**Categorical Data Encoding**

As we found during our initial data analysis, here were few categorical features in the given dataset. There are many machine learning algorithm which can support categorical feature in computation without any manipulations but there are many more which do not support. Machine learning algorithm use for this project does not support the categorical feature directly and requires further manipulation in the data. Therefore we had to figure out how to turn these categorical feature into numerical features for algorithm processing.

There are many ways to encode the categorical features into numerical. As with many other aspects of data science, there is no best approach for categorical data encoding. Every approach has its trade-off and potential impact on analysis outcome. Therefore we tried multiple ways to encode our data and measured its outcome by running logistic regression classifier with cross validation technique.

For our implementation, we used a built-in python package "**CategoryEncoders**", which provides different techniques to encode the categorical data - One-Hot, Ordinal, Binary, Backward Difference, BaseN, Hashing, Helmert, Leave-One-Out, Polynomial, Sum encoding and default dummy encoding. We then used the encoded data to run logistic classifier with cross validation to measure the encoding technique impact. Under mentioned is the output of our observation with random forest imputed dataset with z-score normalization.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Encoding Type** | **Dimensionality** | **accuracy** | **precision** | **recall** | **f1-score** |
| BackwardDifferenceEncoder | 89 | 81.70% | 28.13% | 83.42% | 42.08% |
| BaseNEncoder | 29 | 85.53% | 60.76% | 73.40% | 66.49% |
| BinaryEncoder | 29 | 85.41% | 60.66% | 73.00% | 66.26% |
| HashingEncoder | 14 | 84.31% | 48.62% | 76.36% | 59.41% |
| HelmertEncoder | 89 | 85.34% | 64.53% | 70.81% | 67.53% |
| LeaveOneOutEncoder | 13 | 85.36% | 56.21% | 75.57% | 64.47% |
| OneHotEncoder | 96 | 85.28% | 64.17% | 70.80% | 67.32% |
| OrdinalEncoder | 13 | 84.51% | 48.31% | 77.68% | 59.57% |
| PolynomialEncoder | 89 | 85.28% | 63.10% | 71.28% | 66.94% |
| SumEncoder | 89 | 85.42% | 64.30% | 71.19% | 67.57% |
| DummyEncoder | 89 | 85.51% | 64.12% | 71.58% | 67.65% |

According to the above observation, BaseNEncoder type encoding is giving us the highest performance. However, when running the same model on different versions of dataset, DummyEncoder results in consistent performer over other encoder types. Therefore, We choose Dummy Encoder to encode our categorical features.

**<Do we need to write description about the each encoding ?>**