**Three Critical Elements of Data Preprocessing — Part 3**

The backbone of modeling in data science.

In this article, I will discuss data transformation, a critical element of the data preprocessing step in the data science project life cycle.

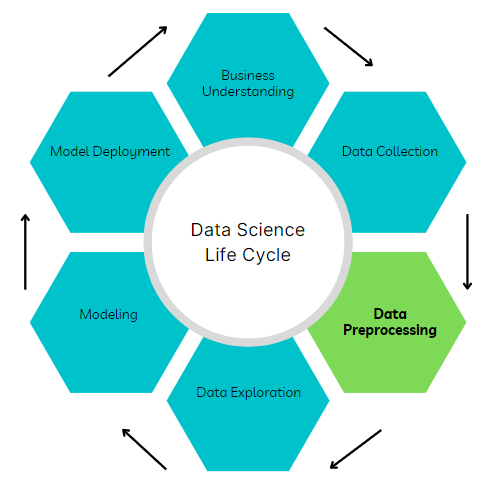
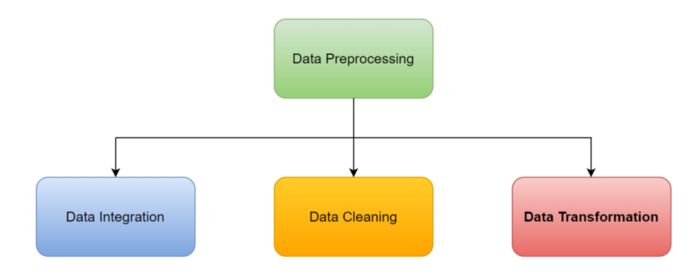


Image by author

Data preprocessing is the process of converting raw data from different sources into a refined form that can be used to derive actionable insights. It entails integration, cleaning, and transformation.



In part 1 of this series, we discussed data integration, combining data from different sources to obtain a dataset with all available relevant features and examples like a salad bowl 😊. Details can be found below:

**[Three critical elements of data preprocessing — Part 1](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-1-3c16f46f8ced" \t "_blank)**

[The backbone of modeling in data science.](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-1-3c16f46f8ced" \t "_blank)

[towardsdatascience.com](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-1-3c16f46f8ced" \t "_blank)

Subsequently, we explored data cleaning, the process of finding and fixing duplicated, corrupted, and missing data in a collected dataset. Details can be found here:

**[Three Critical Elements of Data Preprocessing — Part 2](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-2-2078110ae8e7" \t "_blank)**

[The backbone of modeling in data science.](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-2-2078110ae8e7" \t "_blank)

[towardsdatascience.com](https://towardsdatascience.com/three-critical-elements-of-data-preprocessing-part-2-2078110ae8e7" \t "_blank)

Here, I will cover the last piece of the data preprocessing puzzle, data transformation, from the point of view of modeling in data science.

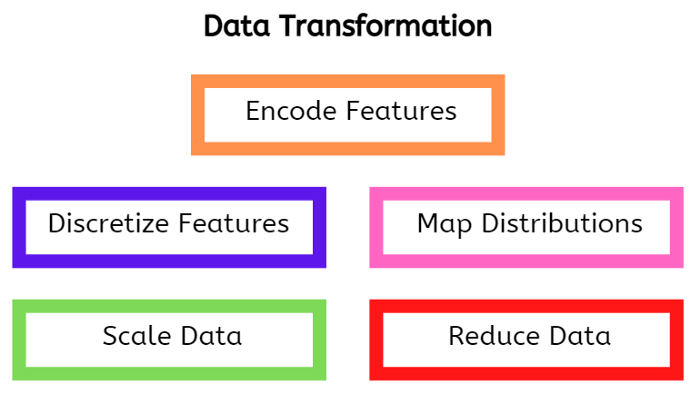
**Data Transformation**

In the context of machine learning, data transformation is the process of converting data into a suitable format or structure that best represents the data patterns and is amenable to model fitting.

We can squeeze more juice out of the data if we properly apply transformations before modeling.

Machine learning algorithms accept different data formats and types although numerical data is the most acceptable format. Where numerical data are numbers that can be decimals or integers and have values that range from — infinity to + infinity.

By contrast, categorical data are qualitative features that are represented as strings (e.g. a status feature may have “on” or “off” as unique categories). Datasets may also contain the Boolean data type with “True” or “False” feature values.



The following data transformations are generally applicable and highly valuable for modeling in data science.

**Feature Encoding**

This is the process of converting categorical data into numerical data. Two main approaches exist namely, ordinal and one-hot encoding.

*Ordinal encoding:*In this case, the hierarchy in the categorical data is maintained after transformation. For example, a risk level feature may be transformed as follows: Low risk — 0, Medium risk — 1, High risk — 2. The Pandas library’s [ordinal encoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html) can be used for this operation.

*One-hot encoding:*This approach is used when there is no order in the categorical feature and it has few unique categories (low cardinality). Since a new column is created for each category, the data size can grow significantly when this method is applied to a feature with high cardinality. Cardinal encoding may be applied in this case as described [here](https://pycaret.gitbook.io/docs/get-started/preprocessing/data-preparation" \l "cardinal-encoding" \t "_blank).

