HR Allocation Report

Overview Diagram:

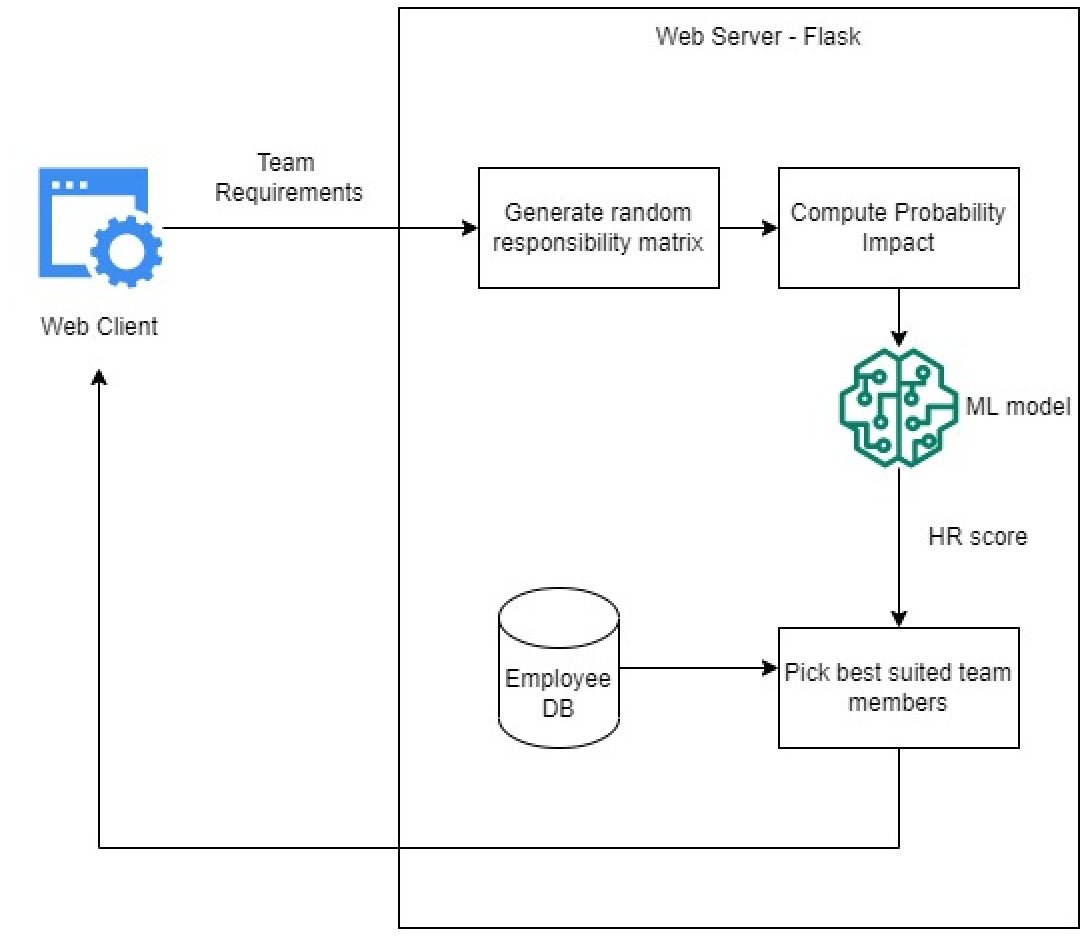

Web Server - Flask
Team Requirements
Web Client
Generate random responsibility matrix
Compute Probability Impact
ML model
HR score
Employee DB
Pick best suited team members

Fig 1. Representation of the workflow of the web application

System requirements

The specifications of the system used to train the model and develop the web application have been specified below.

- RAM: 8.00 GB
- Processor: intel core i5 8th gen
- OS: Windows
- Machine learning model trained on Google collab

Machine learning model:

The heart of the HR allocation application is the prediction model. A multi-output regression model based on the decision tree algorithm was deployed to predict the team structure. Prior to this however, a synthetic dataset was generated and several models were tested to find the best one.

