Prescriptive Algorithms
Assignment 3: Group project

Chigozie Ifepe 2109365

Vishal Shegal 2109374

Pedro Villadangos Benavides 1930079

JADS Jheronimus Academy of Data Science

1.1- Modeling

Given the provided datasets, the main objective of this first part of the project is to give the best possible estimation for the "*Remaining Useful Life"* of airplane engines, from now on written as ***RUL***.

As stated in the dataset, airplane engines work time is measured in cycles. For airplane engines, RUL measures the number of cycles an airplane engine can work before it needs maintenance.

It is important to notice how the engines provided in the training dataset are at different moments of their useful time, as can be seen if looking at the distribution for engine's *RUL* in *Fig.1*

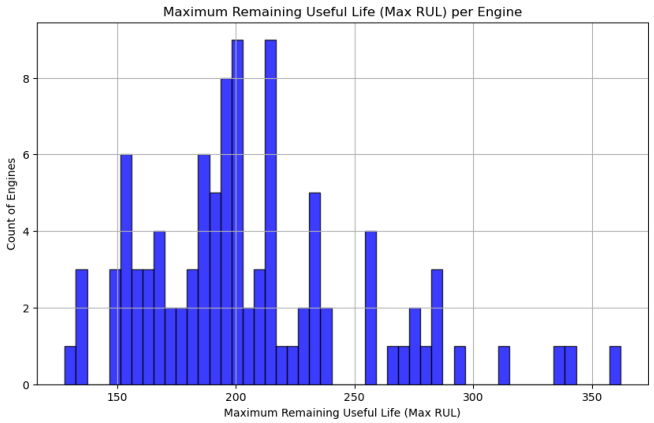

Fig.1 RUL across engines in the dataset

RULs are calculated by fetching the maximum cycle present in the dataset and subtracting from the current cycle in each of the rows for each of the engines.

Dataset contains the values for 3 sets and 21 sensors that measure unknown insights in the engines for each run cycle. As seen in Fig 2, some of these sensor values are not related across RUL fluctuations hence can be deprecated for the model as they are not related.

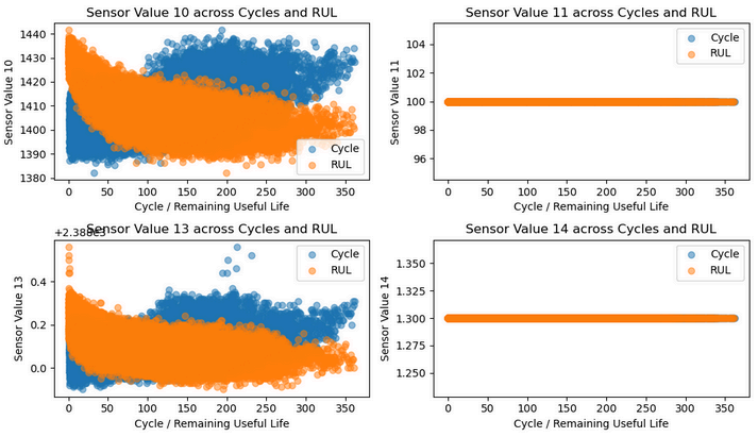

Fig.2 Sensor Values across RUL and Cycles

At first, without considering feature selection for training and hyperparameters. A basic random forest model is calculated using all features and standard hyperparameters. The model performance is evaluated using *Mean Absolute*, *Root Mean Squared* and *Mean Squared errors*. All subsequent models tested are going to be compared using these metrics for the sake of selecting the best model for prediction.

For simplicity, in feature selection all the features whose feature importance is 0, calculated by the *feature_importance* method are going to be deprecated in the models. For the hyperparameters both *gridsearch* and *randomsearch* are performed. The hyperparameters given by the gridsearch are going to be used as they perform better in all error metrics mentioned earlier compared with the rest of the models.

1.2 Prediction

Using the selected trained best model given by feature and hyperparameter selection we are going to make the predictions for the second dataset. Rounding their RUL values obtained for the sake of easy maneuver and comparison in future steps.

As RUL is calculated by each of the cycles we have opted to calculate a single value for each of the engines averaging the weight obtained in the predictions for each cycle, obtaining like this a single predicted value for each engine.

When comparing the results obtained in the prediction with the ones of the consultancy company it catches an eye how for every engine the RULs given by the consultancy are by general lower than our predictions. When carrying a statistical test to check for significance in the variations between the values a p value of 5 e^-28 is obtained.

