**Regression**

In a *regression* problem, we aim to **predict the output of a continuous value**, like a price or a probability. Contrast this with a *classification* problem, where we aim to select a class from a list of classes.

## Dataset used : The Auto MPG dataset

The dataset is available from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/).

## The model

### **Build the model**

Let's build our model. Here, we'll use a Sequential model with two densely connected hidden layers, and an output layer that returns a single, continuous value. The model building steps are wrapped in a function, build\_model, since we'll create a second model, later on.

### **Inspect the model**

Use the .summary method to print a simple description of the model.Now try out the model. Take a batch of 10 examples from the training data and call model.predict on it.

### **Train the model**

Train the model for 1000 epochs, and record the training and validation accuracy in the history object.

Let's update the model.fit call to automatically stop training when the validation score doesn't improve. We'll use an *EarlyStopping callback* that tests a training condition for every epoch. If a set amount of epochs elapses without showing improvement, then automatically stop the training.

## Conclusion

This notebook introduced a few techniques to handle a regression problem.

* Mean Squared Error (MSE) is a common loss function used for regression problems (different loss functions are used for classification problems).
* Similarly, evaluation metrics used for regression differ from classification. A common regression metric is Mean Absolute Error (MAE).
* When numeric input data features have values with different ranges, each feature should be scaled independently to the same range.
* If there is not much training data, one technique is to prefer a small network with few hidden layers to avoid overfitting.
* Early stopping is a useful technique to prevent overfitting.