

Imputing missing data

An data science oriented project made with the focus on acquiring missing data from Maternity Foundation



Figure 1: Maternity foundation logo - Maternity.dk

1. Semester project

Ms.c. Business Data Science

Aalborg University 2022

Characters:

XX/XX

Supervisor: Daniel S. Hain

dsh@business.aau.dk

Alamgir Hossain

ahossa20@student.aau.dk

Christian Nielsen

cnie19@student.aau.dk

Kasper Raupach Haurum

khauru18@student.aau.dk

Md. Raiyan Alam

malam22@student.aau.dk

Sadiksha Baral

sbaral22@student.aau.dk

Abstract:

The aim of this project seeks to accomplish the feat of utilizing data imputation for the company, Maternity Foundation, in their problems regarding missing data. This is an objective we seek to complete through the means of the tools we have learned in this 1st semester in Data Science. The missing data itself stems from the application that Maternity Foundation has developed "Safe Delivery App", which is a platform that offers its users knowledge and training in giving assistance to women with various maternal health ailments. This missing data originates from the information that the users gave to the developers, such as their geographic location (District, State, etc), professional background, and various incomplete data entries. While this may seem insignificant at first glance, it can carry consequences that may affect business operations when considering making decisions made on behalf of that data provided.

The data in this project originates from the Safe Delivery App users in the territories of India and Ethiopia, whereas the importance of reducing the data by removing NAs, missing data, and cleaning the data, is important as the phone devices in these countries may not be able to contain larger data, or otherwise affect performance on the phones when having to process the data on their devices. Therefore not only by conducting imputation of data there will be smaller degrees of uncertainty, it will also reduce the size of the data for the users.

In this project we will first be doing a brief overview by doing an EDA (Exploratory data analysis), which will show key figures and reveal findings that may be relevant to stakeholders. Afterwards we will then move onto conducting a PCA (Principal Component Analysis) on the assigned dimensions that are most relevant by significance. In the case of this project, we are interested in investigating the users' interactions with the modules within the Safe Delivery App, how long the users are interacting with the aforementioned modules, and the results that they are able to achieve.

Next up we will move onto the section dealing with the imputation itself, based on the dimensions we have previously assigned, and showcasing the coding that will be used to achieve this.

We will then in the last parts of the project reflect upon our efforts making this project, how we reached the results we ended up with, and lastly potential future research that can be made based on our findings. Likewise we will also give a brief reflection on the teamwork collaboration using the Agile project management theory, and our experience in delegating the project into the phases of envision, speculate, explore, adapt and the closing phase. In the final we will round it all up with a conclusion upon our project, and go over what improvements could have been made.

Introduction	6
Summary of the business problem being solved	6
The technical aspect of the project	8
Overview of data imputation being researched elsewhere	9
Why is this a relevant problem to tackle	10
Structure of the project paper	11
Background & Theory	12
Agile project management theory	12
Data science techniques	14
Examples of business cases where data imputation matters	16
Ethical concerns	17
Case description	20
Why can this project benefit Maternity Foundation	21
Data / Method	23
Exploratory Data Analysis	24
Data preparation and preprocessing	29
Feature importance method	32
Test setup and evaluation of results	33
Technical implementation	35
Which models are used, and what are their inputs and outputs?	35
Random Forest	35
KNN	37
MICE imputation	38
Project implementation:	38
Analysis of results	39
Analysis of features importance	39
Analysis of results from model testing	43
Discussion	50
Conclusion	52
Appendix:	52
References:	53
Illustrations	55

Introduction

Summary of the business problem being solved

Missing data is a common issue in data science and can greatly impact the accuracy and reliability of a study or analysis. To accurately analyze and interpret data, it is essential to have complete and accurate information (Barnard J, 1999). Imputing missing data, or filling in missing values with substitute values, is a common method for addressing this issue. This project focuses on dealing with the missing data in the application “Safe Delivery App”, developed by the organization Maternity Foundation.

The Maternity Foundation is a non-profit organization from Denmark that provides support and resources for midwives and nurses in primarily regions of the world with inadequate facilities to train them (**Maternity foundation source, website**). The company collects a wealth of data on its users, including demographic information, medical experience, and access to resources. However, due to various factors such as incomplete surveys or data entry errors, some of the data is missing. This may be due to their users being from countries such as India, and Ethiopia where acquiring clean data may be more problematic, and/or due to the way the application is designed.

The goal of this project is to develop a method for imputing missing data in the Maternity Foundation's database. This will allow the organization to make more informed decisions and better serve its users, and to reduce the data that is stored on the users devices to offer better performance to individuals that may not be equipped with the latest hardware that may affect the performance of the app.

There are several approaches to imputing missing data, including the use of multiple imputation methods and single imputation methods. Multiple imputation methods involve generating multiple imputed datasets, while single imputation methods involve replacing missing values

with a single estimated value. (Rubin, 1987). To develop a method for imputing missing data, we will first conduct a thorough analysis of the Maternity Foundation's database to identify patterns and relationships in the data by doing EDA. We will then use machine learning techniques to develop a predictive model that can generate substitute values for missing data. We expect that our method will be able to impute the missing data in the Maternity Foundation's database, thereby improving the organization's ability to make informed decisions and better serve its clients.

The results of this project have the potential to be significant for the Maternity Foundation and other organizations that deal with missing data. By developing a reliable method for imputing missing data, we can improve the accuracy and reliability of data-driven decision making, leading to better outcomes for individuals and communities. The results of our coding will also provide valuable insight into the performance of different multiple imputation methods that can be likewise utilized.

We believe by accomplishing this task, the Maternity foundation will greatly benefit as a result (Barnard J, 1999). As such, we see that performing data imputation can achieve a number of goals, including:

- Improved data quality: By imputing missing or incomplete data, Maternity Foundation can improve the overall quality of their data. This can enable more accurate and reliable analysis, leading to better insights and decision-making.
- Increased data availability: By imputing missing data, Maternity Foundation can make use of data that would otherwise be unusable. This can enable the analysis of a larger and more complete dataset, leading to more accurate and comprehensive insights.
- Enhanced modeling and analysis: By imputing missing data, Maternity Foundation can improve the performance of their predictive models and other statistical analysis. This can enable more accurate predictions and insights, leading to better decision-making and performance.

- Reduced bias: By imputing missing data, Maternity Foundation can reduce the bias that can result from missing or incomplete data. This can improve the fairness and accuracy of their analysis, leading to better outcomes and results.

The technical aspect of the project

In this project the technical problem presented to us by Maternity Foundation is the problem of having a dataset that contains multiple missing values, or incomplete data entries. This is a, while a common problem, something that can hinder making accurate data science operations, such as machine learning as the train/validation sets, because the data provided may be skewing the resulting predictions because of the inaccuracy inherited inside it. It will be difficult to give an exact label to the nature of this problem as it overall affects not only machine learning, but all operations made on the data, so it would be prudent to say this problem is a fundamental challenge that data scientists undertake when performing their duties. We feel that by taking on such a problem it can not only be beneficial as us as it will be a likely challenge we have to deal with in our future careers, but also the Maternity Foundation as this project may inform them with the knowledge in how to prevent incomplete data entries by being informed how to tackle it.

This is important because missing data can cause problems with data analysis and decision-making, such as making it difficult to draw meaningful conclusions from the data or introducing biases into the results. Data imputation involves using statistical techniques to estimate the missing values in a dataset, typically by using the values of other data points in the dataset to make predictions about the missing values. The goal of data imputation is to fill in the missing values in a way that is as accurate and reliable as possible, so that the resulting dataset can be used for analysis and decision-making without the problems that can arise from missing data.

Overview of data imputation being researched elsewhere

The current state of the art in data imputation is focused on developing and evaluating new methods for imputing missing data. Many researchers are exploring the use of machine learning algorithms, such as neural networks and decision trees, for imputing missing data (Little & Rubin, 2002). These methods have been shown to produce accurate imputed values, and are particularly useful for dealing with large datasets with complex patterns of missing data. When doing data imputation it can also involve the use of multiple imputation methods, which have been shown to produce more accurate results than single imputation methods (Rubin, 1987). Multiple imputation methods involve generating multiple imputed datasets, which are used to replace missing values in the original dataset.

One of the most commonly used multiple imputation methods is the Markov Chain Monte Carlo (MCMC) method. The MCMC method generates imputed datasets by simulating values for the missing variables based on the observed data. This method has been widely used in a variety of fields, including health research, economics, and social sciences (Little and Rubin, 2002).

Another commonly used multiple imputation method is the Fully Conditional Specification (FCS) method. The FCS method involves fitting separate regression models for each variable with missing data, and then using these models to generate imputed values (Van Buuren, 2012). The FCS method has been shown to produce accurate results in a variety of settings, including health research and survey data analysis (Van Buuren, 2012)

Within the world of research in data imputation is the development of methods for assessing the quality of imputed values. These methods are important for ensuring that imputed values are accurate and reliable, and can help researchers to avoid making decisions based on biased or inaccurate data (Buuren & Groothuis-Oudshoorn, 2011). For example, the mean squared error (MSE) and the fraction of missing information (FMI) are commonly used metrics for evaluating the quality of imputed values.

In addition to the development of new imputation methods and evaluation metrics, there is also a focus on understanding the factors that impact the performance of imputation methods. For example, researchers have studied the impact of the amount of missing data on the accuracy of imputed values (Little & Rubin, 2002), and the impact of the type of data being imputed on the performance of different imputation methods. This research is important for helping researchers to choose the most appropriate imputation method for their specific dataset and research question.

Overall, the current state of the art in data imputation is focused on developing and evaluating new methods for imputing missing data, and understanding the factors that impact the performance of these methods.

Why is this a relevant problem to tackle

Imputing missing data is a relevant and important topic for companies and organizations because it can help to improve the accuracy and reliability of their data analysis incomplete or missing data can lead to inaccurate or biased results, which can have negative impacts on decision making and overall business performance By imputing missing data, a company or organization can ensure that their data is complete and accurate, which can in turn improve their ability to make informed and effective decisions.

Additionally, imputing missing data can help to increase the efficiency and effectiveness of data analysis. When data is incomplete, it can require more time and effort to analyze and interpret by imputing missing data, a company or organization can reduce the time and effort required for data analysis, which can in turn improve their ability to generate insights and make decisions (Van Buuren, 2012).

Overall, imputing missing data is a relevant and important topic for companies and organizations because it can improve the accuracy and reliability of their data analysis, and increase the efficiency and effectiveness of their decision making by implementing effective imputation methods, a company or organization can ensure that their data is complete and accurate, which can help them to make more informed and effective decisions.

Structure of the project paper

This project seeks to go over the contents of our findings which will be presented in a chronological order, starting by the introduction all the way to the end with our conclusion based on the findings made in this project. The order that it is presented is to allow the reader to see the way we end up with the results we have ended with, while also seeing the preceding theory behind the models and data science tools when operating on the data. This can also allow someone from a non-technical background, such as an individual who has not studied data science, to also be able to interpret the knowledge contained in this project without being armed with a dictionary or internet search engine. However, to give an abridged explanation, a summary of the project will be provided here.

The introduction will provide a motivation why we felt it was prudent for us to select the subject we did, and what the overall theme is in relation to data science. This will also give a short explanation of what kind of organization that Maternity Foundation is, and what the application “Safe Delivery App” provides in the form of the data that we will be working upon. Lastly a brief resume of what data imputation is will likewise be explained.

Background & Theory

Agile project management theory

Agile project management is a flexible, iterative approach to project management that was developed in the late 1990s as an alternative to traditional, waterfall-style project management methods (Agile Manifesto, n.d.). The Agile method was created by a group of software developers who were frustrated with the inflexibility and inefficiency of traditional project management approaches (Schwaber & Sutherland, 2017).

In February 2001, a group of 17 software developers met in Snowbird, Utah to discuss their experiences with agile software development. They published the Agile Manifesto, a set of principles for agile software development that emphasizes collaboration, flexibility, and adaptability (Agile Manifesto, n.d.). The Agile Manifesto has since been adopted by organizations in a variety of industries, including data science, as a guide for agile project management.

