1. **Data preparation**

**Numpy:** allows us to work with arrays.

**Matplotlib:** allows us to plot charts and graphs

**Pandas:** allows us to plot the dataset and create a matrix of features or preprocess your dataset

**Features/independent variables –** variables containing the properties that you will use to make a prediction which is your **dependent variable vector.**

**iloc (locate indexes) –** take the indexes of the columns and rows we want to extract from the dataset. A trick to take all the rows is by using a **colon** (**:**) which stands for a range. To separate the rows from the columns, use a **comma**

A range in python includes the lower bound but includes the upper bound

dataset.iloc[:, :-1].values : in iloc the order is rows, columns

**ct = ColumnTransformer(transformers=, remainder=”passthrough”) : transformers-** Specifies the kind of transformation we want to do and on which indexes of the columns we want to transform, remainder – specify that we actually want to keep the columns that won’t be applied some transformations. **Passthrough** enables us to **not only** keep the three columns resulting from one hot encoding eg 100, 010, 001

**fit\_transform** method doesn’t return it’s output as a numpy array whereby models as a must need the data to be in the form of a numpy array, so you use

**Feature scaling**: scaling all your variables or features to make sure they all take values in the same scale.

Don’t do feature scaling before splitting the dataset to prevent information leakage to the test set

**Note: Standardization works all the time –** puts values of features between -3 and +3 or -2 and +2, while **normalization works when you have a normal distribution of your values –** puts values between 0 and 1