Data Preparation, Analysis and Manipulation:

1. **Introduction:**

Our data preparation steps are outlined in the following diagram:

Type-checking and Define and Identify Missing Values

Removal of Zero Variance column

Removal of Correlated Columns

Outlier Clipping

Missing Value Imputation

Feature Engineering

Data Cleaning post- Feature Engineering

‘Customer Churn’ Data from Kaggle

Encoding

Final Clean Data

Scaling

Fig-1: Summary of the Data Preparation Steps

1. **Data Cleaning Approaches for the Customer Churn Dataset:** 
   1. Type checking and consistency of the fields:

No field with inconsistent data type was found in our base data set. Values across the columns were consistent in the sense that there was no type mismatch between the values in the same column.

* 1. Define and Identify Missing Values:

We programmatically identified the missing values in the columns. We found almost 1.57% of the cell values as missing or ‘NA’. We imputed the missing values using various algorithm as explained later in this section.

* 1. Removal of Zero Variance Columns:

We programmatically inspected the data set for any zero variance columns, as zero variance column will not add any feature in determining the output classification. However, we didn’t find any column having near zero variance (within a reasonable threshold). So, no column was eliminated out of this step.

* 1. Removal of Correlated Columns:

We created the correlation matrix for all numerical fields to check if there is any significant correlation present between the fields and removed the fields that had significant correlation. Correlated features can impact the performance of the classifier, and removing these fields help removing the redundancies from the feature set. We applied this step only for numerical variables.

* 1. Data imputation for missing values:

Imputation is the process of estimating or deriving values for fields where data is missing. There is a vast body of literature available on various imputation techniques. For our data-preparation, we took scenario-based approaches for each field/feature that had missing values. Fig-2 schematically outline our data imputation techniques that we adopted.

Max Frequency

Identify the Missing Values

Numerical Features

Categorical Features

Mean Value

Summary and Statistics

Random sampling of values with highest frequencies

**Sci-kit Learn**

**Univariate Imputer: Simple Imputer, Multivariant Imputer:** [**Iterative Imputer**](https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html#sklearn.impute.IterativeImputer)

Fig-2: Outline of our Missing Value Imputation Algorithm

**Data Imputation for Numerical Variable:**

Mean: For numerical variables, we used mean of the values in the corresponding column to impute the missing values.

**Data Imputation for Categorical Variable:**

For Categorical variables we used the value with highest frequency and at times randomly sampled from the values wit highest frequency to impute the missing value.

**Package Based Data Imputation:**

We also selectively used the simple imputer and iterative imputers from Python Sci-kit learn package for imputation of missing values for univariate and multivariate features respectively.

* 1. Feature Engineering:

We used various feature engineering techniques to boost the dataset and evaluated classifier performance based on the feature rich dataset. Depending on the outcome, we re-engineered the features, and at times, eliminated a few features and/or retained some others.

**Deriving new features based on existing features:**

We tried to derive new features based on the existing features that presumably could enrich the feature set. For example, we derived a few new fields as shown below:

*#mou\_Mean/rev\_Mean --> mean\_per\_minute\_charge*

telo\_df['fe\_mean\_per\_minute\_charge']=telo\_df['mou\_Mean']/telo\_df['rev\_Mean']

*#rev\_Mean-totmrc\_Mean-- > mean\_extra\_amount\_they\_pay*

*telo\_df['fe\_mean\_extra\_amount\_they\_pay']=telo\_df['rev\_Mean']-telo\_df['totmrc\_Mean']*

The initial derived features are tabulated in Table-1.

|  |  |  |  |
| --- | --- | --- | --- |
| Seq # | Derived Features | Source Features | Derivation Formula |
| 1 | mean\_per\_minute\_charge | mou\_Mean, rev\_Mean | mou\_Mean/ rev\_Mean |
| 2 | mean\_extra\_amount\_they\_pay | rev\_Mean, totmrc\_Mean | rev\_Mean - totmrc\_Mean |
| 3 | mean\_allocated\_calls | mou\_Mean, ovrmou\_Mean | mou\_Mean - ovrmou\_Mean |
| 4 | mean\_profit | ovrrev\_Mean, vceovr\_Mean, datovr\_Mean | ovrrev\_Mean + vceovr\_Mean + datovr\_Mean |
| 5 | tot\_mean\_failed\_calls | drop\_vce\_Mean, drop\_dat\_Mean, blck\_vce\_Mean, blck\_dat\_Mean | drop\_vce\_Mean + drop\_dat\_Mean + blck\_vce\_Mean + blck\_dat\_Mean |
| 6 | perc\_of\_success\_calls | comp\_vce\_Mean, plcd\_vce\_Mean | comp\_vce\_Mean/ plcd\_vce\_Mean |
| 7 | mou\_cvce\_diff\_vce\_mean | mou\_cvce\_Mean, comp\_vce\_Mean | mou\_cvce\_Mean - comp\_vce\_Mean |
| 8 | mou\_rvce\_diff\_vce\_mean | mou\_rvce\_Mean, recv\_vce\_Mean | mou\_rvce\_Mean - recv\_vce\_Mean |
| 9 | tot\_revenue\_per\_call | totrev, totcalls | totrev/ totcalls |
| 10 | tot\_mou\_per\_call | totmou, totcalls | Totmou/ totcalls |
| 11 | tot\_charge\_per\_call | tot\_revenue\_per\_call, tot\_mou\_per\_call | tot\_revenue\_per\_call \* tot\_mou\_per\_call |
| 12 | tot\_revenue\_adj | totrev, adjrev | totrev - adjrev |
| 13 | tot\_mou\_adj | totmou, adjmou | totmou - adjmou |
| 14 | tot\_calls\_adj | totcalls, adjqty | totcalls - adjqty |
| 15 | avg\_revenue\_per\_call | avgrev, avgqty | avgrev/ avgqty |
| 16 | avg\_mou\_per\_call | avgmou, avgqty | avgmou/ avgqty |
| 17 | avg\_charge\_per\_call | avg\_revenue\_per\_call, avg\_mou\_per\_call | avg\_revenue\_per\_call \* avg\_mou\_per\_call |

Table-1: Feature Engineering and Derivation of New Features

**Evaluate the Classifier Performance based on the new features:** We evaluated the classifier performance based on the input dataset new features and checked whether there is any marked improvement, and in case there was no marked improvement we discarded the feature as redundant.

**Finalize the Feature Set:** We iteratively evaluated the classifier performance and finalized the incremental feature set based on the classifier performance in each iteration.

Retain the feature

Derive new features based on heuristics

Feature Boosting

ML Algorithms

Classifier Performance Evaluation

Eliminate and Re-engineer

Improvement

No Improvement

Fig-3: Feature Engineering and Feature Boosting Algorithm

* 1. Data Cleaning post- Feature Engineering:

Once the incremental features have been added to enrich the dataset, we further removed the outliers based on a threshold (5%). We again, removed any zero variance columns and/or correlated columns from feature-rich dataset.

* 1. Encoding:

We used following encoding schemes for categorical variables:

**Frequency Encoding:** For some of the nominal features, we used frequency encoding.

**One-Hot encoding:** For non-ordinal categorical variables, we have used one-hot encoding to split columns with multiple categorical values into multiple columns with values ‘0’ and/or ‘1’.

**Weight of Evidence:** For a few nominal features, we leveraged Weight of Evidence encoding scheme based on percentage or weights of a particular value of the feature.

* 1. Scaling:

Various scaling mechanism are in use for Machine Learning input dataset. The most commonly used ones are: Min-max normalization, Mean Normalization, Z-score normalization and Scaling to unit length.

**For our dataset, we used min-max normalization:**

Reference:

1. Edwin de Jonge, Mark van der Loo: An introduction to data cleaning with R, Statistics Netherlands
2. <https://www.kdnuggets.com/2018/05/packt-tackle-common-data-cleaning-issues-r.html>
3. <https://blog.dominodatalab.com/manual-feature-engineering/>
4. <https://en.wikipedia.org/wiki/Feature_engineering>
5. https://en.wikipedia.org/wiki/Data\_transformation