Since its inception in the late 1990s, agile project management has been widely adopted in a variety of industries, including:

- **Software development:** Agile project management was originally developed for software development projects, and it remains a popular choice in this field. The Agile method allows software development teams to quickly respond to changes in technology and customer needs, and to deliver software in small, incremental increments.
- **Data science:** Agile project management can be particularly useful for data science projects, where the complexity and uncertainty of the work can make traditional project management approaches difficult to use. By breaking the project down into smaller, iterative pieces and regularly checking in with stakeholders, the Agile method can help data science teams effectively manage missing data imputation and other challenges.

- Marketing: Agile project management can be useful in marketing projects, where the goals and objectives may change frequently. The Agile method allows marketing teams to respond quickly to changing customer needs and market conditions, and to deliver marketing campaigns in small, iterative increments.

Agile project management is a flexible, iterative approach to project management that is particularly well-suited to data science projects (Agile Manifesto, n.d.). The Agile method emphasizes collaboration, frequent iteration, and the ability to adapt to changing requirements and priorities (Kanbanize, n.d.).

One key aspect of Agile project management is the use of sprints, which are short, focused periods of work that typically last one to four weeks (Project Management Institute, 2017). During each sprint, the team defines a set of specific goals and works to achieve them. This allows for regular progress check-ins and the ability to course-correct as needed.

In the context of data science projects, Agile can be particularly useful for managing missing data imputation. Missing data is a common challenge in data science, as it can arise for a variety of reasons, including data entry errors, incomplete data collection, and data loss. Imputing missing data involves using statistical techniques to estimate the missing values, and this can be a time-consuming and complex process.

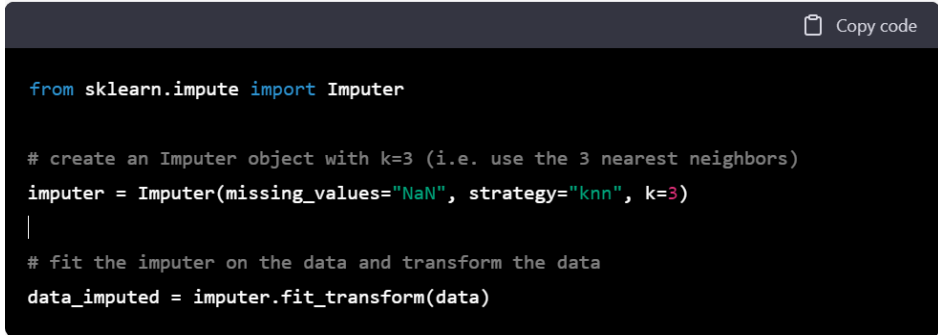
Using the Agile method, the data science team can break the imputation process down into smaller, more manageable tasks and work on them in iterative sprints (Project Management Institute, 2017). This allows for more flexible and efficient project management, as the team can prioritize the most important tasks and respond to any challenges or setbacks as they arise.

Agile project management is characterized by a number of key principles, including:

- Iterative development: Agile projects are broken down into small, incremental chunks of work called "iterations" or "sprints." Each iteration involves the team completing a set of specific goals and delivering a usable product.
- Collaboration: Agile projects are built on collaboration between the project team and stakeholders. This allows for regular feedback and the ability to respond to changing requirements and priorities.
- Flexibility: Agile projects are designed to be flexible and adaptable, allowing for course corrections and changes as needed.
- Visibility: Agile projects prioritize transparency and visibility, with regular progress check-ins and the use of tools such as burn-down charts to track progress

Data science techniques

One of the most popular methods of imputation is called k-nearest neighbor imputation, which involves using the values of the k-nearest neighbors of a missing value to fill in the missing value. This method can be useful when the dataset has a spatial or temporal component, as it can preserve the relationships between nearby values. The Imputer class in scikit-learn can be used to perform k-nearest neighbor imputation, as shown in the following example:



```
from sklearn.impute import Imputer

# create an Imputer object with k=3 (i.e. use the 3 nearest neighbors)
imputer = Imputer(missing_values="NaN", strategy="knn", k=3)

# fit the imputer on the data and transform the data
data_imputed = imputer.fit_transform(data)
```

(Figure 2: Example of code for using KNNImputer - Own creation)

There are many other methods of imputation that can be used in Python, including more advanced techniques such as KNN imputer. KNN Imputer is a method of data imputation that uses the values of the k-nearest neighbors of a missing value to fill in the missing value. This method can be useful when the dataset has a spatial or temporal component, as it can preserve the relationships between nearby values. KNN imputer can be implemented in Python using the Imputer class from the scikit-learn library, which allows the user to specify the number of neighbors (k) to use and other options such as the distance metric and imputation strategy.

KNNImputer method to resolve missing data by doing data imputation, for example, would look the following when attempting to resolve the aforementioned issue:

```
import numpy as np
from sklearn.impute import KNNImputer

# create a sample dataset with missing values
X = np.array([[1, 2, np.nan], [3, 4, 3], [np.nan, 6, 5], [8, 8, 7]])

# create the KNNImputer object
imputer = KNNImputer(n_neighbors=2, weights="uniform")

# fit the imputer object on the dataset
imputer.fit(X)

# transform the dataset to fill in the missing values
X_transformed = imputer.transform(X)

# view the transformed dataset
print(X_transformed)
```

(Figure 3: Another example for using KNNImputer - Own creation)

In this example, the KNNImputer method is used to fill in the missing values in the sample dataset X. The n_neighbors parameter specifies the number of nearest neighbors to use when imputing the missing values, and the weights parameter specifies the weighting scheme to use when calculating the imputed values. The fit method is then used to fit the imputer object to the dataset, and the transform method is used to fill in the missing values and create a transformed dataset, X_transformed. The resulting dataset contains no missing values, and the imputed values are calculated based on the values of the nearest neighbors.

IterativeImputer can be a strong model for data imputation, since it enables the data scientist to use the Multivariate Imputation by Chained Equations (MICE) method for estimating the true values (Hannah S Laqueur, Aaron B Shev, Rose M C Kagawa(2022)). The idea of this method is first use the best simple estimator for the missing values, and then using an external model, predict the missing values in an iterative fashion. This means that after the first initial prediction, this predicted value becomes the new estimate for a new prediction. This pattern continues until either a threshold tolerance for the difference level, or the maximum set number of allowed iterations have been performed. The purpose is to in a round robin fashion, incrementally get closer to the best possible estimator for the individual missing value (Pedregosa et al.(2022)). In our case, we will be fitting the IterativeImputer with a KNNRegressor model, as well as a MinMaxScaler to scale the data down between 0 and 1. This should increase the ability to generalize from the data patterns. This is due to KNN being a distance based algorithm, meaning that large scale numbers can affect the performance (Pulkit Sharma (2019)).

Examples of business cases where data imputation matters

Data imputation is being used in several areas, such as in business in the field of customer relationship management (CRM). CRM systems often rely on data about customer interactions, preferences, and behaviors, and this data is often incomplete or missing (Srivastava, Dhar, & Bucklin, 2002). Data imputation can be used to fill in these gaps, improving the accuracy of customer segmentation and targeting, and enabling businesses to make more informed decisions about how to engage with their customers.

Another example of data imputation being used in business is in the development of predictive analytics models. Predictive analytics models rely on large amounts of data to make accurate predictions about future events, and missing data can significantly impact the accuracy of these models (Liaw & Wiener, 2002). By using data imputation techniques, businesses can improve the accuracy of their predictive models, enabling them to make more informed decisions about future trends and opportunities.

Data imputation has also been used in the development of machine learning models, which are commonly used in a variety of business contexts, including marketing, finance, and operations.

Machine learning models rely on large amounts of data to learn patterns and make predictions, and missing data can significantly impact the accuracy of these models (Chawla et al., 2002). By using data imputation techniques, businesses can improve the accuracy of their machine learning models, enabling them to make more informed decisions about a variety of business issues.

In summary, data imputation is a powerful tool that has been utilized in a variety of business contexts to improve the accuracy and reliability of data analysis and decision-making. It has been used in the development of CRM systems, predictive analytics, and machine learning models, among other applications, and it has helped businesses to launch products, software, and other initiatives more effectively.

Ethical concerns

As the field of data science continues to advance and expand, it is imperative that we also consider the ethical implications of our research and data collection practices. In this section, we will delve into the topic of meta-data and the ethical concerns surrounding its collection and use. Meta-data, or data about data, can provide valuable insights and context for understanding and interpreting larger datasets. However, the collection and handling of meta-data also raises important ethical questions, particularly in regards to privacy and consent.

One example of data being used for malicious purposes is the Cambridge Analytica scandal. Cambridge Analytica was a political consulting firm that used data analytics to target political messages to specific groups of voters (Duhigg, 2018). The company acquired data on millions of Facebook users through a personality quiz app and used this data to create targeted political advertisements.

The use of this data for political purposes was deemed unethical for several reasons. First, the data was collected without the knowledge or consent of the users, and it was used for purposes that they were not aware of. This raised concerns about privacy and the manipulation of people's opinions and behaviors without their knowledge.

Second, the use of this data was seen as unethical because it was used to manipulate and influence the outcome of elections. The targeted political advertisements were designed to appeal to people's biases and emotions, and they were intended to sway people's opinions and behaviors in ways that were not transparent or fair.

Finally, the Cambridge Analytica scandal was considered unethical because it exposed the vulnerability of social media platforms to manipulation and abuse. The fact that a company was able to acquire data on millions of users without their knowledge or consent raised concerns about the security and privacy of social media platforms.

The Cambridge Analytica scandal is an example of data being used in an unethical context. The use of data to manipulate and influence people's opinions and behaviors was deemed unethical, as was the collection of data without the knowledge or consent of the users.

Another thing to reflect upon is the General Data Protection Regulation (GDPR), which is a set of rules and regulations that were implemented by the European Union (EU) in 2018 to protect the personal data of individuals and give them greater control over how their data is collected, used, and shared. The GDPR applies to any organization that processes the personal data of EU residents, regardless of whether the organization is based within the EU or not.

Data science, which involves the collection, analysis, and interpretation of large amounts of data, has the potential to conflict with the GDPR in several ways. One potential conflict is in the way that data is collected and used. The GDPR requires organizations to obtain explicit consent from individuals before collecting their personal data and to use that data only for the purpose for which it was collected. Data scientists may be working with data sets that were collected for one purpose, but they may be interested in using that data for another purpose, such as conducting research or building a predictive model. This could potentially conflict with the GDPR's requirements around the purpose of data collection and use.

Another potential conflict is in the way that data is shared. The GDPR requires organizations to be transparent about how they share personal data and to take steps to protect the privacy of individuals whose data is being shared. Data scientists may be working with data sets that are

shared with them by other organizations, and they may be interested in sharing those data sets with others for research or other purposes. This could potentially conflict with the GDPR's requirements around data sharing and privacy protection.

To avoid conflicts with the GDP), there are several steps that organizations and individuals can take (Dataconomy, 2018):

1. Obtain explicit consent: Before collecting personal data, organizations should obtain explicit consent from individuals and make sure that they are aware of how their data will be used. Data scientists should also be aware of the purpose for which the data was collected and ensure that it is only used for that purpose.
2. Use de-identified or aggregated data: Whenever possible, data scientists should use de-identified or aggregated data sets that do not contain personal information. This can help to minimize the potential for conflicts with the GDPR.
3. Implement appropriate safeguards: Organizations should implement appropriate safeguards to protect the privacy of individuals whose data is being used or shared. This could include measures such as encryption, access controls, and secure storage.
4. Be transparent: Organizations should be transparent about how they collect, use, and share personal data, and they should provide individuals with clear information about their rights under the GDPR. Data scientists should also be transparent about the data sets they are using and the methods they are using to analyze the data.
5. Seek legal advice: If you are uncertain about whether your data science activities are in compliance with the GDPR, it is important to seek legal advice to ensure that you are operating within the law.

In the case of this project we have taken steps to avoid trespassing any of these possibilities, such as signing a NDA-agreement that would prevent us from detailing any meta-data of the users to

people that are not involved with the Maternity Foundation. Likewise, upon the creation of an account on the Safe Delivery App, the data that is collected will be after that you accept user agreement, and you can also request to avoid sharing data too.

