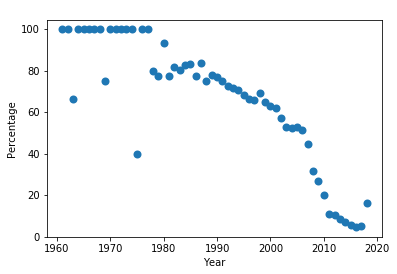
# **How to deal with missing values in a data set ?**

The real-world datasets consist of missing values, and a data scientist spends a major amount of time on data preparation, including data cleaning. Missing Value can be a result of unrecorded observations or data corruption.

**Types of Missing Data**

* Missing at Random (MAR) — It means that there is a relationship between the proportion of missing values and the observed data. For example,in the below graph we see that the proportion of missing values in the mileage column is correlated to the car’s manufacturing year.Therefore,this type of missing values in the data set can be predicted using other features.



Relationship Between Percentage of Missing Values and the Manufacturing Year of Car.

* Missing Completely at Random (MCAR)-It means that the proportion of missing values is unrelated to any observation in the data.An example of it is a weighing scale that ran out of batteries and as a result some of the data will be missing .
* Missing Not at Random (MNAR)- It means that the missing data is related to the factors that are unknown to us. For example, the weighing scale mechanism may wear with time, thereby producing more missing values as time progresses, but we might not note the same.If we have MNAR missing mechanism ,we need to understand why the data is missing, rather than straight away imputing them.

**Detecting Missing Values in Python**

Missing Values can be in the form of an empty string, NA or N/A or None.Pandas in python identifies all the NA or blank values in the data as NaN values.However, it doesn’t identify na, ?, n.a. or n/a.The NA or blank value formats in the data set can be detected using **df.isnull()** command. This method returns boolean response True in the presence of missing values in the data.

However,sometimes there might be a case where the missing values are in a different format in the data .For Instance, a column has missing values in n/a , \_ \_ or na format.An easy way for pandas to detect the missing values in a non- standard format in the data set while importing the data is to put all the types of missing value as a list.

missing\_values = ["n/a", "na", " \_ \_"]

df = pd.read\_csv("loan data.csv", na\_values = missing\_values)

In the data set **df.isnull().sum()** command is used to find the total number of missing values for each feature in the data.

**Visualizing Missing Values in Python**

Visualizing the missing values gives the analyst a good understanding of the distribution of NaN values in the data set.

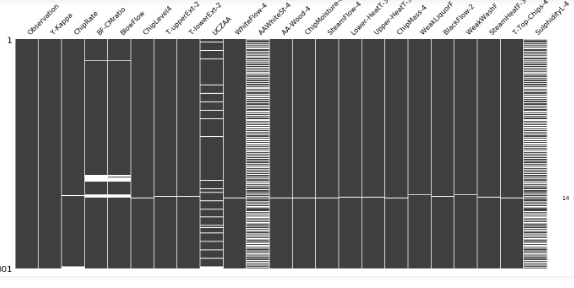
# Import the library

pip install missingno

import missingno as msno

# Visualize the missing values using a matrix

msno.matrix(df)

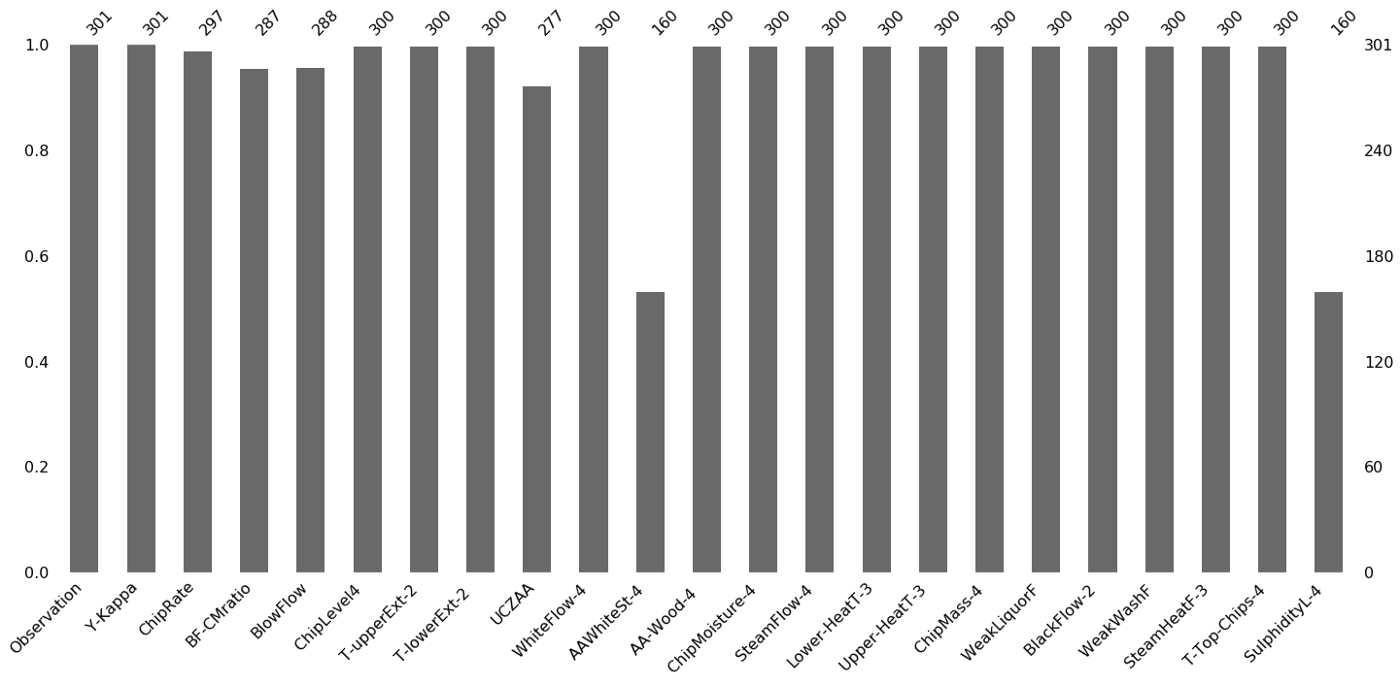


The pattern of missingness is similar for AAWhiteSt-4 and SulphidityL-4 columns

The count of missing values in each column is represented using a bar chart.

# Visualize missing values as a bar chart

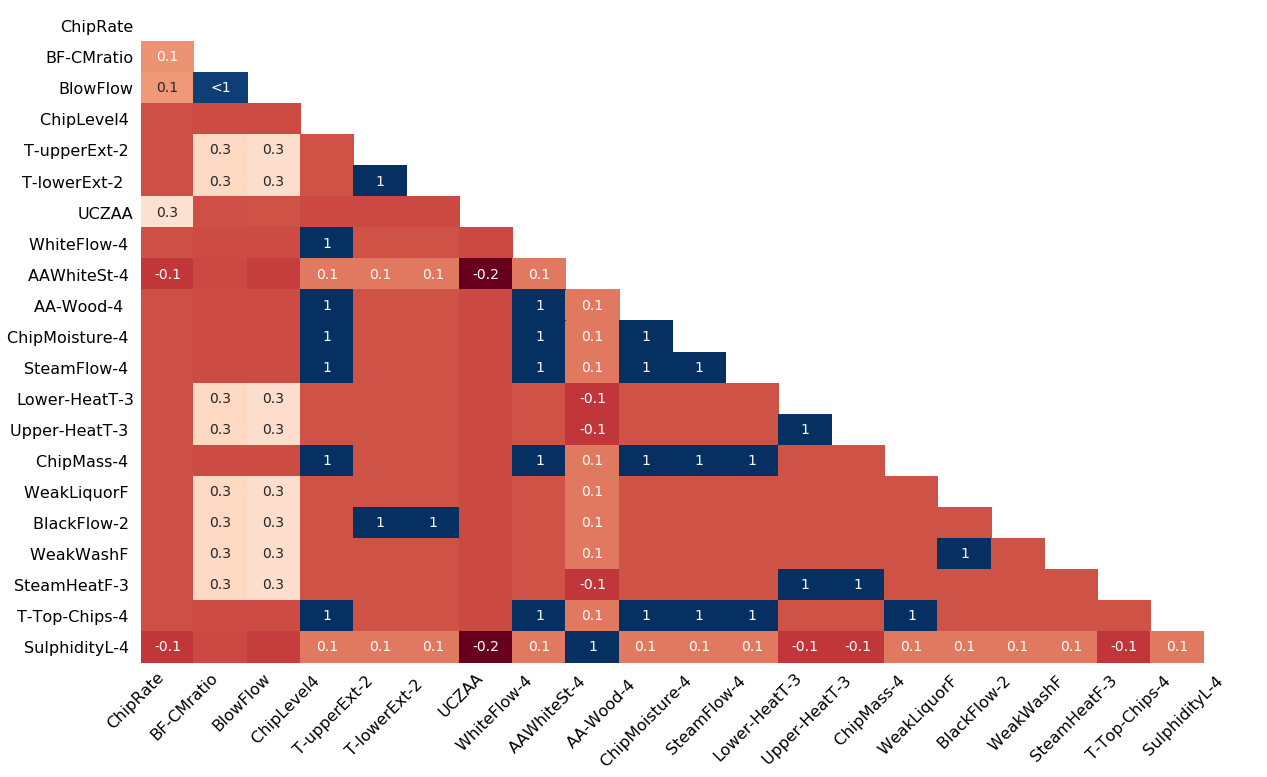
msno.bar(df)



A heat map represents the correlation between missing values in every column.

A value corresponding to -1 indicates that variable A in the data set results in missing values in its variable B.

msno.heatmap(df)



**Approaches to Handle Missing Values**

**1 Drop Columns and Rows Containing Missing Values**

Remove the columns and rows containing missing values in MCAR data. However, the problem with this approach is the loss of information.It is recommended to delete a particular column if the number of missing values in the data is more than 70–75 percent.Also, when we have large datasets , we can delete rows containing null values. Although,it is not recommended if the percentage of missing values in the data set is greater than 30 percent.

#Drop the rows with at least one element missing

df.dropna(inplace = True)

# Drop the rows with all the elements missing

df.dropna(how='all',inplace = True)

# Drop the rows with missing values greater than two

df.dropna(thresh=2, inplace = True)

# Drop the rows with at least one missing value in the columns specified in the subset function

df.dropna(subset=['age', 'fare'])

# Drop the columns with at least one missing value

df.dropna(axis= 1 , inplace = True)

# Drop the columns containing all the elements missing

df.dropna(axis= 'columns',how = 'all', inplace = True)

**2 Imputing missing values in the data with mean,median,and mode**

We can replace the missing value in the data set with mean, median or mode of that particular feature but this method can lead to underestimation of variance and can add bias in the data .This approach is ideal when data size is small as it helps prevent information loss ,but it doesn’t take into account the correlation between the variables while doing the mean or median imputation in python as it is a uni-variate approach.

For Example, in a data set consisting of columns age and fare and there are missing values in age feature. If we impute the missing values with mean age then it might result in increase in bias due to positive correlation between age and fare feature.

from sklearn.impute import SimpleImputer

mean\_imp = SimpleImputer( strategy='mean')

# For Mode replace strategy with most\_frequent

# For Median replace strategy with Medianmean\_imp.fit(train)

train\_df = mean\_imp.transform(train)

If we want the data to be first treated for missing values and then used by our model then we can use Pipeline as this prevents data leakage.

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

# define the imputer

imputer = SimpleImputer(strategy='mean')

# define the model

lda = LinearDiscriminantAnalysis()

# define the modeling pipeline

pipeline = Pipeline(steps=[('imputer', imputer),('model', lda)])

# define the cross validation procedure

kfold = KFold(n\_splits=3, shuffle=True, random\_state=1)

# evaluate the model

result = cross\_val\_score(pipeline, X, y, cv=kfold, scoring='accuracy')

**3 Imputation using k-NN**

The *k* nearest neighbors is an algorithm where new point is assigned a value based on its close resemblance to the points in the training data set. This method can be used for imputing the missing values for each feature by the non-missing values which are in the neighborhood to the observations with missing data.Depending on the data set it can give more accurate results than mode,median or mean imputation.However, this method is computationally expensive as the entire training data set is stored in the memory ,and unlike SVM it is sensitive to outliers in the data.

from sklearn.impute import KNNImputer

# define imputer

imputer = KNNImputer(n\_neighbors=5, weights='uniform', metric='nan\_euclidean')

# fit on the data set

imputer.fit(X)

# fit on the data set

imputer.fit(X)

# transform the data set

Xtrans = imputer.transform(X)

**4 Multiple Imputation by Chained Equations(MICE)**

MICE is a multiple imputation technique to replace the missing values in the data set with MAR missing mechanism.It uses the other features in the data to make the best prediction for each missing value. In this algorithm each missing value is modeled on the observed values in the data.To know more about MICE algorithm check “ [*MICE algorithm to Impute missing values in a dataset*](https://medium.com/swlh/mice-algorithm-to-impute-missing-values-in-a-dataset-c55d555b6fbe) “.This algorithm can be implemented using the Scikit-learn Iterative Imputer.

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

After this we will find how the values in the X are correlated to decide which algorithm to use to impute the null values.

X.corr()

lr = LinearRegression()

imp = IterativeImputer(estimator = lr ,verbose = 2 ,max\_iter = 30,tol = 1e-10,order = 'roman')

imp.fit(X)

imp.transform(X)

**Conclusion**

Missing data can lead to invalid results due to the absence of complete information. They are handled as training an ML model on a data set consisting of missing values can result in an error as the python library, including, Scikit learn doesn’t support them.