**Discretization**

Continuous data may be better presented to an algorithm by creating class intervalsthat discretize the data. For example, a set of age ranges (0–12: 0, 13 — 19: 1, 20–35: 2, 35+:4) may be created from continuous age data with the transformed data having better predictive power. Useful Pandas methods for discretization include [cut](https://pandas.pydata.org/docs/reference/api/pandas.cut.html" \t "_blank) and [qcut](https://pandas.pydata.org/docs/reference/api/pandas.qcut.html" \t "_blank).

In addition, binarization is a special type of discretization which involves assigning feature values to any of two groups namely zero or one. The [binarize](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.binarize.html" \t "_blank) tool in the Scikit-learn preprocessing module can be used for this operation.

**Distribution Mapping**

Some machine learning algorithms perform better when the input data has specific distributions (e.g normal distribution). Notable approaches for mapping distributions include:

*Uniform mapping:* This involves mapping the data to a uniform distribution with equally likely outcomes. More details can be found [here](https://scikit-learn.org/stable/modules/preprocessing.html" \l "mapping-to-a-uniform-distribution" \t "_blank). In addition, the [quantile transformer](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html" \l "sklearn.preprocessing.QuantileTransformer" \t "_blank) tool in Scikit-learn can be used for this action.

*Gaussian mapping:* Here, the data is mapped as close as possible to a normal distribution with the mean, median, and mode being approximately the same. More details can be found [here](https://scikit-learn.org/stable/modules/preprocessing.html" \l "mapping-to-a-gaussian-distribution" \t "_blank). Additionally, the [power transformer](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html" \l "sklearn.preprocessing.PowerTransformer" \t "_blank) tool in Scikit-learn can be used for this operation.

Arguably, it is not always a good idea to transform the data distribution due to unintended effects such as masking the true behavior of the residuals. Additional explanations may be found [here](https://medium.com/peter-flom-the-blog/why-you-should-probably-not-transform-your-data-9ea098dcaddf).

**Data Scaling**

This process ensures that features with different units and magnitude ranges are converted to the same scale to avoid misrepresentation of the data to the model. Several data scaling methods exist but I will only briefly describe a few here for brevity with a link provided for further study if interested.

*Standardization:* This involves subtracting the mean and dividing by the standard deviation. It ensures that the data is centered around zero and scaled with respect to the standard deviation. More details can be found [here](https://scikit-learn.org/stable/modules/preprocessing.html" \l "standardization-or-mean-removal-and-variance-scaling" \t "_blank).

*Normalization:*This method ensures that the data values have a unit norm either for each observation or each feature. More details can be found [here](https://scikit-learn.org/stable/modules/preprocessing.html" \l "normalization" \t "_blank). Additionally, the [normalize](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html" \l "sklearn.preprocessing.normalize" \t "_blank) method in the Scikit-learn preprocessing module can be used.

*Scaling to a range:*Here, the data values for a given feature are mapped to a specific range. It typically involves subtracting by the mean and dividing by the difference between the min and max value of the feature. More details can be found [here](https://scikit-learn.org/stable/modules/preprocessing.html" \l "scaling-features-to-a-range" \t "_blank).

Other scaling methods include log scaling and clipping values using minimum and maximum thresholds. This free Google course gives a more in-depth explanation of data scaling (normalization). The effects of different scaling methods on data with outliers can be [here](https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html" \l "sphx-glr-auto-examples-preprocessing-plot-all-scaling-py" \t "_blank).

**Data Reduction**

Many new features may be created during data transformation for very good reasons. However, as the number of features increases, computational cost and time increase as well. In addition, an excessive number of features may result in issues such as overfitting. This issue is often called the curse of dimensionality.

Hence, we may need to reduce the number of features to efficiently build machine learning models while improving predictive performance.

Multi-collinearity (where two or more independent features are highly correlated) is another reason why features may be pruned. It affects the model performance and interpretability of especially non-tree-based algorithms such as linear regression. More details about multi-collinearity can be found [here](https://towardsdatascience.com/multicollinearity-why-is-it-bad-5335030651bf" \t "_blank).

Some methods for data reduction include:

*Principal component analysis:*This is a very useful method for mapping data features to a lower orthogonal dimensional space while preserving as much information in the data as possible. However, the transformed features are not as interpretable as the original features. Hence, they may not be used in some applications. More details can be found [here](https://scikit-learn.org/stable/modules/decomposition.html" \l "principal-component-analysis-pca" \t "_blank). In addition, the decomposition module in Scikit-learn has a [PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) tool that can be used for this action.

*Feature elimination:*In this case, the least relevant features are dropped while keeping the features with the most predictive power. The original features which are intuitive and mostly interpretable are used and this avoids the issue of interpretability encountered with PCA. This method can be performed recursively as described [here](https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html" \l "recursive-feature-elimination" \t "_blank).

**Conclusions**

In this article, we covered data transformation, the process of converting data into a format that is amenable to model fitting. This process may also include some data reduction strategies as discussed.

The data transformations explored here are relevant to most data science modeling applications but not exhaustive. It should be noted that there are other transformations that are specific to different fields such as tokenization for natural language programming and the Featurecreation of lag features for time series analysis.

This brings us to the end of this series on data preprocessing. Yay!! I hope you enjoyed the articles, until next time. Cheers!