BHT Berliner Hochschule für Technik

Quantifying the influence of imputing missing values of Bike Share Data.

vorgelegt von

Hinakoushar Tatakoti

Matr.Nr.: 915584

dem Fachbereich VI — Informatik and Medien
der Berliner Hochschule für Technik Berlin.
vorgelegte Masterarbeit Master Thesis
zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Studiengang

Data Science (M.Sc.)

Tag der Abgabe March 31, 2023

Studiere Zukunft

Supervisors

Prof. Dr. Timothy Downie	Berliner Hochschule für Technik
Prof. Dr. Felix Biessmann	Berliner Hochschule für Technik

Declaration

Abstract

Missing data is one of the most common challenges experienced by researchers, data analysts, and data scientists. Therefore, managing missing data is an essential task before obtaining any results; otherwise, results may not be valid. In this thesis, we demonstrate various imputation methods using the MCAR missingness pattern in the bike sharing dataset. On the other hand, bike sharing systems have witnessed exceptional growth and significant attention in the transportation domain in recent years. However, it experiences a prevalent issue of missing data due to various reasons, such as the failure of data acquisition devices or the user's participation, among others. The missing data led to increased uncertainty in the modeling and planning of bike transportation systems, so obtaining complete imputing of the missing data is a crucial task. This study quantifies the influence of missing values in bike-sharing data using various standard imputation methodologies by introducing distinct missingness proportions. To experiment, I collected a real-world complete bicycling dataset from New York City (Citi Bikes), and for the simulation, I used a common algorithm to introduce missing values for univariate and multivariate imputation techniques.

Experiment conducted with heterogeneous data and realistic missingness condition MCAR,Using both univariate and multivariate imputation methods when both training and testing are affected by missing data. Each imputation strategy is evaluated regarding the imputation quality and the impact it has on a subsequent ML task. This thesis findings provide vital insights into the presence of a variety of imputation methodologies under realistic conditions

Keywords: imputation, missing data, MCAR, citi bike, univariate, multivariate

Acknowledgement

I would like to take this opportunity to thank my supervisor, Prof. Dr. Timothy Downie, for the constant encouragement, valuable advice, and patient guidance. He has provided throughout my time as his student. I have been truly fortunate to have had a supervisor who cared so much about my work, and who promptly responded to my questions and enquiries. My sincere gratitude and appreciation are never ending for all the support he has shown and given to me throughout my study journey. I would not have made it without his guidance and expertise. Finally, I would like to thank the Berliner Hochschule für Technik for giving me the opportunity to excel and making use of their valuable resources, campus experience and professional expertise.

Contents

List of Figures

List of Tables

List of Algorithms

,

Chapter 1

Introduction

"Missing data are just data with uncertainty. Treat them accordingly." - [Rubin, 1996]

The objective of imputation is not to create perfect data but plausible data, and imputing missing values is one of the finest strategies. The key to successful imputation is comprehending the nature of missingness and the appropriate statistical method to use. The visualization of the dataset helps to comprehend the structure of missing observations in dataset (see Figure A.1) and could suggest which variables could potentially be beneficial for the imputation of missing entries. There are many libraries available in the market for Python as well as in R, for instance, miceForest, naniar, ggplot2, Missingno, etc.

1.1 Background

The missing values are quite common in many real-world datasets; it's a main pain point for data analysts and data scientists as missingness can reduce the statistical power of a study and generate biased estimates, leading to invalid conclusions [Korean, 2013]. Therefore, a common question is: what are the reasons producing missing values, and how should we manage these missing data? , a further question can be posed as follows: "Can I just drop missing values, treat with standard imputation methods, or use some other strategy?".

One of the most frequent data quality problems is missing values [Kumar and Boehm, 2017]. Reasons for incomplete data are manifold in real world dataset.

1. The user does not provide information regarding certain fields or want to share personal data. For example, some may not be comfortable sharing information about their birth year, age, salary, gender, etc. during the data capture procedure; these fields are left vacant.

2. In some cases, data is gathered from various prior records and not explicitly collected. In this circumstance, corrupting data is a significant issue. Due to minimal maintenance, some portions of the data are corrupted, giving rise to missing data.

3. Inaccuracies during the data collection procedure also contribute to missing data. For example, in manual data entry, it is challenging to completely avoid human errors.

4. Equipment inconsistencies lead to incorrect measurements, which in turn cannot be used.

Throughout the past few decades, researchers from diverse communities have been contributing to an increasingly large set of methods to impute missing values.

From previous discussion, one straightforward and simple solution is removing a row or column of missing values. While this is a basic and quick method, it comes with an enormous disadvantage. As a consequence, it reduces the size of the dataset and results in the loss of some critical information required for model construction. However, the aggregate percentage of data that is missing is essential. Generally, if less than 5 percent of values are lacking, it is permissible to neglect them [William et al., 2022]. However, the total percentage of data lacking alone is not enough; we also need to pay attention to which data is missing. Often, it might have been necessary to contemplate deleting cases (participants) or individual variables that were lacking a ton of values. This phase alone may significantly enhance the integrity of the data and reduce the aggregate percentage of missing values in the dataset.

Another simple method is replacing missing values with constant mathematically valid values; while this might seem like a valid solution to ensure the better functioning of models, such approaches may often reduce the amount of available data for analysis or model building and, depending on the missingness patterns, might also bias downstream applications (regression) [Stoyanovich et al., 2020] and thus further decrease data quality [Little and Rubin, 2002]. Poor data quality can rapidly deteriorate applications or destroy the anticipated behavior of applications and cause application breakdowns, often resulting in significant economic expenditures. Moreover, poor data quality can encourage unjust automated judgments that disadvantage minorities or have other negative societal impacts [Stoyanovich et al., 2020]. For this reason, many researchers started investigating to what degree surveillance of data integrity could be computerized [Baylor et al., 2017]. While some components of such surveillance, such as the consistency of data classifications, are straightforward to automate, others, such as philosophical correctness, are still the subject of ongoing research [Biessmann et al., 2021]. However, even if automated surveillance tools, such as those recommended in the work of [Schafer and Graham, 2002], were used, a fundamental challenge remained: the open question is how can we automatically resolve the discovered data quality issues?.

Statisticians established the theoretical foundations for missing value imputation [Rubi, 004b] by distinguishing different patterns of missingness (more specifics are provided in Section (3.3). Statistical techniques have been suggested to handle missing values [Schafer and Graham, 2002]. Allan and Wishart [Allan and Wishart., 1994] were the first to develop a statistical technique to substitute a missing value. They supplied two algorithms for approximating the worth of a single lacking measurement and recommended putting in the estimate in the data. The beginning of multiple interpolation has been chronicled by Fritz Scheuren [Fritz, 2005]. Multiple Imputation was developed as a response to a practical problem with the absence of income statistics in the March Income Supplement to the Current Population Survey (CPS). Also, Rubin came up with the concept of using numerous versions of the comprehensive dataset, something he had already investigated in the early 1970s. The original 1977 article establishing the concept was published in 2004 in the History Corner of the American Statistician by [Rubi, 004b].

Some of statistics community focuses on multiple imputation (MI) [Rubin, 1987]. In MI, one substitutes missing values with numerous projections from an imputation algorithm. Those more than 1 imputed comprehensive databases can be used to evaluate the variability of estimated values. The most prominent and extensively used MI methodology is multiple imputation by chained equations (MICE) [Rubi, 004b], which is very adaptable and can be implemented with different models. While some applications can benefit from this uncertainty information, incorporating this uncertainty precision into data processes can be challenging. From a practitioner's point of view, point estimates are much simpler to integrate into conventional data processes. This is why we constrain our research to point estimate imputations. Note, however, that all the experiments performed in this study could, in principle, also be evaluated with respect to their uncertainty estimates in a MICE context, using the investigated interpolation techniques as the model underpinning the MICE estimator.

Most imputation of missing values studies provide considerable experimental evidence that the respective suggested method in the application settings investigated outperforms other competitors' baselines. Yet, it remains challenging to evaluate which imputation technique continuously performs best in a broad spectrum of application domain and datasets under practical missingness circumstances. In particular, most benchmarks do not systematically record and evaluate both imputation quality and the effect of the imputation on downstream applications with baselines in a broad variety of circumstances [Jäger et al., 2021].

In this thesis, simulation is conducted on simple, straightforward, and complex multiple imputations, applied to bike share data under plausible missingness circumstances with respect to imputation quality and the effect on the forecasting performance of downstream ML regression models. For the experiment, used a publicly available bike share dataset which contains heterogeneous data (numerical and categorical). Simulation is conducted by artificially introducing missing values in terms of various fractions using one of the missingness pattern (MCAR see Section 3.3). Measured both the imputation effectiveness in 3 scenarios: 1) Univariate missing value imputation performance, 2) Multivariate missing value imputation on combinations of columns(Multiple Regression and Mice Strategy), 3) train and test on complete dataset.

1.2 Problem Statement

Mobile application-based transportation services are revolutionizing the developed transportation businesses of both the established and developing countries. They generate vast quantities of data, which have the potential to provide deeper insights into urban travel activity than ever before. The bike-sharing service (BSS) industry is developing at an overwhelming rate, with new service providers entering in the same domain. However, several BSS start-ups have collapsed in many countries in recent years. All these instances have one element in common: user dissatisfaction because of insufficient or ineffective re-balancing approaches. The BSS administrators depend on data insights to drive their policies, strategies, and planning. Therefore, the user's journey statistics as well as exterior variables such as temperature, optimal hours, weekdays versus weekends, working versus not working, etc. play important parts in the bike sharing sphere. However, the data generated by these services is discovered to have several insufficient entries; the explanation might be one of these (see background 1.1). As most BSS modelling concentrates on trip beginning and destination, completely disregarding journeys with incomplete information results in the loss of important data that is still present in other recorded variables, which include trip duration, date and time of the trip, and so on.

It is very important to handle these missing data elements to acquire the complete data without jeopardizing on valuable information, which is necessary for data analysis to make business decisions. Therefore, this thesis quantifies missing values in bike share data using simulations of univariate and multivariate interpolation techniques under plausible missingness patterns.

1.3 Aim of the work

The goal of this thesis is to quantify the impact of missing value imputation on bike-sharing services (BSS) by applying simple univariate and complex multivariate imputation techniques under realistic conditions such as MCAR missingness pattern, with different fractions of missing values artificially inserted into complete dataset. In the further chapters, methodology (see section 3), implementation, and exploration (see section 4) provide comprehensive descriptions of methodologies used for simulation and how experimental setup is done and implemented. Data from a BSS that operated in New York City, USA, known as "City Bike", is used to illustrate the proposed approaches. This is followed by a brief summary of the findings and a comparison of the experiments.

1.4 Content

This thesis's structure seeks to be as straightforward and explicit as possible while still considering all scientific standards. Chapter 2 includes all the essential fundamentals to understand the current work. The fundamental concepts are summarized, and references to related literature are given. The primary component to conduct the simulation is described in the chapter 3, which will encompass all the essential phases from data to the final description, and Chapter 4 highlights some noteworthy implementation and explorations aspects. Afterwards, the assessment technique and findings are described in chapter 5, and some common discussion of the results are followed by the last chapter 7 containing the Conclusion and outlook for this undertaking. The pertinent literature and the appendix can be located at the end of this document.

Chapter 2

Related work

The following section is dedicated to providing an overview of all related work but is mainly focused on the understanding of imputation in the bike share dataset (time series domain) and research papers, some of which focus on presenting new or improved imputation methods and comparing them with existing and baseline approaches in broader settings.

In bike sharing domain, there are different types of imputation methods being used. The task [Milan et al., 2022] worked on two techniques for imputing missing data: (i) a probabilistic strategy based on Bayes' theorem and (ii) a machine learning approach based on the k-nearest neighbor algorithm to compute journey duration, date, and time of the trip. Also, the study determined that when the Bayesian method was contrasted with the k-NN method for imputation, the k-NN method outperformed the Bayesian method with a better degree of imputation precision.

Similarly, [Xiao et al., 2022] concentrated on using the capabilities of a generative adversarial network (GAN), which learns the distribution of missingness and generates data close to ground-truth values, to substitute the missing numbers from a bike-sharing system. The suggested techniques assume missing traffic counts when considering two scenarios: the "not missing at random (NMAR) problem and the "missing absolutely at random" (MCAR) problem. They determined that the suggested techniques demonstrate resilience with increasing missingness ratios in the dataset. They also used the RMSE values to evaluate the missing data imputation accuracy, which is smaller than 0.15, while the missingness ratio increases from 20 to 80 percent. The suggested methods are stable and effective for the missing data imputation challenge for a bike-sharing system compared to other baseline imputation methods.

This thesis, which performs an experiment on time series data for the bike sharing system, makes it essential to discuss a literature review related to imputation approaches used for time series data. The study [Hyun et al., 2022] concentrated on experimenting with time series data using different imputation techniques. Therefore, several time series forecasting models are learned using separate training datasets produced using the MCAR, MAR, and MNAR interpolation techniques. However, they proved that the k-nearest neighbor technique is the most effective in reconstructing missing data and contributes positively to time series forecasting compared with other imputation methods.

The researchers [Anil et al., 2019] examines seven differnt imputation techniques (random, median, k-NN, predictive mean matching, Bayesian linear regression, linear regression, and non-Bayesian) without optimizing their hyperparameters based on five small and quantitative datasets (max. 1,030 observations). The researchers describe different missingness patterns but do not specify which one they used in their experiments. However, they evaluated the methods' imputation effectiveness for 10% to 50% unknown values. Again, the authors demonstrate that k-NN interpolation is optimum when it is independent of the dataset and missingness percentage. Even the researchers [Woźnica and Biecek, 2020] evaluates and contrasts seven differnt imputation techniques (random, mean, softImpute, miss-Forest, VIM kknn, VIM hotdeck, and MICE) combined with five categorization models regarding their prognostic performance. Therefore, they use 13 binary classification datasets with missing values in at least one column, which is why they do not know the data's missingness structure. The proportion of unknown numbers varies between 1% and about 33%. In contrast to the work of [Jason and Rafael, 2018] and [Anil et al., 2019], the authors could cope with the circumstance where only fragmentary data were accessible for training. In their situation, they could not determine a singular optimum replacement technique. However, they demonstrate that the combination of the imputation technique, downstream model(regression or classification), and measure the estimates (F1 or AUC) influences the findings.

The paper [Zhang et al., 2018] discuss about the EM (expectation-maximization) algorithm, they implemented incremental EM technique that learns and optimizes a hidden representation of the data distribution, and parameterized by a deep neural network, to conduct imputation. They use ten classifications and three regressions task datasets and 11 imputation baselines (zero, mean,median, MICE, miss-Forest, softImpute, k-NN, PCA,autoencoder, denoising autoencoder, and residual autoencoder) for comparison. The authors completed both assessments, imputation, and downstream tasks performance, with 25%, 50%, and 75% of MNAR missing values and demonstrated that their approach outperformed the baselines.

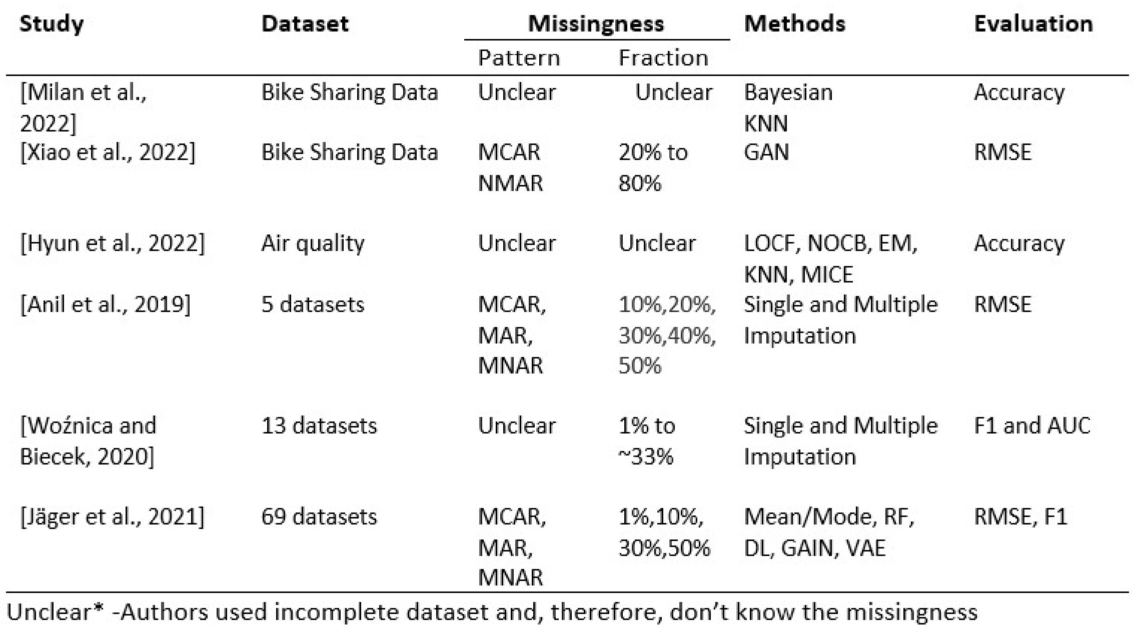
To the best of my understanding, [Biessmann et al., 2019] provided the biggest and most comprehensive comparison, although they concentrated on introducing an interpolation algorithm and exhibiting its improvements. The suggested algorithm cross-validates the choosing of the optimum imputation method out of k-NN, SVM, or tree-based imputation methods, where the hyperparameters are also cross-validated. The authors then benchmarked their strategy on 84 classification and regression tasks against five imputation methods: mean, pmm (predictive mean matching) , Bayesian PCA, k-NN, and iterative k-NN. They measured the imputation and downstream task performance on 10% to 50% MCAR and MNAR missing values. The authors demonstrate that the suggested approach outperforms the baselines, closely followed by k-NN and incremental k-NN.

[Jäger et al., 2021] conducted a comprehensive suite of experiments on 69 datasets with heterogeneous data and realistic missingness conditions such as MCAR, MAR, and MNAR with different missingness fractions ranging between 1% and 50%, comparing both novel deep learning approaches and classical ML imputation methods when either only test data or both train and test data are affected by missing data. Each estimation technique is evaluated in terms of its quality and the effect it has on a succeeding ML assignment. Their findings provide crucial insights into the performance of a variety of estimation techniques under practical circumstances.

To recapitulate the mentioned publications and associated benchmarks (see Table 2.1). Most benchmarks use broad missingness percentages but lack practical missingness conditions or a large number of heterogeneous datasets. Furthermore, one article discussed imputation methods trained on complete and fragmentary data, and the rest did not. To summarize the contributions of this work, complement the existing research by providing a broad and exhaustive simulation of simple and multivariate imputation methods with regard to the following dimensions:

1. Large Time series dataset : Bike share dataset around 0.3M rows with numeric and categorical columns.

2. Downstream tasks: Regression
3. Realistic missingness patterns and the amount of missing values: focused MCAR missingness patterns and 1%, 5%, 10%, 20%, 30%, 40% missing values.
4. Imputation methods: Used 5 imputation methods for Univariate and 2 for Multivariate methods.
5. Evaluation on imputation performance and impact on downstream task performance: Systematically compared the imputation methods based on their imputation performance and how they impact the performance of a downstream model (regression).
6. Training on complete and imputed data: Simulated and compared the performance of imputation models with ground truth estimations.



