Data Preparation, Analysis and Manipulation:

1. **Introduction:**

Our data preparation steps are outlined in the following diagram:

Type-checking and Defining and Identifying the Missing Values

Outlier Clipping

Removal of Zero Variance Columns

Missing Value Imputation

Feature Engineering and Encoding

Scaling

‘Customer Churn’ Data from Kaggle

Final Clean Data

Removal of Correlated Columns

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:

The very first step was validating the data sanity. 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. Defining and Identifying the Missing Values:

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

* 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 outlines our data imputation techniques that we adopted.

Identify the Missing Values

Numerical Features

Categorical Features

Summary and Statistics

Mean Value

Max Frequency

Random sampling of values with highest frequencies

Custom

Package Based

Univariate (Simple Imputer)

Multivariate(Iterative Imputer)

Fig-2: Outline of our Missing Value Imputation Algorithm

**Data Imputation for Numerical Variables:**

**Mean**: For numerical variables, we used ‘mean’ of the non-missing values in the feature dimension to impute the missing values.

**Data Imputation for Categorical Variables:**

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

**Package Based Data Imputation:**

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

* **Univariate Feature Imputation:** The **[SimpleImpute](https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html" \l "sklearn.impute.SimpleImputer" \o "sklearn.impute.SimpleImputer)**[r](https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html" \l "sklearn.impute.SimpleImputer" \o "sklearn.impute.SimpleImputer) class from sci-kit learn package provides basic strategies for imputing missing values leveraging an univariate algorithm. The univariate algorithm imputes values in the i-th feature dimension in the feature space using the non-missing values in that feature dimension only. Missing values can be imputed with a provided constant value, or using the statistics (mean, median or most frequent) of each column in which the missing values are located. This class also allows for different missing values encodings. For univariate imputation, we used mostly mean of the non-missing values.
* **Multivariate Feature Imputation:** The [**IterativeImputer**](https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html#sklearn.impute.IterativeImputer) class from sci-kit learn package imputes missing values leveraging a multivariate algorithm. A multivariate feature imputation algorithm models each feature with missing values as a function of other features, and uses that estimate for imputation. It does so in an iterated round-robin fashion: at each step, a feature column is designated as output y and the other feature columns are treated as inputs X. A regressor is fit on (X, y) for known y. Then, the regressor is used to predict the missing values of y. This is done for each feature in an iterative fashion, and then is repeated for max\_iter imputation rounds. The results of the final imputation round are returned, and based on any other external filter, is used as an imputed value for the missing values corresponding to a feature dimension.
  1. Feature Engineering and Encoding:

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.

**Feature Engineering for Numerical Features:**

For numerical features, we adopted an iterative approach to identify and retain the best features that we believed will enrich our dataset and help in classification. We leveraged following 3-steps approach:

* **Deriving new features based on existing features using aggregate functions and statistics:**

We tried to derive new features based on the existing features that presumably could enrich the feature set. We created new features with the statistics aggregated by the parent variables for all numeric features. Each observation of the parent variable will have one row in the dataset with the parent variable as the index followed by removal of duplicate values. The derived features were then fed to an iterative flow of classifier performance evaluation and decision making as to whether the feature will be retained or discarded. An example of how we derived a few custom features is 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']*

* **Evaluate the Classifier Performance based on the new features:** We then evaluated the classifier performance based on the input dataset enriched with the 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.

Fig.3 schematically represents our feature engineering scheme for numerical features

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 for Numerical Features

**Feature Engineering for Categorical Features:**

* **Encoding:**

We first encoded the categorical features using 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 used one-hot encoding to split columns with multiple categorical values into multiple columns with values ‘0’ and/or ‘1’.

* **Feature Extraction based on Feature Interactions:** We attempted to derive some new categorical variables based on feature interaction between encoded categorical variables.
* **Evaluate the Classifier Performance based on the new features:** Similar to numerical feature engineering, we evaluated the classifier performance based on the enriched dataset to determine whether we would retain the categorical feature or drop based on the improvement/deterioration of the classifier performance level.

Fig.4 schematically represents our feature engineering scheme for categorical features

Retain the feature

Encoding Categorical Features

Derive New Features leveraging Feature Interactions

ML Algorithms

Classifier Performance Evaluation

Eliminate and Re-engineer

Improvement

No Improvement

Fig-4: Feature Engineering for Categorical Features

**Derived Features:** Based on our feature engineering approach elucidated earlier, we derived following new features to enrich our dataset.