Case description

The Maternity Foundation is a non-profit organization that works to improve maternal and newborn health around the world. The organization was founded in Denmark in 2003, and has since expanded to work in more than 20 countries across Asia, Africa, and Latin America. In the context of this project for example, the countries that we looked at are India, and Ethiopia where they likewise have operations ongoing.

The Maternity Foundation's mission is to reduce maternal and newborn mortality and morbidity by providing high-quality, evidence-based education and knowledge to mothers and newborns in low- and middle-income countries. The organization focuses on a number of areas, including improving access to care, increasing the availability of skilled birth attendants, and providing education and training to health workers.

The Maternity Foundation works in partnership with local organizations, governments, and health systems to implement its programs and initiatives. The organization also conducts research and evaluation to assess the impact of its work and to inform the development of new strategies and interventions.

One of their initiatives is the application known as Safe Delivery App, which is a software program that is available for smartphones to be downloaded. The application itself is an educational tool for women that are working as nurses or midwives, whereas the application provides instructional videos and online courses to be taken that can act as supplemental to their education. This application in this project has been the primary source of data, as the data that we have been utilizing has been sourced from the user traffic of the users on the application.

The specific scope that we have been looking at is the users that are located in India, and Ethiopia, as the Maternity Foundation wanted to be able to do data imputation upon these users

as the data that is provided from the users in these region may not be as “clean” as one would want it to be. As such, we have in this case been working to accomplish that objective by using tools such as KNNImputer, and other data science techniques.

Why can this project benefit Maternity Foundation

We believe that researching data imputation and missing data for the Maternity Foundation, that this topic is of great importance and offers significant potential value for the organization.

Data imputation is a statistical technique used to estimate and fill in missing values in a dataset. This is important because missing data can significantly impact the accuracy and reliability of analysis and decision making. By using data imputation techniques, we can improve the quality and completeness of the data available to the Maternity Foundation, which in turn can lead to more informed and effective decision making.

One business model in which data imputation can be particularly useful is predictive modeling. By using imputed data, we can build more accurate and reliable predictive models that can help the Maternity Foundation anticipate and address potential issues related to their business operations and gain a stronger insight in the metrics of their users to better be able to accustom their software to that customer segment.

The value created by data imputation can be seen in the improved accuracy and reliability of the data available to the Maternity Foundation, as well as the potential for better decision making and improved outcomes in maternal health. This value can be measured in several ways, including using metrics such as accuracy and precision of the imputed data, and the effectiveness of the predictive models built using the imputed data.

To determine whether data imputation is worth pursuing, it is important to carefully assess the potential benefits and costs. This can be done through a cost-benefit analysis, which considers both the financial costs of implementing data imputation techniques and the potential value that can be generated. By conducting a thorough analysis, we can determine whether the potential benefits of data imputation outweigh the costs and make an informed decision about whether to invest in this approach.

Overall, we believe that data imputation is a valuable and promising approach for the Maternity Foundation. By improving the quality and completeness of the data available, we can enable more effective decision making and overall ensure that a clean data framework is provided for further usage. By carefully evaluating the costs and benefits of data imputation, we can determine whether this approach is worth pursuing and make a confident decision about investing in it.

In addition to the potential benefits mentioned above, data imputation can also help the Maternity Foundation to comply with relevant regulations and standards. For example, many health organizations are required to report data on maternal health outcomes to government agencies or other regulatory bodies. To do this accurately and reliably, it is important to have complete and accurate data. By using data imputation techniques, the Maternity Foundation can ensure that it is able to report complete and accurate data, which can help to meet these requirements and maintain compliance.

Furthermore, data imputation can also help to improve the credibility and reputation of the Maternity Foundation. By providing high-quality data, the organization can demonstrate its commitment to evidence-based decision making and transparency, which can enhance its reputation and credibility among stakeholders. This can be particularly important in the field of maternal health, where trust and credibility are crucial for building strong relationships with other stakeholders.

In conclusion, data imputation is a valuable and promising approach for the Maternity Foundation. By improving the quality and completeness of the data available, the organization can enable more effective decision making, improve outcomes in maternal health, and maintain compliance with relevant regulations. By carefully evaluating the costs and benefits, the Maternity Foundation can make an informed decision about whether to invest in data imputation and ultimately drive value for the organization and its stakeholders.

Data / Method

The data utilized in this project was received to us through the organization the Maternity Foundation. They have collected the data through their application, Safe Delivery App, whereas the application has collected it through the user registration process, and the prolonged usage by the user. As previously mentioned, the Safe Delivery app is a software program that the user has downloaded down onto their smartphone device, and by using it, it collects a number of different data points. These data points include such as last login time, geographic location, and user ID. This data based upon usage and registration is then stored, and that data is what we have been using as our material to base the project on. As far as the different types of data, there are overall three in total:

1. Activity data – The data stored in this folder is divided into 44 different .parquet files, each registering when the application is opened by a user. This can result in one user having multiple entries as it registers activity, thereby one user can potentially have many activities that can be registered here.
2. MyLearning - This data stored here contains the information relating to the application specific modules, and the learning chapters that are related to the Safe Delivery app.
3. User data – This data is a registry of the users based on the answers they gave upon the creation of their profile on Safe Delivery App. This includes data such as their district, state, profession, and educational level.

Exploratory Data Analysis

The data which has been utilized in this project has been sourced through the Maternity Foundation by means of their application, which is the Safe Delivery App. This application has been used across the globe by people of various backgrounds, however, chiefly with the objective of providing midwives and nurses supplementary knowledge for various maternal-related healthcare subjects.

This project mostly involved three different kinds of data. One was the activity data, the other was the user data and the last one was my learning data. The initial thought was to look at the data first and identify the missing values from the datasets, as well as the amount of them. Followed by the initial step, the team chose some specific features to work with within the user data. Data activity was planned to be used to find a way to impute the missing user data by using other dimensions.

The team started the task with an exploratory data analysis of the user data. The data itself is split into different categories. One of these categories in the user-data information, which is a parquet file listing the following features:


```

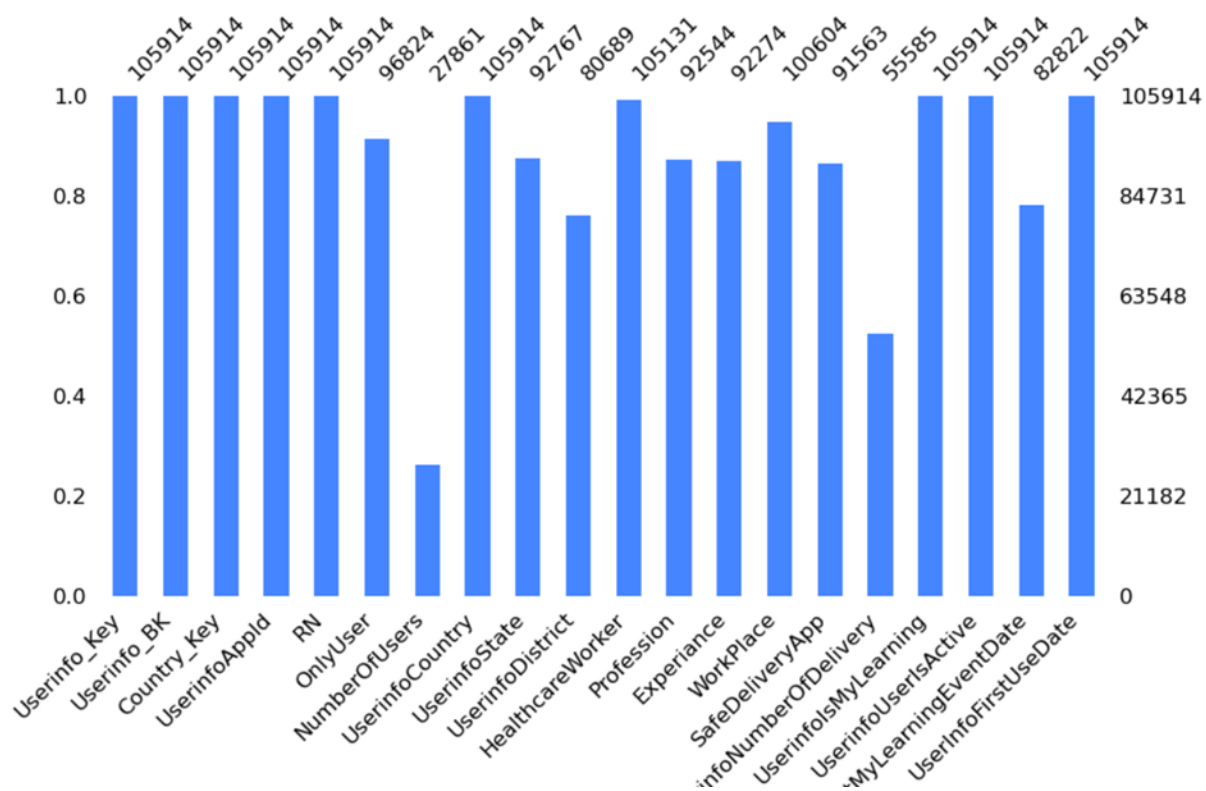
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105914 entries, 0 to 105913
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Userinfo_Key                          105914 non-null object
1   Userinfo_BK                           105914 non-null object
2   Country_Key                           105914 non-null object
3   UserinfoAppId                         105914 non-null object
4   RN                                     105914 non-null float64
5   OnlyUser                              105914 non-null object
6   NumberOfUsers                         105914 non-null object
7   UserinfoCountry                       105914 non-null object
8   UserinfoState                         105914 non-null object
9   UserinfoDistrict                     105914 non-null object
10  HealthcareWorker                      105914 non-null object
11  Profession                             105914 non-null object
12  Experiance                            105914 non-null object
13  WorkPlace                             105914 non-null object
14  SafeDeliveryApp                       105914 non-null object
15  UserinfoNumberOfDelivery              105914 non-null object
16  UserinfoIsMyLearning                  105914 non-null float64
17  UserinfoUserIsActive                  105914 non-null float64
18  UserinfoFirstMyLearningEventDate      82822 non-null  datetime64[ns, UTC]
19  UserinfoFirstUseDate                  105914 non-null datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](2), float64(3), object(15)
memory usage: 16.2+ MB

```

(Figure 4: List of features in user_data - Own creation)

While exploring the data, some values were found to contain the string 'Unknown'. These 'Unknown' strings were replaced with NaN for the purpose of analysis. The label Unknown could have an administrative, but were treated as equivalent to NaN in the EDA. After turning the 'Unknown' strings into NaN, the next step was finding the missing values across the data set, as it will give the best insight as to which dimensions would be more valuable to impute from a business perspective.

Through the use of the missingno module, the amount of NaNs were identified for each of the user attributes. Among them, OnlyUser, UserinfoState, UserinfoDistrict, Workplace, Profession, SafeDeliveryApp, Experience, HealthcareWorker are the main ones.



(Figure 5: Distribution of missing values within user data. - Own creation)

Followed by the last operation, it was found that the attribute named 'NumberOfUsers' has no use. As it has just count of the users. Added to that, there is a significant difference between being an OnlyUser and a MultiUser as far as user characteristics goes. Using these dimensions to describe characteristics could provide insight, but due to the limitations in the 'mapping' of the values, it was impossible to utilize them properly.

The next exploratory analysis focused on testing the dependence between dimensions. A chi-squared test was performed between the UserinfoCountry feature and each of the 4 target variables. This was done to determine the underlying missing data mechanism. In other words, whether the data is Missing Completely at Random (MCAR), Missing At Random (MAR) or Missing Not At Random (MNAR). These data mechanisms carry meaning in terms of the choice of method. After performing the chi-squared test with Workplace, Profession, SafeDeliveryApp, Experience, and HealthcareWorker it was found that there is a significant association between the missing data and UserinfoCountry. The underlying mechanism is therefore either MAR or MNAR. Based on where the data is coming from originally, it is likely MNAR.

