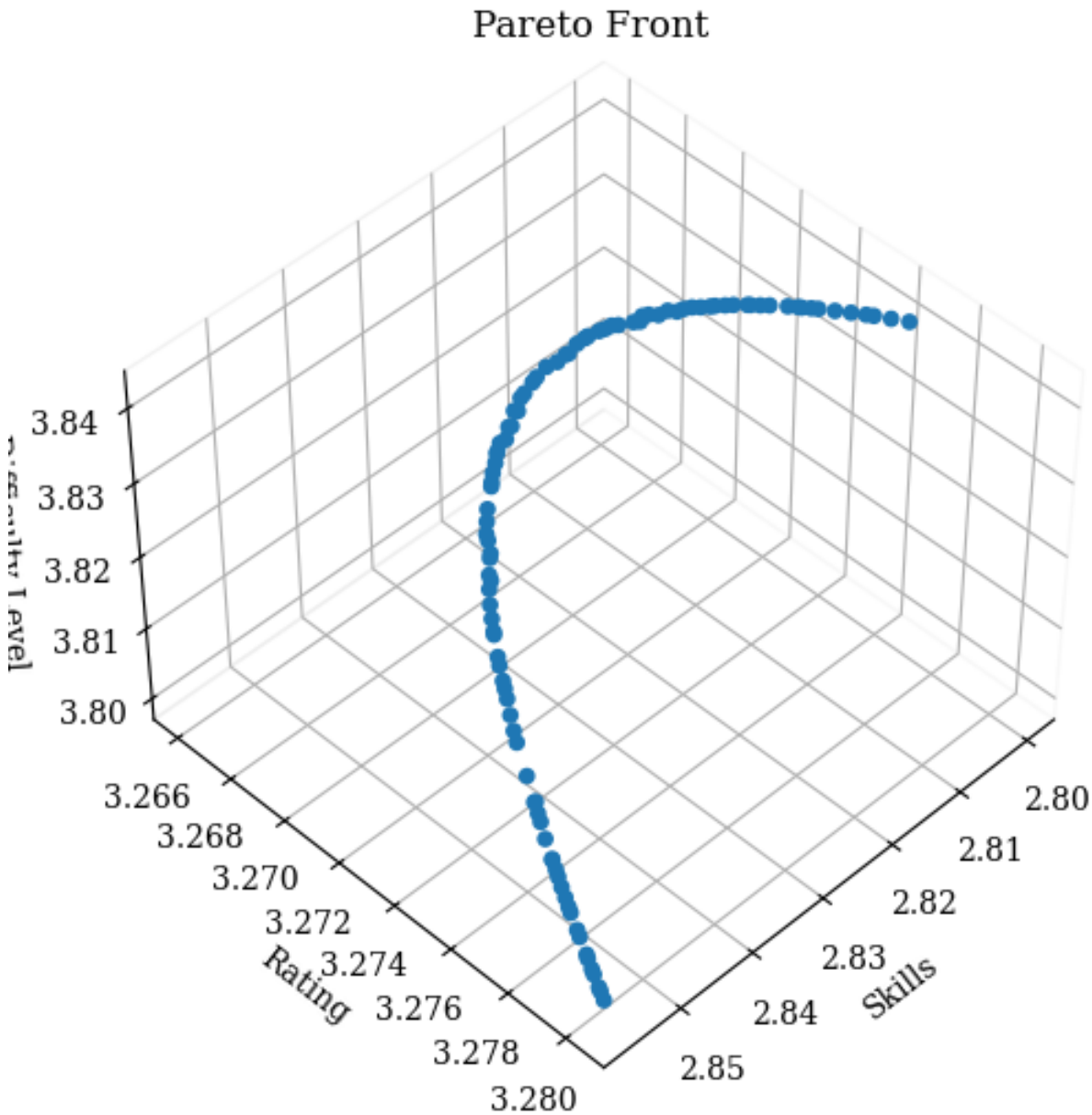


Figure34 : Pareto front of course recommendation problem optimization in problem class

In pareto of optimization problems lower values of the objectives are considered more important if the problems is define using minimization. On the other hand, if the problem is defined using maximization, then the higher values of the objective functions are considered. Course recommender system API used minimization to solve the optimization problem. Therefore, the lower values of objectives have higher preference. As you can see the most important factor is the skills (with objective values in range of 2.80 and 2.85) because it has lower values compared to other two objectives that is why it is at front of the pareto. Rating

has values in range of 3.266 and 3.280 whereas the difficulty level has values in range of 3.80 and 3.84.

Evaluation

Evaluation of number of records used in testing

Evaluation of the output of recommender systems

The ElementwiseProblem class gives multiple recommendation each time the user send request based on dataset. The Recommender performs best for fewer records from the dataset. For example, when it was tested with fifty records it took almost no time and the result were also accurate. When it was tested with 500 records it took slightly longer but the results were still accurate. However, when it was tested using the all the dataset of Coursera the results were slightly more inaccurate and precision was low compared to when tested using fewer records from coursera dataset. All tested used in report are on 500 records from courser dataset gather all 3500 records.

Evaluation of course recommendation on skills and course description columns of the dataset

As I said earlier initially it was multi-objective recommender system but because of some code errors and time limitation I must restrict it to single objective optimization. However, the API can recommend course using single objective but on two different fields of coursera dataset (Skills and Course descrtiption). These two fields have similar keywords that can be used for recommendations. The API can recommend courses by matching the similarity of skills that the user entered with Skills column as well as the course description column. However, only one of the columns can be used at one time to find similarity with skills that the user entered by commenting/uncommenting a section of the code. The tests were performed by finding the similarity between the skills entered by the user and with both Skills and course description column of coursera dataset but one at time and the results were noted down which are available in test section of this report.

