DAT410 / DIT728 - Design of AI Systems
Assignment 1: AI Problem Solving

Group Members: Nazmiyeh Sadiyeh & Saghar Alvandi

1. Predict the Temperature

When dealing with temperature we usually like to have date, timestamp and temperature values. In this case, we can assume that we might have a dataset that includes previous temperature values, timestamps and the date. Additional data is also relevant like the location, and time of the year. Basically, the data that we might be dealing with is numeric and continuous.

The amount of data is also relevant where we might need years of temperature data to find trends. This has a big effect on the accuracy of our results. There is another factor that might affect the final result like unpredictable weather events. Take this winter in Sweden for example. Usually based on previous years, it usually snows more often with most days having temperatures below zero. Even the snow amount is less this year.

To predict the temperature for the next weekend and for the next year, we need to think about short-term and long-term changes.
For the next weekend, the focus is on short-term changes and the data comes mostly from the past few weeks. These can be affected by local weather conditions like humidity, wind, and air pressure**.**
For the same date next year, the focus is on seasonal patterns and yearly trends. It is affected by long-term climate changes. The data covers several years to understand trends and seasonality.

A modelling approach for both long-term and short-term temperature forecasting can be represented mathematically as :

$$T_{t+n} = f(T_t, T_{t-1}, \ldots, X)$$

Where:

- T = temperature
- X = external factors (humidity, wind speed, location, etc.)
- t = current time
- n = forecast horizon (short-term prediction: few days, long-term prediction: one year)

An effective model for this task is Long Short-Term Memory (LSTM), which can process multiple time steps as input to capture patterns and dependencies in the data. The model is trained using historical temperature data that is relevant to the specific forecasting period, allowing it to learn from past trends and improve prediction accuracy.

2. Bingo Lottery Problem

We have 1000 lottery tickets where each lottery ticket is 5x5 with 25 randomly placed numbers between 1 to 25. We need to select a set of numbers that will give as close as possible to 120 winning tickets. A winning ticket has a horizontal, vertical or diagonal line filled with winning numbers. We also need to make sure that no ticket wins more than once. Based on the problem description we already know the number combination of each ticket by looking at them. We need to analyse them and see which combination of numbers can provide 120 wins without repeated wins. We have 1000 tickets and 25 numbers to choose between for each ticket, giving 25 000 numbers in total. ($1000\ tickets \times 25\ numbers\ per\ ticket\ =\ 25,000\ numbers$)
This means there are many possible combinations of numbers and winning tickets to choose from.

This problem has constraints. It also has a discrete set of numbers and the set is finite. It also involves finding combinations of numbers that maximize winning tickets without violating the constraints. Additionally, the number structure is already printed and fixed.

It sounds like an optimization problem with constraints where the goal is to maximize the number of winning tickets but with constraints.

- We want exactly 120 wins.
- No ticket can win more than once.
- The numbers must form horizontal, vertical, or diagonal lines on each ticket.

A possible way to solve this problem could be using Integer Linear Programming. This method helps to find the best solution when dealing with an optimization problem with constraints.

We need to define a binary variable for each number and ticket combination.

Let x_{ij} be a binary variable representing whether a number i is selected for ticket j's winning row.

$x_{ij} =$ 1 if number i is part of the ticket j's winning row

$x_{ij} =$ 0 Otherwise

Where

$i = 1, 2, \ldots, 25$ represents the 25 possible numbers.

$j = 1, 2, \ldots, 1000$ represents the 1000 tickets

Then, we need to decide on our objective for the method. In our case, we want to maximize the number of the winning tickets with regard to the constraints. Let y_i be a binary variable that

represents whether the ticket j wins (1 if it wins, 0 if it does not). This gives us the following Objective function:

$$Maximize \sum_{j=1}^{1000} y_j$$

Where

$j = 1, 2, ..., 1000$ represents the 1000 tickets.

Finally, we need to define our constraints.

1. It should be as close as possible to 120 wins.

$$\sum_{j=1}^{1000} y_j = 120$$

2. No ticket wins more than once
3. A winning combination is received for a horizontal, vertical or diagonal row.

3. Public Transport Departure Forecast

To deal with this problem we need to have a dataset including features like time of the day, day of the week, previous delays and additional historical information regarding departure times for each transport line. The data should be like a time series including both categorical features like (line numbers, time of the day etc) and numerical features (previous delays, waiting times etc).

The goal for this task is to predict the future departure time based on earlier observation. The problem can be defined as a regression task. To solve this problem, we have chosen to use Gradient boosting regression (XGBoost). The reason for this choice was the ability of this model to handle both categorical and numerical features and capturing complex nonlinear relationships in data (GeeksforGeeks, 2025).

Let :

$$\mathbf{x_t} = \{\text{line number}, \text{day of week}, \text{time of day}, \text{previous delays}, \text{weather conditions}, \dots\}$$

The goal is to learn the function below:

$$\hat{y}_{t+1} = f(\mathbf{x_t})$$

X_t is the set off input features at time t
$\hat{y}_{t+1}$ is the predicted departure time for the next period

XGBoost builds an ensemble of decision trees to iteratively reduce the error in predictions. Thus, the final prediction might be presented as below:

$$\hat{y}_{t+1} = \sum_{m=1}^{M} \gamma_m h_m(\mathbf{x_t})$$

M is the number of decision trees.
h_m represents the m-th tree in the ensemble.
γ_m is the weight assigned to each tree.