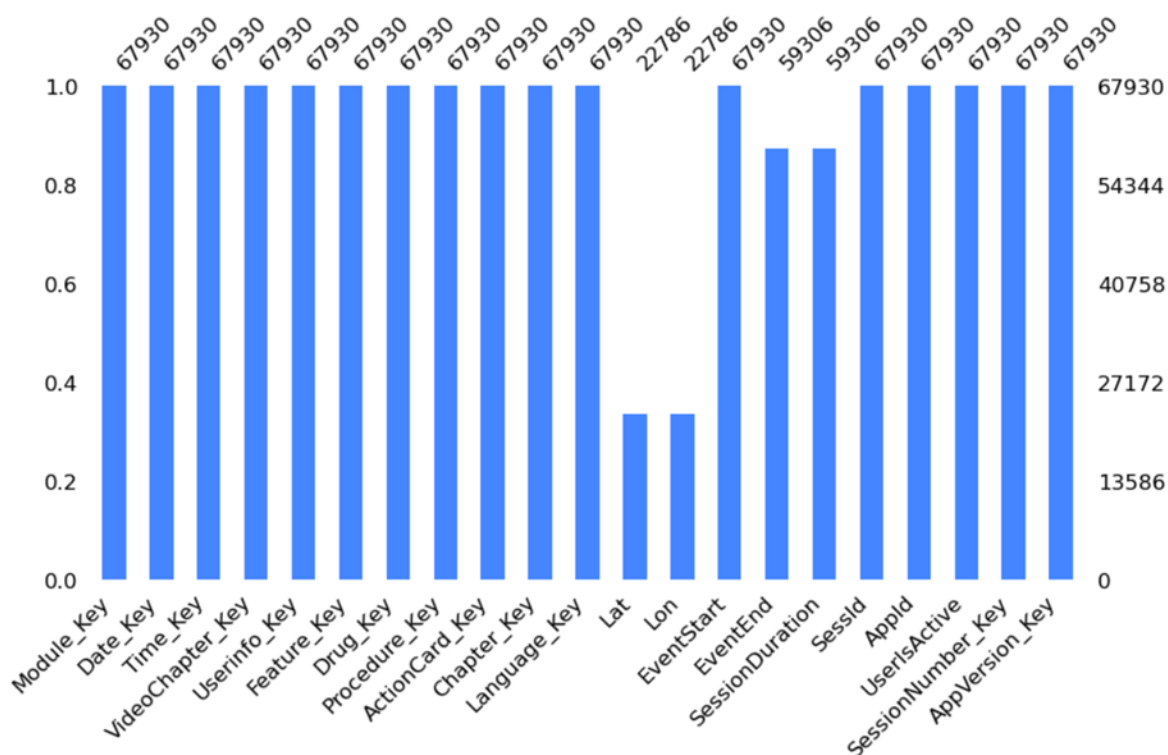
Followed by the last operation, the next focus is on the exploratory analysis of activity data to see whether the file contains any important dimensions that can be used to fill out the missing data of user data. The data itself is split into different categories. One of these categories is the activity-data information, which is a total of 44 parquet files listing the following variables:

```
[8]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25336286 entries, 0 to 25336285
Data columns (total 21 columns):
 #   Column                Dtype
---  -
 0   Module_Key            object
 1   Date_Key              object
 2   Time_Key              object
 3   VideoChapter_Key     object
 4   Userinfo_Key         object
 5   Feature_Key          object
 6   Drug_Key              object
 7   Procedure_Key        object
 8   ActionCard_Key       object
 9   Chapter_Key          object
10  Language_Key         object
11  Lat                   object
12  Lon                   object
13  EventStart            datetime64[ns, UTC]
14  EventEnd              datetime64[ns, UTC]
15  SessionDuration       float64
16  SessId                object
17  AppId                 object
18  UserIsActive          object
19  SessionNumber_Key     object
20  AppVersion_Key        object
dtypes: datetime64[ns, UTC](2), float64(1), object(18)
memory usage: 4.0+ GB
```

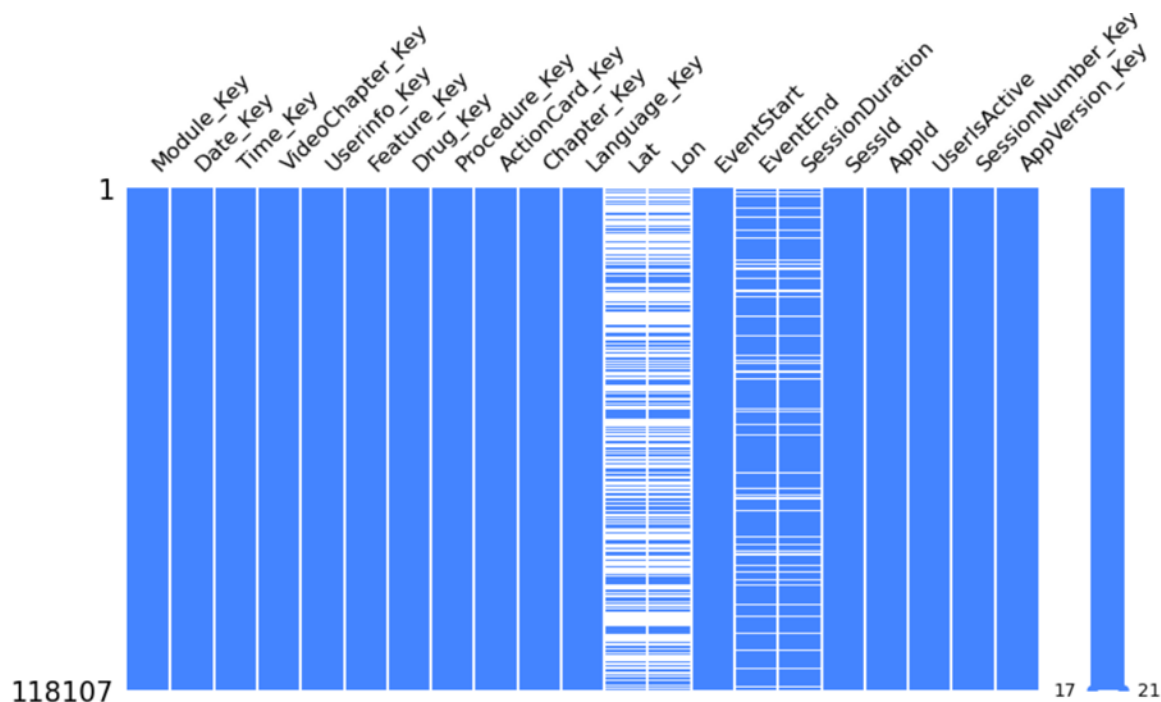
(Figure 6: List of features in data_activity - Own creation)

After screening all the data from all the 44 files, the missingno python was used to check the distribution of missing values for all 44 data activity files. In all 44 files, a similar trend has been identified. Only 4-four columns have missing values. These are Latitude, longitude, EventEnd Date, and SessionDuration. Obviously, when a user forgets to stop the session, the event end date column will be blank, and so will the session duration. On the other hand, it is possible that the user started, but did not complete the session. Or even that they are somehow reluctant to finish the session. In addition to that, it seems that the app does not ask for location while using the app. Added to that, the app can be run offline too. Here is the scenario of missing data:



(Figure 7: Distribution of missing values in data_activity - Own creation)

Since all the activity_data files have a similar type of missing data pattern. The row-wise analysis will yield a more concrete scenario of what is missing. In this operation, the fascinating output is the missing distribution. The result is shown below:



(Figure 8: Missing data patterns in data_activity - Own creation)

After the exploratory analysis of activity data, it was found that there is not enough variable to predict the latitude and longitude so as the data of district and state dimensions from the user data. Additionally, the data activity files are limited to a specific timeframe.

Data preparation and preprocessing

There were a lot of missing district and state values in the user dataset and to try to impute the values using some sort of mapping method, other datasets were explored to see if there was anything linking users to a particular location. There were latitude and longitude coordinates in the user activity dataset having multiple entries for the same user and those had different Latitude and Longitude values. To get around this problem we simply took the most frequent Latitude and Longitude as a user's coordinate. Then using geopy for reverse geocoding, we converted the coordinates into district and state names. But for 130 users, reverse geocoding resulted in empty strings as district and state names, so we had to manually search the district and state using coordinates and google maps. Then merging the obtained results with our original user dataset we had a bit higher number of users who had the location data included. Using this dataset we went further with the imputation and feature extraction of the user dataset.

Even with the expanded dataset we didn't get much better results for imputation of user data. However, if more data had been provided initially, perhaps more user data could have been found, or more relevant dimensions can be identified. But in addition to that, in the activity data, there seems to be two necessarily important dimensions, and the dimensions are the average time spent on the sessions by the user and the most frequent module key. To get information about the active time value of the user average time per month per user by aggregating the session time of each user by month is calculated and then used mean operation to get the mean session time of each user, which gave a very good estimation of how long each user was spending on the app and to also get how the time was being spent. For every user, the most frequently used module key is used as that model represents the user's attributes more.

Looking into another dataset (mylearning), where the features `KlpResultScore` and `KlpResultLevel` seem to represent the users achievement scores. Both of them seemed viable to be merged into the user dataset but `KlpResultScore` had way too many missing values so only feature `KlpResultLevel` were taken and proceeded with. But the problem here was also that each user had multiple scores as they attempt multiple modules. However the mean value of `KlpResultLevel` is taken as the score for each user.

Dimensions achieved from the two different dataset were merged to the user dataset. The problem with trying to merge was the difference in number of users in my learning, activity and user dataset. The user dataset had 105914 unique users, activity had 33099 unique users and there were only 14285 unique users in my learning dataset. This meant just merging would result in a lot of missing values. To try and get around this problem, time and klp result level value are set to 0 and module key as -1. After filling the datasets we were able to merge the datasets.

Some of the features from the user dataset didn't make sense for the imputation and some of the other features had a lot of missing values in comparison. So, dropping some of the features from them

- `UserInfoFirstMyLearningEventDate` and `UserInfoFirstUseDate`: This was a date value and in this case, it wouldn't make sense to impute based on. This is because the data value does not directly bear relevance for the user characteristics. This is the reason why we decided to remove this feature.

- Userinfo_Key : It was a feature with unique values to identify each user. It didn't have any relation with other features. It was merely used in linking other datasets to this dataset which we have already done. So, we were able to remove this feature as well.
- Userinfo_BK : This feature probably acted as a secondary key in the system. Similar to UserInfoKey it was also an unique identifying feature, with no relations to other features.
- UserinfoAppId : As was the case with the two previous features, this feature functions merely as an identifier for the individual user AppId. For this reason, it didn't carry particular feature importance either.
- UserinfoState & UserinfoDistrict : We had originally kept State and District information but as they have restrictions like a district needs to be inside a particular district and a state inside Country, we realized it probably made more sense to leave them out of imputation and decided to remove the two features.
- Country_Key : There was another feature called UserInfoCountry which had the same information as Country_Key. So we decided to only keep UserInfoCountry and label encode the feature. Keeping both of these, could potentially create a bias, or enforce an existing one.
- NumberOfUsers: This feature describes the number of users per application login. It had 73% missing data, so we decided there wasn't enough preexisting data to go off of. Also imputation of this feature doesn't make sense as it functions more as an indicator for the amount of users of the individual user account.
- OnlyUser : Same as NumberOfUsers, except that it instead indicates when only a single person is using the app user account.
- UserinfoNumberOfDelivery : It had 47% of missing data which is quite higher than the other features, so we decided to get rid of the feature.

To train and test our algorithms the data need to be numeric but some of the features in the dataset were in strings that represented different categories/classes. However labelEncoder from scikit-learn preprocessing package allows the encoding of the string categorical values into numerical labels, and also allows the reversion back to string categorical values if incase needed in future.

Feature importance method

Within data science, a feature is an individual measurable property or characteristic of a phenomenon being observed. When working with data, it is often necessary to identify which features are the most important for a particular task of analysis. This process is known as feature selection.

Feature significance scores can be calculated for problems involving the prediction of a numerical value (referred to as regression problems) and problems involving the prediction of a class label (referred to as classification problems). Feature importance scores are significant to understand the dataset, a model and identify the important features by reducing redundant variables.