Study	Dataset	Missingness		Methods	Evaluation
		Pattern	Fraction		
[Milan et al., 2022]	Bike Sharing Data	Unclear	Unclear	Bayesian KNN	Accuracy
[Xiao et al., 2022]	Bike Sharing Data	MCAR NMAR	20% to 80%	GAN	RMSE
[Hyun et al., 2022]	Air quality	Unclear	Unclear	LOCF, NOCB, EM, KNN, MICE	Accuracy
[Anil et al., 2019]	5 datasets	MCAR, MAR, MNAR	10%,20%, 30%,40%, 50%	Single and Multiple Imputation	RMSE
[Woźnica and Biecek, 2020]	13 datasets	Unclear	1% to ~33%	Single and Multiple Imputation	F1 and AUC
[Jäger et al., 2021]	69 datasets	MCAR, MAR, MNAR	1%,10%, 30%,50%	Mean/Mode, RF, DL, GAIN, VAE	RMSE, F1

Unclear* -Authors used incomplete dataset and, therefore, don't know the missingness

Figure 2.1: An overview of related benchmarks

Chapter 3

Methodology

The main purpose of thesis is to evaluate the missing value imputation methods for real-world data from the bike sharing system. The thesis mainly targeted the New York City bike public transportation system, such as Citi Bikes. The following sub-sections provide comprehensive details about the dataset requirements, the various data preprocessing techniques involved, the all the standard missing patterns available, and which one is considered for the experiment. Then follows a detailed description of the compared imputation methods and the evaluation metrics used for comparison.

3.1 Datasets

The study centers on contemplating a quite large dataset with characteristics including continuous, discrete, qualitative, and at least one numerical variable that can be handled for the outcome. The fundamental prerequisite of the experiment is a complete dataset with no missing values, where missing data have been introduced in various percents via the MCAR missingness pattern on different imputation techniques.

The historical data is collected from the official website of Citibike [1] for the year 2018, which is public transportation data. The rationale behind selecting particularly 2018 is that Citibike supplied intriguing information, such as gender and year of birth of the cyclist, up until a later date in 2018 that were not included. The datset contains 15 features with rows 353,892, which matches all of previously defined requirements. In theory, bike usage is highly affected by weather changes, which could affect the preferred mode of transportation [Salma, 2020]; also, working or non-working days influence the trip duration. As many researchers have experimented with the prediction of trip duration with the help of weather, hence for the experiment i have also included some external factors such as weather and holidays data; therefore, obtained an hourly rate historical weather information from an external website (visual website)[2] and holidays from the python PyPI library, then joined it with bike share data based on time dimension. The detailed information about the individual dataset and it's attributes names, types and descriptions are mentioned in the table format as below 1) bike share tripdata (see table 3.3), 2) weather data (see table 3.3) and 3) US holiday data (see table 3.3).

[1] `https://ride.citibikenyc.com/system-data`

[2] `https://www.visualcrossing.com/weather/weather-data-services`

Variable	Variable Description	Type
tripduration	total ride time in seconds	int
starttime	time and date ride started	object
stoptime	time and date ride stopped	object
start station id	start ride station id	int
end station id	end ride station id	int
start station name	start ride station name	string
end station name	end ride station name	string
start station latitude	ride start geo location latitude	float
start station longitude	ride start geo location longitude	float
end station latitude	ride end geo location latitude	float
end station longitude	ride end geo location longitude	float
bikeid	rented bike id	int
usertype	Customer and Subscriber	object
birth year	year of birth	int
gender	(Zero=unknown; 1=male; 2=female)	int

Table 3.1: Trip Data

Variable	Variable Description	Type
date	the day of the 365 days	object
temp	°c temperature of the day	float
feelslike	°c temperature feel of the day	float
dew	dew °c temperature	float
humidity	humidity in the air	float
precip	total rainfall depth in mm	float
precipprob	probability of rainfall depth in degrees	int
snow	amount of snowfall in cm	float
snowdepth	depth of the snowfall in cm	float
windspeed	speed of wind in Kph	float
winddir	direction of wind in degrees	float
sealevelpressure	pressure of sea level in mbar	float
cloudcover	power density of sunlight in W/m2	float
visibility	object or light can be clearly discerned in km	float
solarradiation	solar radiation in W/m2	float
uvindex	solar UV radiation intensity in W/m2	int
conditions	weather condition clear, snowy, raining	object

Table 3.2: Weather Data

date	holiday
2018-01-01	New Year's Day
2018-01-15	Martin Luther King Jr. Day
2018-02-19	Washington's Birthday
2018-03-30	Good Friday
2018-05-28	Memorial Day
2018-07-04	Independence Day
2018-09-03	Labor Day
2018-11-22	Thanksgiving Day
2018-12-25	Christmas Day

Table 3.3: Holiday Data

3.2 Data Preprocessing And Exploration

Data preprocessing and investigation are two of the important phases in the machine learning life cycle or data mining process to accomplish the intended outcomes of ML algorithms. As a precondition for the bike dataset, different preprocessing and exploration steps are applied before obtaining completely structured data; therefore, on the bike share data below, preprocessing and exploration steps are applied:

1. Data integration: Different sources of data are combined to produce a single, consistent, comprehensive data set for the experiment, where bike ride data is united with weather information on the month, day, and hour dimensions. Also, holidays in NYC are combined based on the date dimension. When combined data is used for the purpose of research, the whole technique is called data integration.

2. Data transformation: Raw data is transformed into a meaningful form such that new attributes are created using existing attributes; for example, if the start time is in the complex format (2018-01-01 02:06:17.541), which includes compressed data, this time can be separated into date, month, day, hour, and minute. Also, using the start and end [latitude and longitude] coordinates, Manhattan distance can be calculated using the Haversian library; age is calculated with birthyear; seasonality [spring, summer, autumn, winter] is extracted based on the month dimension; the gender variable is converted into a categorical variable; the trip duration is converted from seconds to minutes; and measured units are converted into the appropriate form.

3. Data cleaning: The data has many outliers when investigated through box plot representation, as shown in figures 3.1 and 3.2. Therefore, considering the expenses of the cycle rentals (citi bike rental information)[3], it's very impractical to state that the ride duration is more than a day or half a day (more than 5000-35000 minutes see Figure 3.1) tripduration hence, those rides are chopped off from the dataset. Similarly, for the age variable, some records are recorded as older than 100, so when the ride duration is more than 180 minutes and the rider age is more than 65, assumed these are outliers and also, only the 90^{th} percentile data is used for the further experiment to make it realistic, therefore, all the mentioned outliers are omitted from the data set.

[3]https://citibikenyc.com/how-it-works/bike-rental-nyc

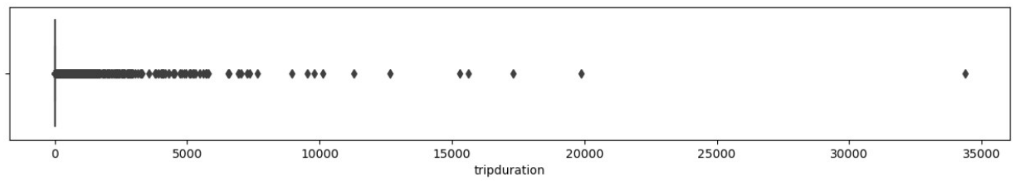


Figure 3.1: Tripduration boxplot

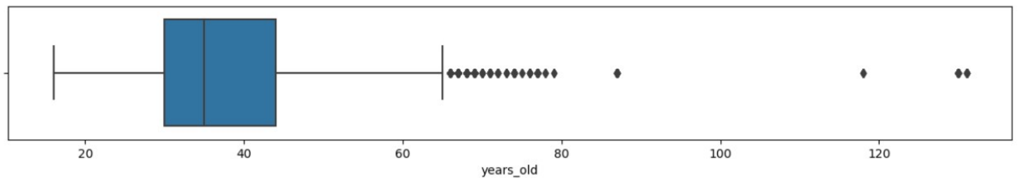


Figure 3.2: Years old boxplot

The attribute values day have names for seven days of the week, but people are more interested in seeing weekend and weekday information, so it's reduced to weekday and weekend. Similarly, holidays are reduced to two categorical levels: holidays and working days. Also, weather condition dimension categories are reduced from 10 to 4 by combining comparable meaning values. Also considered is the 90^{th} percentile of data, which incorporates tripduration less than or equal to 16.53, the following figure (3.3) shows having 90^{th} percentile tripdurations.

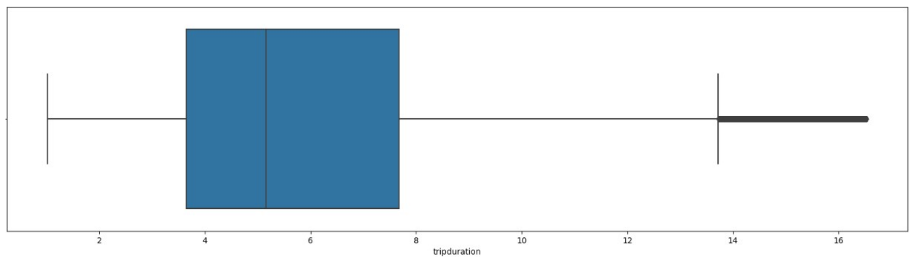
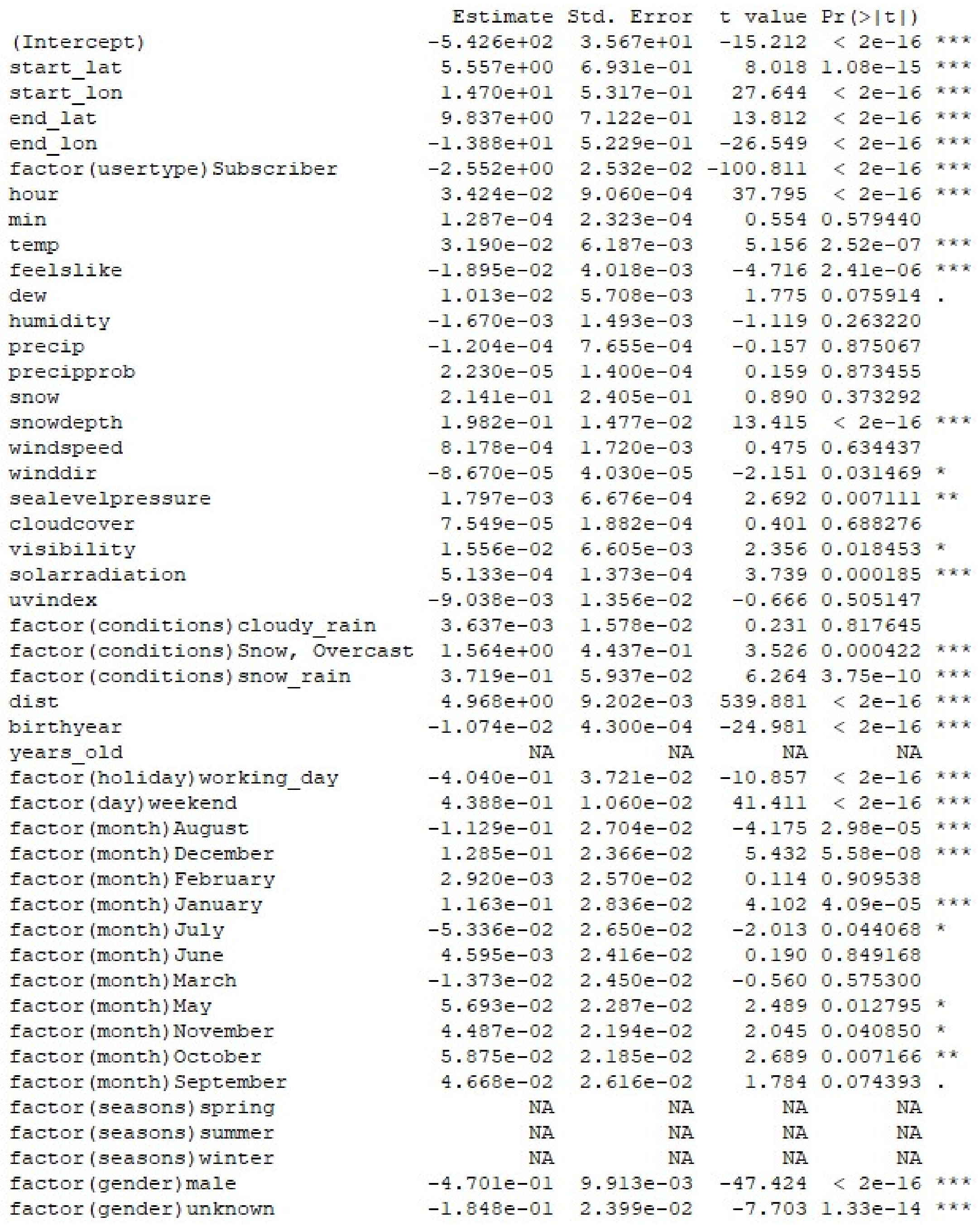


Figure 3.3: 90^{th} Percentile Tripduration

4. Data reduction: The dimensionality reduction also plays an important role in selecting only significant attributes for the final model. Our experiment performs a linear regression model on trip duration vs. the independent variable to identify the p-value, with p-values less than or equal to 0.05 considered significant features for model construction; otherwise, the variables are disqualified and hence not included in the experiment. The p-value for each variable (see Figure 3.4).

Also, it's important to comprehend what specifically a p-value is and why we need to use it before implementing any ML model on all the variables. P values (probability values) are most often used by researchers to say whether a certain pattern they have measured is statistically significant or not, statistical significance under the assumption that the null hypothesis is correct, or how likely it is that the data could have occurred under the null hypothesis. It does this by determining the probability of the test statistic, which is the value determined by a statistical test using the data. It's another way of stating that the p value of a statistical test is small enough to refute the null hypothesis of the test. The most frequent criterion is $p < 0.05$, that is, when it is anticipated to find a test statistic as extreme as the one determined by the test only 5% of the time [P, 2008].

The p-value for each variable can be seen in the last column of Figure 3.4 . The special symbols on the last column (***, **, *,.) indicates significance of the column. It shows that whose p-value is < 0.05 are significant; otherwise not, to fit a model, these significant columns are included. Examples of such columns excluded are min, $humidity$, $precip$, $precipprob$, $snow$, $windspeed$, $cloudcover$, $uvindex$, $years\_old$, and $seasons$ as there is no special symbol appears for them. When it comes to categorical columns, if one categorical level is significant then whole column is significant i.e. month columns, some categorical values(months) are significant and some are not but as at least one is significant so included in the model building.



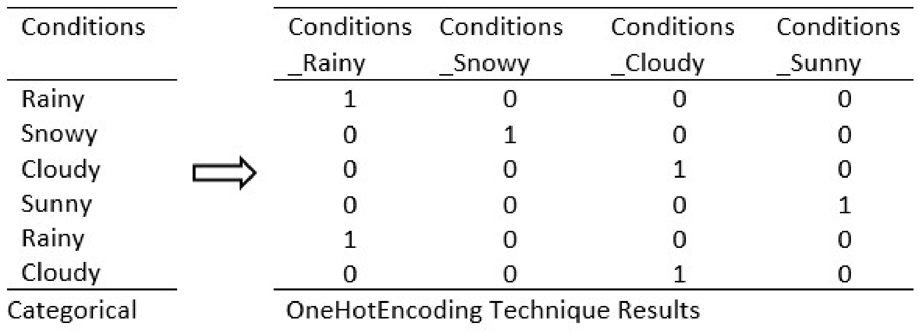
```
                                  Estimate Std. Error  t value Pr(>|t|)
(Intercept)                     -5.426e+02  3.567e+01  -15.212  < 2e-16 ***
start_lat                        5.557e+00  6.931e-01    8.018 1.08e-15 ***
start_lon                        1.470e+01  5.317e-01   27.644  < 2e-16 ***
end_lat                          9.837e+00  7.122e-01   13.812  < 2e-16 ***
end_lon                         -1.388e+01  5.229e-01  -26.549  < 2e-16 ***
factor(usertype)Subscriber      -2.552e+00  2.532e-02 -100.811  < 2e-16 ***
hour                             3.424e-02  9.060e-04   37.795  < 2e-16 ***
min                              1.287e-04  2.323e-04    0.554 0.579440
temp                             3.190e-02  6.187e-03    5.156 2.52e-07 ***
feelslike                       -1.895e-02  4.018e-03   -4.716 2.41e-06 ***
dew                              1.013e-02  5.708e-03    1.775 0.075914 .
humidity                        -1.670e-03  1.493e-03   -1.119 0.263220
precip                          -1.204e-04  7.655e-04   -0.157 0.875067
precipprob                       2.230e-05  1.400e-04    0.159 0.873455
snow                             2.141e-01  2.405e-01    0.890 0.373292
snowdepth                        1.982e-01  1.477e-02   13.415  < 2e-16 ***
windspeed                        8.178e-04  1.720e-03    0.475 0.634437
winddir                         -8.670e-05  4.030e-05   -2.151 0.031469 *
sealevelpressure                 1.797e-03  6.676e-04    2.692 0.007111 **
cloudcover                       7.549e-05  1.882e-04    0.401 0.688276
visibility                       1.556e-02  6.605e-03    2.356 0.018453 *
solarradiation                   5.133e-04  1.373e-04    3.739 0.000185 ***
uvindex                         -9.038e-03  1.356e-02   -0.666 0.505147
factor(conditions)cloudy_rain    3.637e-03  1.578e-02    0.231 0.817645
factor(conditions)Snow, Overcast 1.564e+00  4.437e-01    3.526 0.000422 ***
factor(conditions)snow_rain      3.719e-01  5.937e-02    6.264 3.75e-10 ***
dist                             4.968e+00  9.202e-03  539.881  < 2e-16 ***
birthyear                       -1.074e-02  4.300e-04  -24.981  < 2e-16 ***
years_old                               NA         NA       NA       NA
factor(holiday)working_day      -4.040e-01  3.721e-02  -10.857  < 2e-16 ***
factor(day)weekend               4.388e-01  1.060e-02   41.411  < 2e-16 ***
factor(month)August             -1.129e-01  2.704e-02   -4.175 2.98e-05 ***
factor(month)December            1.285e-01  2.366e-02    5.432 5.58e-08 ***
factor(month)February            2.920e-03  2.570e-02    0.114 0.909538
factor(month)January             1.163e-01  2.836e-02    4.102 4.09e-05 ***
factor(month)July               -5.336e-02  2.650e-02   -2.013 0.044068 *
factor(month)June                4.595e-03  2.416e-02    0.190 0.849168
factor(month)March              -1.373e-02  2.450e-02   -0.560 0.575300
factor(month)May                 5.693e-02  2.287e-02    2.489 0.012795 *
factor(month)November            4.487e-02  2.194e-02    2.045 0.040850 *
factor(month)October             5.875e-02  2.185e-02    2.689 0.007166 **
factor(month)September           4.668e-02  2.616e-02    1.784 0.074393 .
factor(seasons)spring                   NA         NA       NA       NA
factor(seasons)summer                   NA         NA       NA       NA
factor(seasons)winter                   NA         NA       NA       NA
factor(gender)male              -4.701e-01  9.913e-03  -47.424  < 2e-16 ***
factor(gender)unknown           -1.848e-01  2.399e-02   -7.703 1.33e-14 ***
```

Figure 3.4: P-Value chart

3.2.1 OneHotEncoding

The dataset contains categorical columns, which need to be addressed before inputting data into machine learning models; therefore, it's necessary to convert all the categories into numerical values before model fit. The experiment adapted the OneHotEncoding algorithm of the scikit-learn library for the conversion technique. There are other techniques to convert categorical data into numerical data, such as Dummy Variable Encoding, Label Encoding, OneHotEncoding, etc. But one-hot encoding is a simple, non-parametric method that can be used for any kind of categorical variable without any assumptions about its values. For example If the categorical feature weather condition has 4 distinct values (rainy, snowy, sunlight, and overcast), these features will divide into 4 numerical features, each of which corresponds to a distinct value. For these four novel features, only one of them has a value of 1, while the others are all 0 (see figure 3.5).