	Skills Entered	**Course Rating**	**Difficulty Level**	**Total Recommendations**	**No of correct recommendations**	**No of incorrect recommendations**	**Precision**	**Overall Precisions**
Case 1	Python	5	2	5	4	1	1.0	

Case 2	Python, Neural Networks, Machine Learning	5	2	5	3	2	0.8	
Case 3	Python, Neural Networks, Programming	5	2	5	3	2	0.6	
Case 4	Neural Networks	5	2	5	4	1	0.6	
Case 5	Neural Network	5	2	5	4	1	0.8	
Case 6	Software Engineering	5	2	5	3	2	0.8	
Case 7	Sound D	5	2	5	0	5	0.0	
Result				35	21			21/35 = 0.6

Figure35 : Table of precision of the system using Course Description search

	Skills Entered	**Course Rating**	**Difficulty Level**	**Total Recommendations**	**No of correct recommendations**	**No of incorrect recommendations**	**Precision**	**Overall Precisions**
Case 1	Python	5	2	5	5	0	1.0	
Case 2	Python, Neural Networks, Machine Learning	5	2	5	4	1	0.8	
Case 3	Python, Neural Networks, Programming	5	2	5	3	2	0.6	
Case 4	Neural Networks	5	2	5	3	2	0.6	
Case 5	Neural Network	5	2	5	4	1	0.8	
Case 6	Software Engineering	5	2	5	4	1	0.8	
Case 7	Sound	5	2	5	2	3	0.0	
Result				35	25			25/35 = 0.714

Figure36 : Test Results of Recommenders system with Skills column similarity

The table in figure above shows the results of tests on course recommender system API. Although the table is showing difficulty level and rating were entered by the user, but they

were not used by the API in its recommendations because the current system is single objective recommender system. From observing the tests done on the system using keywords search in skills column the overall accuracy is 0.714 which shows that the course recommender system's accuracy is 71 percent. When the recommender was test for same skills entered by the user but compared with Course Description column similarity. The result showed the precision of 0.6. The overall accuracy of the system for Course Description column is 60 percent.

The results shows that the current recommender system API which is using single objective, is more accurate in its recommendation with Skills column. It is observer during the testing that the course recommender system always recommends courses even when no course with relevant skills entered by the used was present in the Coursera dataset. For example, when the "sound design" was searched the number of incorrect recommendations were 3 and the number of correct recommendations were 2. This can be optimized by refining the algorithm and the dataset. The problem is that if the skill is a multi-word skill, then it considers each word of the skill separately and try to find same word in Course Description or Skills column. Which is wrong because the skill is multi-word and whole skill should have been matched, not just one or two words. This can be optimized but for that the dataset would need more cleaning and pre-pre-processing and the algorithm needs to be changed a bit.

6. Conclusions and Discussion

The course recommender system API is developed using multi-object optimization algorithm and python framework flask. The API is using Coursera dataset for recommendations. The API has a user-friendly interface developed using HTML, CSS and jinja template of flask. The design of the graphical user interface is simple and intuitive. The API is based on

objective based recommendations principle and methodology. The user enters his objectives which are Skills, Rating, and Difficulty Level; the app recommends a course from the Coursera dataset. Although the API was supposed to use all three objectives for recommendations but because of coding errors it is not using all three objectives and is just using one objective which is skills that the user enters. API recommends five courses each time a user sends a request to the API for course recommendations. The course recommendations are displayed to the user on same interface. The user gets recommendations from the API within 9 seconds when all the data in Coursera is used for recommendations. However, if 500 hundred records from the Coursera dataset are used for recommendations then the API sends recommendations within 2 seconds. The API is tested, and results are satisfactory for single objective recommendations. The course recommender system API never recommended duplicate courses in five recommendations that it made. The overall accuracy of the API for Skills objective when compared with skills column in Coursera dataset is 71 percent. The API have been tested on Windows and IOS operating system and works fine. It is web application, and it is tested web browsers such as Safari, Chrome and Microsoft Edge and it works fine on all these web browsers. The API has achieved all the objectives apart from one which is it could not implement all the three objectives in course recommendations. The multi-objective optimization algorithm can increase the accuracy in recommendation systems as well as it can use multiple objectives in objective based recommendations. The API used minimization technique of optimization and ElementwiseProblem class of Pymoo in solving the optimization problem. The research provided an opportunity to explore the optimization techniques to solve problems. It also provided opportunity to undertake research on recommender systems and how they can help in many fields today. APIs have gained popularity in recent past, and their use have increased. These API can be used in different systems to perform sub tasks.

Future Work

In future more work can be done to improve the API. It is using single objective in recommendations; it can be improved by implementing multiple objectives in course recommendations. The data can also be pre-processed, further, and algorithm can be improved in a way that if a skill has multiple words, it should compare the muti-word skill with muti-word skill in Coursera dataset. I will improve the accuracy of the recommendation algorithm.

Python library Pymoo has other classes that can also be used in solving optimization problems.

Keywords used
Recommender System, Multi Objective Optimization, Machine Learning, Genetic Algorithms, Pymoo NSGA II

7. References

Hug, N., 2020. Surprise: A Python library for recommender systems. *Journal of Open Source Software*, *5*(52), p.2174.

