**Introduction**

* **What is feature engineering?**

*“How do you get the most out of your data for predictive modeling?”*

This is the problem that the process and practice of feature engineering solves.

The feature is an attribute useful or meaningful for our modeling task. For a model problem, feature engineering is an important part of observation for learning about the structure of the problem. For instance, In computer vision, an image is an observation, but a feature could be a line in the image. In-text recognition, an article could be an observation, but a feature might be a single word.

Therefore, feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

* **Importance of feature engineering**:

Better features mean flexibility: the flexibility of good features will allow you to use less complex models that are faster to run, easier to understand and easier to maintain.

Better features mean simpler models: With good features, you could use the best characterize that underlying problem.

* **Process of feature engineering**

Machine learning process: problem definition, data processing, model preparation, model evaluation and model tuning. The feature engineering applies in the middle of data processing.

Data processing includes:

Select data (integrate data, normalize data..)

Preprocess data (format, clean, make the data sample to work with)

Transform data (Feature engineering happens here)

Model Data: Create models, evaluate them and tune them

Feature engineering process:

Feature extraction

Feature selection

Model evaluation: estimate model accuracy on unseen data using the selected features.

**Text data**

**Image data**

**//Binarizing:** converts the image array into 1s and 0s. This is done while converting the image to a 2D image. Even gray-scaling can also be used. It gives you a numerical matrix of the image.

**//Blurring:** Blurring algorithm takes the weighted average of neighbor pixels to incorporate surroundings color into every pixel. It enhances the contours better and helps in understanding the features and their importance better.

1. Greyscale values as features:

Signal channel image: convert the image array to greyscale, transform the 2D pixel matrix to a 1D array, and use these raw pixels as the features, it is the simplest way to create the features from the image.Three channels image: take the mean pixel value of the three channels as the 2D pixel matrix, then transfer to a 1D array. Each element in the array represents a feature.

Multiple channels image: for a multiple channels image, we can just simplify it into several 2D pixel matrices. After transforming, these matrices can be shown as the arrays we need.

1. Extracting edge features:

People can recognize the objects in an instant – a dog, a car and a cat. What are the features that you considered while differentiating each of these images? The shape could be one important factor, followed by color, or size. For the machine, the similar idea is to extract edges as features and use that as the input for the model.

As we know, an image is represented in the form of numbers. So, we will look for pixels around which there is a drastic change in the pixel values.

There are various kernels that can be used to highlight the edges in an image. For instance, the Prewitt kernel.

1. ORB use Opencv

**Conclusion**