However, there is a drawback to this approach: it yields many predictors, the Figure (3.5) depicts one categorical column converted into four new columns, which it generates based on the number of unique categories present in the variable. If the column has a N number of unique categorical levels, then it generates N columns; after the transformation, the dataset has N numerical (having 0 and 1) columns instead of one categorical column, so it's necessary to identify such categorical columns to reduce the number of unique levels whenever possible. In our case, we adapted the data exploration technique and learned that some categories occurred for the least amount of time or yielded the same meaning, so those levels were merged with existing levels. For example, variable days of the week can be converted into weekday and weekend instead of retaining all the week names; likewise, variable holidays can be converted into working and non-working instead of all the holiday names. In this manner, categories can have meaningful names and a lower number of unique levels to avoid having too many predictors for model construction.



Conditions		Conditions _Rainy	Conditions _Snowy	Conditions _Cloudy	Conditions _Sunny
Rainy		1	0	0	0
Snowy		0	1	0	0
Cloudy	⇨	0	0	1	0
Sunny		0	0	0	1
Rainy		1	0	0	0
Cloudy		0	0	1	0
Categorical		OneHotEncoding Technique Results			

Figure 3.5: OneHotEncoding on Categorical Columns.

Furthermore, as it generates N columns, this might be a problem when importing the resulting data into an unregularized linear regression model or generalized linear models estimated by maximum likelihood (or least squares). If used as is without special treatment, it may cause problems such as multi-co-linearity, singularities, or perfectly co-linearity. To prevent this issue, OneHotEncoding provides a parameter termed "drop". However, removing one category violates the symmetry of the original representation and can therefore induce a bias in downstream models, for instance in penalized linear classification or regression models.

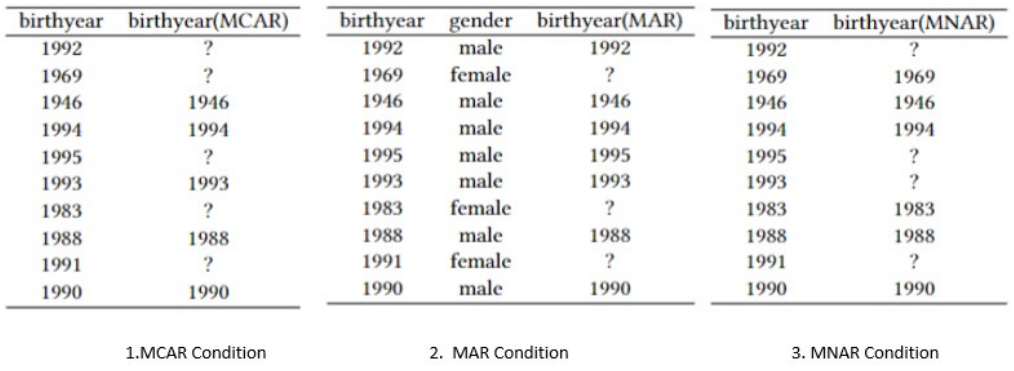
3.3 Missingness Patterns

In general, most research papers concentrated on three distinct categories of missing patterns to impute the missing values. In these instances, the missing data reduce the analyzable population of the study and consequently the statistical power but do not introduce bias.

1. **MCAR:** In the provided condition, the fact that the missing data is independent of the observed and unobserved data In other words, no systematic distinctions exist between participants with incomplete data and those with complete data. For example, in Figure 3.6, the MCAR condition table explains the column birth year missing values as five out of ten, independent of the birth year values. When data are MCAR, the data that remain can be deemed an uncomplicated random sample of the comprehensive data set of interest. MCAR is generally regarded as a strong and often impractical presupposition.

2. **MAR:** When data are MAR, the fact that the data are missing is systematically related to the observed but not the unobserved data. For example, the Figure 3.6 in the MAR condition table explains that birth year values are discarded contingent on values in another column, gender. All discarded birth year values correspond to entries in which the gender was female. That is, if the probability of the rider's trip duration is related to their birth year (which is thoroughly observed) but not their gender, then the data may be regarded as MAR.

3. **MNAR:** When the missingness mechanism is neither MCAR nor MAR, it is MNAR, in which case associations with the observed data cannot explain all systematic differences between the observed and missing data [Hughes et al., 2019]. For example, in the given Figure 3.6, one missingness is classified as MNAR, and the birth years of those born after 1990 are missing, therefore it could produce much more missing information as time progresses.



birthyear	birthyear(MCAR)
1992	?
1969	?
1946	1946
1994	1994
1995	?
1993	1993
1983	?
1988	1988
1991	?
1990	1990

1.MCAR Condition

birthyear	gender	birthyear(MAR)
1992	male	1992
1969	female	?
1946	male	1946
1994	male	1994
1995	male	1995
1993	male	1993
1983	female	?
1988	male	1988
1991	female	?
1990	male	1990

2. MAR Condition

birthyear	birthyear(MNAR)
1992	?
1969	1969
1946	1946
1994	1994
1995	?
1993	?
1983	1983
1988	1988
1991	?
1990	1990

3. MNAR Condition

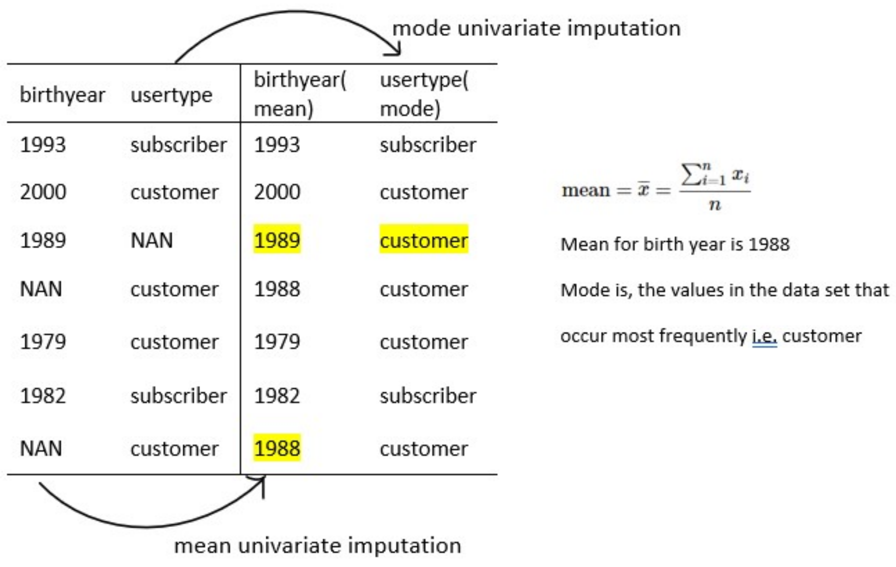
Figure 3.6: Missing Patterns Types

3.4 Imputation Methods

This section discusses the various imputation methods used for the experiment. The overall objective of an imputation method is to train a model on 315,433 observations and 22 features. To isolate crucial processes such as preprocessing the data (see 3.2) and validating the imputation method's confidence interval using the bootstrap, the thesis defined a common algorithm for univariate and multivariate that is employed by all of the respective imputation approaches.

3.4.1 Univariate Imputation

Single-variable imputation at a time is known as univariate imputation, where in a column it selects non-missing values to complete the missing values in the same column. For example, Figure 3.7 depicts that missing values are imputed using various imputation methods to acquire complete data. The birth year can be provided using the available value, i.e., 1988. The categorical columns are managed using mode imputation, where the most frequently occurring categories are replaced with the missing values. This section elaborates on the various types of univariate imputation techniques being used in the experiment. To simulate univariate imputation, missing values were introduced to one column at a time, and each of the below imputation types was applied to each of the variables with varying proportions (1%, 5%, 10%, 20%, 30%, 40%) on both the training and test sets independently.



birthyear	usertype	birthyear(mean)	usertype(mode)
1993	subscriber	1993	subscriber
2000	customer	2000	customer
1989	NAN	1989	customer
NAN	customer	1988	customer
1979	customer	1979	customer
1982	subscriber	1982	subscriber
NAN	customer	1988	customer

$$\text{mean} = \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Mean for birth year is 1988

Mode is, the values in the data set that occur most frequently i.e. customer

Figure 3.7: Univariate Illustration

1. **Drop Imputation:** When missing values are observed, the straightforward technique used is to drop all the missing data using the standard Pandas function dropna() to get complete data for the experiment.

2. **Mode Imputation:** In which the missing values are replaced with the mode value or most frequent value of the entire feature column. When the data is distorted, it is wise to consider using mode values to replace the missing values. For mode imputation, the Python function mode() is used to acquire the variable mode. Especially categorical features make use of this procedure.

3. **Mean Imputation:** In the case where the missing values are substituted with the mean value of the entire feature column, observe that imputing missing data with mean values can only be done with numerical data. The Python function mean() is used to replace missing values.

4. **Sample Imputation:** Extract a random sample from the train and test independently using the values that are not missing; indexes of missing values are stored because these would be provided by a previously selected random sample. Now re-index the randomly extracted sample, just as you did with the missing index in the original data set, and ultimately replace the missing values with the re-indexed random sample. The python random() function is used to generate the random sample. To better understand how imputation of a random sample works, refer the code snippet (see Figure 4.3) in the chapter 4 under 4.2.

Bad Imputation:

As the name signifies, missing values are imputed with subpar values (which are not in the range of the given column values). For example, if location latitude or longitude values are lacking, those missing values are imputed with some out-of-range geo-locations. Similarly, for $usertype$ (subscriber or customer), the column is imputed with some arbitrary value like 'agent'. This technique is applied to train and evaluate equally with various proportions (1% to 40%). A later model is adapted to train and predict to compute the evaluation metrics $RMSE$ and $R2$.

Bootstrap Interval for Univariate Estimates:

Bootstrap is predominantly used for resampling, also called the ensemble method; it's either used for estimating the variance of that estimate or for obtaining a confidence interval for the estimate. In the experiment, obtained 95% confidence estimates by calculating a confidence interval between 0.25 and 0.975 quantiles. Resampled , train, and test with replacement of the same sample size independently. These bootstrap samples are fit and predicted with linear regression to determine $RMSE$ and R^2. This resampling technique is repeated 500 times to generate 500 model estimates, after which a 95% confidence interval for the $RMSE$ and R^2 estimates is computed.

3.4.2 Multivariate Imputation

Multivariate imputer is the one that estimates each feature from all the others; it's a strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion[Pedregosa et al., 2011].

The IterativeImputer API from Scikit-Learn is ideally adapted for the experiment; it claims that it's been inspired by the standard R library [van Buuren and Groothuis-Oudshoorn, 2011] but differs from it by returning a single imputation instead of multiple imputations. However, iterative imputation can also be used for multiple imputation by applying it repeatedly to the same dataset with varying random inputs and sample sizes. If the sample_posterior parameter is true **Caution:**, a call to the transform method of IterativeImputer is not permitted to alter the number of samples; therefore, multiple imputations cannot be achieved by a single call to transform [Pedregosa et al., 2011], but applying the method multiple times can be achieved.

It's essential to comprehend how the IterativeImputer functions works internally; it's a robust and computationally expensive method. The blog [T., 2022] explains the granular level step-by-step explanation of IterativeImputer functionality. It accepts an arbitrary sklearn estimator and attempts to impute missing values by modeling other features as a function of features with missing values.
The below steps depicts internal working of IterativeImputer method,

1. The selected regressor is passed to the transformer.
2. The first feature (column1) with missing values is chosen.
3. The regressor is fit on all the other variables as inputs and with column1 as an output.
4. The regressor predicts the missing values.
5. The transformer continues this process until all features are imputed.
6. Steps 1–5 are called a single iteration round, and these steps are carried out multiple times as specified by the max_iter transformer parameter.

IterativeImputer accepts different input parameters [scikit library, 2022a], which are listed in the below points

- $estimator$**:** Estimator object to use at each step of the round robin; by default it uses BayesianRidge(); this can be set to different values (RandomForestRegressor(), Ridge(), KNeighborsRegressor(), etc.).
- $sample\_posterior$**:** It's a boolean whose default value is false; for multiple imputations, it is set to true, and it shows whether to sample from the (Gaussian) predictive posterior of the fitted estimator for each imputation, the estimator must support $return\_std$ in its predict method.
- max_iter**:** It's an int value, and the default is 10. It shows the maximum number of imputation rounds to perform before returning the imputations computed during the final round. A round is a single imputation of each feature with missing values. The stopping criterion is met once $max(abs(X_t - X_t - 1))/max(abs(X[known_vals]))\ tol$, where X_t is X at iteration t. Note that early stopping is only applied if $sample\_posterior = False$.
- tol**:** Tolerance of the stopping condition, it's a float by default, is set to $1e^{-3}$.
- $imputation\_order$**:** The order in which the features will be imputed. Possible values are ascending, descending, roman, random, and arabic. More information can be found on the scikit-learn website [scikit library, 2022a].
- $random\_state$**:** The integer seed of the pseudo-random number generator to use randomizes selection of estimator features if the $imputation\_order$ is random and the sampling from the posterior ($sample\_posterior$) is set to true.

Simulation focused on two forms of multivariate imputation. 1. Multiple Regression (see Figure 3.8); 2. Multivariate Imputation via Chained Equations (see Figure 3.9). Both the experiments can be conducted by using the IterativeImputer; a comprehensive explanation of implementation can be found in Section 4.3.

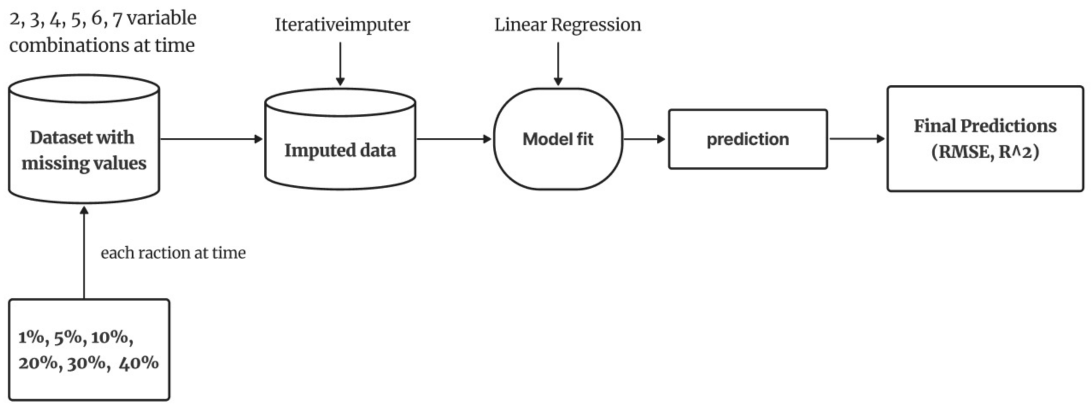


Figure 3.8: Schematic illustration of Multiple Regression

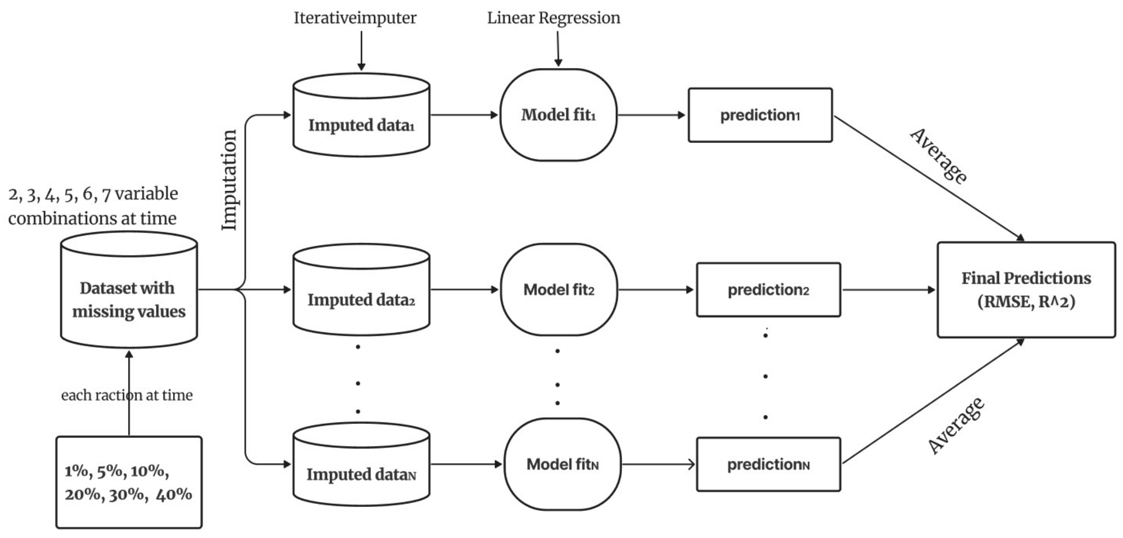


Figure 3.9: Schematic illustration of MICE

3.4.3 Train on Complete Dataset

One of the requirements of the thesis is compiling complete data in order to compare estimates of the imputed estimates with ground truth estimates. Therefore, training on comprehensive data (without any incomplete values) is very essential. In the simulation process, the data is split into train and test (80/20), converted from categorical values into numerical using the OneHotEncoder technique, and then implemented into a supervised linear regression model to fit the train set and validated on unobserved test data to derive the $RMSE$ and R^2 metrics. These results are contrasted with imputed univariate and multivariate estimates; detailed implementation is available in Chapter 5.

3.4.4 Machine Learning Algorithm

For the experiment, which extensively used multiple linear regression, it attempted to characterize the relationship between 22 variables by fitting a linear equation to observed data, where 21 variables classify as explanatory variables, and tripduration acts as a dependent variable. As we know, the fundamental linear regression equation can be described as follows: A linear regression line has an equation of the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The inclination of the line is b, and an is the intercept (the value of y when x = 0).

This can be expanded for multiple linear regression features by expanding the equation for the number of variables accessible within the dataset. The equation for multiple linear regression is analogous to the equation for a simple linear equation, i.e., $y(x) = p_0 + p_1 x_1$ plus the additional weights and values for the different characteristics, which are represented by $p(n)x(n)$. The formula for multiple linear regression can be expressed as: $y(x) = p_0 + p_1 x_1 + p_2 x_2 + . + p(n)x(n)$. The machine learning algorithm utilizes the above formula and different weight values to create lines that fit. Moreover, to determine which line best matches the data, the model considers different weight combinations that best fit the data and which establishes a powerful relationship between the variables [Vijay, 2022].
There are many benefits of using linear regression to build a model,

- **Easiness:** It's one of the easiest, simplest, yet most effective models in the entire machine learning field, as it's computationally simple to implement and does not require a lot of engineering expenditures, neither before the model's introduction nor during its maintenance.