Konak, A., Coit, D.W. and Smith, A.E., 2006. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, *91*(9), pp.992-1007

Jannach, D., Manzoor, A., Cai, W. and Chen, L., 2021. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, *54*(5), pp.1-36.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H. and He, Q., 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, *34*(8), pp.3549-3568.

Khanal, S.S., Prasad, P.W.C., Alsadoon, A. and Maag, A., 2020. A systematic review: machine learning based recommendation systems for e-learning. *Education and Information Technologies*, *25*, pp.2635-2664.

Melville, P. and Sindhwani, V., 2010. Recommender systems. *Encyclopedia of machine learning*, *1*, pp.829-838.

Verma, V. and Aggarwal, R.K., 2020. A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective. *Social Network Analysis and Mining*, *10*, pp.1-16.

Alamdari, P.M., Navimipour, N.J., Hosseinzadeh, M., Safaei, A.A. and Darwesh, A., 2020. A systematic study on the recommender systems in the E-commerce. *Ieee Access*, *8*, pp.115694-115716.

Kulkarni, P.V., Rai, S. and Kale, R., 2020, January. Recommender system in elearning: a survey. In *Proceeding of International Conference on Computational Science and Applications: ICCSA 2019* (pp. 119-126). Singapore: Springer Singapore.

Mokarrama, M.J., Khatun, S. and Arefin, M.S., 2020. A content-based recommender system for choosing universities. *Turkish Journal of Electrical Engineering and Computer Sciences*, *28*(4), pp.2128-2142.

Yanes, N., Mostafa, A.M., Ezz, M. and Almuayqil, S.N., 2020. A machine learning-based recommender system for improving students learning experiences. *IEEE Access*, *8*, pp.201218-201235.

Lü, L., Medo, M., Yeung, C.H., Zhang, Y.C., Zhang, Z.K. and Zhou, T., 2012. Recommender systems. *Physics reports*, *519*(1), pp.1-49.

Wu, S., Sun, F., Zhang, W., Xie, X. and Cui, B., 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, *55*(5), pp.1-37.

Geng, B., Li, L., Jiao, L., Gong, M., Cai, Q. and Wu, Y., 2015. NNIA-RS: A multi-objective optimization based recommender system. *Physica A: Statistical Mechanics and its Applications*, *424*, pp.383-397.

Zaizi, F.E., Qassimi, S. and Rakrak, S., 2023. Multi-objective optimization with recommender systems: A systematic review. *Information Systems*, p.102233.

Vanconcelos, R., (2021), A guide to ML model serving. Available at: https://ubuntu.com/blog/guide-to-ml-model-serving [Accessed: 10-02-2024]

FastAPI (2024), FastAPI frae work, high performance,easy to learn, fast to code, ready for production. Available at: https://fastapi.tiangolo.com [Accessed on: 10-02-2024]

J. Blank and K. Deb, (2020), pymoo: Multi-Objective Optimization in Python, in IEEE Access, vol. 8, pp. 89497-89509, Available at:

https://ieeexplore.ieee.org/document/9078759 [Accessed on: 10-02-2024]

Manu (2024). *Embeddings and Distance Metrics in NLP*. [online] Medium. Available at: https://medium.com/@manuktiwary/embeddings-and-distance-metrics-in-nlp-7a000c96d7db [Accessed 25 Apr. 2024].

pymoo.org. (n.d.). *pymoo: Multi-objective Optimization in Python*. [online] Available at: https://pymoo.org/index.html.

www.kaggle.com. (n.d.). *Coursera Courses Dataset 2021*. [online] Available at: https://www.kaggle.com/datasets/khusheekapoor/coursera-courses-dataset-2021.

Vancouver

8. Appendices

Appendix A: Project Definition Document

Project Definition Document

Multi-objective optimization for API of course recommender system

Computer Science BSc

Student: Taha Jomehpour Arashlou

Taha.jomehpour-arashlou@city.ac.uk

Consultant: Mirela Reljan-Delaney

mirela.reljan-delaney.2@city.ac.uk

Proposed By: Taha Jomehpour Arashlou

Description: This project aims to create a recommender system with use of multi objective optimization instead of traditional methods with the help of pymoo and fastAPI.

This project will create a user-friendly system to help to find suitable courses to students or users who are looking to learn new materials or expand their skillsets.

No outside arrangement has been placed for this project.

Word Count: 1062

Table of Contents

Problem to be solved ***5***

Project Objectives ***5***

Beneficiaries ***6***

Work Plan ***47***

Steps **47**

Methodology **12**

Dataset: 12

Algorithms 48

Pymoo Python Library 20

fastAPI 20

Timeline **49**

Risks ***49***

References ***49***

ETHICS FORM: ***51***

Problem to be solved

Given the rapid expansion of information on the internet, recommender systems play a crucial role in mitigating information overload (Wu et al., 2022). Recommender systems are used for recommendation to users based on some set criteria. These recommender systems are quite common in retail. Lots of e-commerce websites are using recommender system for

recommending most popular products or recommending similar products to the current product (Lu, et al., 2012). However not much have been done in areas such education or academia etc. Many students struggle to find a course that can expand their skills without researching extensively to reach the right course within their range of ability. This project will help students to comfortably choose a course from the online learning platform Coursera to expand their knowledge on various topics. Usually, the recommender systems use traditional algorithms like support vector machine, random forest, gradient biased and neural networks. Another method that can be used multi-objective optimization that can used for recommendations based on multiple objectives. MOP optimize several objectives which are not even similar and are incomparable to each other to get one optimal solution or recommendation (Geng et al., 2015). The research will develop an API that will use the model developed using multi-objective optimization machine learning algorithm to make recommendation to students about future courses based on the skills, difficulty level and university partner of the course etc. The research will use data collected from Kaggle which is an online community of data scientists. The API will be developed using a python framework fastAPI.