We assigned some of the variables as the target variables. Experience, Profession, SafeDeliveryApp and Workplace are the features we decided to focus the imputation on, since these are the characteristic attributes of the users. Furthermore, for each of them we conducted feature importance analysis individually to find the important features that we will use in the imputation. After we had accomplished that, we decided to split the dataset for the cross-validation and a 20 % test size was randomly selected from the dataset.

```
Shape of X_train dataset: (72807, 11)
Shape of y_train dataset: (72807,)
Shape of X_test dataset: (18202, 11)
Shape of y_test dataset: (18202,)
```

(Figure 9: Shapes of the training sets - Own creation)

RandomForestRegressor was used to find the important features. In terms of the RandomForestRegressor where the number of estimators is 25 used. Mean square error is used to evaluate the prediction. *GridSearchCV* function is used to perform hyper parameter tuning of the model. The best evaluated model based on the cross-validation is chosen as the best estimator for the fit.

Same method was applied to each of the target features, in order to identify the feature importance, with a split of 80% train data and 20% test data. Finally we confirm the feature importance of the dataset.

Test setup and evaluation of results

Before starting the test, it was necessary to create a control for comparison after performing the imputation. To accomplish this, a dataset containing the extra dimensions was discovered to increase the accuracy. The dataset was then cleaned, where afterwards two copies of the dataset were used for the test. One of these copies would then, based on the calculated amount of missing values for each of the original features, have the same percent of missing values randomly placed within them. Finally, this dataset would be imputed using the MICE method with KNNRegressor as the build in estimator. To test the quality of our estimates, and thereby our imputation method, three different methods were utilized to give a full perspective on the estimations. This is due to each method providing a slightly different perspective on the quality of the estimate.

The first method was calculation of the Mean Square Error (MSE) between the imputed data values, and the corresponding true values, and as well as the variation of the true variables.

$$Var = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{X})^2 \quad MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \theta(X))^2$$

(Figure 10: Equations for variation and MSE. Source: (Beretta, L., Santaniello, A.(2016))

This enables the possibility to evaluate the relative value distribution match between our true, and imputed data. The lower the MSE value for each variable is, the closer the estimates should be to the true mean distribution. Furthermore, MSE can be compared to the true variable variation, in order to evaluate how good the estimations are. If the MSE value is close to, or lower than the variation value, that would indicate a good estimate. If on the other hand the MSE value is significantly higher than the true variation, this would indicate that the imputations have

a significantly different value distribution (Beretta, L., Santaniello, A. (2016)). An issue with this method approach however, is that when it's applied to large datasets this measurement value can tend to be very small, due to the measurement being performed across a large population.

Since the comparison between MSE and variation mostly provides insight about the fit to the true mean distribution, a second more direct method will also be taken into use. This involves what is known as a difference matrix between the imputed and true data values. From start to finish, it simply involves subtracting the imputed dataframe (or matrix) from the true data. This can be done for either the individual dimensions for an individual difference score, or for the entire difference matrix as a whole. The theory behind it is that the closer to the true values that the estimates come for each dimension, the closer to 0 the sums will get. This therefore gives a more quick and direct comparison of how good the overall estimate is for the actual values.

The second method however suffers from the problem that redistributions from high to low values or vice versa, can cause high difference scores, whilst still being fairly accurate overall. Therefore the third method will simply be to compare the distributions of the true and imputed label values. This method was chosen to get a direct measurement of the model accuracy. Due to the functionality of IterativeImputer in Python, an existing method for testing the accuracy of imputations couldn't be located. This meant that observing and calculating from the real redistributions of the labels, was the most precise measure for accuracy. We used the following basic equations to directly calculate the accuracy.

$$\left(\frac{\text{number of wrong labels}}{\text{original number of missing values}} \right) * 100 = \% \text{ misplaced labels}$$
$$100 - \left(\left(\frac{\text{number of wrong labels}}{\text{original number of missing values}} \right) * 100 \right) = \% \text{ accuracy}$$

(Figure 11: Equations for accuracy measure. - Own creation)

By taking the sorted value counts of each true feature value, and subtracting those counts from those of the imputed values, we end up with a table showing the direct label redistributions. This

we can then use to calculate the percent of misplaced labels, and thereby in turn the individual feature prediction accuracy as a percentage.

Technical implementation

As shown earlier in the project paper, there is a difference between the two demographic groups included in the dataset, which is Indians and Ethiopians in this case. Due to this imbalance it may not be as statistically significant what sample you can derive from the users that are from Ethiopia, then in comparison to those users who are in India. Additionally, some of the variables are not fulfilled by the users in Ethiopia, such as UserinfoDistrict, whereas the large majority of the users in India have decided to include an answer to the variable.

This dissimilarity that exists between the two user groups can potentially then lead to a bias in the user attributes, as the Indians are largely represented than in contrast to the users that are from Ethiopia. Secondly, with a smaller sample group than in comparison to the users that are from India, it can potentially lead to issues when deriving features and/or doing PCA on the attributes that are tied to the Ethiopian users. The reasoning behind why it can be a potential issue, is that there is a smaller data sample group to base the PCA on, that in contrast to those users who are from India, which may lead to a bigger risk of potentiality for skewed results, or that it is less accurate.

Which models are used, and what are their inputs and outputs?

Random Forest

One of the widely used machine learning algorithms is Random Forest. It is mainly used for regression and classification problems. This algorithm uses different samples to build decision trees and pick up their majority referendum for classification and mean in case of regression. This algorithm's one of the most significant features is that it can handle the categorical variable of a dataset for classification and continuous variable for regression.

There are a few steps in the working procedure of random forest. Firstly, it picks n number samples of data from a dataset which has k amount of data. Followed by the operation, separate decision trees are constructed based on each sample. Then the output gets generated from the decision tree. The final result or output is chosen from the majority of the referendum or averaging for Classification and regression accordingly.

There are five different types of features in the random forest. They are:

Diversity: Every tree is different in the random forest. Not all the variables were taken while building an individual tree.

Immune to the curse of Dimensionality: The previous feature mentioned that it does not count every variable. So, it automatically reduced the range of variables, and space got limited in the model. Thus, the algorithm can work well and can avoid the curse of dimensionality.

Parallelization: While creating the tree, random forests do it individually and separately. As this operation is done separately, the random forest can make full use of the power of the CPU.

Train-Test split: There is no need to split the data into train and test. Because when decision trees are made, it already skips 30% of data for the test. Thus, the segregation of the train and test of data can be avoided.

Stability: As the majority of the referendum is used over here; thus, stability over the result can be achievable.

In the random forest algorithm, there are some crucial hyperparameters. These hyperparameters are used to boost the prediction power and the speed of the model. Out of them, `n_estimators`, `max_features`, and `mini_sample_leaf` are used mainly for power prediction, and `n_jobs`, `random_state`, and `oob_score` are used for speeding up the model.

KNN

The K-Nearest Neighbors (KNN) imputation method is a technique for imputing missing values in a dataset by using the values of the nearest neighbors of each missing value.

Here is an outline of the steps involved in the KNN imputation method:

1. Determine the number of nearest neighbors (K) to use for the imputation process.
2. For each missing value in the dataset, find the K nearest neighbors of that value based on some distance metric, such as Euclidean distance.
3. Use the values of the K nearest neighbors to impute the missing value. This can be done using the mean, median, or mode of the K nearest neighbors, or by using some other statistical method.
4. Repeat the imputation process for all missing values in the dataset.

The choice of K can have a significant impact on the accuracy of the imputed values. A larger K may result in a smoother imputation, but may also reduce the influence of individual points and lead to less accurate imputations. A smaller K may result in more accurate imputations, but may also be more sensitive to noise in the data. Also different distance metrics may be more or less appropriate depending on the characteristics of the data. For example, the Euclidean distance may be a good choice for continuous variables, while the Manhattan distance may be more appropriate for categorical variables.(P. Jonsson and C. Wohlin, 2004)

The KNN imputation method is often used when the missing data is believed to be "missing at random," meaning that the missing values are not related to the observed values in any systematic way. If this assumption is not met, the imputed values may be biased or inaccurate. It can be a useful method for imputing small amounts of missing data, but may not be as accurate as more sophisticated imputation methods for large amounts of missing data. It can also be computationally intensive, for large datasets with a large number of missing values. Additionally, the KNN imputation method can be sensitive to the presence of outliers in the data, as the imputed values may be influenced by the values of the nearest neighbors, including any outliers. This can lead to imputed values that are significantly different from the true underlying values.

MICE imputation

The MICE method is an iterative imputation process for creating strong estimates of true values. The first step in this method is the initial simple imputation strategy, that is chosen based on the distribution of the data. These strategies often vary between using the mean, median or mode values of the target feature for the initial imputation estimate. The next step is to apply an algorithm, such as the ones mentioned above, to predict the true values of the same estimate, and compare them to the first initial round. These new predicted values become our second round of imputations. The algorithm will then be applied again on the predicted values from the second round, creating a third round of estimated values. This third round gets compared to the previous (second) round of estimates, and the iterative cycle continues on. The order in which the imputations happen can be changed to better fit the data structure etc. The default for IterativeImputer for example is a descending order, meaning it will start with the first column containing missing values, find the first row with a missing value and then impute these row by row, column by column (Hannah S Laqueur, Aaron B Shev, Rose M C Kagawa(2022)).

Project implementation:

In this project we decided upon using the Agile project management theory as a platform to base our work upon. We felt that in this project it was important to retain a degree of flexibility as we knew that during the time we had to research and write the project, there would be a likelihood that we were going to change our efforts accordingly to the challenges that we would stumble upon. When we originally started the project we had the intention to base the data which contained the geographic information (Longitude, latitude) to impute missing user location. This was the case leading up until a midpoint of the writing period when we realized that imputing upon this would be too statistically insignificant, leading us to the decision to forfeit this effort. This whole period with us attempting to impute this would be a sprint within the agile project management tool, and in our project writing we have had several sprints when we completed one part of the project, to then leading it to take on another part. This was a great boon to us as when we in this project experienced several times that we had to change, alter, and modify our programming code, our findings, and objectives accordingly to those roadblocks we stumbled

upon.

Like the software developers that met in Snowbird, Utah two decades ago, we two can testify to the gains we have enjoyed with the elasticity that is offered by the agile project theory, rather than in contrast to the rigid structure that is found in the waterfall theory.

The waterfall model in the field of data science often is lacking because it does not take into account the iterative and exploratory nature of data analysis. Data science projects often involve a lot of trial and error, and the rigid structure of the waterfall model can make it difficult to accommodate this type of experimentation and discovery.

In contrast, other approaches to software development and data analysis, such as agile methodologies, are designed to be more flexible and adaptable to change. These approaches are better suited to the iterative and exploratory nature of data science and allow for more flexibility and agility in the development process. During this project, that is the overall experience as often ideas & concepts did not carry out in reality when working with the data.