- **Interpretability:** As it's very simple to understand and comparatively straightforward compared to other deep learning models (neural networks), the interpretation of the findings makes it easy to explain which input variable causes the output variable to change.
- **Scalability:** Linear regression is technologically inexpensive and, therefore, works well in instances where scaling is important. For example, the algorithm can expand well with respect to increased data amount (big data).
- **Optimal for online settings:** The model can be learned and retrained with each new example to generate predictions in real-time, unlike neural networks or support vector machines that are computationally expensive and require plenty of processing resources and significant latency periods to retrain on a new dataset. All of those factors make such compute-intensive models expensive and inappropriate for real-time applications [Vijay, 2022].

In the experiment, used the sklearn library LinearRegression model with $fit\_intercept$ true, which depicts, whether to compute the intercept for the model. If set to False, no intercept will be used in calculations (i.e., the data is expected to be centered) [scikit library, 2022b].

3.5 Evaluation Metrics

To understand the machine learning algorithm, performance assessment measures play a vital role. It is necessary to acquire accuracy from training data, but it's also important to get an authentic and approximate outcome from unforeseen data; otherwise, the model is of no use. Therefore, to develop and implement a generalized model, it's important to evaluate that simulation on various measures, which helps us better maximize the performance, fine-tune it, and obtain a better outcome [Jäger et al., 2021]. To evaluate the experiments, used two metrics: root mean square error($RMSE$) and R squared(R^2).

The RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y_i})^2}$$

where N is the number of observations, y_i is the observed values,and $\hat{y_i}$ is the predicted values. R^2 is a statistical measure that tells us how well our model is making all its predictions on a scale of 0 to 1. In the experiment used R^2 score to determine the accuracy of our model in terms of distance or residual.

The R2 score is defined as:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Where R^2 is coefficient of determination, RSS is sum of squares of residuals and TSS is total sum of squares between y_i is the observed values, and $\hat{y_i}$ is the predicted values. Where $RSS = \sum\left(y_i - \hat{y_i}\right)^2$ and, $TSS = \sum\left(y_i - \hat{y_i}\right)^2$. For regression tasks involving imputing columns and to evaluate model accuracy, $RMSE$ and R^2 metrics being used. The $RMSE$ is an error metric; a reduced value indicates greater performance. R^2 is an accuracy measure: the higher the value, the greater the accuracy.

Chapter 4

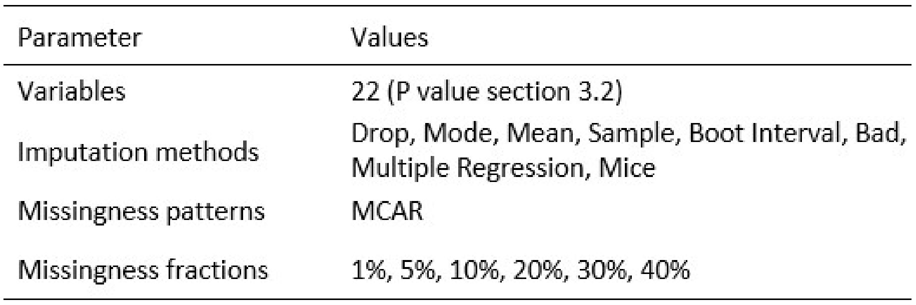
Implementation and Experiments

This section provides a comprehensive explanation of the implementation of the specified methodology and the experimental setting to perform the simulation. As described in section 3.4, for univariate and multivariate imputation, a common set of algorithms is defined where it produces the missing values with different proportions on each column at a given time for train and test separately, it applies imputation techniques on these missing value columns to produce complete data set before applying a regression model to the train set and the test set to produce evaluating metrics. Research was performed using python's scikit-learn version 1.0.2 and R to verify the p values and fit a mini model on both Python and R to compare the estimates. Used CPU with 8GB RAM and BHT clusters with a P100 GPU for the experiment.

In the first place, creating an 80/20 train-test split that is mainly focused on the missingness pattern MCAR, to artificially induce the missing values as described in the section 3.3 is necessary to observe the behavior of the simulation at different fractions of missingness. Also, different imputation methods [drop, mode, mean, sample, bad, multiple regression, mice] have been applied to each induced missing value proportion, i.e., 1%, 5%, 10%, 20%, 30%, and 40%.

4.1 Experimental Settings

The experimental settings are listed in table format (see Figure 4.1), and each imputation method is simulated on all 21 variables one at a time (which doesn't include the dependent variable) using the missingness pattern MCAR with different missing fractions. The experiment is split into three parts. 1. Simulation of Univariate Imputation (see section 4.2), 2. Simulation of Multivariate Imputation (see section 4.3), and 3. Training on Complete Data (see section 4.4).



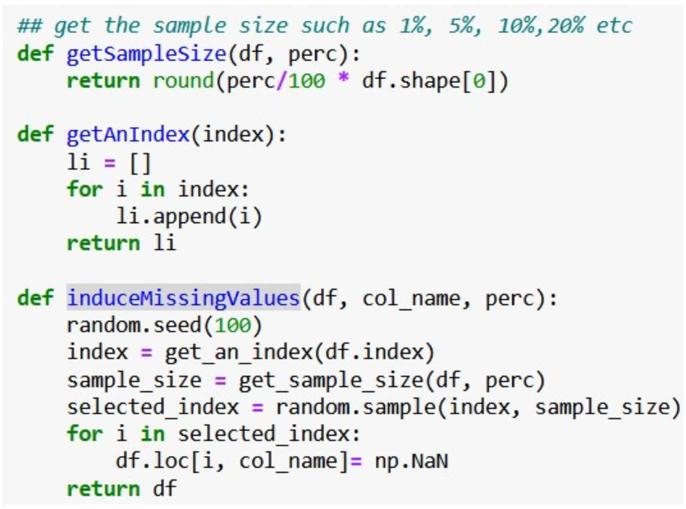
Parameter	Values
Variables	22 (P value section 3.2)
Imputation methods	Drop, Mode, Mean, Sample, Boot Interval, Bad, Multiple Regression, Mice
Missingness patterns	MCAR
Missingness fractions	1%, 5%, 10%, 20%, 30%, 40%

Figure 4.1: Overview of experimental settings.

Before stepping into 3 different simulations, an experiment is conducted (mini model) on Python and R using 4 independent variables (i.e., distance, hour, birthyear, and temperature) and one dependent variable (tripduration) to check the $RMSE$, R^2, and model coefficient. The data is divided into train and test (80 and 20), respectively, and the estimates are obtained using Linear Regression. The figures A.2 for Python and A.3 for R represent the estimates. These estimates shows exactly the same values, the $RMSE$ is 2.326, R^2 is 0.4948, Intercept is 28.97, and the coefficients are $[dist : 4.9883305, hour : 0.0426766, birthyear : 0.0139814, temp : 0.0231373, snowdepth : 0.2046418]$. As per the results confident that the Linear Regression estimates are not dependent on programming platform or programming language, it may change for other adding some more features or categorical variables, hence can't strictly claim that LR is programming platform independent. This small test is conducted basically to understand, how LR varies depending on programming language.

4.2 Simulation of Univariate Imputation

As demonstrated in Chapter 3 (Methodology) (see section 3.4), how univariate can be conducted and which all imputation techniques involved in this thesis are all illustrated with concise. As described it's a simulation of single column imputation, is run on all the 21 independent columns using different imputation methods with different proportions each at a time on the all columns except dependent variable, artificially introduced missing values ranging from 1% to 40%. The code for artificially introducing missing values (see figure 4.2) according to MCAR, method $induceMissingValues()$. The function $getSampleSize()$ and $getAnIndex()$ are responsible for obtaining sample indices for specified percentages (see Figure 4.2).

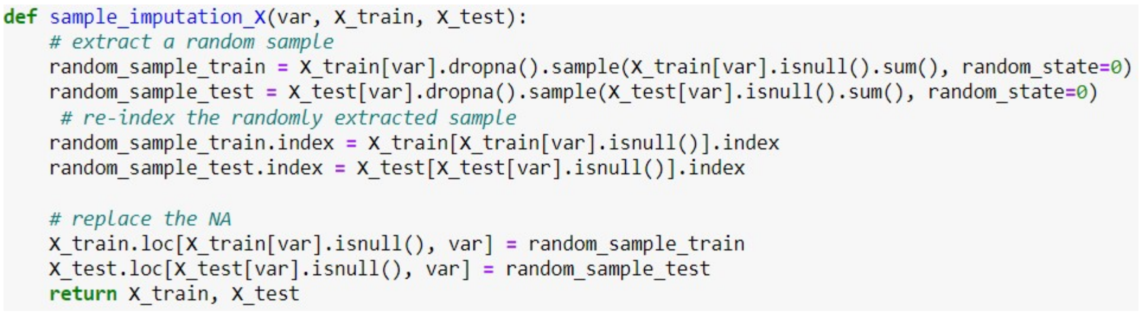


```
## get the sample size such as 1%, 5%, 10%,20% etc
def getSampleSize(df, perc):
    return round(perc/100 * df.shape[0])

def getAnIndex(index):
    li = []
    for i in index:
        li.append(i)
    return li

def induceMissingValues(df, col_name, perc):
    random.seed(100)
    index = get_an_index(df.index)
    sample_size = get_sample_size(df, perc)
    selected_index = random.sample(index, sample_size)
    for i in selected_index:
        df.loc[i, col_name]= np.NaN
    return df
```

Figure 4.2: MCAR Pattern for Univariate missingness.

The understand it more precisely see algorithm 1, which provides a clear overview of the simulation procedure. It is a common unit of code for univariate simulation that executes on all columns with the following inputs parameters 1) df (dataframe) which contains all 22 columns, 2) X is 21 independent column names. When starts simulating it iterate through each independent column X, where x is one column at a time, dataset is divided into train (80%) and test (20%), fraction is the percentage of missingness (MCAR) code, which is above explained as single inducing missingness 4.2, and $induceMissingValues()$ is applied equally on both train and test. The missing value data of the train and test is imputed using $imputeUnivariate()$ method by selecting the type of imputation method (drop, mode, mean, sample, bad); however, the mean() method is only applied to the numerical column and the rest of the imputation techniques are applied to all of the columns. The code for sample imputation is mentioned in the figure 4.3, as discussed in the

3.4.1 univariate imputation. The method OneHotEncoding is applied to both train and test data individually, to convert the categorical values into numerical values, later the model is fit using transformed train data and validated using unseen transformed test data to derive the $RMSE$ and R^2, so each column's $RMSE$ and R^2 are logged into a dictionary in each iteration, and a final dictionary holding all these estimates is returned, this dictionary can be used to analyse the results at alter stage.



```python
def sample_imputation_X(var, X_train, X_test):
    # extract a random sample
    random_sample_train = X_train[var].dropna().sample(X_train[var].isnull().sum(), random_state=0)
    random_sample_test = X_test[var].dropna().sample(X_test[var].isnull().sum(), random_state=0)
     # re-index the randomly extracted sample
    random_sample_train.index = X_train[X_train[var].isnull()].index
    random_sample_test.index = X_test[X_test[var].isnull()].index

    # replace the NA
    X_train.loc[X_train[var].isnull(), var] = random_sample_train
    X_test.loc[X_test[var].isnull(), var] = random_sample_test
    return X_train, X_test
```

Figure 4.3: Random Sample Imputation Code Snippet

The Complete Univariate simulation can be found in below algorithm.

Algorithm 1: Simulation of Univariate Imputation methods

```
Data: Dataframe(df) and all the dependent feature names(X).
Result: Univariate of RMSE and R2 of each Columns
begin
    for $x \longleftarrow X$ do
        $train, test \longleftarrow split(df)$;
        /* different fractions 1%, 5% , 10%, 20%....                    */
        $train, test \longleftarrow induceMissingValues(x, fraction, train, test)$;
        /* type of imputation method drop, mean, mode....               */
        $train.imputed \longleftarrow imputeUnivariate(X, type, train)$;
        $test.imputed \longleftarrow imputeUnivariate(X, type, test)$;
        $train.trasformed \longleftarrow OneHotEncoding(train.imputed)$;
        $test.trasformed \longleftarrow OneHotEncoding(test.imputed)$;
        $model \longleftarrow model.fit(train.trasformed)$;
        $predict \longleftarrow model.predict(test.trasformed)$;
        $RMSE, R2 \longleftarrow predict$ /* log RMSE and R2 of each variable   */
```

4.2.1 Bootstrap Interval for Univariate Estimates

As discussed in the section 3.4.1, the experimental setup is conducted to determine the boot interval of each univariate imputation 3.4 on each column. The algorithm (see algorithm 2) provides a clearer comprehension of how to acquire boot interval (2.5 and 97.5) estimates ($RMSE$ and R^2) for each column.

The detailed explanation of the common algorithm used on all the imputation method to obtain the bootstrap interval estimations. The function inputs parameter are 1) df (dataframe) which contains all 22 columns, 2) X is 21 independent column names, when algorithm starts running it iterates through each column X in a for loop, assigning each column to x, where data is divided into train (80%) and test (20%), and $fraction$ is the percentage missingness (MCAR), to induce artificially on each column at a time (x). Similarly same steps (split dataset, artificially insert missing values for various fractions, impute missing values, OneHotEncoding) are followed as mentioned in the Univariate simulation 4.2 for the imputation methods, however, as simulation required to run one extra step where it requires to capture the bootstrap interval for these univariate imputation estimates. The one extra step required one more iteration to get the boot strap samples hence to obtain bootstarp sample B (500 samples), for each boot iteration (i), $sampleWithReplacement()$ method is called on both train and test, producing the same-sized sample as train and test but with replacement. When linear regression is fit and validated on test samples, boot estimates are logged for all 500 iterations. The final $RMSE$ and R^2 are calculated using the percentile 2.5 and 97.5 intervals; note: these boot estimates are logged for each variable, each imputation and each fractions of missingness.

The algorithm for boot interval estimation of Univariate imputations as follows.

Algorithm 2: Boot Strap Interval for Univariate Imputation methods

```
Data: Dataframe(df) and all the dependent feature names(X).
Result: Boot interval of RMSE and R2
begin
    for $x \longleftarrow X$ do
        /* x is a column ...                                              */
        $train, test \longleftarrow split(df)$;
        /* different fractions 1%, 5% , 10%, 20%....                      */
        $train, test \longleftarrow induceMissingValues(x, fractions, train, test)$;
        /* type of imputation method drop, mean, mode....                 */
        $train.imputed, test.imputed \longleftarrow imputeUnivariate(x, type, train, test)$;
        $train.trasformed, test.trasformed \longleftarrow$
          $OneHotEncoding(train.imputed, test.imputed)$;
        for $i \longleftarrow B$ do
            /* where B is number of boot iterations produce B samples     */
            $train.sample \longleftarrow sampleWithReplacement(train.trasformed)$;
            $test.sample \longleftarrow sampleWithReplacement(test.trasformed)$;
            $model \longleftarrow lm.fit(train.sample)$;
            $predict \longleftarrow model.predict(test.sample)$;
            $boot.estimates \longleftarrow predict$;
        $percentile(boot.estimates, 2.5)$ and $percentile(boot.estimates, 97.5)$ ;
    return $RMSE$, $R2$
```

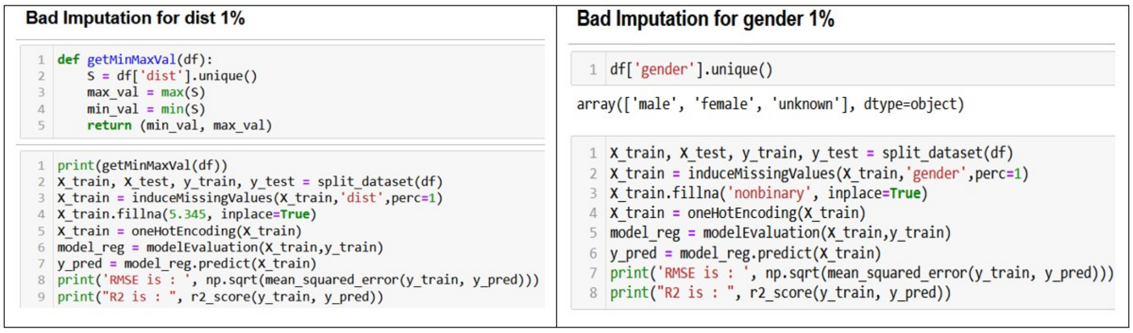
To summarize the boot intervals of each column are contrasted with the original univariate imputation method estimates of each column, each imputation method, and each percentage of missingness. However, if the original univariate imputation estimates are within the boot interval estimates, then it's an acceptable or good estimate; otherwise, it's termed as biased estimate.

4.2.2 Bad Imputation for Univariate.

As explained briefly in the section 3.4.1, to see how bad values (which are not in the range of given column values) imputation impact model estimates, the experiment is conducted using both train and test data of different missingness proportions(1%, 5%, 10%, 20%, 30%, 40%). For the experiment selected all 21 columns as independent variable and 1 dependent variable.

The bad imputation, is imputing missing values with constant values but which are not within the given column value range (outliers), first step is to identify what would be the outlier value for each columns, for numerical columns technique used is obtaining the minimum and maximum value, and likewise, for the categorical columns identify the categorical levels. Based on the first step after initial identification, second step would be, select any out of range value to impute for each respective column according to the min and max value criteria and categorical levels.

For instance, the min and max values for the dist column are 0.0 and 3.962001591390758 respectively, so imputed an outlier value of 5.345 which is not the range of given values. Similarly, for gender, there are 3 levels of category: male, female, and unknown, so imputed an outlier value of $nonbinary$. Eventually, linear regression is fit on train data and evaluated using test data. For each column simulation is performed, the $RMSE$ and R^2 estimates are logged to compare with univariate imputation and ground truth values to verify how bad value imputation impacts estimates. The above bad imputation simulation description can be seen in the code snippet as follows (see figure 4.4).


Bad Imputation for dist 1%

```
def getMinMaxVal(df):
    S = df['dist'].unique()
    max_val = max(S)
    min_val = min(S)
    return (min_val, max_val)
```

```
print(getMinMaxVal(df))
X_train, X_test, y_train, y_test = split_dataset(df)
X_train = induceMissingValues(X_train,'dist',perc=1)
X_train.fillna(5.345, inplace=True)
X_train = oneHotEncoding(X_train)
model_reg = modelEvaluation(X_train,y_train)
y_pred = model_reg.predict(X_train)
print('RMSE is : ', np.sqrt(mean_squared_error(y_train, y_pred)))
print("R2 is : ", r2_score(y_train, y_pred))
```

Bad Imputation for gender 1%

```
df['gender'].unique()
```

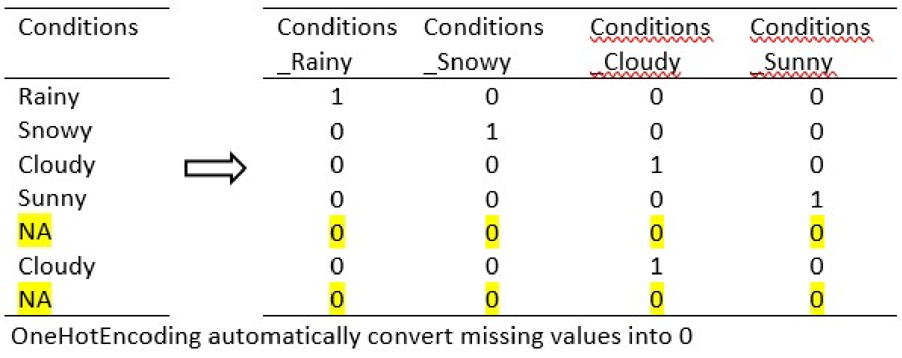
```
array(['male', 'female', 'unknown'], dtype=object)
```

```
X_train, X_test, y_train, y_test = split_dataset(df)
X_train = induceMissingValues(X_train,'gender',perc=1)
X_train.fillna('nonbinary', inplace=True)
X_train = oneHotEncoding(X_train)
model_reg = modelEvaluation(X_train,y_train)
y_pred = model_reg.predict(X_train)
print('RMSE is : ', np.sqrt(mean_squared_error(y_train, y_pred)))
print("R2 is : ", r2_score(y_train, y_pred))
```

Figure 4.4: Bad Imputation Simulation.