Project Objectives

5. Efficiency and usability of the system.
    a. The Recommender system shall be efficient and use a suitable algorithm to provide personalised and reasonable options to the user.
    b. The system shall help the user to choose a suitable course from the online learning platform coursera.
    c. The system must give 5 recommended courses that matches the most with the description and the skills provided by the user.
    d. The system must run without any problems and must not contain any empty or harmful materials.

e. The system must consider multiple factors to recommend a suitable course to the user.

6. User friendliness of the system.
    a. The system must implement an easy and accessible interface.
    b. The system needs to have a feedback and refresh option, so it does not show duplicated or identical courses.
    c. The system must use an API to connect to the user.

The system must provide description, course rating, university of the course, difficulty level and the URL so the user can easily choose and access.

    d. The system must be able to run on most devices without any problem.

7. Resource optimization
    a. The system shall have a credible source such as the database and the research.
    b. The system must not be financially costly to run and operate.
    c. The system shall have large enough database to not limit the users' choice.
    d. The system must give results in a reasonable time.

Beneficiaries

- **Students**: students who want to learn new skillsets or expand their pre-existing knowledge can use the system to get recommendation and compare courses relevant to them so they can more confidently choose a course.

- **Employers:** employers can use this system to get relevant courses recommendations and trainings relevant to their business to train employees.

- **Graduates**: graduated who want to expand their CVs or need an insight and learning before they secure a role, can get recommendation of what they need to learn before getting into their desired field and increases their employability.

Work Plan

Steps

By getting the data from Kaggle and training it using machine learning and Pymoo, after the model is trained and API will be created by the help of FastAPI to connect the user to the database, and at last system will be tested thoroughly.

Methodology

The research will use prince2 methodology which is an agile methodology. The agile methodology develops a prototype and improve it based on user's recommendations and continue doing that until the final product is not ready.

Dataset

The data is a csv file that is collected from the online platform Kaggle about the Coursera. Coursera is an online platform that offers variety online courses. The data is originally collected from publicly available data of Coursera. The key features of the dataset are course name, university partner, course rating, difficulty of the course, description of the course, skills associated with the course and course URL.

Algorithms

We will use multi-objective optimization algorithm for course recommender system. The traditional methods don't consider multiple objectives which are crucial to improve user experience. Multi-objective optimization can balance the multiple objectives (Zaizi et al., 2023). The research will use a python library Pymoo to implement the course recommender system.

The research will develop a machine model in python that will be trained and tested on data collected from Kaggle.

Pymoo Python Library

To implement the multi-objective optimization algorithm, we will use python library Pymoo which is a python framework that helps in creating Multi-objective Optimization .

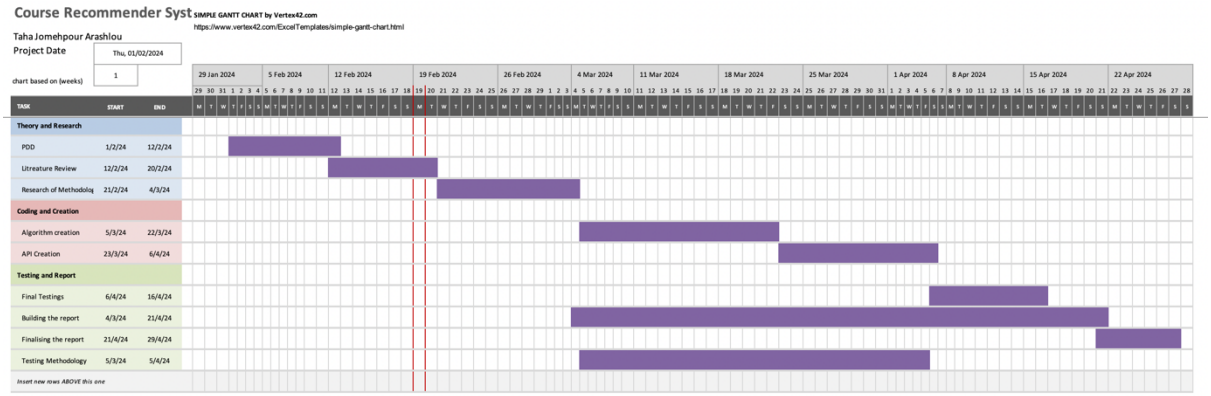
fastAPI

fastAPI python framework will be used to develop the API. FastAPI is a modern, high performant web framework for building APIs with python 3.8+. Key features of FastAPI include data validation and enabling creation of endpoints of handling recommendation

requests and response. FastAPI is also one of the fastest python frameworks available and very easy to use. These features makes it a great tool to choose in creating a recommender system API (FastAPI, 2024).

Timeline

Gantt chart:



This chart shows the planned schedule of the project.

