In a big city, congestion (traffic) often occurs in the morning or evening. As a result, the duration of travel to or from work is extended and still varies by about 30%. It can be observed that in the rain, congestion increases (travel time is on average X% longer), and in the event of snow, it increases even more (travel time on average Y% longer). We would like to predict such situations each morning one-two hours before the congestion. What kind of intelligent technology and how could it be used here?

Understand the problem (identify / explain where uncertainties may arise when developing / implementing a problem solving algorithm, where there are several ways to solve a problem, and it is unclear which way is better if one sees a commercial system solving this kind of problem; the tool was able to resolve).
Problem locations requiring a solution must be described. Identify the difficulties in solving the given problem. Which parts of the solution to the problem are not obvious. (0,25)

The problem consists in calculating the estimated average travel time stretching in one or two hours from the current moment. This value changes based on the starting time and the weather, while other parameters, such as location or transport mean are not considered or can be ignored. This also means that we cannot rely on data of the current state of the traffic but only on historic data.

Supposing the given values(30%, X%, and Y%) are averages, in order to get a more precise solution(compared to a probabilistic calculation), there is the necessity to calculate the correlation between the intensity of the causes and their effects.

In particular:

1. How much is time affected from the starting time? Which kind of distribution does the probability follow?
2. How much does the intensity of the rain or snow impact the final time?

Without the data used to calculate the given statistics, we can either create a very simple probabilistic formula(estimating the probability of those events as gaussian distributions or similar), or create our own model which will train and adjust as the time passes, whose efficiency can be then checked by its correlation to the given values.

As the problem is about calculating estimated time, and not weather forecasting, we are assuming that weather type, intensity and timing are already given as input information.

Divide the solution into several tasks. (0,25)
Provide clear and comprehensible formulations of tasks for different professionals (meaning not to formulate tasks for different professionals, but to explain tasks so that their meaning, the need to solve them can be seen by the project manager, financier, manager and, for example, HR manager). (0,5)

Tasks:

1. If possible: require the data used to get the statistical values
   This would give us the distribution of the probabilities, and allow us to use a simple probabilistic formula.
   In any case, they could be used to train our model.
2. Identify the best model
   As a prediction model, it will need to have a strong final correlation to the given percentages. There are no temporal dependencies that would affect our output, so our model won't need memory.
3. Create the model
   Create the mathematical model and code it accordingly
4. Retrieve the data
   Retrieve any past data or start collecting and recording new one. The data required is a lot, so it could be needed to use various gathering strategies.
5. Train the model
   Based on the acquired data, train the model to adjust its weights and give the most accurate solution
6. Test the model
   Use part of the data, as well as the given percentages to check the model accuracy

2. Ability to provide several reasoned solutions. (2)
Name 2-4 ways to solve the problem (to identify which methods or algorithms can be applied to the above problems; only those tasks where the intelligent system is required / possible to be selected). (0,5)

We don't require any long or short term memory, as there is no indication in the formulation of the problem, so for the same input we are expecting the same output. No internal state is required. For this reason I would use a Feedforward Neural Network for our prediction.

Some of the most popular FNN algorithms used for predictions are:

1. Backpropagation (Gradient Descent):

Backpropagation is the fundamental algorithm for training neural networks. It uses gradient descent to minimize the error between predicted and actual outputs by adjusting the weights of the network.

As shown by:
Tsong-Lin Lee,
Back-propagation neural network for long-term tidal predictions,
Ocean Engineering, Volume 31, Issue 2, 2004,
and:
Tsung-Lin Lee,
Back-propagation neural network for the prediction of the short-term storm surge in Taichung harbor, Taiwan,
Engineering Applications of Artificial Intelligence, Volume 21, Issue 1, 2008,

backpropagation has a great scalability factor both in terms of size of data processed than in applications.

2. Mini-Batch Gradient Descent:

Mini-batch gradient descent combines the efficiency of SGD (Stochastic Gradient Descent) with the stability of batch gradient descent by updating the model parameters using a small batch of training samples.
Pros: balances efficiency and stability of training.
Cons: Requires a large number of data
As shown by Wu Lizhen, Zhao Yifan, Wang Gang, Hao Xiaohong, in
A novel short-term load forecasting method based on mini-batch stochastic gradient descent regression model, Electric Power Systems Research, Volume 211, 2022,
MBGD can be used to elaborate a short term prediction based on data which have input and output size similar to ours (5 parameters in input and 1 as output), but requires a lot of data.

3. Levenberg-Marquardt:

Levenberg-Marquardt is a specialized optimization algorithm commonly used for training neural networks with a small to medium number of parameters. It combines aspects of gradient descent and the Gauss-Newton method.

In Sadig Mammadli,

Financial time series prediction using artificial neural network based on
Levenberg-Marquardt algorithm,
Procedia Computer Science, Volume 120, 2017,
It is shown how this algorithm can be applied for a prediction strategy, with a limited number of input parameters and few output results.

3. Ability to prove that your proposed solution is the best. (3)
Select the best solution from alternative ones. (0,5)
Provide advantages of the selected solution compared to other possible solutions (give reasons why the solution you choose is better than others. The other solutions are worse or less suitable for solving the problem). (2,5)

Let's review our necessities:
We have three main inputs: time, probability of rain, and probability of snow.
We can assume that both of them have a limited number of parameters, so the numeric inputs will be a limited number.
We can also assume that time will probably have a Gaussian-like distribution effect on the output, as there will be more people leaving around the critical hours, as well as some of the numeric parameters of the weather inputs(e.g. snow intensity).

Also, we don't know how our training data will be composed or retrieved.

With these considerations, my strategy choice is the Levenberg-Marquardt algorithm.
The Backpropagation (Gradient Descent) algorithm is the least efficient of the three, and its performance can be further degraded by a large dataset.
The MBGD algorithm, while converging faster than the normal Gradient Descent, relies heavily on mini-batch sets of data, which, while it's true that the focus is on a specific time period, introduces the probability of biased results, as there will probably also be more data collected during those criticals hours.
The LM algorithm is highly related to the gaussian model, which has a strong correlation with our inputs, is efficient with a small number of input parameters and relies on the entire dataset for its training.

4. Clearly explain, how the proposed solution should be implemented by a programmer. (4)

Name the inputs and outputs (when formulating the task it is necessary to predict what type and what data will be presented at the solution input and what data will be received at the output, and how these output data should be interpreted. we had one output, and its values of 0 or 1 indicated which class the image belongs to, whose attributes we sent to the entrance). (0.5)

Inputs:

1. Time of predicted leaving (from 0 to 1, normalized from 0 to 1440, indicating the minute of leaving)
2. Probability of rain at that time (from 0 to 1, where 1 is 100%)
3. Eventual intensity of rain at that time(from 0 to 1, where 1 is the max intensity)
4. Temperature at that time (from 0 to 1, where 0 is -20 and 1 is 40, Celsius degrees)

Output:

1. Percentage of how long the estimated time would increase(from 0 to 1, where 1 is 100%)

Draw a diagram of the solution algorithm (here you should write or draw something similar to the algorithm's diagram, albeit not 100% detailed). (1)