4.3 Simulation of Multivariate Imputation

As demonstrated in Chapter 3 (Methodology) (see section 3.4.2), how multivariate can be conducted and which all imputation techniques involved in this section are all illustrated with diagrams. The multivariate imputation simulation is conducted two phases, i.e., 1. multiple regression (see Figure 3.8) and 2. MICE (multivariate imputation via chained equations) (see Figure 3.9), as discussed in section multivariate imputation, only numerical columns are considered to artificially induce the missing values according to the MCAR missingness pattern with different fractions of missingness, where categorical columns are discarded. The reason for not selecting categorical columns in the experiment is that, as OneHotEncoding produces N columns as explained (see Figure 3.5) in the section 3.2.1, but while producing N columns if any missing values are present in that columns while applying OneHotEncoding, it automatically converts the missing values into 0, then all the rows for that missing value (see Figure 4.5) becomes 0. Due to this behavior, it's challenging to identify which are missing and which are not, particularly when the drop=$first$ methodology is applied.

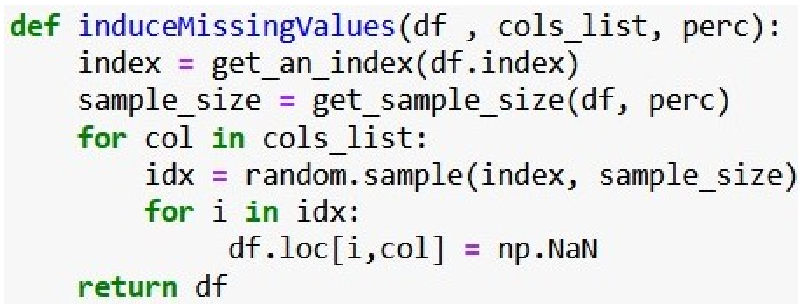


Conditions	Conditions _Rainy	Conditions _Snowy	Conditions _Cloudy	Conditions _Sunny
Rainy	1	0	0	0
Snowy	0	1	0	0
Cloudy	0	0	1	0
Sunny	0	0	0	1
NA	0	0	0	0
Cloudy	0	0	1	0
NA	0	0	0	0

OneHotEncoding automatically convert missing values into 0

Figure 4.5: OneHotEncoding Behaviour in Missingness.

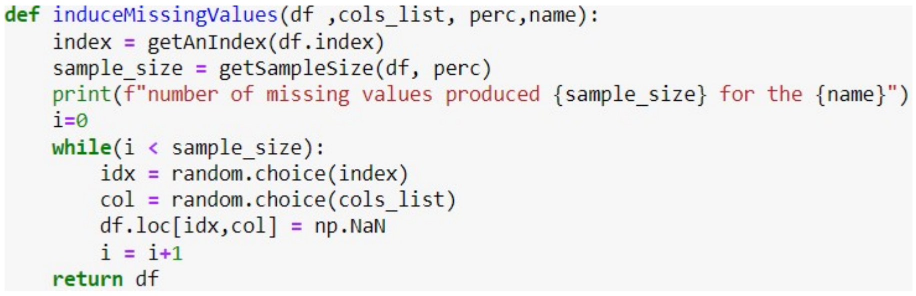
It's also essential to comprehend how missing values are incorporated into multivariate imputation. The below code snippet is similar to the univariate missing value code snippet (see 4.6), but the only minor change is that it takes a list of columns (a different set of combinations), and for each column in the set of combinations ["dist", "birthyear"], independently fractions of missing values are inserted. For instance, if the columns dist and birthyear are passed as a set of 2 column combinations and the percentage is 10%, then the "dist" column has 10% of the missing values, and similarly, "birthyear" has 10% of the missing values, hence it's referred as artificially inserting missing values "independently". However, these set of columns combinations changes from 2 to 7 and different fractions 1% to 40%. Consider another instance if simulation is conducting for 5 set of columns ('dist', 'birthyear', 'start_lat', 'start_lon', and 'end_lat'), with fractions 40% then the insertion of missing values would be "dist" columns has 40%, "birthyear" has 40%, "start_lat" has 40%, "start_lon" has 40% ,"end_lat" has 40% independently.



```
def induceMissingValues(df , cols_list, perc):
    index = get_an_index(df.index)
    sample_size = get_sample_size(df, perc)
    for col in cols_list:
        idx = random.sample(index, sample_size)
        for i in idx:
            df.loc[i,col] = np.NaN
    return df
```

Figure 4.6: MCAR Multivariate missingness Independent of columns.

Another scenario for multivariate imputation, where artificially inducing missing values "dependently". The code above 4.6 can be extended for the inducing missing values dependently scenario by adapting some minor code changes. The below final code snippet for the artificially inserting missing values dependently (see Figure 4.7). For the description consider, 2 set of columns [$start\_lat$ and $start\_lon$] with 20% of missing values fractions, now it inserts 20% of null values on both column together randomly, similarly for set 5 columns ('dist', 'birthyear', 'start_lat', 'start_lon', and 'end_lat') with fraction 20%, then together all 5 columns, in total 20% of missing values are inserted. hence it's called as inducing missing values dependently. However, these set of columns combinations changes from 2 to 7 and with only 20% fractions. In this case, a dist feature was not included in the column combinations (which notably generate missing values) but included in the model construction with complete data.



```
def induceMissingValues(df ,cols_list, perc,name):
    index = getAnIndex(df.index)
    sample_size = getSampleSize(df, perc)
    print(f"number of missing values produced {sample_size} for the {name}")
    i=0
    while(i < sample_size):
        idx = random.choice(index)
        col = random.choice(cols_list)
        df.loc[idx,col] = np.NaN
        i = i+1
    return df
```

Figure 4.7: MCAR Multivariate missingness Dependent on columns.

4.3.1 Multiple regression

The architecture of the multiple regression imputation is explained in Figure 3.8. To make use of scikit learn library method $Iterativeimputer$ for multiple regression it's requires certain tweaks in the standard method parameters, by default $Iterativeimputer$ uses $BayesianRidge()$ as an imputer estimator, so for the experiment adapted same default estimator. The max_itr equals 1, which produces only one imputed dataset, and $sample\_posterior$ is set false(which means not using sampling from the (Gaussian) predictive posterior of the fitted estimator for each imputation), see Figure 4.

The simulation conducted in two phases 1.) artificially inserting missing values independently using all the fractions(1% to 40%) (see Figure 4.6) on all the set of columns combinations(2,3,4,5,6,7) and, 2.) artificially inserting missing values dependently using only 20% on all the set of columns combinations(2,3,4,5,6,7) (see Figure 4.7).
For the first scenario, the common algorithm used to simulate the multiple regression is 3, where it takes input parameters 1.) df (dataframe) with all the 22 variables 2.) set of column combinations at time $[x_1, x_2..]$, $[x_2, x_3..]$...$[x_{(n-1)}, x_n..]$ (2 to 7 set of columns), where it iterates through given set of column in loop, where it assigns each list to $[x_1, x_2]$ in single iteration, the dataset is divided into train (80%) and test (20%), then artificially inducing missing values independently on given set columns $[x_1, x_2]$, for the given fractions (varies between 1% to 40%) (see Figure 4.6) on both train and test, the method $induceMissingValues()$ is responsible to it. Now the $OneHotEncoding()$ is applied to produce numerical columns on both train and test individually, it's time to impute missing values using standard Iterativeimputer method according to the above specified criteria on both train and test set, and then imputed data is returned by $fit\_transform$. This imputed complete data is fit using linear regression to derive the $RMSE$ and R^2 estimates, this whole procedure is repeated for next set given of columns. However, simulation is only included numerical columns. Also the algorithm for multiple regression imputation using Iterativeimputer function is explained in 4.

For second scenario, also the same common algorithm used see Algorithm 3, same multiple regression setup for Iterativeimputer, same set of columns (see Algorithm 4), but only difference is in 1.) artificially introducing missing values, 2.) the fractions of missing values 3.) dist variable not included in the missing value set. As explained already, how to insert missing values dependently see Figure 4.7. As considered only 20% of missing values dependently on all the combinations (2, 3, 4, 5, 6, 7) at a time with same set of common algorithm to produce final estimates $RMSE$ and R^2.

Algorithm 3: Simulation of Multiple Regression Imputation methods

```
Data: Dataframe(df) and combinations of columns [[x1,x2],[x2,X3]....[xn-1, xn]].
Result: Estimates RMSE and R2
begin
    for $[x_1, x_2] \longleftarrow [[x_1, x_2], [x_2, x_3]...[x_{(n-1)}, x_n]]$ do
        $train, test \longleftarrow split(df)$;
        $train, test \longleftarrow induceMissingValues([x_1, x_2], fraction, train, test)$;
        $train.transformed \longleftarrow OneHotEncoding(train)$;
        $test.transformed \longleftarrow OneHotEncoding(test)$;
        $y\_predict \longleftarrow$
          $multiple\_imputation(train.transformed, test.transformed, random\_state)$;

        $predictions.append(y\_predict)$;
        $RMSE, R^2 \longleftarrow predictions$;
        /* log RMSE and R2 of each column combinations */
```

Algorithm 4: IterativeImputer algorithm for Multiple Regression

```
Data: train and test, randomvalue.
Result: predictions
begin
    $imputer \longleftarrow IterativeImputer(estimator = BayesianRidge(), max\_iter =$
      $5, sample\_posterior = False, random\_state = randomvalue, tol = 1e^{-1})$;
    $train.imputed \longleftarrow imputer.fit\_transform(train)$;
    $test.imputed \longleftarrow imputer.fit\_transform(test)$;
    $model \longleftarrow model.fit()$;
    $y\_predict \longleftarrow model.predict()$;
```

4.3.2 MICE (Multivariate Imputation via Chained Equations)

The MICE imputation architecture as explained in the previous chapter 3 (see Figure 3.8) is adapted in MICE experiment, used the Python variant Iterativeimputer() from the scikit-learn library extensively used for multiple imputation with prior configurations on it to adopt MICE requirements. The iterative estimator employs the parameters discussed in 3.4.2, a Bayesian-Ridge() default estimator, and max_itr, which is the utmost number of imputation rounds to perform before returning the imputations computed during the final round, the default value is 10, but for the experiment designated value is 5. If $sample\_posterior$ is set true for multiple imputations, this means sampling from the (Gaussian) predictive posterior of the fitted estimator for each imputation. However, simulation is only included numerical columns. The above mentioned criteria is be explained with help of algorithm 6, same is adopted for mice simulation.

Similar to multiple regression imputation, MICE simulation is also conducted in two stages 1.) artificially inserting missing values independently using all the fractions(1% to 40%) (see Figure 4.6) on all the set of columns combinations(2,3,4,5,6,7) and, 2.) artificially inserting missing values dependently using only 20% on all the set of columns combinations(2,3,4,5,6,7) (see Figure 4.7).

For the first scenario, the common algorithm used to simulate the MICE is 3.4.2, where it takes input parameters 1.) df (dataframe) with all the 22 variables 2.) set of column combinations at time $[x_1, x_2..], [x_2, x_3..]...[x_{(n-1)}, x_n..]$ (2 to 7 set of columns), where it iterates through given set of column in loop, where it assigns each list to $[x_1, x_2]$ in single iteration, the dataset is divided into train (80%) and test (20%), then artificially inducing missing values independently on given set columns $[x_1, x_2]$, for the given fractions (varies between 1% to 40%) (see Figure 4.6) on both train and test, the method $induceMissingValues()$ is responsible to it. Now the $OneHotEncoding()$ is applied to produce numerical columns on both train and test individually, it's time to impute missing values using standard Iterativeimputer method according to the above specified criteria, but to adopt Iterativeimputer() to MICE setup, $fit\_transform$ required to run multiple times with different random state, therefore, "mice_imputation()", with differnt random state every iteration, is running for 6 times (both on train and test included in each run), to produce 6 differnt imputed dataset. The method "mice_imputation()" calls the algorithm 6 to impute and apply Linear Regression model on imputed train dataset and validate on imputed test data, to produce predicted tripdurations. These 6 predictions are stored in a list, at the end of iteration predictions average obtained to get final $RMSE$ and R^2 estimates. The same above procedure is run for next set of columns.

For second scenario, also the same common algorithm used see Algorithm 3.4.2, same MICE setup for Iterativeimputer, same set of columns (see Algorithm 6), but only difference is in 1.) artificially introducing missing values, 2.) the fractions of missing values 3.) dist variable not included in the missing value set. As explained already, how to insert missing values dependently see Figure 4.7. As considered only 20% of missing values dependently on all the combinations (2, 3, 4, 5, 6, 7) at a time with same set of common algorithm to produce final estimates $RMSE$ and R^2. The Algorithm for Mice Imputation as follows below.

Algorithm 5: Simulation of Multivariate(MICE) Imputation methods

```
Data: Dataframe(df) and combinations of columns [[x1,x2],[x2,X3]....[xn-1, xn]].
Result: Estimates RMSE and R2
begin
    for $[x_1, x_2] \longleftarrow [[x_1, x_2], [x_2, x_3]...[x_{(n-1)}, x_n]]$ do
        $multiple\_predictions \longleftarrow Null$;
        $train, test \longleftarrow split(df)$;
        $train, test \longleftarrow induceMissingValues([x_1, x_2], fraction, train, test)$;
        $train.transformed \longleftarrow OneHotEncoding(train.imputed)$;
        $test.transformed \longleftarrow OneHotEncoding(test.imputed)$;
        for $i \longleftarrow range(6)$ do
            $y\_predict \longleftarrow$
              $mice\_imputation(train.transformed, test.transformed, random\_state)$;

            $multiple\_predictions.append(y\_predict)$;
        $predictions\_average \longleftarrow avg(multiple\_predictions)$;
        $RMSE, R^2 \longleftarrow predictions\_average$;
        /* log RMSE and R2 of each column combinations */
```

Algorithm 6: IterativeImputer algorithm for Mice

```
Data: train and test, randomvalue.
Result: predictions
begin
    $imputer \longleftarrow IterativeImputer(estimator = BayesianRidge(), max\_iter = 5, sample\_posterior = True, random\_state = randomvalue, tol = 1e^{-1})$;
    $train.imputed \longleftarrow imputer.fit\_transform(train)$;
    $test.imputed \longleftarrow imputer.fit\_transform(test)$;
    $model \longleftarrow model.fit()$;
    $y\_predict \longleftarrow model.predict()$;
```

4.4 Training and Testing on Complete Dataset

As discussed in the section 3.4.3, it's necessary train a model and test on unseen data to obtain the ground truth estimates. So to compare the imputed results, ground truth estimates are required; thus, a model was trained and validated using various supervised regression techniques such as Linear Regression, Lasso Regression, Ridge Regression, Gradient Boosting Regressor, XGBOST, and Random Forest; each of these produces different estimates, but Linear Regression model is best for comparing the estimation of imputed dataset, hence Linear regression is chosen for the final simulation. For the experiment, selected a continuous variable tripduration as a prediction feature and the remaining 21 columns as independent attributes. The comprehensive data estimates are contrasted to imputed estimates, such as $RMSE$ and $R2$. If imputed $RMSE$ and R^2 are close to ground truth estimates then it's considered as a good imputation method for missing values.

The complete data is split into 80% and 20%, where tripduration is dependent feature and rest are independent features. As it's contains the categorical values, before fitting Linear Regression converting categories into numerical values is mandatory, hence used OneHotEncoding method to transform into numerical columns (see Figure 3.5), also used drop='first' special parameter to drop a first column in the produced N columns to avoid the multi co-linearity . Now the Linear Regression model (fit_intercept = True) is trained on train data and tested on test data to obtain the final ground truth estimates $RMSE$ and R^2.

Chapter 5

Results

This chapter examines the results of various imputation methods on missing values from bike share data, including univariate imputation (drop, mode, mean, sample, bad) and multivariate imputation (multiple regression and mice). The chapter 4 provides a comprehensive description of the approach of the experimental arrangement. The univariate imputation procedure is to replace a missing values by a single imputation value without taking into consideration of the uncertainty of the imputation. Also, the multiple imputation method is to take into consideration the uncertainty of the imputation by executing the iterative imputer (with prerequisites for multiple regression and mice) multiple times so that it can provide a precise estimate of missing values. The experimented results are described using visualization, predominantly scatter plots or line plots for all experimental results; these enables to get a reasonable impression of the estimates.

This thesis is concentrated on the imputation methods and their quality, not on attaining a reasonable or perfect prediction of the trip duration; instead, it evaluates the $RMSE$ and R^2 of the imputation methods. Whether the prediction score is low or high doesn't matter; the estimate numbers matters for the evaluation of these distinct imputation methods.

5.1 Univariate Imputation Results

Already detailed description of methods and it's implementation is discussed in previous chapters (see section 3.4.1 and 4.2), how simulation is conducted is elaborated in the form of algorithm and with examples, and based on this simulation, $RMSE$ and R^2 estimates are obtained and visualized in the form of a line graph (see Figure 5.1). Plotted $RMSE$ and R^2 plots for each column, run on different fractions using different univariate methods, these methods are presented in the legends.

The pointplots for Univariate simulation (see Figure 5.1), depicts visualization of imputation estimates(left R^2 and right $RMSE$) and full (ground truth) estimates. Where imputation methods (drop, mode, mean, sample, bad) and full means ground truth values are represented with differnt coloured and marked lines, according to legends can be identified these imputation lines. The x-axis presents simulation fractions(0% to 40%), 0% for no missing values i.e ground truth estimates and rest for imputation fractions.

When imputation estimates align straight line with full estimate values (red dot), it means the imputed estimates are same or very close to ground truth estimates, so it's considered good imputation method for that particular variable.