Risks

1. Learning difficulties: using the python library pymoo can be challenging in a limited time and can cause delays in the timeline.
2. Creating an API using fastAPI: can be challenging as it is a newer framework, and it takes time to learn and use comfortably.
3. Unforeseen Dependencies: as it can cause a delay or malfunction in the system or my personal functionality to deliver the project with quality and efficiently.
4. Affordability: The Cloud servers can become too expensive to operate on after a time.
5. Limited resources: for a better result in the system a more advanced and broader data might be needed.
6. The system might not recommend a very suitable course for the user.
7. The courses might become outdated and not available anymore.

References

Lü, L., Medo, M., Yeung, C.H., Zhang, Y.C., Zhang, Z.K. and Zhou, T., 2012. Recommender systems. *Physics reports*, *519*(1), pp.1-49.

Wu, S., Sun, F., Zhang, W., Xie, X. and Cui, B., 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, *55*(5), pp.1-37.

Geng, B., Li, L., Jiao, L., Gong, M., Cai, Q. and Wu, Y., 2015. NNIA-RS: A multi-objective optimization based recommender system. *Physica A: Statistical Mechanics and its Applications*, *424*, pp.383-397.

Zaizi, F.E., Qassimi, S. and Rakrak, S., 2023. Multi-objective optimization with recommender systems: A systematic review. *Information Systems*, p.102233.

Vanconcelos, R., (2021), A guide to ML model serving. Available at: https://ubuntu.com/blog/guide-to-ml-model-serving [Accessed: 10-02-2024]

Vancouver

FastAPI (2024), FastAPI frae work, high performance,easy to learn, fast to code, ready for production. Available at: https://fastapi.tiangolo.com [Accessed on: 10-02-2024]

J. Blank and K. Deb, (2020), pymoo: Multi-Objective Optimization in Python, in IEEE Access, vol. 8, pp. 89497-89509, Available at: https://ieeexplore.ieee.org/document/9078759 [Accessed on: 10-02-2024]

ETHICS FORM

Research Ethics Review Form for BSc and MSci Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate students undertaking their final project in the Department of Computer Science must consider the ethics of their project work and ensure that it complies with research ethics guidelines and the law for data protection. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

To ensure that they give appropriate consideration to ethical issues, all students must complete this form and attach it to their project definition document (PDD). There are two parts:

PART A: Ethics Checklist. All students must complete this part.
The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete part B as well. The project supervisor or consultant has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned work with human end user participants as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor or consultant as the project progresses and details are established. You must obtain ***full approval*** in writing, before recruiting and engaging with human end users participants for your project.

A.1 If you answer YES to any of the questions in this block, your consultant/supervisor must have obtained approval for the project from an appropriate external ethics committee, and you need to have received written confirmation of this from him/her. Students cannot themselves apply for ethics approval in this case as the project is considered high risk". This type of research is not covered by City's process, and external approval from an appropriate institution is required.		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)?	**NO**
1.2	Will you recruit participants who are covered by the Mental Capacity Act 2005?	**NO**
1.3	Will you recruit any participants who are covered by the Criminal Justice System, for example, people on remand, prisoners and those on probation?	**NO**
A.2 If you answer YES to any of the questions in this block your consultant/supervisor must have obtained appropriate ethics committee approval		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about protected characteristics (as identified by the Equality Act 2010)? *For example: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and*	**NO**

	proceedings	
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the Senate Research Ethics Committee (SREC), you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://researchmanager.city.ac.uk/. Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee (SREC).		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**NO**

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be ***provisional***. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with ***full approval*** of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module and/or result in an academic misconduct investigation.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions must be answered fully. All grey instructions must be removed.		*Delete as appropriate*
1.1.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES / NO**
1.2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**YES / NO**
1.3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES / NO**
1.4	Will consent be obtained from the participants in your project? Consent from participants **MUST** be obtained if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc. *If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained.* *You must also retain the completed forms for subsequent inspection.* *Failure to provide the completed consent request forms will result in withdrawal of any earlier ethical approval of your project.*	**YES / NO**
1.5	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	**YES / NO**

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? *If **YES**, you must provide details of how your safety will be ensured.*	**YES / NO**

B.3 Attachments **ALL of the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**	***YES***	***NO***	***Not Applicable***
Details on how safety will be assured in any non-University location, including risk assessment if required (see B2)			
Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) *Any personal data must be acquired, stored and made accessible in ways that are GDPR compliant.*			
Full protocol for any workshops or interviews**			
Participant information sheet(s)**			
Consent form(s)**			
Questionnaire(s)** *sharing a Qualtrics survey with your supervisor is recommended.*			
Topic guide(s) for interviews and focus groups**			
Permission from external organisations or Head of Department** *e.g. for recruitment of participants*			

***If these items are not available at the time of submitting your project proposal, then **provisional approval** can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. **All** such items **must** be seen and approved by your supervisor before the activity for which they are needed begins. Written evidence of **final approval** of your planned activity must be acquired from your supervisor before you commence.*

Changes

If your plans change and any aspects of your research that are documented in the approval process change as a consequence, then any approval acquired is invalid. If issues addressed in Part A (the checklist) are affected, then you must complete the approval process again and establish the kind of approval that is required. If issues addressed in Part B are affected, then you must forward updated documentation to your supervisor and have received written confirmation of approval of the revised activity before proceeding.

Templates for Consent and Information

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. You **must** adapt them according to the needs of your project before you submit them for consideration.

Participant Information Sheets, Consent Forms and Protocols must be consistent. Please ensure that this is the case prior to seeking approval. Failure to do so will slow down the approval process.

