**Data Cleaning:**

1. **Dealing with missing values**
2. **Dealing with outliers**
3. **Correcting typos**
4. **Grouping sparse classes**
5. **Dropping duplicates.**

**Missing values:**

<https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>

There are three main types of missing data:

1. Missing completely at random (MCAR)
2. Missing at random (MAR)
3. Not missing at random (NMAR)

* **Do Nothing:** Let the algorithm handle the missing data. Some algorithms can factor in the missing values and learn the best imputation values for the missing data based on the training loss reduction (ie. XGBoost). Some others have the option to just ignore them (ie. LightGBM — use\_missing=false). However, other algorithms will panic and throw an error complaining about the missing values (ie. Scikit learn — LinearRegression). In that case, you will need to handle the missing data and clean it before feeding it to the algorithm by either dropping rows/columns.

* **Using Mean/Median:**

Pros:

* 1. Easy and fast.
  2. Works well with small numeric datasets.

Cons:

1. Doesn’t factor the correlations between features. It only works on the column level.
2. Will give poor results on the encoded categorical features(do not use it on categorical features).
3. Not very accurate.
4. Doesn’t account for the uncertainty in the imputations.

* **Using Constant/Zero:**

Pros:

* 1. Works well with categorical features.

Cons:

1. Doesn’t factor the correlations between features. It only works on the column level.
2. Can introduce bias in the data.

* **Imputation using k-NN:** The k nearest neighbours is an algorithm that is used for simple classification. The algorithm uses ‘feature similarity’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. This can be very useful in making predictions about the missing values by finding the k’s closest neighbours to the observation with missing data and then imputing them based on the non-missing values in the neighbourhood.

It creates a basic mean impute then uses the resulting complete list to construct a KDTree. Then, it uses the resulting KDTree to compute nearest neighbours (NN). After it finds the k-NNs, it takes the weighted average of them.

Pros:

* 1. Can be much more accurate than the mean, median or most frequent imputation methods (It depends on the dataset).

Cons:

1. Computationally expensive. KNN works by storing the whole training dataset in memory.
2. K-NN is quite sensitive to outliers in the data (unlike SVM)

* **Multivariate Imputation by Chained Equation:**
* **Imputation using Deep Learning(Datawig)**

Pros:

* 1. Quite accurate compared to other methods.
  2. It has some functions that can handle categorical data (Feature Encoder).
  3. supports CPUs and GPUs.

Cons:

1. Single Column imputation.
2. Can be quite slow with large datasets.
3. You have to specify the columns that contain information about the target column that will be imputed.

* **Regression Imputation:**
* **Stochastic Regression Imputation:** It is quite similar to regression imputation which tries to predict the missing values by regressing it from other related variables in the same dataset plus some random residual value.
* **Fancy Imputer:**
* **Hot-Deck Imputation:**

**Dealing with Outliers:**

* Boxplots.
* Cook’s distance.
* Z-Score imputation.
* Remove outliers
* Winsorizing – replacing extremes with minimum and maximum percentiles.
* Discretization – binning – dividing continuous variable into discreet groups.

**Grouping Sparse Classes:** Categorical features can often have a large number of distinct values, some of which have a low frequency. Many machine learning algorithms can struggle with too many features, this is referred to as the curse of dimensionality. To address this, you may wish to group qualitatively similar values. This is likely to be a manual effort working with a subject matter expert.

**Transformations:**

<https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py>

* Categorical Encoding: One-hot, Label, Hashing, Target encoding etc.
* Dealing with Skewed data
* Bias mitigation: Examples of preprocessing bias mitigation algorithms are Reweighing, Optimized preprocessing, Learning fair representations and Disparate impact remover.
* Scaling
* Rank transformation
* Power functions.

**Dimensionality Reduction:**

1. PCA
2. ICA
3. Feature Selection