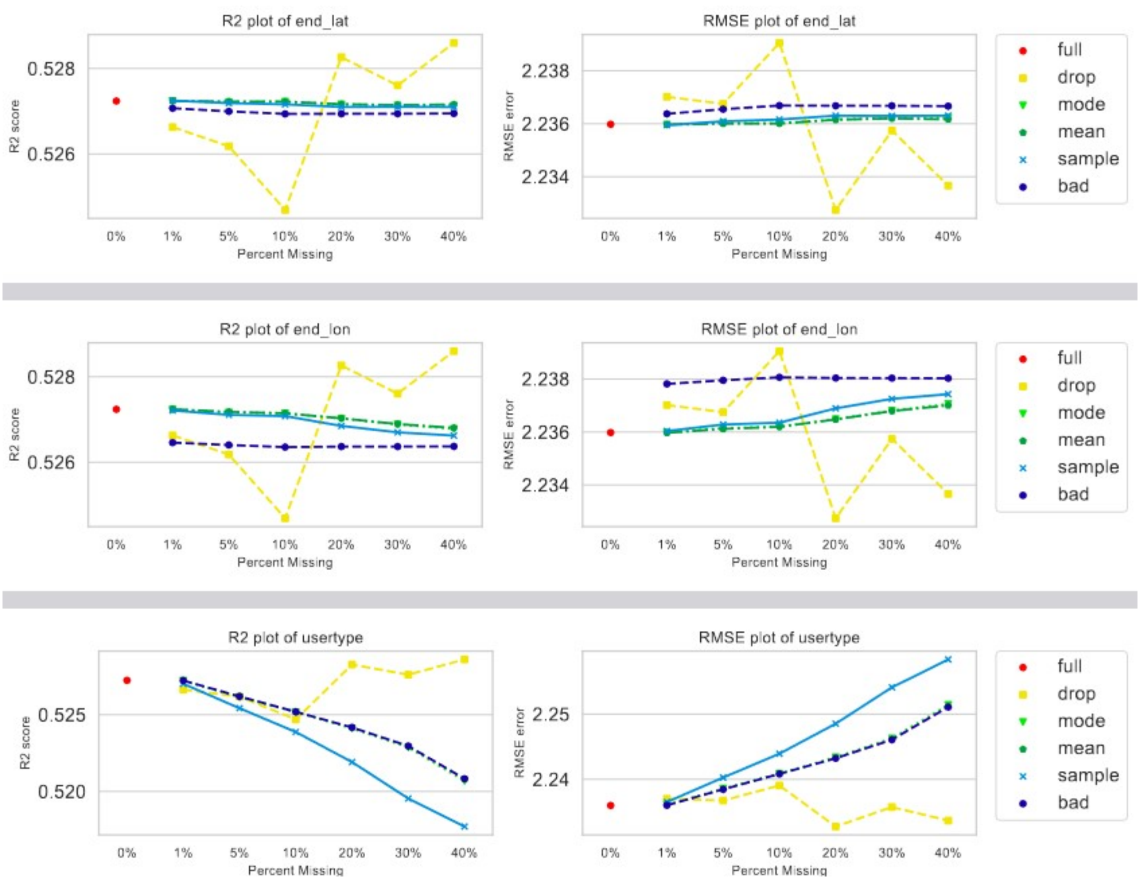
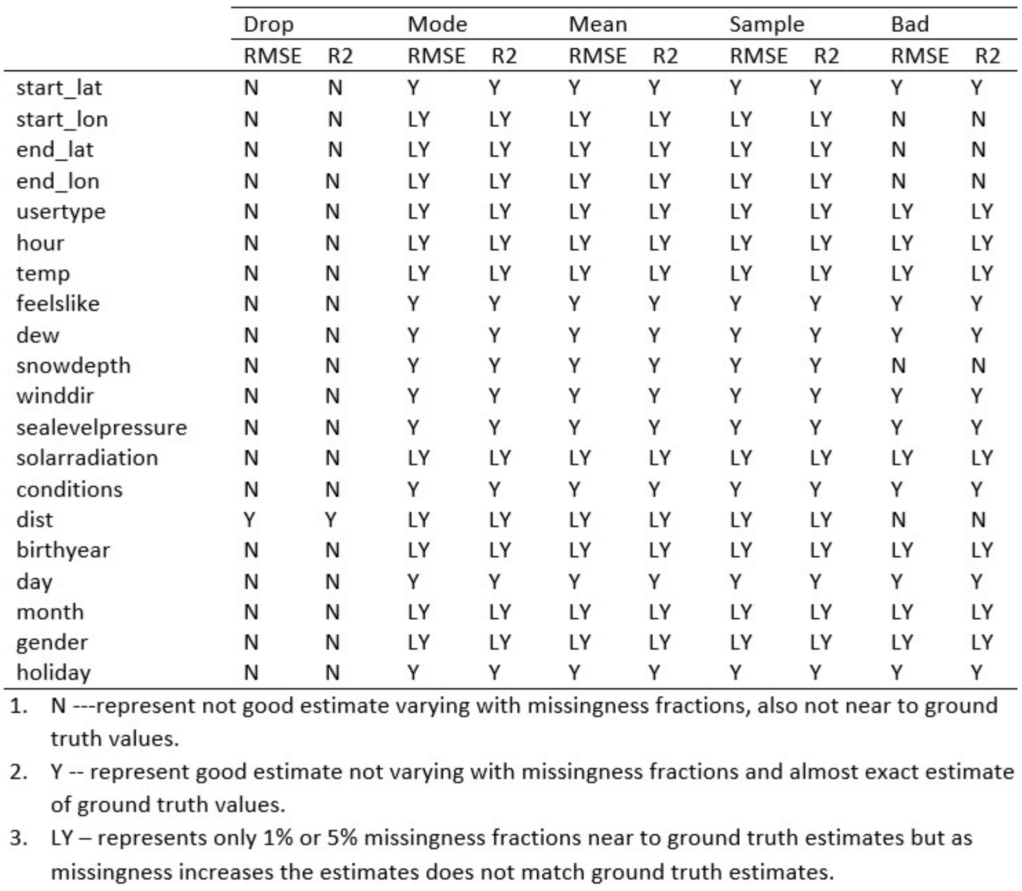


Figure 5.1: Univariate RMSE and R^2 Estimate Plots.

When plot is examined, the mean line (i.e., the usertype column in Figure 5.1) is missing, it's as expected because for all the categorical variables, mean imputation is not applicable; also as expected, the graph shows $RMSE$ and R^2 are indirectly proportional. The plots for all columns can be seen in the github repository [1] under the results folder. The single dots represents the ground truth estimate $RMSE$=2.2359767726747313 and R^2 = 0.5272377063160185 with legend full.

To make analysis easy, each column imputation performance are captured in table format when estimates are compared with ground truth values see Figure 5.2. The columns contains each imputation method and respective estimates ($RMSE$ and R^2), each rows represents the each variable performance with respect to imputation methods, where N represents not performing well compared to ground truth values in all the fractions of missingness, Y indicates showed very close estimates as ground truth estimates in all the fractions of missing values, and LY depicts that showed good estimates when there is 1%, 5% missing values are present, but for higher percentages didn't perform well.

[1] https://github.com/Hinakoushar-Tatakoti/Quantifying-the-influence-of-imputing-missing-values-Bike-Sharing-System



	Drop		Mode		Mean		Sample		Bad	
	RMSE	R2	RMSE	R2	RMSE	R2	RMSE	R2	RMSE	R2
start_lat	N	N	Y	Y	Y	Y	Y	Y	Y	Y
start_lon	N	N	LY	LY	LY	LY	LY	LY	N	N
end_lat	N	N	LY	LY	LY	LY	LY	LY	N	N
end_lon	N	N	LY	LY	LY	LY	LY	LY	N	N
usertype	N	N	LY	LY	LY	LY	LY	LY	LY	LY
hour	N	N	LY	LY	LY	LY	LY	LY	LY	LY
temp	N	N	LY	LY	LY	LY	LY	LY	LY	LY
feelslike	N	N	Y	Y	Y	Y	Y	Y	Y	Y
dew	N	N	Y	Y	Y	Y	Y	Y	Y	Y
snowdepth	N	N	Y	Y	Y	Y	Y	Y	N	N
winddir	N	N	Y	Y	Y	Y	Y	Y	Y	Y
sealevelpressure	N	N	Y	Y	Y	Y	Y	Y	Y	Y
solarradiation	N	N	LY	LY	LY	LY	LY	LY	LY	LY
conditions	N	N	Y	Y	Y	Y	Y	Y	Y	Y
dist	Y	Y	LY	LY	LY	LY	LY	LY	N	N
birthyear	N	N	LY	LY	LY	LY	LY	LY	LY	LY
day	N	N	Y	Y	Y	Y	Y	Y	Y	Y
month	N	N	LY	LY	LY	LY	LY	LY	LY	LY
gender	N	N	LY	LY	LY	LY	LY	LY	LY	LY
holiday	N	N	Y	Y	Y	Y	Y	Y	Y	Y

1. N ---represent not good estimate varying with missingness fractions, also not near to ground truth values.
2. Y -- represent good estimate not varying with missingness fractions and almost exact estimate of ground truth values.
3. LY – represents only 1% or 5% missingness fractions near to ground truth estimates but as missingness increases the estimates does not match ground truth estimates.

Figure 5.2: Comparison with ground truth values of each column.

Also, dropping missing values is not a viable solution to handle the missing value dataset, however, when the missing percentage changes the estimates changes rapidly either decrease or increase, totally unpredictable, also it fails to provide acceptable estimates for all the variables except $dist$, when comparing each estimates with ($RMSE$ and R^2) to ground truth estimates; however, for $dist$ (boot interval estimates are very close or narrow interval see Figure 5.3), it displays the same as ground estimates regardless of the fractions of missingness ($RMSE$, R^2). Mode performed well for some variables regardless of missingness, but it performed poorly for others when there was 1% or 5% missingness. Similarly, mean and sample imputations performed well for some features regardless of missingness, but when the missing value was between 1% and 5%, they performed well. Last but not least, bad imputation doesn't perform well for $start\_lon$, end_lon, and $dist$; however, it showed good results for $start\_lat$, $feelslike$, dew, $winddir$, $sealevelpressure$, $conditions$, day, and $holiday$; and for the rest of variables, it showed good estimates when missingness is 1% or 5%.

5.1.1 Boot Interval Estimates

The bootstrap interval estimates for the each of the imputation is calculated according to the algorithm (see Algorithm 2) as discussed in the section 4.2.1, the following graph gives the idea of each imputation bootstrap estimate interval (0.25 and 0.95) (see Figure 5.3). The graph contains the information of boot intervals as well as the imputation of each method with different fractions of missingness(1% to 40%) on the x-axis, two subplots left R^2 and right $RMSE$ estimates, each imputation methods represented as lines with differnt colours and markers (legend), where the dots above and below the line graph represents the boot interval estimates of respective imputation methods. All the variables boot intervals estimates can be seen in github folder results.

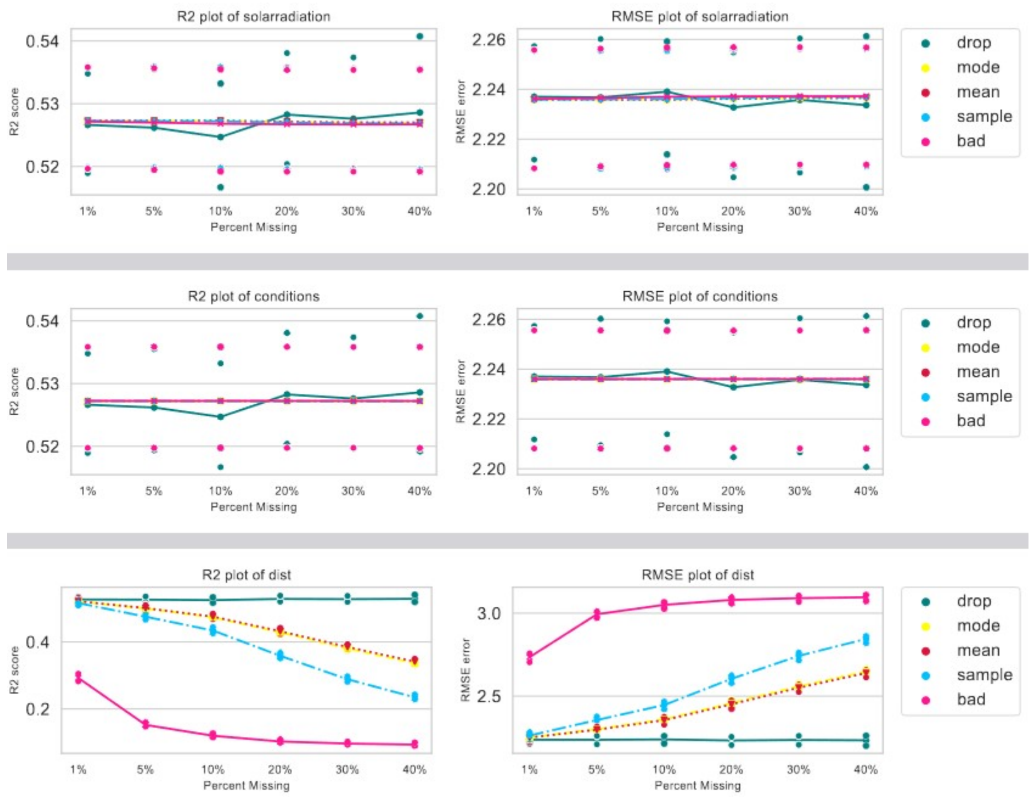


Figure 5.3: Univariate Imputation Bootstrap Estimate Interval.

After the analysis of boot estimation of each variable, the $dist$ variable indicates the boot estimates intervals are very narrow, for each imputed estimates and, for each fractions (see Figure 5.3). However, for the rest of the variables boot interval is broader for each imputation and for each fractions. All the imputation methods with all fractions are within the bootstrap interval, hence these columns estimates are not biased.

5.2 Multivariate Imputation Results

Already a comprehensive description of methods and their implementation is discussed in previous chapters (see section 3.4.2 and 4.3). How multivariate simulation is conducted, is elaborated in the form of an algorithm and with examples, and based on this simulation, $RMSE$ and R^2 estimates are obtained and visualized in the form of a line graph. These graphs provides better understanding of the estimations to compare with the ground truth estimates.

As the simulation is conducted in two phases for both multiple regression (4.3.1) and mice imputation (4.3.2) by artificially inserting missing values independently (1% to 40%) and artificially inserting missing values dependently (20%).

For first scenario artificially inserting missing values independently (1% to 40%), the both multiple regression and mice graphically using lines, two subplots left R^2, and right $RMSE$, x-axis represents different fractions of missingness and legends describes as set of columns (2, 3, 4, 5, 6, 7), as already described in the previous sections, it start with 2 columns having missing values, then 3 columns has null values, 4 columns having missing values , and so on until 7 columns have missing values. The columns keep on adding to 2 set of columns till 7 columns, each of these set produced estimates are plotted on the graph. These set of columns are represented on legends with differnt colours and markers in the plot. The Figure 5.4 and 5.5 depicts the about the given explanation.

For the second scenario artificially inserting missing values dependently (20%), the both multiple regression and mice graphically using scatter plots, two subplots left R^2, and right $RMSE$, x-axis represents different set of columns (0, 2, 3, 4, 5, 6, 7), where 0 represents ground truth estimates (no missing values) and rest are imputed set of columns. The three estimates 1.) ground truth estimates, 2.) multiple regression and 3.) mice estimates are plotted on the graph with differnt colours as represented in the legends. The Figure 5.6 depicts the above explanation.

5.2.1 Multiple Regression Imputation

For the first scenario of inserting missing values independently, after the analysis of the visualization plot 5.4, for each set of columns and each fraction, it can be summarized as all the graphs looking the same because the obtained estimates showed a slight change in the imputed estimates, for example, $RMSE$ = 2.248695709534565 for 2 sets of columns, $RMSE$ = 2.2486934530162555 for 3 sets of columns, and $RMSE$ = 2.248669242740133 for 4 sets of columns.

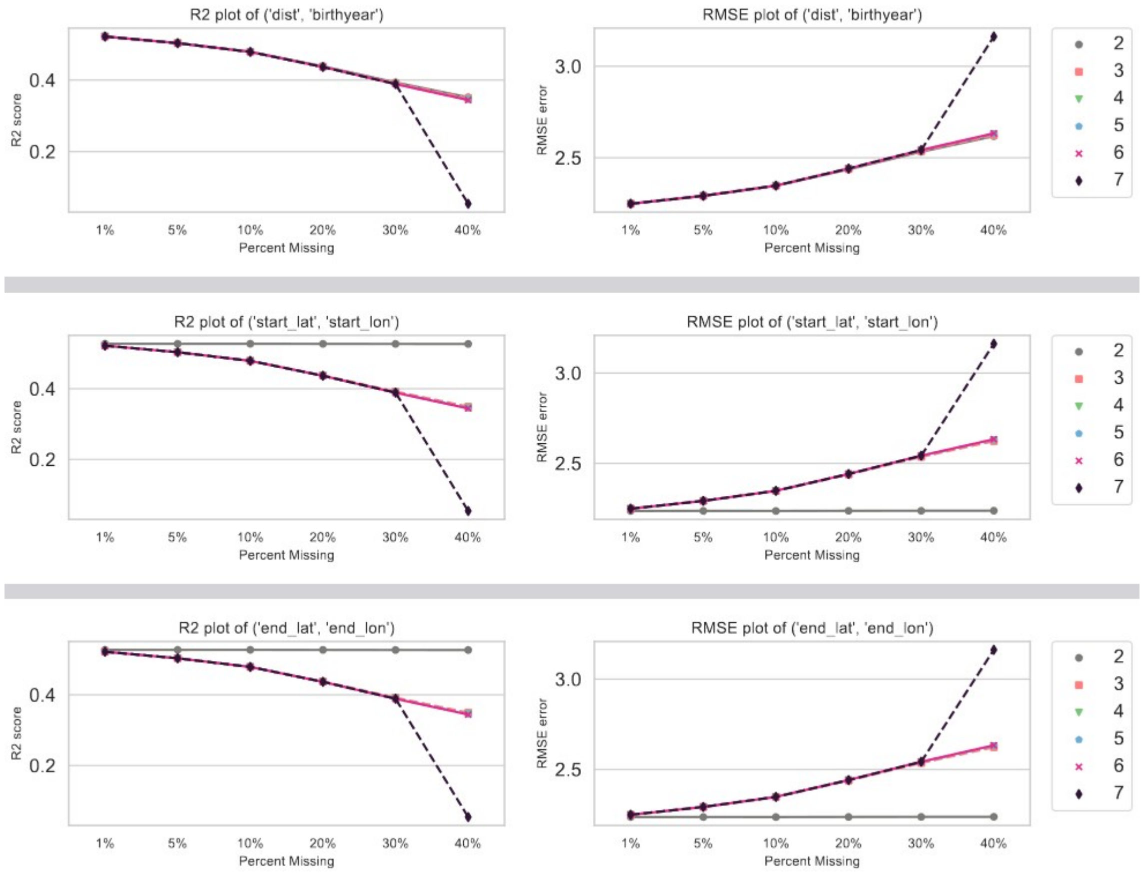


Figure 5.4: Multiple Regression Estimates

For the set of 2 columns, with all missing fractions, it showed decent estimates compared to ground truth estimates; excluding "dist" columns in the set of 2 columns, it showed the same estimates (a straight line), irrespective of fractions. When the number of fractions of missing values and the number of sets of columns are increasing (from 2 to 7), the estimation score (R^2)

is decreasing and the $RMSE$ is increasing, but when the number of missing values in the 7th set of columns increased to 40%, suddenly the estimates showed poor results. Whenever the "dist" column is included in the set of columns, the imputed estimates are not acceptable even for two sets of columns; it behaves as mentioned above, irrespective of the fractions or set of columns missingness.

5.2.2 Mice Imputation

For the first scenario inserting missing values independently, when graphs are observed keenly, it shows that especially when the dist' variable is added into the 2 sets of columns (included for the insertion of missing values set), this combination of sets heavily influences the results, in which it reduces the R^2 score steadily and increases the $RMSE$ value steadily as missing fractions (5% to 40%) increase, but when 1% of the missing values are present, it shows better results, however, after that sudden decreases in the estimates when it reaches 5%.

Overall, two sets of columns with missing values have closer results to ground truth estimates in all the fractions (1% to 40%) of missingness except when the dist variable is included in the set. Also, as the number of missing fractions increases and the estimates are moving further away from the ground truth estimates (the R^2 score is decreasing and the $RMSE$ is increasing), most of the different variable set column combinations (number of missing value in the columns set) provided better results when there were only 1% of missing values. However, when missing values are included in 6 or 7 columns, the estimation is also have showed some good estimations compared to other combinations of columns.