We strongly recommend using Qualtrics to produce digital information sheets and consent forms.

Further Information

https://www.city.ac.uk/about/governance/committees/cs-research-ethics
https://www.city.ac.uk/research/ethics/how-to-apply/participant-recruitment
https://www.city.ac.uk/research/ethics

Appendix B: Reuse Summary

No reuse code been used, although the classes are following the format of the webites (handbook of the libraries) which are mentioned inside the code.

pymoo.org. (n.d.). pymoo: Multi-objective Optimization in Python. [online] Available at: https://pymoo.org/index.html.

freeCodeCamp.org. (2022). How to Center a Div with CSS. [online] Available at: https://www.freecodecamp.org/news/how-to-center-a-div-with-css [Accessed 30 Apr. 2024].

Appendix C: Changes to original submission of code

There has been few minor changes after the code submission on 23rd of April

```
new_dict[i['Course Name'], i['Skills'], i['Difficulty Level'], i['Course Rating'],
i['Course URL']] = i #comment and uncomment if you need to change the output
display
    #new_dict[i['Course Name'], i['Course Description'], i['Difficulty Level'],
i['Course Rating'], i['Course URL']] = i
```

This outputs more details of the recommended courses and we can change if we want to see course description or the skills

```
courses_data = new_dict
#comment and uncomment where need to search from description or skills (preferably
skills)
'''
course_skills = [data["Skills"] for data in courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(course_skills)
'''

course_descriptions = [data["Course Description"] for data in
courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(course_descriptions)
```

this part is also a modified version of the previous file that a Tfidf vectorizer for skills has been added that searches in the skills coloumn and gives us the ability to either search in skills or course description

```
#to run the application use command flask --app appv6
#this is an updated version of the appv5 (updated after 23rd of April)
#some coding in this part has been inspired from the pymoo official website
availble at: https://pymoo.org/index.html

import pymoo
import numpy as np
import pandas as pd
from pymoo.core.problem import Problem, ElementwiseProblem
from pymoo.algorithms.moo.nsga2 import NSGA2
# from pymoo.factory import get_sampling, get_crossover, get_mutation
from pymoo.optimize import minimize
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)


course_data = pd.read_csv("Coursera.csv") # loadig data from csv file. course_data
is the dataframe
course_data = course_data[course_data['Course Rating']!="Not Calibrated"] #
removing  not calibrated form the data set
course_data = course_data.head(500) # we can select fewer records rather than the
whole dataset

course_data['Course Rating'] = course_data['Course Rating'].astype(float) # getting
the float
data = course_data[['Course Name','Difficulty Level','Course
Description','Skills']] # we have selected just (Course Name','Difficulty
Level','Course Description','Skills') column

course_data.index = course_data['Course Name']# we are storing the indexing the
courses
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'Size' column
course_data['Difficulty'] = label_encoder.fit_transform(course_data['Difficulty
Level'])
course_data.index = course_data['Course Name']
course_data=course_data.to_dict('records')# to convert out data into rocodes so
that we can go thoug each record in dataframe and create didctionay
new_dict = {}
for i in (course_data):# looping throught the records and creating dictioanry
    new_dict[i['Course Name'], i['Skills'], i['Difficulty Level'], i['Course
Rating'], i['Course URL']] = i #comment and uncomment if you need to change the
output display
    #new_dict[i['Course Name'], i['Course Description'], i['Difficulty Level'],
i['Course Rating'], i['Course URL']] = i

courses_data = new_dict
#comment and uncomment where need to search from description or skills (preferably
skills)
'''
course_skills = [data["Skills"] for data in courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(course_skills)
'''

course_descriptions = [data["Course Description"] for data in
courses_data.values()]
```

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(course_descriptions)

# Define the course recommendation problem
def cosine_similarity_function(x, skills):
    keywords_vector = tfidf_vectorizer.transform([skills])
    return -sum(x*cosine_similarity(keywords_vector, tfidf_matrix).flatten())


class CourseRecommendationProblem(ElementwiseProblem):# ElementwiseProblem is
defined in pymoo we are inheriting from ElementwiseProblem

    def __init__(self, skills, rating, difficulty):# __init__ is constructor in
python
        self.skills = skills
        self.rating = rating
        self.difficulty = difficulty
        xl = np.full(len(courses_data), 0.0)
        xu = np.full(len(courses_data), 1.0)
        #using super we are calling ElementwiseProblem class constructor
        super().__init__(
            n_var=len(courses_data), n_obj=1, n_constr=0, xl=xl, xu =xu)
    #_evaluate is a overridden funciton
    def _evaluate(self, x, out, *args, **kwargs):
        #distances = np.array([[np.linalg.norm(np.array([x[i][0], x[i][1]]) -
np.array([courses_data[course]["rating"], courses_data[course]["difficulty"]])) for
i in range(len(x))] for course in courses_data])
        #out["F"] = np.column_stack([np.min(distances, axis=0), np.mean(distances,
axis=0), np.max(distances, axis=0), cosine_similarity_function(x, keyword)])

        out["F"] = cosine_similarity_function(x, self.skills)# Define the course
recommendation problem instance



def get_recommended_course(skills, rating, difficulty):
    algorithm = NSGA2(pop_size=100)
    problem = CourseRecommendationProblem(skills, rating, difficulty)# creating
object of CourseRecommenderProblem
    res = minimize(problem,
                algorithm,
                ('n_gen', 100),
                seed=1,
                verbose=True,
                save_history=True,)
    subject_requirements = [skills, rating, difficulty]  # Example requirements
(skills, rating, difficulty level)
    if res.X.ndim>1:
        a = res.X[0]
    else:
        a = res.X
    print(res.X)
```