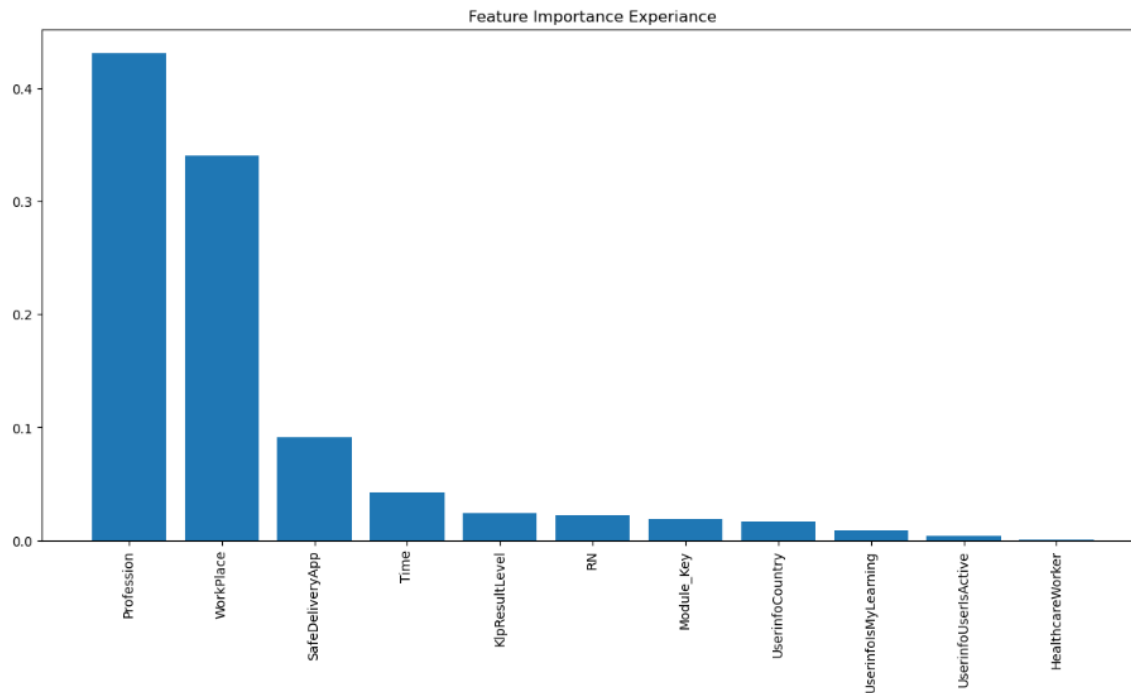
Analysis of results

Analysis of features importance

When imputing missing data, identifying important features is one of the significant tasks in the machine learning process. We can better understand the data when we consider feature importance. As we mentioned earlier, `RandomForestRegressor` and `GridSearchCV` functions are used to perform the optimal hyper parameter of the model. As we've demonstrated, using feature importance analysis may improve the model's overall performance. Even while some models, like `RandomForestRegressor`, choose features for us, it is still crucial to understand how a particular feature affects the model's performance because it offers us more control over the task we're attempting to complete.

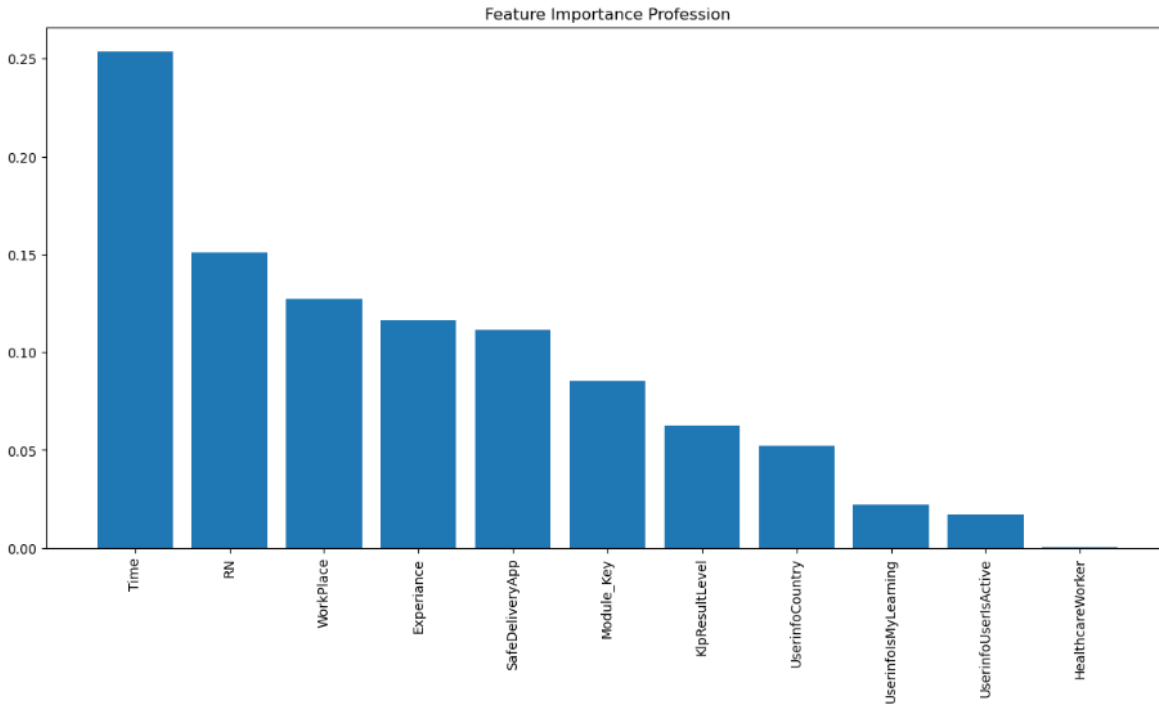
The **figure 12** below illustrates that certain features are really not employed at all whereas others have a significant effect on *Experience*. However, through selecting a portion of the most crucial

features, we can reduce the total number of features. The graph also shows Profession, WorkPlace, SafeDeliveryApp, Time and so on are the highly important features.



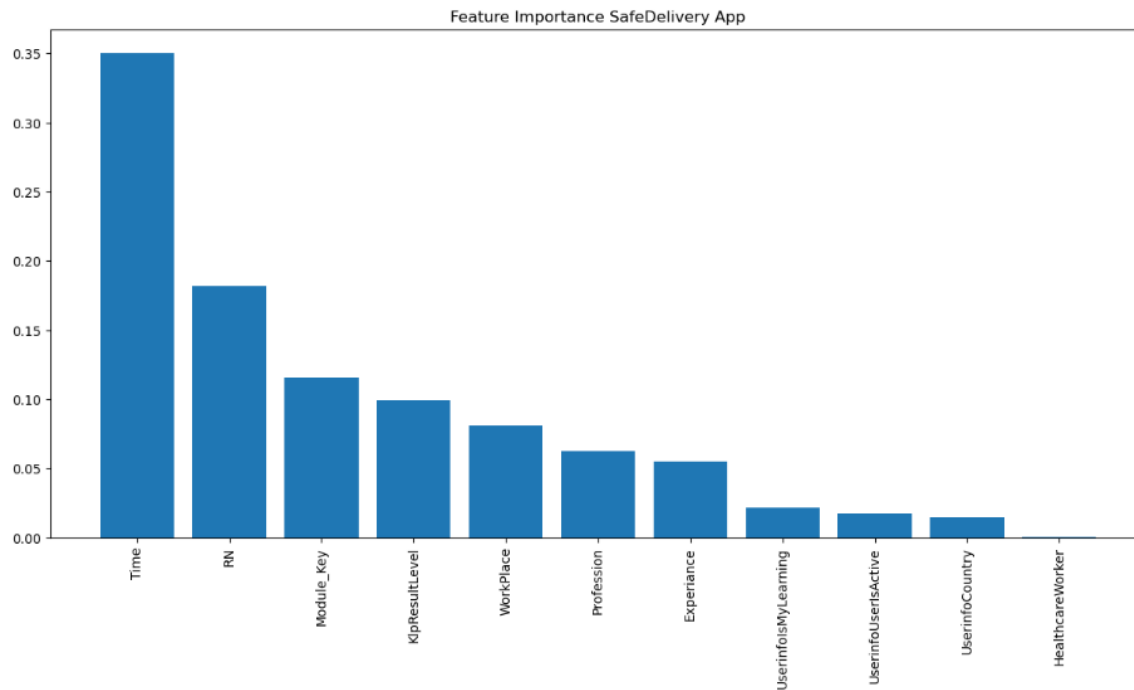
(Figure 12: Feature importances for Experience dimension - Own creation)

The figure 13 shows *Profession* as a target feature and as we plot the features based on RandomForestRegressor and Hyper-parameter tuning methods, Time is interestingly one of the top 11 features. Furthermore, RN, Workplace, Experience, SafeDeliveryApp, Module_key and KlpResultLevel are highly important features of the dataset.



(Figure 13: Feature importances for Profession dimension- Own creation)

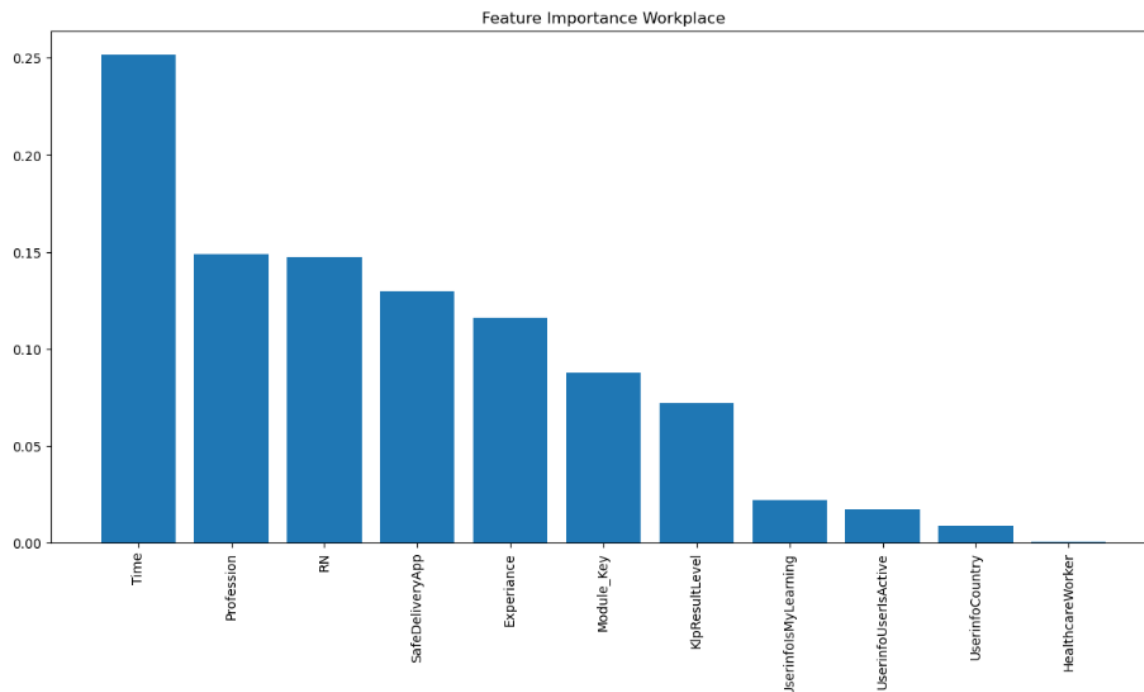
The figure 14 explains the *SafeDeliveryApp* as a target feature, whereas it shows 11 important features from the entire dataset. *Time* is the most important feature among all features; RN, Module_key, KlpResultLevel, Workplace, Profession and Experience comes consecutively as important features.



(Figure 14: Feature importances for SafeDeliveryApp dimension - Own creation)

The figure 15 is the last feature importance selection graph where *Workplace* is the target feature. It also shows the important features such as Time, Profession, RN, SafeDeliveryApp, Experience, Module_key and so on.

Overall, Profession, Experience, Time, RN, Workplace, SafeDeliveryApp and Healthcareworker are the important features. However, among the important features, some of the features data are missing. Hence, missing data imputation techniques will be applied to the missing features only.



(Figure 15: Feature importances for WorkPlace dimension - Own creation)

Analysis of results from model testing

The final hyperparameter setup that was set for the model pipeline was the following. As mentioned earlier, we used a KNNRegressor as the estimator for a MICE imputation. Through testing the performance of the model with different hyperparameters, we found that above a certain point, the number of neighbors used for the estimation didn't bear much different results in terms of the accuracy measures used. The weight parameter indicates which metric is used to measure nearby neighbors and the decision power each of them carry for the prediction. Both a distance based as well as the chosen uniform parameter was tested. Neither of these made a big difference in terms of accuracy either.