The figure below shows the various steps involved in predicting the team structure starting from the input obtained from the front end.

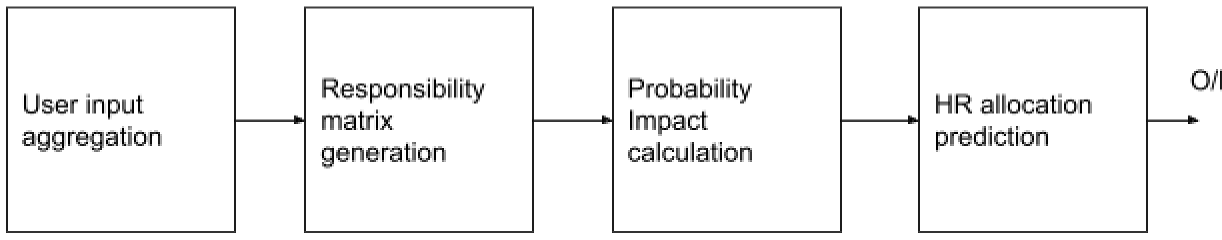

User input aggregation
Responsibility matrix generation
Probability Impact calculation
HR allocation prediction
O/P

Fig 2. Depicts the overall flow of operations in the application

1. User input aggregation:
   The user of the application will give the details of the following to the application:
   - Number of teams required
   - Maximum required level of the four categories of engineers per team

   The prediction model will convert these details into a list, with the maximum required of each of the four engineers represented by a number ordered by the team number. This input will then be passed to the function for generating the responsibility matrix.

2. Responsibility matrix generation:
   The next phase of the prediction model involves generating the responsibility matrix. A responsibility matrix is a two dimensional matrix where the rows represent the challenges and the columns represent the team structure. A total of eight challenges were identified for any generic software engineering task:
   - Understanding of the network requirements
   - Knowledge gathering
   - Technical understanding
   - Detection identification
   - Architecture decision
   - Traceability
   - Test specification
   - Documentation

   The entries in the responsibility matrix are values strictly less than one. The only condition is that the sum of each row of the responsibility matrix should be one. The responsibility matrix is generated randomly using the user input obtained from step 1. Provisions have been made to replace this function with a deterministic one for generating the responsibility matrix. The end result is that the user input is converted to a two-dimensional responsibility matrix. There are 8 rows and 160 columns. (4 categories of engineers * 4 maximum levels * 10 maximum teams) It is to be noted that the entries in the responsibility matrix for teams less than the maximum number of teams required and for engineers who aren’t required in certain teams are 0. The responsibility matrix will then be used to calculate the probability impact scores.

3. Probability impact calculation:
   The probability impact matrix has 1 row and 160 columns. It’s entries are the sum of the corresponding columns of the responsibility matrix. The third phase of the prediction model deals with generating the probability impact matrix from the responsibility matrix. The probability impact matrix is given as an input to the Machine Learning algorithm.

4. HR allocation prediction:
   The final phase of the prediction model is the HR allocation prediction. A multi-output decision tree regressor is used to predict the team structure. The model takes the probability impact matrix as an input and predicts a 1*160 vector. The entries in this vector indicate the number of employees of a particular profession and particular level that are required (HR score). The vector can be read as follows:
   - ➔ Every 16 entries represent a team (4 categories * 4 maximum levels)
   - ➔ Among these 16, every block of 4 entries represents a engineer category
   - ➔ Each entry in the block of 4 represents the number of employees of a particular level

   These details can then be used to identify the appropriate engineers from an employee database.

Dataset generation:
Since the prediction model makes use of a regression algorithm, it becomes necessary to train the model prior to predicting new outputs. A data set containing 100,000 rows and 320 columns was made for this very purpose. The first 160 columns contain the probability impact values. The rationale behind having 160 columns for this is as follows: There are a maximum of 10 teams. Each team has a maximum of 4 different classes of employees. Each employee class has a maximum of 4 levels. Hence, there are 4*4*10=160 columns. The next 160 columns in the dataset contain the HR score corresponding to the previous 160 columns. The HR scores are computed from the probability impact matrix according to a standard staircase function. The dataset was stored as a CSV file.

