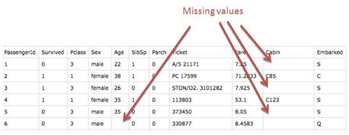

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Pic Credit : AnalyticsIndia Magazine

Handling Missing Data

Missing Values occur because of various reasons, such as, corrupt data, failure to load the information, or incomplete extraction. Handling the missing values is one of the greatest challenges faced by analysts, because making the right decision on how to handle it generates robust data models. Missing values cause a lot of problems when using the dataset for any machine learning algorithm.They hinder with data analysis and data visualization.

In this article , lets look at the following methods to handle missing data

1.Drop all missing Values
2.Drop the values above a certain threshold
3.Imputation using mean ,median and mode
4.Imputation using forward fill and backward fill
5.sklearn Imputer for Numerical Variables
6.sklearn Imputer for Categorical Variables

Drop all missing Values

In this method , we delete a particular row if it has a null value for a particular feature .This method is used only when there are enough samples in the data set. It has to be ensured that there is no bias after data deletion.Removing the data will lead to loss of information which will not give the expected results while predicting the output.All observations that are having null values can be deleted using pandas dropna() method.

```
# Number of missing values in the data
df = df[df.columns[0:20]]
df.isna().sum()
```

```
encounter_id                0
patient_id                  0
hospital_id                 0
hospital_death              0
age                      4228
bmi                      3429
elective_surgery            0
ethnicity                1395
gender                     25
height                   1334
hospital_admit_source   21409
icu_admit_source          112
icu_id                      0
icu_stay_type               0
icu_type                    0
pre_icu_los_days            0
readmission_status          0
weight                   2720
albumin_apache          54379
apache_2_diagnosis       1662
dtype: int64
```

```
print("Orginal shape before dropna()" ,df.shape)
drop = df.dropna()
print("Sshape after dropna()" ,drop.shape)
```

```
Orginal shape before dropna() (91713, 20)
Sshape after dropna() (25349, 20)
```

Drop the values above a certain threshold

If the information contained in the variable is not that high, you can drop the variable if it has more than 50% missing values.In this method we are dropping columns with null values above a certain threshold

```
# Drop columns based on threshold limit
threshold = len(df) * 0.60
df_thresh=df.dropna(axis=1, thresh=threshold)
# View columns in the dataset
df_thresh.shape
```

Imputation using mean ,median and mode

This imputation method treats every variable individually, ignoring any interrelationships with other variables and is beneficial for simple linear models and NN. This method may not be suitable for tree based algorithms and this is an approximation which can add variance to the data set. The loss of the data can be negated by this method which yields better results compared to removal of rows and columns.

Null values are replaced with mean/median.mode in this method. This is the statistical method of handling Null values.The mean of the numerical column data is used to replace null values when the data is normally distributed. Median is used if the data comprised of outliers. Mode is used when the data having more occurences of a particular value or more frequent value.

```
impute_df = df[['age','bmi','height']]
mean_age = impute_df['age'].mean()
median_bmi = impute_df['bmi'].median()
mean_height = impute_df['height'].mean()
```

The null values are replaced with mean/median/mode values by using fillna() method

```
# Replace Null Values (np.nan) with mean
impute_df['age'].fillna(mean_age, inplace=True)

impute_df['height'].fillna(mean_height, inplace=True)

# Alternate way to fill null values with mean
impute_df['bmi'].fillna(median_bmi,inplace=True)

impute_df.isna().sum()
```

Imputation using forward fill and backward fill

Null values can also be replaced by it's previous value in the column which is called Backward fill or next occurring value in the column which is called Forward fill.The NaN value will remain even after forward filling or back filling if a next or previous value isn't available or it is also a NaN value.

```
#backward fill
fill_df['gender'].fillna(method='bfill',inplace = True)
# Forward fill
fill_df['ethnicity'].fillna(method='ffill',inplace = True)
fill_df.isna().sum()
```

sklearn Imputer for Numerical Variables

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
# Fit and transform to the parameters
imputer_skdf = pd.DataFrame(imputer.fit_transform(imputer_skdf))
imputer_skdf.columns = ['readmission_status','weight']
# Checking for any null values
imputer_skdf.isna().sum()
```

sklearn Imputer for Categorical Variables

```
# Replacing null values in Embarked with most frequent value
imputer = SimpleImputer(missing_values=np.nan,
strategy='most_frequent')
categ_df = pd.DataFrame(imputer.fit_transform(categ_df))
categ_df.columns =['ethnicity','gender']
# Value counts for Embarked column
print("ICU Stay Type - value Counts : \n "
,categ_df['ethnicity'].value_counts())
print("ICU Type - value Counts : \n " ,
categ_df['gender'].value_counts())
```