For the MICE part, the IterativeImputer was set to a maximum of 100 iterations, which is a relatively big number for the method. If the right dimensions are used for the imputation, anywhere from 5-15 iterations should suffice for a reliable estimate. The reason this parameter was set so high, was due to the built in default tolerance level of IterativeImputer, which wasn't reached with 30 iterations. More iterations should in theory only make the estimate more

accurate, but also require more computation time. At a certain point, approximating the estimate any closer to the theoretical true value becomes redundant for some use cases. Hence, the built in tolerance level, which as mentioned earlier wasn't reached within a low number of iterations. Finally, the initial strategy 'Most frequent' was chosen in order to try and help the model achieve a higher test score.

```
model = KNeighborsRegressor(n_neighbors=100, weights='uniform', n_jobs=-1,)
imputer = IterativeImputer(sample_posterior=False, random_state=seed, estimator=model, max_iter=100, initial_strategy='most_frequent')
scaler = MinMaxScaler(feature_range=(0, 1))
# Building the pipeline
pipeline = Pipeline(
    steps=[
        ('s', scaler), # First we scale the data
        ('i', imputer) # Then we fit the imputer and transform the NaN's
    ]
)
```

(Figure 16: Final imputation pipeline - Own creation)

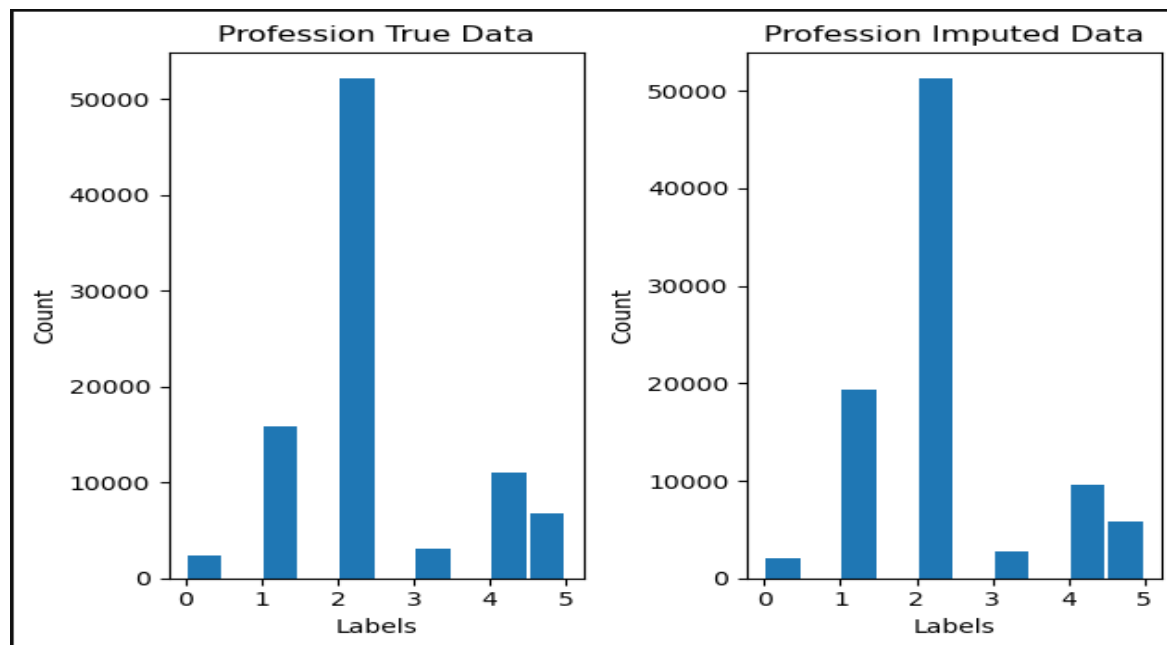
As described in the methodology, 3 different methods for testing were used. The first of these was a comparison of the MSE and variation of the true data.

Variance of true data:		Mean Square Error results between imputation and true data:	
RN	9.581651e+00	RN	0.000000
UserinfoCountry	3.820467e-02	UserinfoCountry	0.000000
HealthcareWorker	1.647935e-04	HealthcareWorker	0.000000
Profession	1.379516e+00	Profession	0.263029
Experience	1.920822e+00	Experience	0.286961
WorkPlace	2.028650e+00	WorkPlace	0.136866
SafeDeliveryApp	7.724805e+00	SafeDeliveryApp	1.182476
UserinfoIsMyLearning	1.848189e-01	UserinfoIsMyLearning	0.000000
UserinfoUserIsActive	4.974457e-02	UserinfoUserIsActive	0.000000
KlpResultLevel	3.685510e-01	KlpResultLevel	0.044518
Module_Key	6.959842e+09	Module_Key	0.002176
Time	7.566961e+07	Time	0.024097
dtype: float64		dtype: float64	

(Figure 17: True data variance - Own creation) (Figure 18: MSE between true data and imputed-Own creation)

This perspective seems to indicate that the estimates are pretty close to the same mean distribution. For Profession, Experience and WorkPlace we get MSE values that are well below their corresponding true variation. The same can be said for SafeDeliveryApp, although this feature seems to have a significantly higher variation as well MSE value. These low MSE values however, as mentioned in the methodology, are most likely due to the high number of users. It

would also seem that the pipeline treats some of the non-missing features, although this is more likely a rounding error somewhere. Next we can examine the distributions of the imputations:



(Figure 19: Histogram over the distribution of labels for Profession feature- Own creation)

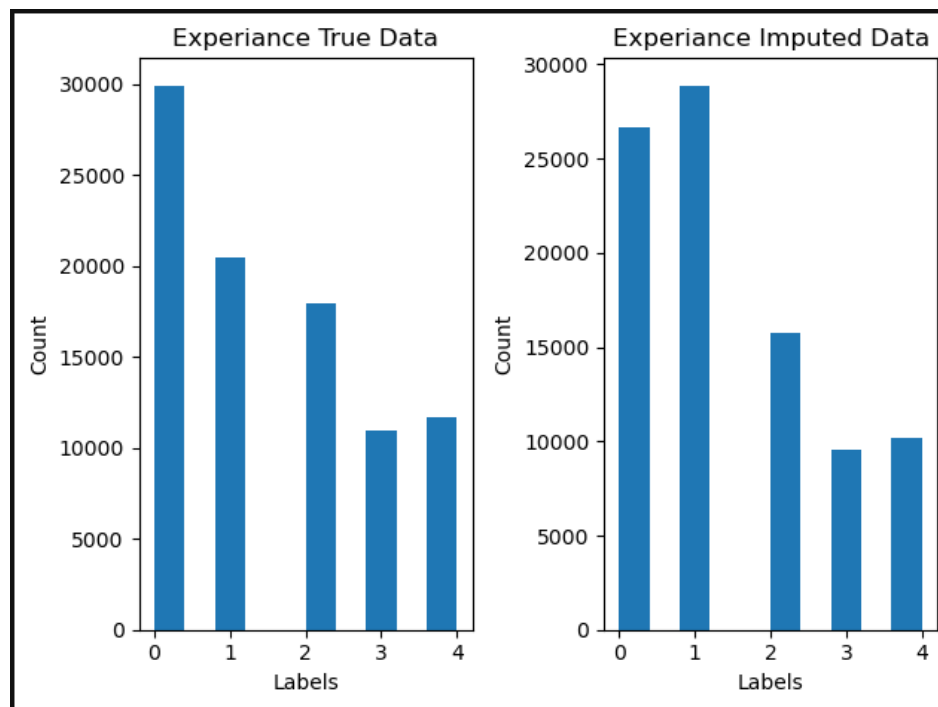
```
0    -307
1    3634
2    -857
3    -281
4   -1364
5    -825
Name: Profession, dtype: int64

print(f' Accuracy for Profession: {100 - (3634 / 0.13)} %')
print(f' Percent misplaced labels: {(3634 / 0.13)} %')

Accuracy for Profession: 69.28452553720385 %
Percent misplaced labels: 30.715474462796156 %
```

(Figure 20: Redistribution and accuracy measures for Profession feature- Own creation)

This seems to indicate that the final estimate for Profession is not completely wrong, but still not as good as it could be. As can be seen in the histograms in figure 18, as well as from the top table in figure 19, a total of 3634 labels were incorrectly relabeled as 1 instead. Out of the original amount of missing data, this gave a total accuracy rating of 69.28% for Profession, which is not an optimal percentage, and ideally should be at least 10% or so higher.



(Figure 21: Histogram over the distribution of labels for Experience feature- Own creation)

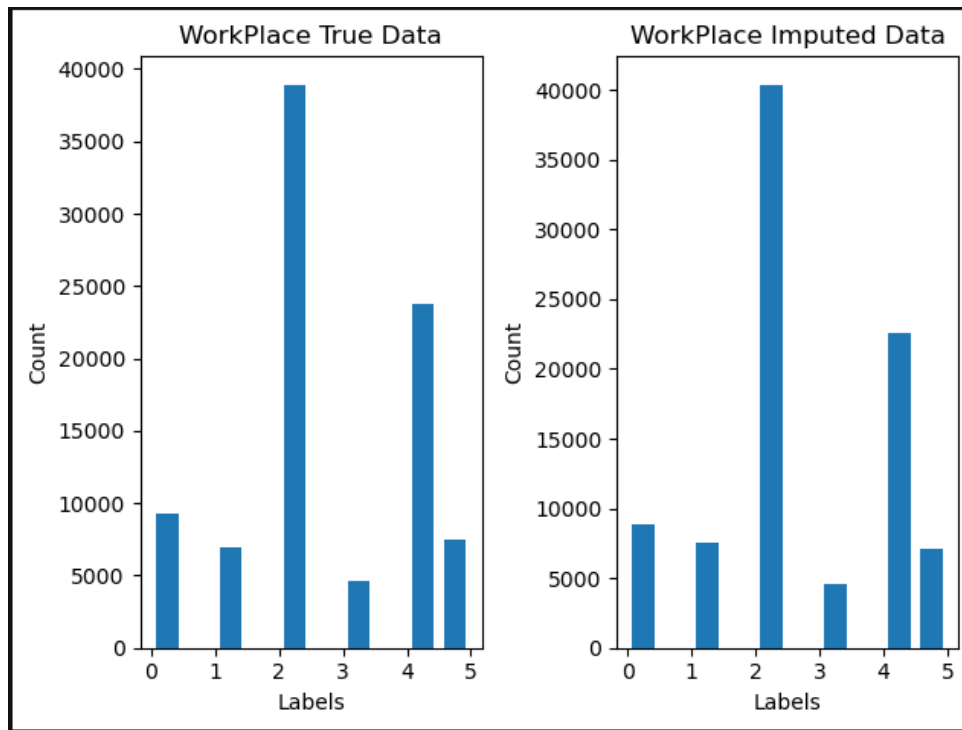
```
0    -3285
1     8411
2    -2193
3    -1399
4    -1534
Name: Experience, dtype: int64

print(f' Accuracy for Experience: {100 - (8411 / (0.
print(f' Percent misplaced labels: {(8411 / (0.

Accuracy for Experience: 27.797319495874035 %
Percent misplaced labels: 72.20268050412596 %
```

(Figure 22: Redistribution and accuracy measures for Experience feature- Own creation)

The results for the imputation of this feature did not look good. Whilst the MSE and variation comparison does seem to suggest a good estimation, it is clear from both the histogram and accuracy measures that the dimensions feature importances most likely weren't captured correctly. Despite testing several hyperparameters and models, the redistributions always reached high values as can be seen from both the histogram in figure 20 as well as the table 21. One of the reasons as to why these prediction accuracy rates so far have been low, will be discussed later.



(Figure 23: Histogram over the distribution of labels for Workplace feature - Own creation)

```
0    445
1   -601
2  -1478
3     74
4    1199
5     361
Name: Workplace, dtype: int64

print(f' Accuracy for Workplace: {100 - ((601 +
print(f' Percent misplaced labels: {(601 + 1478)

Accuracy for Workplace: 54.31221088024261 %
Percent misplaced labels: 45.68778911975739 %
```

(Figure 24: Redistribution and accuracy measures for Workplace feature - Own creation)

The results for the Workplace feature were not good either. Once again we see from the MSE and variation measure, that they indicate a good fit, as well visually looking to be a closer match to the true distribution than Experience. The accuracy score of 54.31% still only barely beats flipping a coin, which once again would seem to indicate that not all of the feature importances were correctly found.