1. **Mean Per Minute Charge:** The new feature represents the mean of charge for the calls per minute. This value has been derived based on numerical features: ‘Mean number of monthly minutes of use’ and ‘Mean monthly revenue(charge)’ and has been derived as:

**Mean Per Minute Charge = Mean number of monthly minutes of use/ Mean monthly revenue(charge)**

1. **Mean Extra Amount that Customer Pays:** The new feature represents the average extra amount customer pays per month. This field is derived as follows:

**Mean Extra Amount that Customer Pays = Mean monthly revenue(charge) - Mean total monthly recurring charge**

This feature provides an insight of how much extra charge a customer is paying and hence eventually can be a determinant for a customer churn.

1. **Mean Allocated Calls:** This feature signifies total minutes of calls per month and is derived from two other features: ‘Mean number of monthly minutes of use’ and ‘Mean overage minutes of use’. The field is derived as follows:

**Mean Allocated Calls = Mean number of monthly minutes of use + Mean overage minutes of use**

1. **Mean Monthly Profit:** This feature signifies total monthly profit of the telecom company and may impact the customer churn. The field is derived as:

**Mean Monthly Profit = Mean overage revenue + Mean revenue of voice overage + Mean revenue of data overage**

1. **Mean Failed/Dropped Calls:** This feature signifies total monthly profit of the telecom company and may impact the customer churn. The field is derived as:

**Mean Failed/Dropped Calls = Mean number of dropped (failed) voice calls + Mean number of dropped (failed) data calls + Mean number of blocked (failed) voice calls + Mean number of blocked (failed) data calls.** This field is an important indicator of overall service quality.

1. **Percentage of successful calls:** This feature is an important indicator of the service quality. The field is derived as:

**Percentage of successful calls = Mean number of completed voice calls / Mean number of attempted voice calls placed**

1. **Unrounded to rounded completed voice call difference:** This feature is an important indicator of the service quality. The field is derived as:

**Unrounded to rounded completed voice call difference = Mean unrounded minutes of use of completed voice calls- Mean number of completed voice calls**

1. **Unrounded to rounded completed received voice call difference:** This feature is also an important indicator of the service quality. The field is derived as:

**Unrounded to rounded completed received voice call difference = Mean unrounded minutes of use of received voice calls - Mean number of received voice calls**

1. **Mean total revenue per call:** This feature signifies mean charge per call. The feature is derived as:

**Mean total revenue per call = Total revenue/Total number of calls over the life of the customer**

1. **Mean Total minutes of use per call:** This feature signifies mean total minutes per call and is an important feature identifying usage. The feature is derived as:

**Mean total revenue per call = Total minutes of use over the life of the customer/Total number of calls over the life of the customer**

1. **Total Revenue Adjustment:** This field is derived as: **Total Revenue Adjustment** = **Total Revenue - Billing adjusted total revenue over the life of the customer**
2. **Total minutes of use adjusted**: This field is derived as: **Total minutes of use adjusted = Total minutes of use over the life of the customer - Billing adjusted total minutes of use over the life of the customer**
3. **Total calls adjusted**: This field is derived as: **Total calls adjusted = Total calls - Billing adjusted total number of calls over the life of the customer**
4. **Average revenue per call**: This field is derived as: **Average Revenue per call = Average monthly revenue over the life of the customer/Average monthly number of calls over the life of the customer.** This field is an important indicator of the average charge the customer is incurring.
5. **Average minute of use per call**: This field is derived as: **Average minute of use per call = Average monthly minutes of use over the life of the customer/Average monthly number of calls over the life of the customer.** This field is an important indicator with respect the usage.
6. **Average charge per call:** This field is derived out of two derived features using the following formula:

**Average charge per call = Average revenue per call \* Average minute of use per call**

* 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. The below code segment removes the zero variance columns from the input dataset with a threshold of 0.01.

telo\_df=rem\_var(telo\_df,0.01)

print(telo\_df.shape)

gc.collect()

telo\_df.head()

* 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.

*#Removing Correlated Columns.*

telo\_df=correlated(telo\_df,0.80)

print(telo\_df.shape)

gc.collect()

telo\_df.head()

* 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>
6. <https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>
7. https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf