HANDLING MISSING VALUES

Exploring Missing Value Imputation Techniques

Prepared For

IS 3005

Prepared By

THARINDU DARSHANA
S16341

Contents

Introduction 2

1. What are the missing values? 2
2. Why is it important to address missing values in data analysis and modelling? 2
3. How missing data can affect statistical analyses and machine learning models? 2
4. Types of missing values 3
5. How missing values represented in a dataset? 3

Techniques for Handling Missing Values 4

1. Regression Imputation 4
2. Iterative Imputation 4
3. Linear Interpolation 5
4. K-Nearest Neighbours (KNN) Imputation 5

Available Packages or Tools 6

KNNImputer (in Python scikit learn library) 6

IterativeImputer (in Python scikit learn library) 6

MICE (in R) 6

Scenarios for Practical Application 8

Introduction

1. What are the missing values?

Missing values can be defined as the absence of data entries for some variable/s in a dataset. Missing values in datasets are very common when working with real world data and handling them accordingly is a vital part of the data science process to ensure the reliability and accuracy of the analysis.

2. Why is it important to address missing values in data analysis and modelling?

Addressing missing values is quite important in both data analysis and predictive model building. If we do not handle the missing values properly, it may distort the underlying distribution of variables, affecting descriptive statistics and leading to inaccurate decision-making. Missing data can introduce bias into analysis especially when missingness is not random. For example, in a survey, if females are less likely to answer the question on monthly income compared to males, there are more missing values from females compared to males and this case is not random and introduces bias to the study.

Many machine learning models often require complete datasets to train properly and give accurate predictions. Models trained with missing values may not give accurate predictions and may introduce bias.

3. How missing data can affect statistical analyses and machine learning models?

Missing values can distort the underlying distribution of the variable thus skewing the measure of central tendency and dispersion and ultimately leading to inaccurate conclusions. It also affects statistical tests like hypothesis testing as it may give invalid results if missing values alter the assumptions of those tests.

Even though some machine learning algorithms can handle missing values, many machine learning algorithms cannot process missing values, requiring handling them. Also, improper handling of missing values can result in poorly trained models, thus reducing the ability to make accurate predictions. Therefore, not just handling the missing values, but properly addressing them using suitable techniques based on the pattern of missingness, and the analysis objectives is very important to ensure accurate and meaningful results.

4. Types of missing values

Missing values in a dataset can occur due to various reasons. Based on the underlying causes, we can categorize them into three types

i. Missing Completely at Random (MCAR)

Missing values occur completely at random. There is no relationship between missing data and any other variables in the dataset.
Eg: After a service, customers would be asked to provide ratings on the service. But not all customers would do this. This occurs at completely random.

ii. Missing at Random (MAR)

The likelihood of value being missing is related to other variables in the dataset. That is missing values do not occur at random and there is a pattern of missingness.
Eg: In a survey, there is a question asking about their income. If females are less likely to answer that question than males, it is not randomly occurring missing values, it depends on the 'Gender'

iii. Missing not at Random (MNAR)

Missing values are not at random and cannot be explained by the observed data. The reasons for missingness are related to the unobserved data. Addressing this type of missing values is challenging as standard imputation techniques do not work for this. It requires advanced methods or domain knowledge.

5. How missing values represented in a dataset?

- **NaN (Not a Number)** – This is how many programming languages and statistical tools represent numerical missing values
- **NULL or None** – In databases missing values are typically recorded as NULL. Many programming languages represent categorical missing values as NULL or None.
- **Empty Strings ("")** – This is common in text-based data files where missing values are left as an empty string
- **Special Indicators** – Sometimes special indicators like -999,9999 are used to represent missing values in a dataset.
- **Blanks or Spaces** – In some text data files, missing values might represented as spaces (' ') or blanks

Techniques for Handling Missing Values

In this section, we will explore the following techniques for imputing missing values.

1. Regression Imputation
2. Iterative Imputation
3. Linear Interpolation
4. K-Nearest Neighbours Imputation (KNN Imputation)

1. Regression Imputation

This method predicts missing values using a linear regression model. It imputes one variable at a time by treating the variable with missing value as the response variable and other variables as predictor variables. This method can only handle numerical missing values. The process of regression imputation follows the following steps.

a) Identify the variable with missing values that need imputation. That is choose the response variable for the regression model
b) Divide the dataset into two parts based on missing values. That is the non-missing observations of the target variable into one set and rows with missing values of the target variable into another set.
c) Train the regression model on non-missing observations of the target variable.
d) Apply the regression model to predict the missing values based on the observed values of other variables and fill the missing values with the predicted values.

References:- https://datasciencestunt.com/regression-imputation/

However, this method highly depends on the relationship between variables and underestimates the variability of the variable

2. Iterative Imputation

This is an advanced method for handling the missing values. It uses a model based approach to iteratively estimate the missing values for each variable in the given dataset. Same as in regression imputation, it builds a model for the variable with missing values, using all other variables in the dataset as predictors. The difference between this method and regression imputation is how the models are built and applied. Regression imputation builds a single model for one variable with missing values while iterative imputation builds a separate model for each variable with missing values by iterating through all variables in the dataset. Also, we have the flexibility of selecting the model for iterative imputation. Not just a regression model, we can use other machine learning models to predict the missing values. Therefore, this method can handle both numeric and categorical missing values. The steps for implement the iterative imputation are as follows.

a) Replace missing values with simple estimates such as mean and median for numeric and mode for categorical data

b) For each variable with missing values treat the variable as target and other variables as predictors to fit a model.
c) Predict the missing values for the target variable and update the dataset.
d) Iterate through all the variables with missing values in the dataset and build a predictive model for each variable to predict missing values.
e) Repeat the whole process until the imputed values stabilize or meet a predefined convergence criterion.

References:- https://machinelearningmastery.com/iterative-imputation-for-missing-values-in-machine-learning/

3. Linear Interpolation

This method is mainly used to impute missing values in time series data. It estimates missing values by assuming a linear relationship between the known data points surrounding the missing value. The steps are as follows.

- Locate the missing value and its immediate non-missing values. That is before and after the missing values.
- For a missing value X at position i, compute the missing value from the following formula,

$$x_i = x_{prev} + \frac{X\ next - X\ prev}{i\ next - i\ prev} \times (i - i_{\ prev})$$

  where X_{next} and X_{prev} are values of the previous point and next point and i_{next} and i_{prev} are the positions of the next and previous points
- Repeat the process for each missing value

References:- https://www.kdnuggets.com/how-to-deal-with-missing-data-using-interpolation-techniques-in-pandas

4. K-Nearest Neighbours (KNN) Imputation

This method imputes the missing values based on the similarity of observations. The algorithm identifies the k (a predefined number) nearest observations for a data point with missing values based on some distance metric, then imputes the missing values using the information from these neighbours. This can handle both numerical and categorical missing values. Following are the steps of the procedure.

- For each value with missing values, calculate the distance between it and other non-missing data points in the feature space using a distance matrix and select the k closest data points.
- Missing value is imputed as mean, median (for numeric data) and mode (for categorical data) of the corresponding feature values in k nearest observations
- Repeat the process for all missing values

References:- https://www.blog.trainindata.com/knn-imputation-of-missing-values-in-machine-learning/

Available Packages or Tools

There are several packages for handling missing values in the most popular statistical tools R and Python. In this section let's focus on Python scikit learnn's **KNNImputer** and **IterativeImputer** and **MICE** in R.

KNNImputer (in Python scikit learn library)

This is the K-Nearest Neighbours imputation implementation in Python. It imputes missing values from the abovementioned steps. This method can be computationally intensive for large datasets. This can handle both numeric and categorical data. But the categories in categorical data need to be encoded before feeding them to the KNNImputer. It's python implementation is as follows

```
# Import the KNNImputer class from the sklearn.impute module
from sklearn.impute import KNNImputer

# Create a KNNImputer object and specify the number of neighbors (k)
imputer = KNNImputer(n_neighbors=5)

# Apply the KNN imputer to the dataset (X) to fill in missing values
imputed_data = imputer.fit_transform(X)
```

Here X is the dataframe with missing values.

IterativeImputer (in Python scikit learn library)

This is the module in scikit learn library for iterative imputation. It imputes missing values as mentioned in the above section. This method also can be computationally intensive for large datasets since it iteratively update values for missing data. It's python implementation as follows.

```
#Import the IterativeImputer class from the sklear.impute module
from sklearn.impute import IterativeImputer

#Create a IterativeImputer object and specify the model to predict missing values
and number of iterations
imputer = IterativeImputer(estimator=random_forest, max_iter=10)

# Apply the Iterative imputer to the dataset (X) to fill in missing values
imputed_data = imputer.fit_transform(dataset_with_missing_values)
```

MICE (in R)

MICE stands for Multivariate Imputation by Chained Equations. It generates multiple imputed datasets by modeling each feature with missing values as a function of other features. By creating several different datasets MICE account for variability of the estimates. This method can handle both numeric and categorical missing values. It's R implementation as follows.

```
#Import
library(mice)

# Perform MICE imputation
imputed_data <- mice(data, m = 5, method = 'pmm')
#here m is the number of imputed datasets to generate and method 'pmm' is
predictive mean matching method for predict numerical data

completed_data <- complete(imputed_data, action = "long", include = TRUE)

# Fit linear model and pool results
model <- with(imputed_data, lm(var1 ~ var2 + var3))
pooled_results <- pool(model)
summary(pooled_results)
```

The following table highlights the key features of these three tools.

Feature	KNNImputation	IterativeImputation	MICE
Type of imputation	k-nearest neighbours	Model based imputation	Multiple imputation
Data types it can handle	Numerical and categorical	Numerical and categorical	Numerical and categorical
Computational cost	Moderate to high for large datasets	Moderate to high for large datasets	High for large datasets
Strengths	• Simple and intuitive. • Suitable for small datasets	• Better at capture relationships between variables. • Supports various kind of models	• Accounts for uncertainty in imputation • Suitable for statistical inference

Scenarios for Practical Application

This section will explore how to implement the abovementioned methods in a real-world dataset.

For this purpose, we will use a car price dataset sourced from Kaggle that has missing values.

- [Link to the dataset]

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage(km/ltr/kg)	engine	max_power	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.40	1248.0	74	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14	1498.0	103.52	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.70	1497.0	78	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.00	1396.0	90	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.10	1298.0	88.2	5.0

Above is the sample view of the dataset. It contains factors to determine car prices.

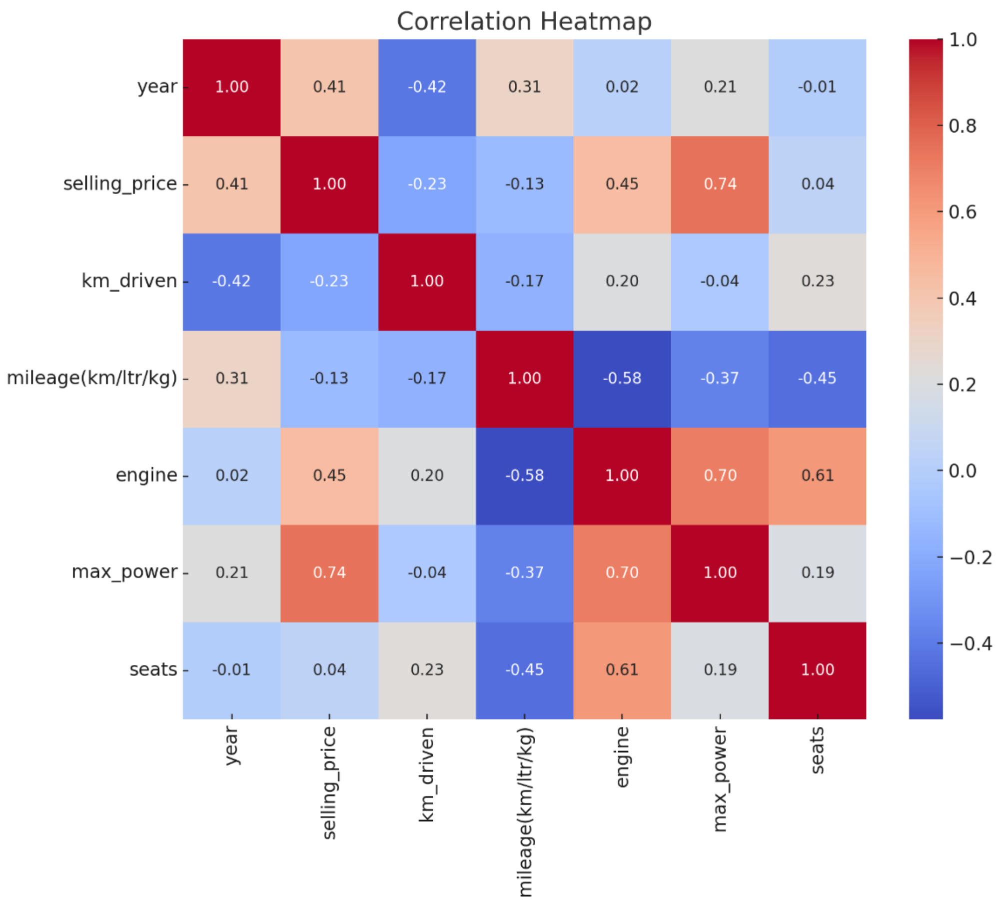
```
df.isna().sum()
```

```
name                    0
year                    0
selling_price           0
km_driven               0
fuel                    0
seller_type             0
transmission            0
owner                   0
mileage(km/ltr/kg)    221
engine                221
max_power             215
seats                 221
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   name                8128 non-null   object 
 1   year                8128 non-null   int64  
 2   selling_price       8128 non-null   int64  
 3   km_driven           8128 non-null   int64  
 4   fuel                8128 non-null   object 
 5   seller_type         8128 non-null   object 
 6   transmission        8128 non-null   object 
 7   owner               8128 non-null   object 
 8   mileage(km/ltr/kg)  7907 non-null   float64
 9   engine              7907 non-null   float64
 10  max_power           7912 non-null   float64
 11  seats               7907 non-null   float64
dtypes: float64(4), int64(3), object(5)
memory usage: 762.1+ KB
```

Here we can see the column names, it's type and the corresponding number of missing values in each column. We have missing values in columns 'mileage (km/ltr/kg)', 'engine', 'max_power' and 'seats'. All the variables with missing values are numerical. We will check if the variables with missing values have strong relationships with other variables in the dataset to choose the best method to impute the missing values.



We can see the variable ‘max_power’ shows a high correlation with ‘selling_price’ and ‘engine’. Also ‘engine’ shows a high relationship with ‘seats’. Therefore the variables with missing values have relationships with other variables in the dataset. So it is better to use model base approach to impute the missing values in the dataset. Here in this case, we will use the Iterative imputation method to impute the missing values because this method uses a model-based approach to impute the missing values and is hence good at handling relationships between variables.

```
df_i.isna().sum()
```

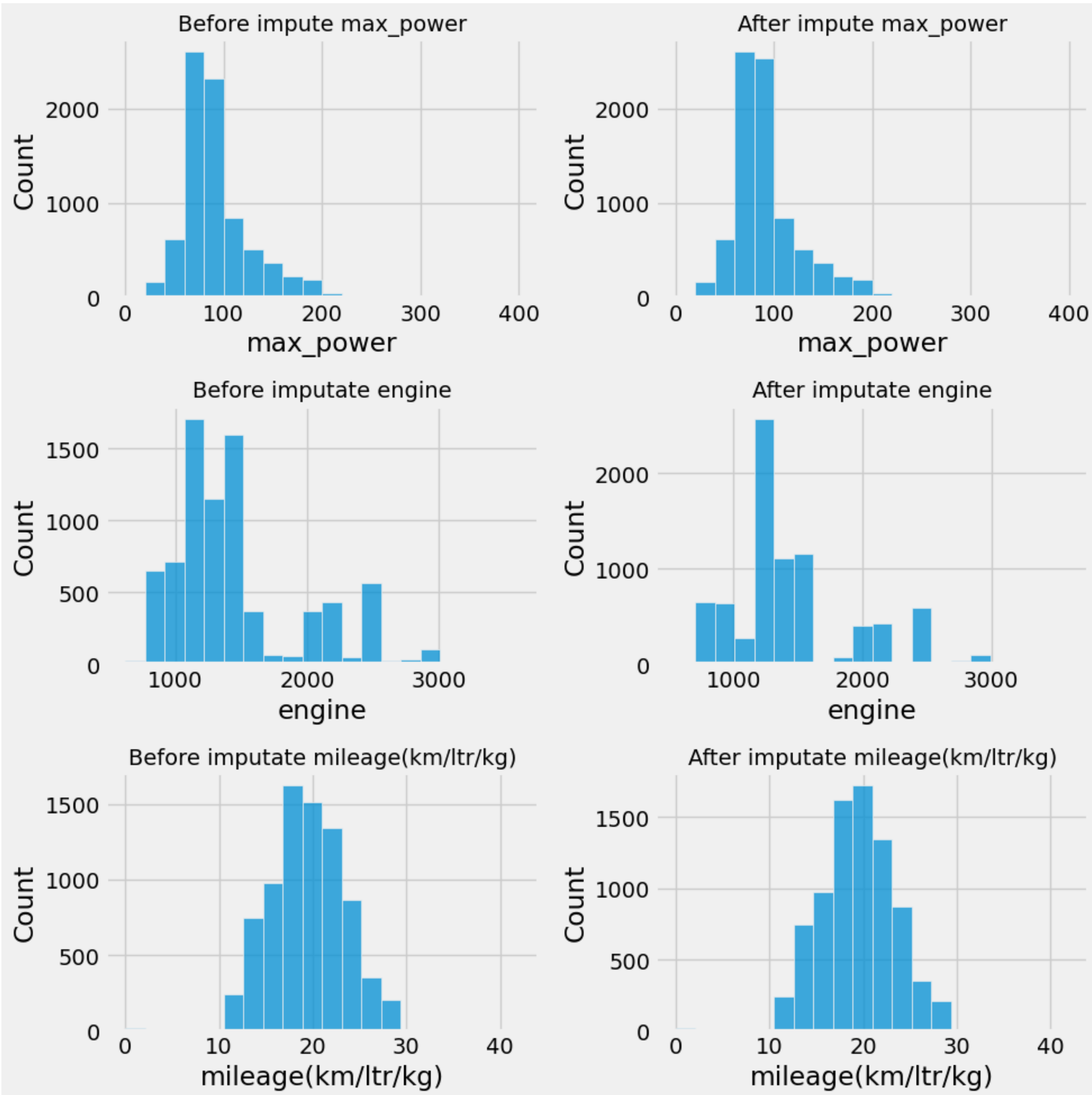
```
name                  0
year                  0
selling_price         0
km_driven             0
fuel                  0
seller_type           0
transmission          0
owner                 0
mileage(km/ltr/kg)    0
engine                0
max_power             0
seats                 0
dtype: int64
```

```
df_i.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   name                8128 non-null   object 
 1   year                8128 non-null   int64  
 2   selling_price       8128 non-null   int64  
 3   km_driven           8128 non-null   int64  
 4   fuel                8128 non-null   object 
 5   seller_type         8128 non-null   object 
 6   transmission        8128 non-null   object 
 7   owner               8128 non-null   object 
 8   mileage(km/ltr/kg)  8128 non-null   float64
 9   engine              8128 non-null   float64
 10  max_power           8128 non-null   float64
 11  seats               8128 non-null   float64
dtypes: float64(4), int64(3), object(5)
memory usage: 762.1+ KB
```

This is the result of the imputation of missing values. We can see there are no missing values left in the dataset and all the missing data has been imputed.

To assess the validity of the imputed values, we can check the distributions of the variables that contained missing values before and after the imputation process.



Here we can see there's not much difference in the original distributions (before imputing the missing values) and the distribution after imputation. This indicates that the imputation process preserved the original data structure and did not distort it. Therefore, the imputed values can be considered valid.