```
    print("\n\n\n\n\n length of Res ", len(res.X),"\n\n\n\n\n\n\n")
    ans = sorted(range(len(a)), key=lambda i: a[i],reverse=True)[:5]
    recommend_courses = []
    for i,x in enumerate(new_dict):
        if i in ans:
            recommend_courses.append(x)
    return recommend_courses

#Define the route for the landing page of flask app/ default rout
@app.route('/', methods=['POST', 'GET'])
def index():
    return render_template('index.html')

# Define the Flask route for recommending courses based on keywords
@app.route('/recommend_courses/',  methods=['POST'])
def recommend_courses():
    skills = request.form['skills']
    rating = float(request.form['rating'])
    difficulty = float(request.form['difficulty'])
    recommended_courses = get_recommended_course(skills, rating, difficulty)
    return render_template('index.html',taskslist=recommended_courses)

    #res = jsonify({'recommended_courses': recommended_courses})
    #return res

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080,debug=True)
```

appv5.py Original Submission Code:

```
import numpy as np
import pandas as pd
from pymoo.core.problem import Problem, ElementwiseProblem
from pymoo.algorithms.moo.nsga2 import NSGA2
# from pymoo.factory import get_sampling, get_crossover, get_mutation
from pymoo.optimize import minimize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)


course_data = pd.read_csv("Coursera.csv")
course_data = course_data[course_data['Course Rating']!="Not Calibrated"]
course_data = course_data.head(500)

course_data['Course Rating'] = course_data['Course Rating'].astype(float)
data = course_data[['Course Name','Difficulty Level','Course
Description','Skills']]
```

```
course_data.index = course_data['Course Name']
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'Size' column
course_data['Difficulty'] = label_encoder.fit_transform(course_data['Difficulty
Level'])
course_data.index = course_data['Course Name']
course_data=course_data.to_dict('records')
new_dict = {}
for i in (course_data):
    new_dict[i['Course Name']] = i


courses_data = new_dict
course_descriptions = [data["Course Description"] for data in
courses_data.values()]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(course_descriptions)

# Define the course recommendation problem
def cosine_similarity_function(x, skills):
    keywords_vector = tfidf_vectorizer.transform([skills])
    return -sum(x*cosine_similarity(keywords_vector, tfidf_matrix).flatten())


class CourseRecommendationProblem(ElementwiseProblem):

    def __init__(self, skills, rating, difficulty):
        self.skills = skills
        self.rating = rating
        self.difficulty = difficulty
        xl = np.full(len(courses_data), 0.0)
        xu = np.full(len(courses_data), 1.0)

        super().__init__(
            n_var=len(courses_data), n_obj=1, n_constr=0, xl=xl, xu =xu)

    def _evaluate(self, x, out, *args, **kwargs):
        # distances = np.array([[np.linalg.norm(np.array([x[i][0], x[i][1]]) -
np.array([courses_data[course]["rating"], courses_data[course]["difficulty"]])) for
i in range(len(x))] for course in courses_data])
        # out["F"] = np.column_stack([np.min(distances, axis=0), np.mean(distances,
axis=0), np.max(distances, axis=0), cosine_similarity_function(x, keyword)])

        out["F"] = cosine_similarity_function(x, self.skills)# Define the course
recommendation problem instance


def get_recommended_course(skills, rating, difficulty):
```

```
    algorithm = NSGA2(pop_size=100)
    problem = CourseRecommendationProblem(skills, rating, difficulty)
    res = minimize(problem,
                   algorithm,
                   ('n_gen', 100),
                   seed=1,
                   verbose=True,
                   save_history=True,)
    subject_requirements = [skills, rating, difficulty]  # Example requirements
(skills, rating, difficulty level)
    if res.X.ndim>1:
        a = res.X[0]
    else:
        a = res.X
    print(res.X)
    ans = sorted(range(len(a)), key=lambda i: a[i],reverse=True)[:5]
    recommend_courses = []
    for i,x in enumerate(new_dict):
        if i in ans:
            recommend_courses.append(x)
    return recommend_courses
#Define the route for the landing page of flask app

@app.route('/', methods=['POST', 'GET'])
def index():
    return render_template('index.html')

# Define the Flask route for recommending courses based on keywords
@app.route('/recommend_courses/',  methods=['POST'])
def recommend_courses():
    skills = request.form['skills']
    rating = float(request.form['rating'])
    difficulty = float(request.form['difficulty'])
    recommended_courses = get_recommended_course(skills, rating, difficulty)
    return render_template('index.html',taskslist=recommended_courses)