Choosing the right algorithm:
A plethora of machine learning and deep learning algorithms were tried before narrowing down on the decision tree based regression model. They are elucidated below. It is to be noted that all the algorithms are regression-based with the capability to predict multiple outputs. All models were quantitatively compared using the mean absolute error metric using k-fold cross validation.
K-fold cross validation:
Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. Thus, the procedure is often called k-fold cross-validation. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.It is a popular method because it is simple to

understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The mean absolute errors of the candidate models are provided below.

Model	**MAE (Mean Absolute Error)**
Multilayer Perceptron model	0.376
Linear regression model	0.272
Support Vector Machine based multiple output regression model	0.215
Decision Tree regression model	0.233

Table 1: Mean absolute errors of candidate models

1. Neural network for multi-output regression
   Since neural networks can support multiple outputs inherently, a multilayer perceptron model (MLP) was developed. The neural network had an input layer with 160 inputs, a dense layer with 160 nodes and an output layer with 160 nodes. The model had a mean absolute error of 0.376. However, this was not the biggest problem. Since the dataset was sparse, the model returned decimal values instead of integers. It was dropped.

2. Linear regression model
   The linear regression model had a mean absolute error of 0.272. Just like the MLP model, the linear regression model returned decimal values instead of integers.

3. Support Vector Machine based multiple output regression model
   Though SVM is a single output regression model, it was chained to make it predict multiple values. The first model in the sequence uses the input and predicts one output; the second model uses the input and the output from the first model to make a prediction; the third model uses the input and output from the first two models to make a prediction, and so on. This model had a mean absolute error of 0.215, the least among all the candidate models. However, the model wasn't able to return integer values.

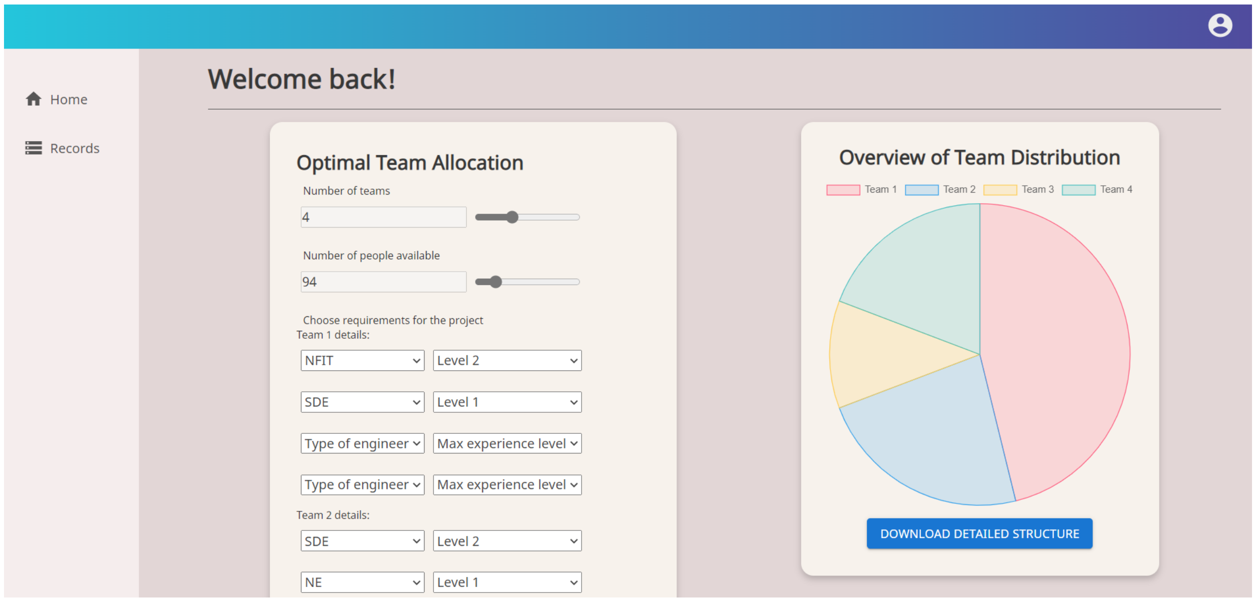
4. Decision Tree regression model
   The decision tree regressor was selected as the algorithm for predicting the team structure, albeit it's fairly high mean absolute error. The reason is that the decision tree regressor returned integer values (as expected). It was also able to correctly map the values for which there were no employees required.