(Figure 25: Histogram over the distribution of labels for WorkPlace feature - Own creation)

```

0      832
1      231
2      590
3     1215
4    -2448
5    -3061
6    -7015
7     4492
8     3311
9     1853
Name: SafeDeliveryApp, dtype: int64

print(f' Accuracy for SafeDeliveryApp: {100 - ((2448
print(f' Percent misplaced labels: {(((2448+3061+7015

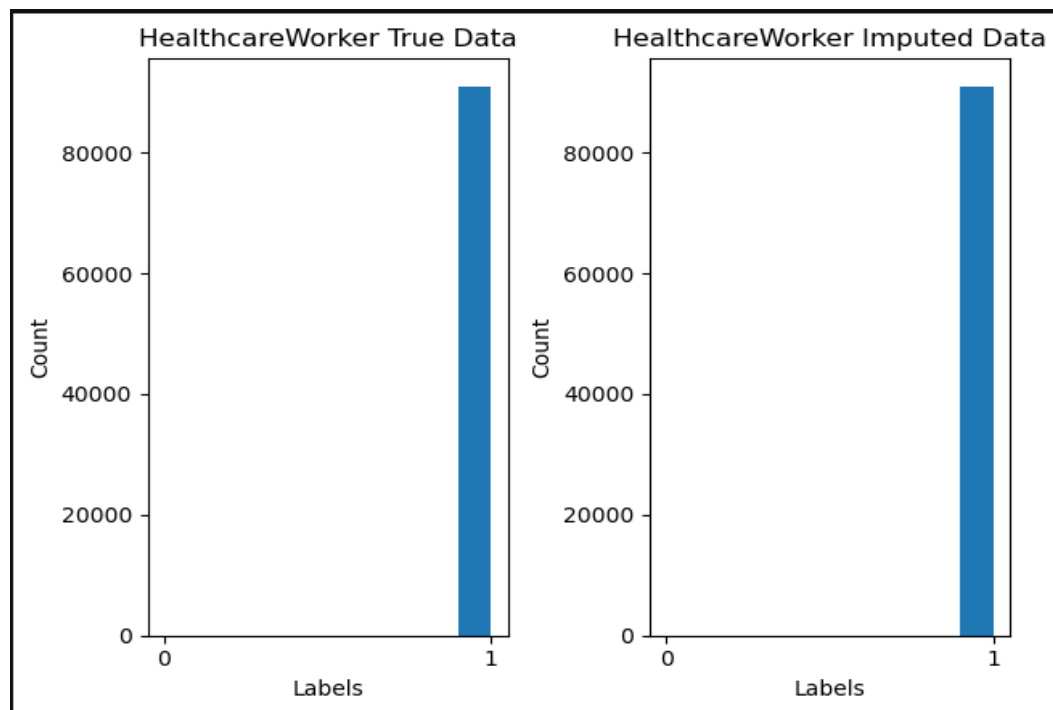
Accuracy for SafeDeliveryApp: 1.7051688765475461 %
Percent misplaced labels: 98.29483112345245 %

```

(Figure 26: Redistribution and accuracy measures for SafeDeliveryApp feature - Own creation)

The results for the SafeDeliveryApp were quite frankly hard to explain. The best reasoning behind this incredibly low score, is the way in which the test set was originally created. Since the values had been label encoded, upon inserting random NaN's values, some of these distributions

may have been altered or slightly skewed. This could cause some of the issues with the results so far, but still doesn't explain the full picture. This will be followed up in the discussion section.



(Figure 27: Histogram over the distribution of labels for HealthcareWorker feature - Own creation)

As the final feature, HealthcareWorker didn't really pose any sort of challenge in terms of imputation. The original number of missing values was less than 1%, making the built model pipeline more than capable to impute this dimension from the offset. With that being said, had more data been missing from the feature, it would likely have seen results more like the other features.

Discussion

As is evident from the results of the imputation tests, the success criteria of building a model to impute the missing feature was in most cases not achieved. Whilst two of the dimensions did achieve accuracy levels close to or above 70%. One of these, HealthcareWorker, was given by the initial low number of missing values. For the second feature, Profession, the current theory is that we managed to find a fitting dimension within the other data, most likely being Time as that achieved a high feature importance for most features. It is believed that more effort should have been put into identifying and testing additional dimensions for feature importances.

Since the KNN algorithm is a similarity-based approach, using the most important features for estimation becomes more central to the algorithms method. Whilst three extra dimensions were found and calculated (Time, KlpResultLevel and Module_Key), only 2 of these really had any feature importance to the user characteristics. The approach for the imputation was in the end to fit_transform the entire dataframe at one time, as this approach had given the best results, compared to a more direct or individual imputation method based on the feature importances of each user characteristic. Had better dimensions been identified however, individually imputing each missing characteristic with its most important features may yield better results for an estimation to the true value.

The next steps to try and improve the model would be to spend more time to find the right dimensions with a high feature importance for each of the target features. These dimensions should then be tested for imputation on the individual target features, as well as a full dataframe imputation, in order to compare the two approaches. Imputation of all the data simultaneously comes with the benefit of less code and less computation. At the same time it is easier to implement a singular model or pipeline into an application, than it is to implement four or five. However, imputing the values using the appropriate important dimensions alone, should in theory lead to a more correct estimate when using a similarity-based algorithm like K-Nearest Neighbor.

Finally for the model, the KNN algorithm could also be the wrong choice in terms of the size of the data. As mentioned in the Data/Method section, the KNN method can be strong when applied to smaller datasets, but accuracy often drops with larger datasets. Another possibility is that the data wasn't sufficiently cleaned for outliers and other types of problems that the KNN algorithm is sensitive to, before testing was performed. A RandomForest algorithm was also tested as the estimator for the IterativeImputer initially, but proved less efficient than KNN in comparison. Using a Classifier rather than a Regressor version of this algorithm may also prove more efficient, but this could not be achieved due to a limitation between IterativeImputer and the Classifier class.

The agile approach proved to be a strong project management tool, but it also gave limited time to both create prototype solutions, analyze results and keep the internal group collaboration active. This especially proved true in conjunction with the weekly meetings with all other groups, as keeping track of this became an extra task to assign effort towards. Getting earlier insight into just the data structure, could possibly have given enough insight to create a framework for the initial analysis of the imputation metrics in a more efficient, front loaded manner.

Conclusion

To conclude, the data imputation that we utilized for this project was not entirely successful in achieving a high level of predictor accuracy. There may be several reasons for this result, including the complexity of the data, the limitations of the imputation method we decided to use, or possibly the risk of unforeseen biases or errors in the data.

Moving forward, it may be necessary to reconsider the imputation strategy, or explore alternative approaches for handling missing data. It may also be useful to conduct further analysis and investigation to identify the accuracy of the predictor.

Overall this project highlights the importance of careful consideration and careful evaluation of data imputation techniques in the context of a specific data set and prediction task.

Appendix:

The list of files to be submitted for the project submission is the following:

- `Data_cleaning.ipynb` - This jupyter lab document contains the preprocessing for the project and data preparation.
- `Feature_importance.ipynb` - This jupyter lab document contains our feature importance.
- `Imputation_pipe_test.ipynb` - This jupyter lab document contains our main data imputation, as well the output that we will base our conclusion primarily upon.
- `Imputation_prelim_EDA.ipynb` - This jupyter lab document contains EDA.
- `Lat-Lon_reverse.ipynb` - This jupyter lab document uses the user data latitude and longitude to find the district of the user and states.
- `Requirements.txt` - This word text document contains the !pip installs that are required to run the various jupyter lab documents.
- `Project_Read_Me.txt` - This word document contains instructions how to run the various jupyter lab documents, and in which order.

References:

Barnard, J.; Meng, X. L. (1999-03-01). "Applications of multiple imputation in medical studies: from AIDS to NHANES". *Statistical Methods in Medical Research*.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*.

Buuren, S. V., & Groothuis-Oudshoorn, K. G. (2011). MICE: Multivariate imputation by chained equations in R. *Journal of Statistical Software*

Little, R. J. A., and Rubin, D. B. (2002). *Statistical analysis with missing data*. John Wiley & Sons.

Van Buuren, S. (2012). *Flexible imputation of missing data*. CRC Press.

<https://www.maternity.dk/safe-delivery-app/>

<https://www.maternity.dk/>

Agile Manifesto. (n.d.). Retrieved from <https://agilemanifesto.org/>

Kanbanize. (n.d.). What is Agile? Retrieved from <https://www.kanbanize.com/agile-methodology/>

Project Management Institute. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Newtown Square, PA: Project Management Institute.

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide*. Scrum.org.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*

Srivastava, R., Dhar, V., & Bucklin, R. E. (2002). Customer relationship management: a database approach. *The Journal of Marketing*

Duhigg, C. (2018). How Cambridge Analytica used your Facebook data to help elect Trump. *The New York Times*. Retrieved from <https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html>

"General Data Protection Regulation (GDPR)." (n.d.). European Commission. Retrieved from https://ec.europa.eu/info/law/law-topic/data-protection/reform/regulation-eu-2016-679-general-data-protection-regulation-gdpr_en

"GDPR and data science: how the two intersect." (2018, May 25). Dataconomy. Retrieved from <https://dataconomy.com/gdpr-and-data-science-how-the-two-intersect/>

<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

P. Jonsson and C. Wohlin, "An evaluation of k-nearest neighbour imputation using Likert data," 10th International Symposium on Software Metrics, 2004. Proceedings., 2004, pp. 108-118, doi: 10.1109/METRIC.2004.1357895.

<https://ieeexplore.ieee.org/document/1357895>

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

<https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html>

Hannah S Laqueur, Aaron B Shev, Rose M C Kagawa, SuperMICE: An Ensemble Machine Learning Approach to Multiple Imputation by Chained Equations, American Journal of Epidemiology, Volume 191, Issue 3, March 2022, Pages 516–525, <https://doi.org/10.1093/aje/kwab271>

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011., <https://scikit-learn.org/stable/about.html#citing-scikit-learn>

Pulkit Sharma: Why is scaling required in KNN and K_Means. Aug 25, 2019.

Medium.com/analytics_vidhya. URL:

<https://medium.com/analytics-vidhya/why-is-scaling-required-in-knn-and-k-means-8129e4d88ed7>

Beretta, L., Santaniello, A. Nearest neighbor imputation algorithms: a critical evaluation. BMC Med Inform Decis Mak 16 (Suppl 3), 74 (2016). <https://doi.org/10.1186/s12911-016-0318-z>

Illustrations

- ❖ Figure 1: Maternity foundation logo - Maternity.dk
- ❖ Figure 2: Example of code for using KNNImputer - Own creation
- ❖ Figure 3: Another example for using KNNImputer - Own creation
- ❖ Figure 4: List of features in user_data - Own creation
- ❖ Figure 5: Distribution of missing values within user data
- ❖ Figure 6: List of features in data_activity - Own creation
- ❖ Figure 7: Distribution of missing values in data_activity - Own creation
- ❖ Figure 8: Missing data patterns in data_activity - Own creation
- ❖ Figure 9: Shapes of the training sets - Own creation
- ❖ Figure 10: Equations for variation and MSE. Source: (Beretta, L., Santaniello, A.(2016)
- ❖ Figure 11: Equations for accuracy measure. - Own creation
- ❖ Figure 12: Feature importances for Experience dimension - Own creation
- ❖ Figure 14: Feature importances for SafeDeliveryApp dimension - Own creation
- ❖ Figure 15: Feature importances for WorkPlace dimension - Own creation
- ❖ Figure 16: Final imputation pipeline - Own creation
- ❖ Figure 17: True data variance - Own creation
- ❖ Figure 18: MSE between true data and imputed - Own creation
- ❖ Figure 19: Histogram over the distribution of labels for Profession feature
- ❖ Figure 20: Redistribution and accuracy measures for Profession feature - Own creation
- ❖ Figure 21: Histogram over the distribution of labels for Experience feature- Own creation
- ❖ Figure 23: Histogram over the distribution of labels for WorkPlace feature- Own creation
- ❖ Figure 24: Redistribution and accuracy measures for WorkPlace feature- Own creation
- ❖ Figure 25: Histogram over the distribution of labels for WorkPlace feature - Own creation
- ❖ Figure 26: Redistribution and accuracy measures for SafeDeliveryApp feature - Own creation
- ❖ Figure 27: Histogram over the distribution of labels for HealthcareWorker feature- Own creation