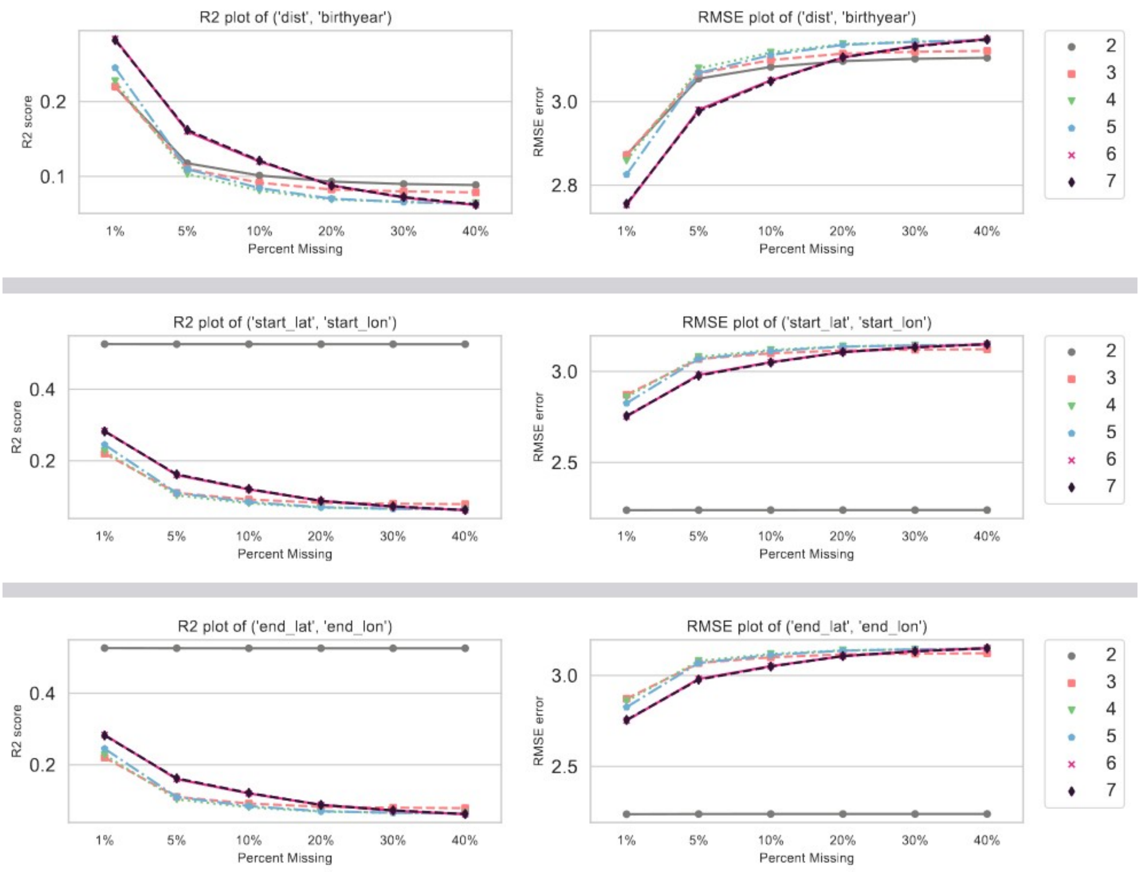


Figure 5.5: Mice Estimates

Multiple Regression Vs Mice

As it's hard to come to the conclusion which one is better performing, hence one more simulation is conducted where keeping 20% of missing values and changing the set of columns (missing value list of column combinations) from 2 to 7, this is the second scenario of artificially inserting missing values dependently, the scatter plots are thoroughly analyzed and it shows multiple regression imputation estimations are closer to the ground truth (full green dot) estimation for all the sets of columns (2 to 7 sets) than the MICE imputations. However, the estimation for MICE exhibited excellent results when the number of missing columns was less (2 to 4 sets). When the number of missing value columns increased from 5, 6, and 7, there was an abrupt increase in the $RMSE$ and a decrease in the R^2 imputed estimates.

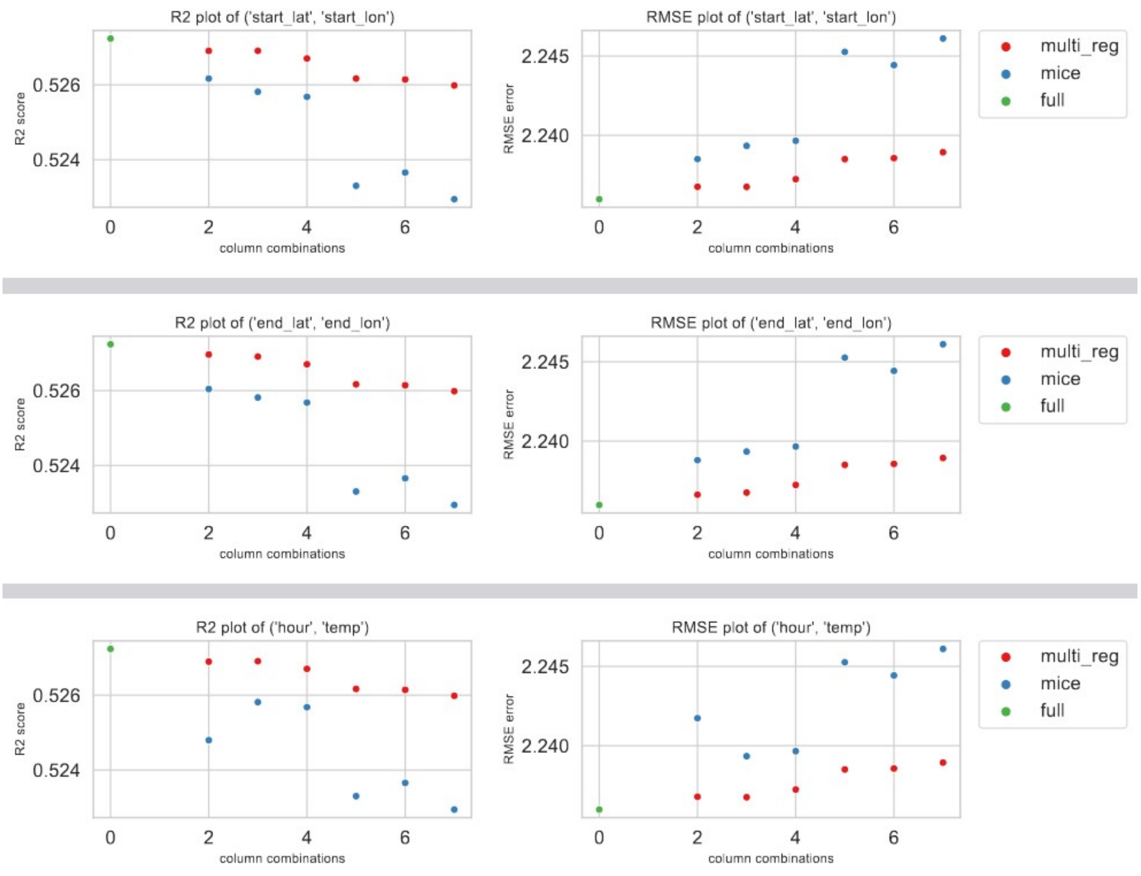


Figure 5.6: Multiple and Mice Estimates

5.3 Training and Testing on Complete Dataset

As discussed in section 4.4, it's necessary to derive $RMSE$ and R^2 to compare with imputed value estimates. After conducting linear regression using 21 independent variables on trip duration, the obtained estimates are $RMSE$ = 2.2359767726747313 and R^2 = 0.5272377063160185, which are considered ground truth estimates that are used to compare with other univariate and multivariate imputed estimates to quantify the missing value imputations. These estimates are depicted along with univariate imputation plots (see Figure 5.1) and also in the multiple regression imputation vs. mouse imputation comparison plots 5.6, which makes it simple to compare imputed estimates. The single point on a graph indicates the ground truth value of the model, which is derived by applying the linear regression model to a complete dataset with no missing values.

Also, for the entire dataset, the bootstrap intervals $RMSE$ estimate is [2.2081787852616945, 2.255565277182137]. and boot R^2 estimate is [0.5197428996061007, 0.5358435016216616].

5.4 Computational Complexity

What is computational complexity of the algorithm?, it is basically measure of the quantity of processing resources (in terms of both area and time) that a specific program uses when it runs. In any study, contemplating computational complexity is essential. For example, if two imputations provide viable answers, identifying which one is the most effective needs evaluation of processing complexity. The findings show that simple interpolation methods are often on level with multiple imputation methods. An important subject in this context is how the different methods rate in terms of their computational complexity. If techniques offer comparable predictive performance, it is better to use those options with the least processing effort.

Table A.4 demonstrates the time taken in seconds for univariate analyses and in minutes for multivariate analyses, along with missingness fractions and imputation methods. The clock began when imputation on the column was initiated and halted when imputation was concluded. For all univariate methods, the mentioned time is the average time taken to impute one column at a time; if there are 21 columns, then 21 * (the time taken for one column) = the total average time taken to impute all the 21 columns; additionally, the time taken to impute higher missingness fractions is a little higher when compared to lower missingness fractions. Also, observed that highest time taken compared other imputation is sample, least is bad imputation and mean, mode, drop almost same time for one column missing value imputation.

Similarly, for multivariate imputation time in terms of minutes, as mentioned in Table A.4, the average time taken for multiple regression imputation is less compared to MICE simulation, but it differs between 0.5 and 0.9 minutes for 2 sets of column imputation including varies fractions of missing values. The mentioned multiple regression time depicts time taken for 1 set of 2 variable combinations; if 7 sets of 2 distinct set of variables are missing, then it would be 7 * (0.9 min) = total time taken for all 7 sets of columns. Also, it was observed that if the number of columns increased, then the time taken for imputation also increased. The MICE simulation recorded a higher time for imputation; the recorded average time represents only one set of two variable imputation and for six iterations (six imputed datasets), consider the average time of 20 minutes, then for six iterations of two variable columns; as we have taken seven sets of two different variable combinations, then it would take 7 * 20 minutes = 160 min to run for all seven sets of two different variables. For 2 variable combinations, the duration varies between 12 and 20 minutes, but when the number of columns such as 3, 4, 5, 6, and 7 increased, the time required to execute these combinations also increased.

To recapitulate, the increasing complexity of the imputation methods is represented in their time duration. As univariate employs straightforward direct imputation, the time taken to complete the simulation for drop, mode, mean, sample, and bad for all the 21 variables is very less compared to multivariate imputations, regardless of fractions of missingness. For multiple regression, as it produces a single imputed dataset when compared to MICE imputation (6 datasets), even still, the time taken for the multiple regression imputation is very little when it's compared to the MICE imputation.

Chapter 6

Discussion

The thesis investigated the efficacy of univariate and multivariate imputation approaches on bike share data that included diverse data points under realistic missingness conditions (MCAR) with various percentages of missing values. In the following, work highlights some of the primary findings. The results can be summarized in two primary findings. First, during univariate simulation and, second during multivariate simulations.

6.1 Univariate Imputation Results

First, during univariate simulation, choosing drop imputation doesn't help much to achieve the desired result; it doesn't help to increase the downstream predictive performance substantially regardless of the missingness conditions (1%, 5%,.... 40%) or type of variable (numerical or categorical) variable (see Table 5.2). Second point is that, when the lower missingness percentages (1% and 5%) showed acceptable predictive estimates for most of the variables using mode, mean, sample and bad imputations. However, bad imputation didn't perform well for some of the variables regardless of percentages and sample imputation computational time is more compared all the other imputations. However, variables ($stat_lat$, $feelslike$, dew, $snowdepth$, $winddir$, $sealevelpressure$, $conditions$, day, and $holiday$) showed excellent estimates compared to ground truth estimates as they align exactly straight to the full estimation point 5.1, but cannot say is it because of how close positively or negatively co related to tripduration.

The bootstrap interval estimates added more value to the varied imputations technique. As shown in Figure 5.3, through this wanted to discover the biased imputation techniques (univariate), somehow failed to prove that any univariate imputation is biased, expected at least "bad imputation" might yield positive results in the biased section.

6.2 Multivariate Imputation Results

The results can be summarized for 1.) multiple regression and 2.) Mice imputation methods in this section.
First, during multiple regression simulations under various fractions 4.6, the results appear the same for different sets of columns, including (2 to 7), as observed estimations only differ in fractions of numbers in their estimations. As already discussed, the incorporation of the dist column in the missing value imputation set altered the estimates (is it because it's significantly positively correlated with trip duration)? As can be seen, as more columns have missing values, the R^2 score declines and the $RMSE$ value increases. Some of the column combinations didn't generate acceptable estimates compared to ground truth values, particularly when the distance variable was included in the column combinations. Secondly, to test how estimations behave

if a highly correlated variable is considered complete (not included in the missing column), another simulation is conducted only using 20% fractions of missingness 4.7; in this case, a dist feature was not included in the column combinations (which notably generate missing values) but included in the model construction with complete data. After the second experiment, results appeared to be improved compared to the first simulation.

Similarly, for mouse imputation, some of the column combinations didn't yield acceptable estimates compared to ground truth values, particularly when the distance property was included in the missing value column combinations (is it because it's highly positively correlated with trip duration), which significantly influenced the final estimations. Similar to multiple regression, for Mice also another simulation called inducing missingness dependently 4.7 is conducted to compare the results with previous MICE results; in this case, a dist feature was also not included in the missing value column combinations (which notably generate missing values) but included in the model construction with complete data. After the second experiment, results appear improved compared to the first simulation, but the primary distinction is that the second one was performed only using 20 percent of the missing fractions with dependently generated missing values.

To recapitulate, when only 20% of the missing fractions with dependently generated missing values were simulated, improved results were obtained and multiple regression performed better (see Figure 5.6) . and also, multiple regression is better performing than mice (even after considering the computation time). When fewer columns and fewer percentages had missing values, performance was improved, particularly for 2- and 3-column combinations and 1% and 5% missing fractions.

6.3 Limitations

One of the primary objectives of this experiment is to quantify missing values using imputation methods on a particular dataset (Bike Share) and, under realistic missingness conditions with various fractions of missing values, make some decisions that limit results. Due to time limitations, the replication of other missingness patterns (MAR and MNAR) is not conducted; only MCAR can be simulation able to manage in this work.

I had to perform the univariate and multivariate imputations many times before selecting the final versions for both. 1.) First, I considered that tripdurations are in seconds; later, I realized that converting to minutes makes analyzing model coefficients straightforward and simple. 2.) Initially, start_station_id, end_station_id, and bike are considered numerical data (with a significant P-value), but later, after discussion with the supervisor, you come to the conclusion that these three variables should be considered categorical variables. 3.) When no anomalies are addressed, the subpar estimates shown, even though it's not essential for this thesis, are still made more realistic by contemplating the 90^{th} percentile data and clipping off one of the age factors also handled. In the final version all (start_station_id, end_station_id, and bikeid) are considered as categorical features and included in the model, even after reducing the number of categorical levels, producing a huge number of columns after OneHotEncoding. Because these assumptions directly affect imputations simulation (specifically multivariate imputations) and therefore increase computational complexity, finally it was determined not to include them in the simulation.

First, focused MCAR missingness patterns as it's easy to make the assumption to artificially insert the missing values, when data is missing completely at random than, other two types of missing patterns. Also, can be considered a more relevant scenario in real-world applications of imputation methods. Thus, MCAR applied for both single and multiple imputations techniques.

Second, the used time series bike share dataset consist of a maximum of 22 features (after various data integration) and 315k observations. For this reason, cannot conclude from the experiments how the imputation methods perform on more larger scale datasets or including external data (weather and holidays data). Furthermore, bike dataset only contain numerical or categorical columns, and time series data. Whether applying simple imputations techniques on time series does provide the good estimates or not hard to answer.

Third, to measure the multivariate imputation impact on the downstream (regression) performance, inserted and imputed values in numerical columns only. Therefore, the impact depends largely on the selected column's importance (e.g., see the work of [Schafer and Graham, 2002]), so for categorical values, estimates may vary when other modern imputation techniques are implemented. Generally, the impact of using an imputation model could vary when multiple columns were affected by missing values, so this work couldn't manage multivariate imputation for categorical data, due to it's OnehotEncoding limitation of converting null values into 0, see detailed discussed in the section 4.3.

Chapter 7

Conclusion and outlook

This chapter discussed the conclusions of the experiment and an outlook for potential enhancements. The value it provides, approaches for improvement, and future work are given.

7.1 Conclusion

In this thesis, performed simulations of imputation techniques comparing univariate and multivariate approaches using bike sharing data under realistic missingness situations. As already discussed the concept of data imputation, data missingness mechanisms, handling heterogeneous columns missing values, single and multiple imputation methods, and then the analysis of the performance of different imputation methods, namely drop imputation, mode imputation, mean imputation, sample imputation, bad imputation, bootstrap intervals for univariate imputations, multiple regression, and MICE imputation.

According to the general results of the $RMSE$ and R^2 estimations offered in the evaluation, the basic, uncomplicated single imputations yield better outcomes when compared to multiple imputations (particularly MICE) in the bike sharing dataset. Also, univariate imputation (mean, mode) outperformed some of the other columns when the computational cost of imputation was taken into consideration. Even when other multivariate imputations such as multiple regression and MICE estimate demonstrated that multiple regression is out formed 5.6 compared to ground truth estimations. However, the variable dist exhibited some influence on the estimates in both univariate and multivariate imputations. I guess one reason would be that this behavior is substantially positively connected with tripduration (0.69 correlation number A.5) than any other variable. When boot interval estimates were accounted for, dist was a relatively tight interval for imputed estimation results, but all the other variables were within broader the range of bootstrap interval estimates, as explained in the section 5.

Moreover, the missing data in bike share, when imputed using fundamental methodologies, exhibited improved results in the MCAR missingness pattern. However, as the inserted missing values are artificial, these might bias the results. Therefore, in general, the distribution of the variables in imputed datasets varies, meaning that when any imputation methods bias the dataset, and accordingly, learning models could also be biased, the different missingness pattern also has an effect on the imputed dataset.

Furthermore, selecting an appropriate method to manage missing values depends on the dataset, the mechanism of missing values, and the missingness rate. This thesis showed evidence about the impact of missing values in common subsequent analyses for the bike sharing dataset.

Finally, comprising with any research, this research has limits, thus we cannot say that the Multivariate imputations are the best approach to manage missing information in all scenarios. Also for Univariate imputations cannot say outperforms in any given dataset. However, the results given in this experiment demonstrates that imputation might possibly be superior for preventing bias in later studies than just eliminating data from datasets with missing values as observed in the drop imputation.

7.2 Value

This work illustrates how missing value imputations (univariate and multivariate) effect the downstream model estimates(regression) compared to ground truth estimates. One who is working on bike sharing systems can make use of these simulation evaluation results to improve their real-time systems to avoid the missingness problem and to yield a complete dataset for improved plans and business decisions. These results can be considered as prior leanings for their experiments especially in the bike share domain.

7.3 Improvements and future works

Used basic univariate and multivariate imputation methods to conduct the experiment, and to achieve the better results, however, the modern approaches such as neural networks (deep learning) techniques and other approaches (kNN imputation, hot deck imputation, etc.) can also be considered for future analysis with all the missingness patterns such as MCAR, MAR, and MNAR with different missing fractions. Similarly for multivariate imputation, the categorical values can also be included in the model as an enhancement, and other strategies or algorithms (such as miss-Forest) can also be tested.