The decision tree regressor was selected because of its relatively low mean absolute error and ability to produce integer values.

Web Application

The web application developed to handle the HR allocation and team formation task uses the following tech stack:

- Frontend: React
- Backend: Flask
- Database: sqlite



Number of teams: 4

Number of people available: 94

Choose requirements for the project

Team 1 details:

NFIT | Level 2
SDE | Level 1
Type of engineer | Max experience level
Type of engineer | Max experience level

Team 2 details:

SDE | Level 2
NE | Level 1

Fig 3. Web application developed to perform HR allocation

The web application caters to the needs of project managers and HRM team members by helping them assign optimal team members based on a particular project's requirement. The project requirements are collected through dynamic form inputs and sent to the backend for some preprocessing. The preprocessing stage gives out the probability impact values which are provided as input to the machine learning model. The prediction results of the model are made downloadable as an excel file in the web application.

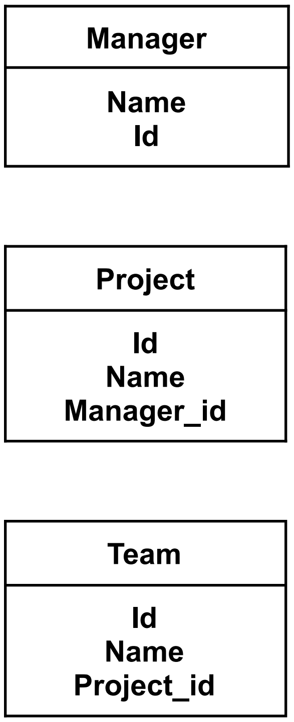
Database:

The figure below shows the schema of our Database from which the employees will be selected based on the level and profession that is required by the teams for that particular project.

Python scripts are written to create the database, by connecting the variable with the database creating a cursor out of it and executing instructions with the help of the cursor. We use data definition language which is a subset of DBMS, to create the tables if they don’t exist. After executing the instructions we commit the changes and close connection that was created for the creation of the database.

Once the database is created we have to insert dummy values into the tables that were created, we use a python library faker and random to generate random names and id’s for the employees and other tables respectively. These generated values are pushed into a list. And this list is inserted into the database using data manipulation language’s insert statement.

And some update and delete functions are also part of the script that are there in case there be a need to alter our database. Where the Id’s that are to be altered and the Id’s that are to be replaced are sent as arguments and the respective changes are made in the database using the data manipulation language statements update and delete.



Employee
Id **Name** **Profession** **Team_id** **Level**

Based on the inputs from the user, the ML model generates the optimum amount of team members which are to be selected for each level for each of the four engineering categories and This is done for each of the 10 teams.

Once we know the level and the engineering category of the employee that's required for the teams we select the employees from the random database that we had previously generated.

Back end:

The number of teams and the max levels of each profession inside a team is sent from the front-end which is then caught and passed to the responsibility matrix generator. The random responsibility matrix generated is used to get the probability impact array, which is then used to predict the HR score(the number of employees at each lvl and profession). If a trained model does not exist, the model is trained on the pre generated set of 100,000 rows of probability impact and HR score to train a decision tree regressor model. With the result of the HR score prediction, the exact number of employees of each level and profession is correspondingly picked up from the database to form team members. Each team is then stacked up as separate sheets to form an excel sheet which is to be sent to the front end.