Being airplane security a big concern with big life and economic risk involved, a consultancy company cannot recommend to manufacturers or airplane companies to push to the limit the maintenance of its engines. A single real life error on this would cause the consultancy company being responsible for a big accident to the eyes of the world. That's why it makes sense to have a significant p value. The consultancy company has to be really conservative with its predictions in this matter.

The obtention of a mean absolute error of *46,94* tells us that on average the consultancy company is conservative by a month and a half with this fact. It appears to be a big error, but when looking at it with perspective to the matter of appliances, and airplane security, it does not appear so big.

2.1- Genetic Algorithm

Using the predicted RULs for the second dataset in the previous part of the assignment, the objective is to find by an heuristic genetic algorithm, good and improving solutions for the scheduling of teams which are going to perform maintenance in engines whose remaining useful time is about to end.

The objective is to minimize the penalty costs incurred by not fixing one of the engines before its useful time comes to an end. A fixed number of teams divided in two different types are assigned to perform the maintenance. Each of the two types of teams takes different time in days to perform maintenance in the engines, depending on the engine also takes more time or not. Different engines have different penalty cost Y for each day passed when the engine is not fixed before their RUL comes to an end.

This maintenance schedule is programmed for a fixed time of T days, in which an engine cannot be maintained more than 1 time in the horizon. Only one team can perform maintenance in each of the engines at the same time.

To construct the genetic algorithm we first define the values of the problem, as number of engines, type of teams, days of the scheduling, maximum daily cost and costs and days needed to perform maintenance for each engine/team type. Then we create classes for fitness and individuals of the population. The fitness value is the objective to maximize by the algorithm. It is associated with the cost penalty for not fixing the engines on time. Using a negative weight when defining the class indicates that the objective is to minimize this value.

Individuals are solutions for the scheduling problem. Represented by tuples that indicates the engine, the team which performs the maintenance, and the day in which the maintenance has to start. They are linked to a fitness value as each scheduling solution might incur a penalty cost for not maintaining the engines on time. Individuals are created randomly by assigning teams to start the maintenance of an engine within the horizon time for these engines which RUL is less in days than the horizon day for the schedule. From these individuals the genetic algorithm is going to select the most “fitted” ones by how it is described in the following steps.

Functions “individual_to_schedule” and “calculate_penalty_cost” add information to the individual when crossing the team type and start day randomly generated by the “create_individual” function by assigning each individual to its schedule and deriving penalty costs for not maintaining the engines on time. Functions “calculate_fitness” and “is_feasible” are to calculate the total penalty cost of an individual and to check whether it satisfies the constraints of the problem respectively.

Once how to create individuals and its linked fitness values are defined we start defining the genetic algorithm by the crossover function “cxTwoPoint” function. What this function does is to establish a point in which the lists from two individuals are going to be swapped, generating a new individual. The offspring. What this function does is to select randomly two individuals from the current generation and divide their maintenance schedules by the same point and then combine the sliced schedules to create a new individual which belongs to the new generation. Selection of parents is random in each generation by the tournament selection method, and is the basis of the genetic algorithm.

With all of this set we use the toolbox object from the DEAP library to run our genetic algorithm. With the “create_individual” function we define the individuals and so the population. A penalty is given to not feasible individuals to discourage its selection by setting its fitness to a high value (10000). The crossover part is defined by using the previous “cxTwoPoint” function in the “toolbox(“mate”)” section. A mutation component is also included in the algorithm with a probability of 0.1 of randomly shuffling its indexes to ensure variability and not fast convergence of individuals.. Selection component is established by a tournament of the size of two individuals.
The hall of fame method is going to be used to save the best individual across each generation so this way the best solution remains.
The initial population is set to 50 individuals randomly generated with the “create_individual” function.
We run the algorithm for 250 generations with a crossover probability of 0.5 and a mutation probability of 0.1.
A small number in the initial population has been chosen because of the feasibility of obtaining optimal solutions in the first generation. This combined by a larger number of generations allows us to see, as in Fig 3, how the genetic algorithm works as its population across generations converges to a minor average fitness value.