Due to time constraint the experiment only simulated on MCAR missingness pattern, to comprehend how other realistic missingness patterns such as MAR and MNAR provide estimations for various imputation methods with varying missing value fractions, this can be considered as an improvement or future work. As the experiment using linear regression model, hyper-parameter optimization is not inspected, if experimented is conducted with other modern models (Neural networks, KNN, Randomforest etc) then optimizing their hyperparameters also accounted to observe the behaviour of the estimates.

Thesis didn't included the ranking on each imputation method on the visualization plots, this also can be added as an improvement or further work. If Bike sharing system is experimenting with other business questions such as, demand of the bikes in particular locations, resource management, predicting the trip start and end station many more, for these extended business questions task (binary classification, multiclass classification, and regression) could also be considered for experiment.

The thesis didn't include the ranking of each imputation method on the visualization graphs; this also can be added as an enhancement or further work. If the bike sharing system is experimenting with other business questions such as demand for the bikes in particular locations, resource management, predicting the trip start and end station, and many more, the other downstream ML tasks of binary classification, multiclass classification, and regression (already experimented work) could also be considered for experimentation.

Bibliography

[Allan and Wishart., 1994] Allan, F. E. and Wishart., J. (1994). Comments on 'missing data, imputation, and the bootstrap. *Journal of Agricultural Science 20*, 89.

[Anil et al., 2019] Anil, J., Dhanya, P., and Krishnan, R. (2019). Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, 33(10):913–933.

[Baylor et al., 2017] Baylor, D., Breck, E., and Cheng, H.-T.and Fiedel, N. (2017). Data management in machine learning. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[Biessmann et al., 2021] Biessmann, F., Golebiowski, J., Rukat, T.and Lange, D., and Schmidt, P. (2021). Missing data imputation for supervised learning. *IEEE Computer Society Technical Committee on Data Engineering.*

[Biessmann et al., 2019] Biessmann, F., Rukat, T., Schmidt, P., Naidu, P., Schelter, S., Taptunov, A., Lange, D., and Salinas, D. (2019). Datawig: Missing value imputation for tables. *J. Mach. Learn. Res.*, 20(175):1–6.

[Fritz, 2005] Fritz, J. S. (2005). *LaTeX: The American Statistician*, volume 59. Taylor and Francis, Ltd., 4 edition.

[Hughes et al., 2019] Hughes, R. A., Heron, J., Sterne, J. A. C., and Tilling, K. (2019). Accounting for missing data in statistical analyses: multiple imputation is not always the answer. *International Journal of Epidemiology*, 48(4):dyz032. https://academic.oup.com/ije/article-pdf/48/4/1294/29162803/dyz032.pdf.

[Hyun et al., 2022] Hyun, A., Kyunghee, S., and Kwanghoon, P. K. (2022). Comparison of missing data imputation methods in time series forecasting. *Computers, Materials & Continua.*

[Jason and Rafael, 2018] Jason, P. and Rafael, V. (2018). Missing data imputation for supervised learning. *Applied Artificial Intelligence*, 32(2):186–196.

[Jäger et al., 2021] Jäger, S., Allhorn, A., and Bießmann, F. (2021). A benchmark for data imputation methods. *Frontiers in Big Data*, 4.

[Korean, 2013] Korean, J. A. (2013). The prevention and handling of the missing data. *Frontiers in Big Data*, 64(5).

[Kumar and Boehm, 2017] Kumar, A. and Boehm, M.and Yang, J. (2017). Data management in machine learning. *Chicago Illinois USA (Association for Computing Machinery)*, 2nd Edition.

[Little and Rubin, 2002] Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data.*, volume 2nd Edition. A JOHN WILEY and SONS, INC., PUBLICATION, 2 edition.

[Milan et al., 2022] Milan, M. T., Ashish, V., and Sai, K. M. (2022). Imputation of trip data for a docked bike-sharing system. *Current Science.*

[P, 2008] P, D. T. (2008). P - value, a true test of statistical significance? *Annals of Ibadan postgraduate medicine.*

[Pedregosa et al., 2011] Pedregosa, G. F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Rubi, 004b] Rubi, D. B. (2004b). The design of a general and flexible system for handling non-response in sample surveys. *The American Statistician*, 58(4).

[Rubin, 1987] Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys.* Printed in the United States of America.

[Rubin, 1996] Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American Statistical Association.*, 91.

[Salma, 2020] Salma, Y. Y. H. (2020). Analysis and prediction of bike sharing traffic flow.

[Schafer and Graham, 2002] Schafer, J. L. and Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychol Methods*, 7.

[scikit library, 2022a] scikit library (2022a). sklearn.impute.iterativeimputer.

[scikit library, 2022b] scikit library (2022b). sklearn.linear.model.linearregression.

[Stoyanovich et al., 2020] Stoyanovich, J., Howe, B., and Jagadish, H. V. (2020). Responsible data management. *Proceedings of the VLDB Endowment*, 13.

[T., 2022] T., B. (2022). In-depth tutorial to advanced missing data imputation methods with sklearn.

[van Buuren and Groothuis-Oudshoorn, 2011] van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67.

[Vijay, 2022] Vijay, K. (2022). What is linear regression? types, equation, examples, and best practices for 2022.

[William et al., 2022] William, J. M., Emily, R., Teri, M., Alyson, M., and Krista, R. (2022). *Applied Statistics in Healthcare Research.*, volume 4.0. Powered by Pressbooks.

[Woźnica and Biecek, 2020] Woźnica, K. and Biecek, P. (2020). Does imputation matter? benchmark for predictive models.

[Xiao et al., 2022] Xiao, X., Yunlong, Z., Shu, Y., and X, K. (2022). Efficient missing counts imputation of a bike-sharing system by generative adversarial network. *IEEE Transactions on Intelligent Transportation Systems*, 23:13443–13451.

[Zhang et al., 2018] Zhang, H., Xie, P., and Xing, E. (2018). Missing value imputation based on deep generative models.

Appendix A

Figures

A.1 Introduction

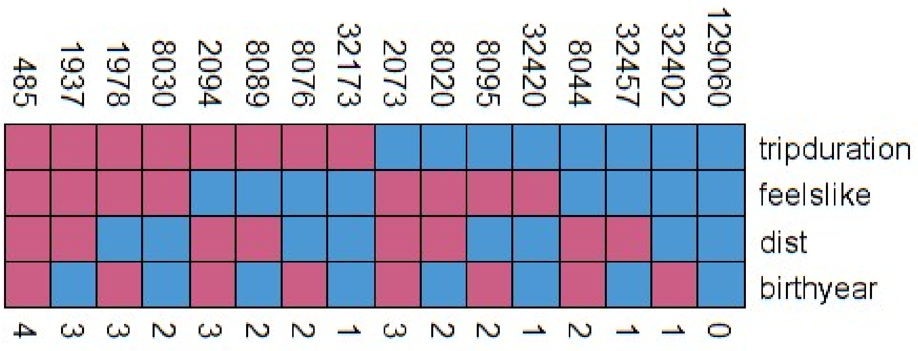
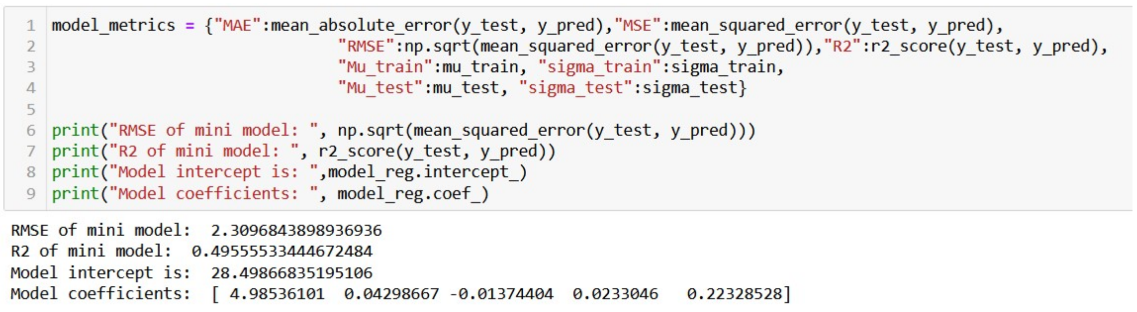


Figure A.1: Missing Value Visualization Example

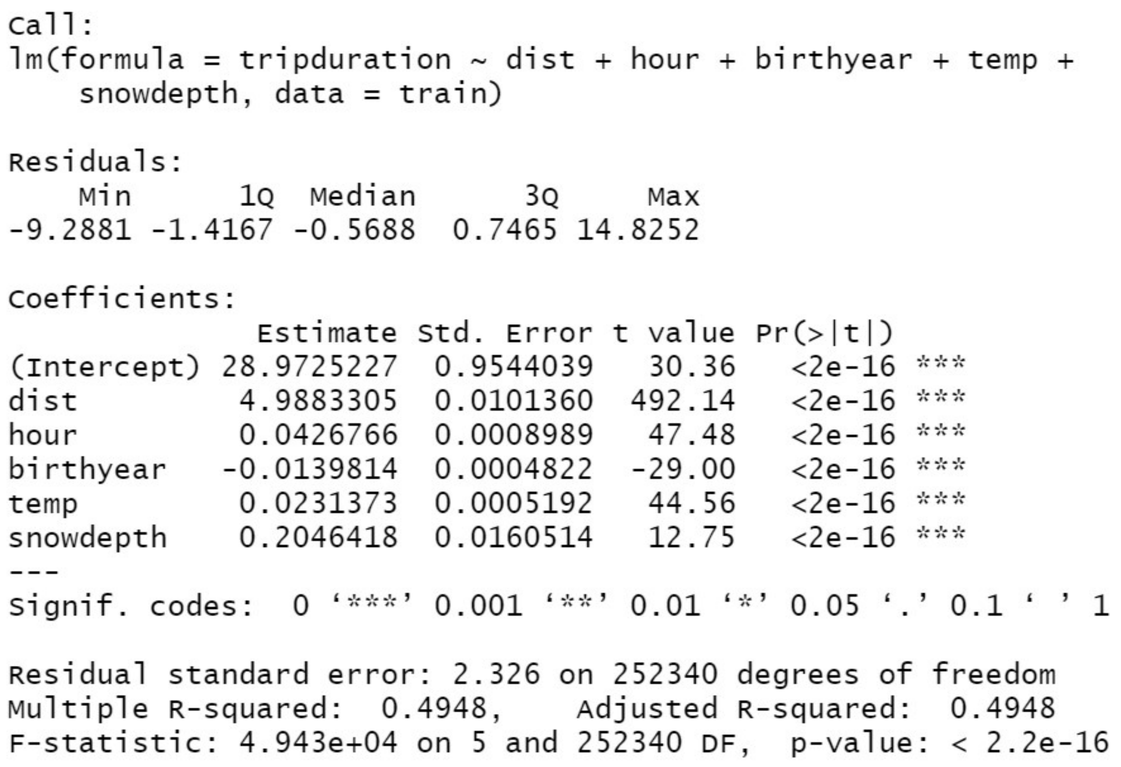
A.2 Implementation and Experiments



```
model_metrics = {"MAE":mean_absolute_error(y_test, y_pred),"MSE":mean_squared_error(y_test, y_pred),
                              "RMSE":np.sqrt(mean_squared_error(y_test, y_pred)),"R2":r2_score(y_test, y_pred),
                              "Mu_train":mu_train, "sigma_train":sigma_train,
                              "Mu_test":mu_test, "sigma_test":sigma_test}

print("RMSE of mini model: ", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 of mini model: ", r2_score(y_test, y_pred))
print("Model intercept is: ",model_reg.intercept_)
print("Model coefficients: ", model_reg.coef_)
```

```
RMSE of mini model:  2.3096843898936936
R2 of mini model:  0.49555533444672484
Model intercept is:  28.49866835195106
Model coefficients:  [ 4.98536101  0.04298667 -0.01374404  0.0233046   0.22328528]
```

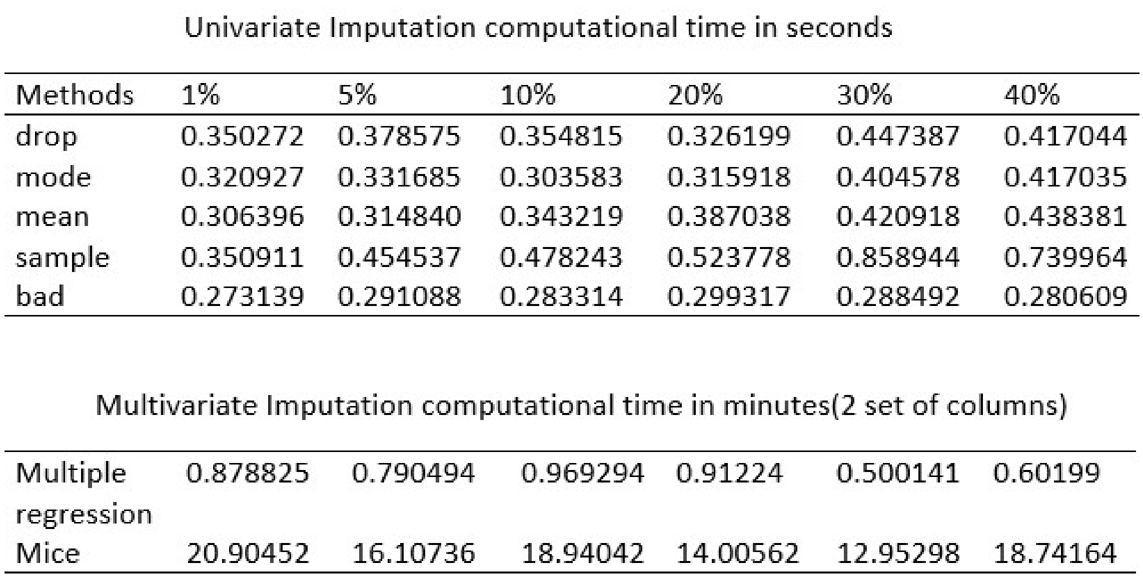
Figure A.2: Mini Model using Python



```
Call:
lm(formula = tripduration ~ dist + hour + birthyear + temp + 
    snowdepth, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.2881 -1.4167 -0.5688  0.7465 14.8252 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 28.9725227  0.9544039   30.36   <2e-16 ***
dist         4.9883305  0.0101360  492.14   <2e-16 ***
hour         0.0426766  0.0008989   47.48   <2e-16 ***
birthyear   -0.0139814  0.0004822  -29.00   <2e-16 ***
temp         0.0231373  0.0005192   44.56   <2e-16 ***
snowdepth    0.2046418  0.0160514   12.75   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.326 on 252340 degrees of freedom
Multiple R-squared:  0.4948,	Adjusted R-squared:  0.4948 
F-statistic: 4.943e+04 on 5 and 252340 DF,  p-value: < 2.2e-16
```

Figure A.3: Mini Model using R

A.3 Results



Univariate Imputation computational time in seconds

Methods	1%	5%	10%	20%	30%	40%
drop	0.350272	0.378575	0.354815	0.326199	0.447387	0.417044
mode	0.320927	0.331685	0.303583	0.315918	0.404578	0.417035
mean	0.306396	0.314840	0.343219	0.387038	0.420918	0.438381
sample	0.350911	0.454537	0.478243	0.523778	0.858944	0.739964
bad	0.273139	0.291088	0.283314	0.299317	0.288492	0.280609

Multivariate Imputation computational time in minutes(2 set of columns)

Multiple regression	0.878825	0.790494	0.969294	0.91224	0.500141	0.60199
Mice	20.90452	16.10736	18.94042	14.00562	12.95298	18.74164

Figure A.4: Imputations Computational Time

A.4 Conclusion and outlook

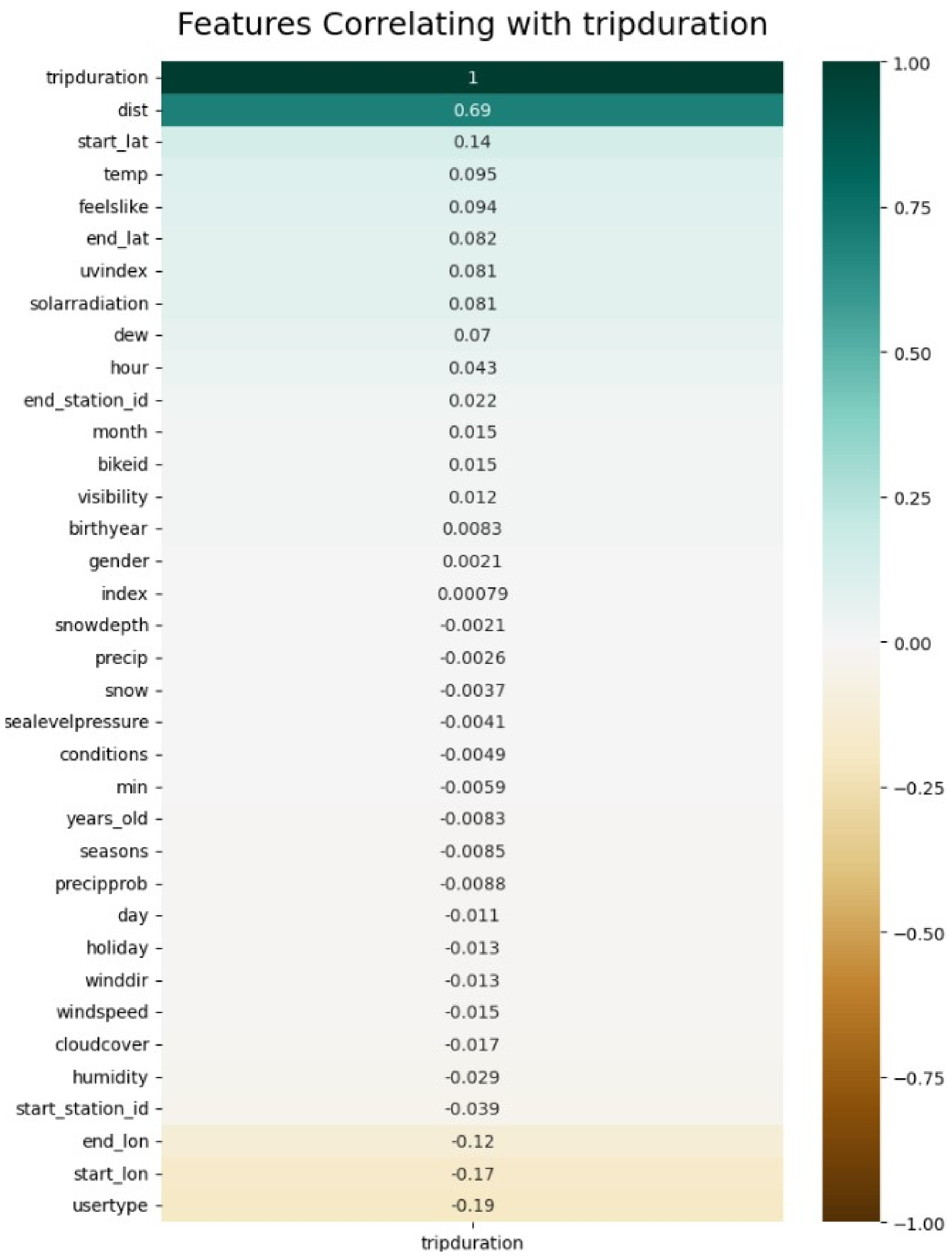


Figure A.5: Correlations Information with Tripduration