    #res = jsonify({'recommended_courses': recommended_courses})
    #return res

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080,debug=True)
```

HTML code had a minor change in the spacing, now after each result it has a break.

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="{{url_for('static',
filename='style.css' )}}">
        </head>
        <body>
```

```
            <div id="mainDiv">

            <h1> Course Recommender System </h1>
            <form action="{{url_for('recommend_courses')}}" method="post">

            <label for="">  Skills </label>
            <input type="text" name="skills"> <br>
            <label for="">  Rating </label>
            <input type="text" name="rating"> <br>
            <label for="">  Difficulty Level </label>
            <input type="text" name="difficulty"> <br>

            <button class="btnn" value="Send" type="submit">Send</button>
            </form>
            </div>
            <div id="resultDiv">
                <h1 > the recommended courses are: </h1>
                    {% for data in taskslist %}
                        <p>
                                    {{data}}   <br>

                        </p>
                        {% endfor %}


             </div>

        </body>
</html>
#this class been inspired from freeCodeCamp.org. (2022). How to Center a Div with
CSS. [online] Available at: https://www.freecodecamp.org/news/how-to-center-a-div-
with-css/ [Accessed 21st Apr. 2024].
```

Style.css has stayed the same

```
#mainDiv{
    width: 500px;
    height: 50px;
    margin: 100px;
    outline: solid 1px black;
    text-align: center;


}
#resultDiv{
    width: 500px;
    margin: 650px;
    height: 800px;
    margin-top: 0px;
    outline: solid 1px black;
```

```
    text-align: center;


}
form {
    border: 3px solid #f1f1f1;
  }

  /* Full-width inputs */
  input[type=text], input[type=password] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    box-sizing: border-box;
  }

  button {
    background-color: #04AA6D;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
  }
  button:hover {
    opacity: 0.8;
  }
```

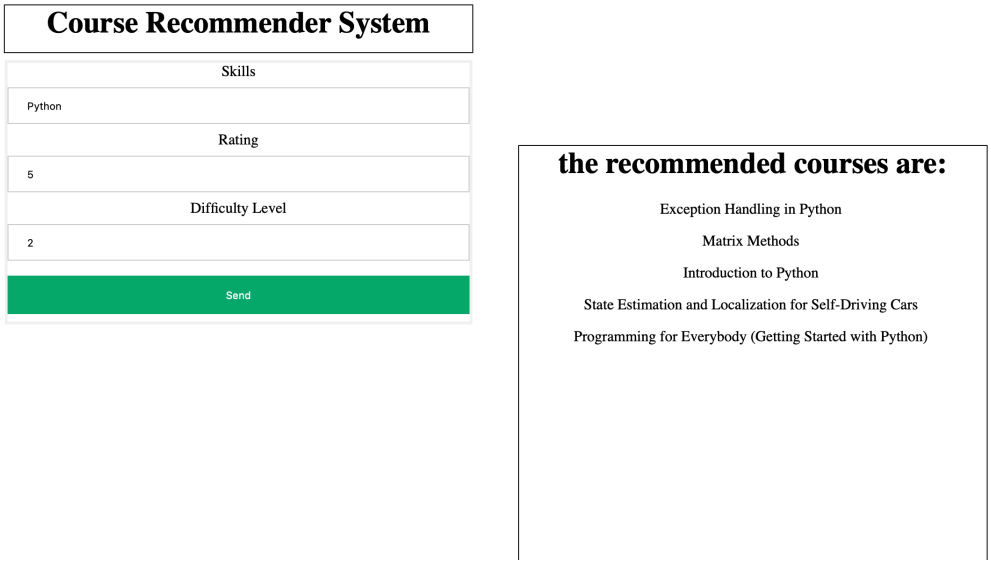


Figure1 in appendix: screenshot of the output produced by appv5 (original submission)

Appendix C Diagrams and Screeshots

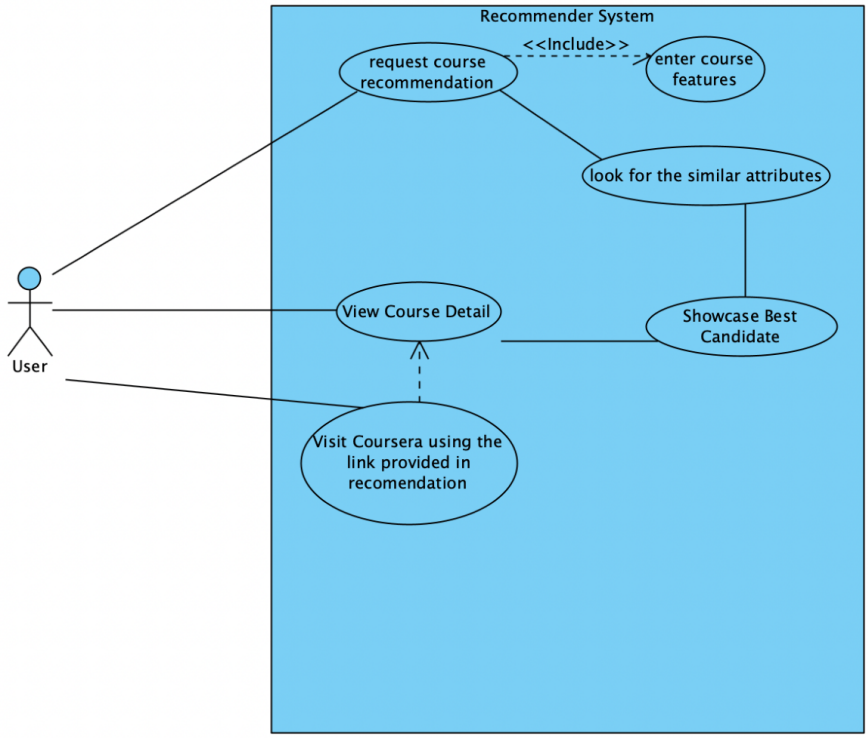


Figure1: Use case diagram