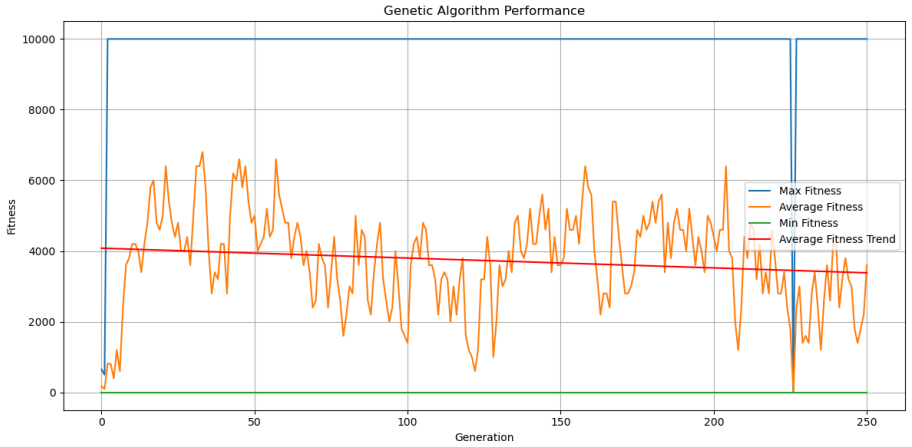

Fig.3 Average fitness and fitness trend across generations of the genetic algorithm

2.2.- Optimization

When running the genetic algorithm to find a good solution for the RUL values predicted in the prediction part we can see how fastly the algorithm converges into an optimal solution.

There are individuals obtained randomly in the first generation that already put the fitness value down to zero. Meaning that there are no penalty costs carried with the schedule solution selected by the algorithm and that it is not a good enough solution but an optimal solution.

Because of planning the horizon of the scheduling for only 30 days, the number of engines in the dataset which has a RUL fewer than that figure is small enough to get carried by the four teams on time.

Schedule:

	RUL	Engine_id	Team	Start_date	End_date	Penalty_cost	Total_penalty_cost
0	29	24	A	3	5	0	0
1	20	31	A	15	17	0	0
2	12	35	A	8	10	0	0
3	15	56	A	1	2	0	0
4	16	66	A	4	5	0	0
5	29	68	B	15	18	0	0
6	26	76	B	9	11	0	0
7	20	81	B	12	20	0	0
8	16	91	B	7	15	0	0
9	21	92	A	2	9	0	0

Fig. 4 Feasible Individual obtained by the GA

As can be seen in *Fig 4* only 10 engines have a RUL of less than 30 days for which the four times are allowed to fix on time.

To better explore how the algorithm can handle the schedule of maintenance, working with a longer horizon of days for the schedule can be tested, or to work with more conservative measures as the one from the consultancy company.

2.3.- Comparison

When running the genetic algorithm in the consultancy predictions results are similar for the scheduling as can be seen in Fig 5

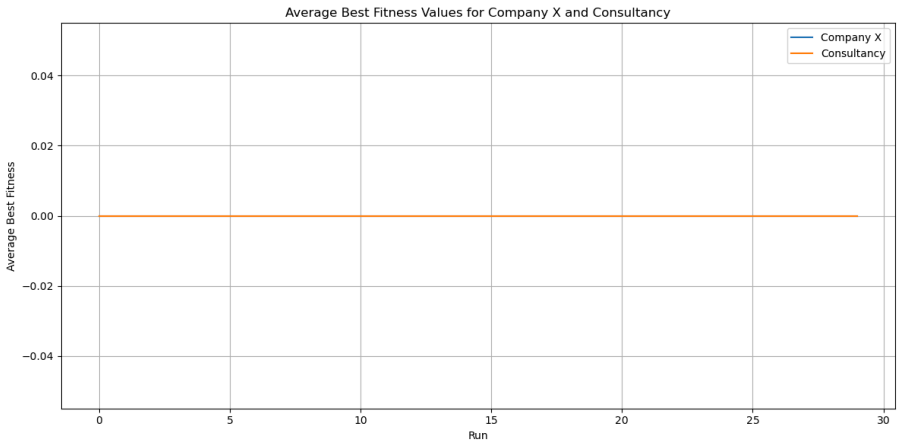

Fig 5. Average Best Fitness across 30 runs for RUL predictions. Ours and consultancy

We can see in the figure how an optimal solution is found by the algorithm for every run in both consultancy and our predictions. If the problem is translated to a real case scenario the conservative predictions from the consultancy would have any penalty cost incurred.