Difference b/w batch Gradient Descent and Stochastic Gradient Descent:

Batch Gradient Descent (BGD)– slower and heavy algorithm as it loads up all the data into the memory and waits for all the rows to complete its calculation. It is deterministic algorithm unlike stochastic which is random in nature.

In BGD if we start the NN with the same set of weights then the number of iterations when starting the BGD every time will be the same and the same results every time we run this algorithm but with stochastic GD even if we start the training with the same weights the next row might get a different weights to work with.

Combo of both these is called mini batch gradient descent method

Pseudo inverse

In Mathematics, inverse is only found for square matrices but not all sq matrixes have an inverse. Therefore, we find a pseudo inverse and work with it.

In machine learning, not all real data have an inverse, necessarily the dataset vectors do not form a square matrix. There are many types of pseudo inverse.

Using Single Value Decomposition (SVD) we can find a pseudoinverse.

Scikit learn - Column Transformer

With ColumnTransformer one can apply many transformations on different columns. As the columns are heterogenous to each other in the dataset, we can apply respective column transformations (different columns having different transformers) using a single ColumnTransformer object. Also, something with the OneHotEncoder here, previously developers to use to first use LabelEncoder on a column before using OneHotEncoder on it. With the ColumnTransformer coming into the picture if we just mention OneHotEncoder as our transformer the library internally takes care of the LabelEncoder part.

sklearn.model\_selection.train\_test\_split

train\_test\_split shuffles the dataset before doing the split into training and test data.

To control this shuffling, you can make use of random\_state parameter.

sklearn.model\_selection.train\_test\_split(*\*arrays*, *test\_size=None*, *train\_size=None*, ***random\_state****=None*, *shuffle=True*, *stratify=None*)

sklearn.preprocessing.StandardScaler

Why only transform() is used in test data for scaling unlike training data which makes use of fit\_transform() ?

fit() is used to compute the parameter needed for transformation and transform() is for scaling the data to convert into standard format for the model.

fit\_transform() is combination of two which is doing above work in efficiently.

Since fit\_transform() is already computing and transforming the training data only transformation for testing data is left, since parameter needed for transformation is already computed and stored, only transformation() of testing data is left therefore only transform() is used instead of fit\_transform().