4. Film festival problem

The goal is to create a schedule for the film festival but also maximizes the number of films an attendee can see. In this film festival, multiple films are scheduled to be shown, and each film is shown at different times. Some films will also be shown at the same time as others. The attendees can also prioritise which films they want to watch and the system will create a schedule for the week. We need to keep in mind that not all films can be watched because some films overlap. On the other hand, some films will be shown multiple times during the festival so the attendees can still watch the film later. This problem could be described as a constrained optimization problem because we want to maximize the number of films people can watch based on their priorities, while also following certain constraints. This problem could also be a Constraint satisfaction problem because we need to find a solution that satisfies all the constraints. The system must make sure that no attendee is scheduled to watch two films at the same time. While also considering their priorities list and the possibility of an attendee to watch some of the films at another time.

Let's assume that attendees can only watch a certain film once, so it gives a chance to other people to also watch it.

Our data could consist of

- Title of each film or a film ID number

- Duration (minutes)
- Start time
- Which dates the film will be shown
- Will the film be shown multiple times?
- Which films overlap
- List of the prioritisation of films by the attendees

One possible method to solve this problem based on the problem structure could be optimization. Our goal is to create a schedule where we try to maximize the number of films watched by attendees, with some constraints to follow. We could use Integer Linear Programming. We could define binary decision variables x_{ij} , where $x_{ij} = 1$ if the attendee watches film j , and $x_{ij} = 0$ otherwise. Our objective function could be

$$Maximize \sum_{i=1}^{n} \sum_{j=1}^{m} p_j * x_{ij}$$

Where

p_j = the priority of film j

$i = 1, .., n$ represent an attendee

$j = 1, ..., m$ represent an individual film

$n = the\ total\ number\ of\ attendee$

$m = the\ total\ number\ of\ films\ each\ attendee\ will\ watch$

Our constraints would be that

- no attendee gets to watch two films at the same time.
- The system needs to take into account the priority list.
- Need also to keep in mind that some films will be shown multiple times.

Another possible method to use to solve this problem could be constraint programming. We could define the films as variables and the available time slots as domains. Our constraints would be the same as the one mentioned above. These constraints need to be satisfied for each attendee.

5. Product Rating in Consumer Test

To be able to determine the score for different dishwashers in a consumer test, we should use some attributes to make the dishwasher quality measurable. We could have attributes like performance, energy efficiency, noise level, price, volume etc. With some combination of these attributes, we could be able to get an overall score for each dishwasher.

This problem could be a Function (Regression) if we try to assign a numeric score as the final rating. On the other hand, it could also be a Function (Classification) if we try to determine the final score by categorical classification. We could have three categories Excellent, Good and Poor. The easiest way to do it is to have some kind of a weighted scoring system where each attribute has an assigned weight. This weight could be determined by how important each attribute is. For example, the noise level can have a weight of 0.4

We are aware that it is hard to decide on a weight for each attribute because it depends on its importance to the consumer which is also subjective and may introduce biases. Such data can be gathered through surveys, some kind of customer feedback system or market research. Let's assume this data is provided to us. Let's also assume that the weight of an attribute is between 0 and 1. On the other hand, the actual rating for attributes is between 0 and 10. This will give us the following mathematical notation

$$Score_j = \sum_{i=1}^{n} w_i * q_{ij}$$

Where

$i = 1, 2, \ldots, n$

$j = 1, 2, \ldots, m$

$Score_j$ = the total score of the dishwasher j

w_i = the weight of the attribute i

q_{ij} = the actual rating of the quality of the attribute i for the dishwasher j

n = the total number of attributes

j = the dishwasher number

Another possible approach is regression, where linear regression can be used if we predict a continuous score for each dishwasher based on its attributes. We could reuse the exact rating system from our weighted scoring system above. We also need as previously the data about the customer rating of each attribute which will be our training data. We can also reuse the same formula for calculating the final score, but instead, the weight for each attribute can be learned

from the data. The regression model will learn the weights for each attribute based on the training data. Then it minimizes the difference between the predicted scores and the actual scores using a loss function, like Mean Squared Error. It will help us determine the importance of each attribute and then be able to get a final score.

6. Constraint Satisfaction and Constraint Programming

a)

Constraint satisfaction problems (CSP) is a mathematical problem (GeeksforGeeks, 2024). The goal is to find values for a set of variables where all specified constraints are satisfied. It is very useful when dealing with artificial intelligence (AI) to solve decision-making problems like scheduling, planning and resource allocation. These problems need to be solved with regard to strict guidelines or constraints.

CSP consist of three main key elements:

1. Variable : Elements that require value assignments
2. Domain: The possible values that each variable can assume
3. Constraints: Rules that define permissible combinations of variable values

There are three different types of CSP:

1. Binary CSPs: Constraint involving two variables
2. Non-Binary CSPs: Constraint involving more than two variables
3. Hard Constraint: Constraint that must be strictly satisfied for the solution to be valid
4. Soft Constraints: Constraint that can be extend and allowing flexibility in the solution

Constraint programming:

Constraint programming is a method used to solve complex problems by defining specific rules known as constraint (Tsvietnov, n.d.). These rules must satisfy all solutions for a problem. Instead of outlining a sequence of steps to find a solution, this approach focuses on specifying the conditions that need to be met. The program searches for solutions that follow all the rules that exist for constraints. This method is particularly effective in areas like scheduling and resource allocation, where multiple variables and limitations must be considered simultaneously.

Sources:

GeeksforGeeks.(2025,January16).*XGBoost*.GeeksforGeeks. https://www.geeksforgeeks.orgxgboost/

GeeksforGeeks. (2024, October 3). *Constraint Satisfaction Problems (CSP) in artificial intelligence*. GeeksforGeeks. https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/

Tsvietnov, O. (n.d.). *Constraint programming by example*. Opensource.com. https://opensource.com/article/19/9/constraint-programming-example