## **Introduction**

# Missing data arise in almost all serious statistical analyses, which is defined as the data value that is not stored for a variable in the observation of interest. The problem of missing data is common even in a well-designed and controlled study. Missing data can reduce the statistical power of a study, produce bias in the estimation of parameters, reduce the representativeness of the samples, complicate the analysis of the study, and lead to invalid conclusions that can be drawn from the data [1].

# Creating and deploying highly predictive models depends on several factors. One of the most common reasons that models perform poorly is the use of data with poor quality. Un-cleaned and incomplete data can have drastic effects on a model. Missing or incomplete data is commonplace in today’s world. Between unanswered surveys, incorrect data gathering, and mistakes, there are large amounts of missing data in datasets. There are several techniques that can be used to deal with this problem including simple imputation, substitution, regression imputation, and multiple imputation.

# This analysis examines the quality of a predictive linear model given various formats of missing data. These formats include Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). The amount of missing data along with the format of missing data can affect the quality and predictive accuracy of a model. Each type of missing data requires different strategies to impute the data [2].

# The dataset we use for this missing data and multiple imputation study is the Boston Housing Dataset with only 506 observations [3]. It was originally described by Harrison and Rubinfeld [4] and obtained from Scikit-learn and has been used extensively throughout the literature to benchmark algorithms.

In this case study, we fit a linear regressor to the Boston data as a baseline for comparison, perform a fit with the imputed data from these types (MCAR, MAR or MNAR) of missing data with varying percentages, and compare the loss and goodness of fit to the baseline. Additionally, we use the MCMC method to compare the difference in performance between imputation with mean and MCMC.

## **Background**

Missing data can be happened because of the errors in data collection, the research design purposes (e.g. survey experiments), incomplete survey question answers, or censored values. To decide how to handle missing data, it is helpful to know how the data came to be missing from the dataset.

In general, there are three types of common missing data mechanisms [2].

### Missing completely at random (MCAR)

### If data are MCAR, the probability of a missing observation () is not related to the value of in dataset. In other words, each datum that is present had the same probability of being missing as each datum that is absent, or no correlation between which data points are missing and which ones are not. This implies that ignoring the missing data will not bias inference. Data is often missing in this format due to errors in obtaining the data or technological errors that may delete data completely at random.

### Missing at random (MAR)

### If data are MAR, the missingness happens in a random process and does not depend on the value of after controlling for another variable. The constraints here that is that missingness may only depend on information that is fully observed. For example, when a certain group of people chooses not to respond to a particular question on a survey. This can be determined if it is related to the group of similar people who chose to leave that particular response blank. MAR is more general than MCAR in that all units do not have equal probabilities of being missing. Therefore, MCAR is a special case of MAR.

### Missing not at random (MNAR)

### For MNAR, missingness depends on the value of . Data that is MNAR is directly related to a value in another column. When a certain condition is met in another column, the data in the column of interest will be missing in this case. This type of data often occurs on surveys as well. For example, in studying income, if conditional on all the observed data, individuals with higher incomes are more likely to withhold information about their incomes, then the missing income data is MNAR. The cases of MNAR data are problematic.

**How to handle missing data with MCAR assumption?**

### Listwise deletion (or complete-case analysis) is to delete any case where some data is missing (i.e., use only cases with complete data, thus omit entire case if any data point is missing). Assuming the missing data are MCAR and/or MAR, it is generally acceptable method when dataset lacks only a few points. In this method, we ignore any observations with missing values on variables necessary to estimate the model. This method is simple to deploy but discarding valuable information can reduce power of statistical tests, especially for a small sample [2].

### Pairwise deletion (Available-case analysis) uses all available data to estimate statistics, assuming the missing data are MCAR. This may lead to parameters of a model based on datasets of different sizes/standard errors and result in intercorrelation matrix that is not positive definite, if elements estimated with all available data. It can therefore create computational problems for certain statistical analyses. In OLS, this means estimating the regression coefficients from the means, variances, and covariances of the variables rather than directly from the observations. While this appears to use more information than complete-case analysis, it can sometimes be *less efficient*. And by basing each statistic of interest on different subsets of the data, results can become nonsensical. Finally, this method is much more difficult to implement outside of OLS to other GLMs. Pairwise deletion is less biased for the MCAR or MAR data. However, this analysis will be deficient if there are many missing observations [2].

**What are the practical imputation methods to handle missing data?**

Imputation refers to filling in missing data with plausible **imputed** values. The completed data set is then analyzed using traditional methods. The imputation methods include single imputation and multiple imputation.

**Single imputation:** replace each missing value with a "likely" value and use that data set to perform analysis. Replace each missing value using some mechanism that offers a reasonable approximation of missing value. For examples: mean (or median, or mode) substitution, dummy variable approach, regression substitution, last value carried forward, hot deck imputation, weighted or propensity approach.

Single Imputation methods may distort associations between variables, lead to overestimation of measured variables precision when data is analyzed after imputation, create bias in certain estimators, and provide overly optimistic results due to increase in sample size.

**Mean Substitution**: (1) substitute a (global) mean for missing value and run analysis as if all data are present, (2) use standard analysis methods, (3) lead to an underestimate of error (reduces variability), (4) cause changes in covariance and correlation, (5) not recommended.

**Dummy Variable Approach:** (1) create a new variable that is marked 0 if a value is missing and 1 if it is present, (2) include missing indicator in regression (or analysis), (3) use all the subjects and may indicate importance of missing pattern as advantage, (4) introduce bias and generally has no theoretical basis as disadvantage.

**Regression Substitution:** (1) linear regression is used to predict the missing score based on other variables, (2) imputed values are dependent on other variables, (3) technique does not add more information but increases sample size and reduces standard error.

**Last Value Carried Forward (LVCF):** use last observed value to fill in missing values at subsequent points. This is often found in longitudinal studies. For example, in a year-long medical study where subject outcomes are measured every month, subject(s) might miss appointments. This method is popular due to ease of use and clarity. However, it assumes the response remains constant which will distort estimates.

**Hot Deck Imputation:** the missing values are replaced with values taken from "nearest neighbor" or "best match" respondents. LVCF a form of hot deck imputation. Can produce biased estimates of correlation and regression coefficients.

**Propensity Score Substitution:** use logistic regression to model "missingness" of data, and Regression generates (as a new variable) propensity score for each subject. This method may underestimate variability.

**Multiple Imputation**

Multiple imputation is another useful strategy for handling the missing data. The missing values are replaced by values calculated via existing values from other variables, and the newly generated values (imputes) stand in for missing values. Imputed values are drawn from distributions that reflect specific uncertainties about the existing values. Multiple datasets are created, each containing a different set of imputes. Standard techniques are then used to analyze each dataset and results are combined into overall analysis [2]. This imputation method can produce valid inference to reveal the uncertainty associated with the estimation of the missing data, even in the presence of high percentage of missing data in a small sample size.

Multiple imputation using Markov Chain Monte Carlo (MCMC) method constructs a Markov chain to simulate draws from the posterior distribution. The missing data are imputed through repeating the imputation step and the posterior step. MCMC has gained popularity in many applications due to the advancement of computational algorithms and power.

## **Method**

**Data description**

The Boston Housing Dataset was obtained from the StatLib archive [3] and has been used extensively throughout the literature to benchmark algorithms. However, these comparisons were primarily done outside of Delve and are thus somewhat suspect. The dataset is small in size with only 506 observations, as originally described by Harrison and Rubinfeld [4].

**Table 1**: **Variable Descriptions and Types**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Range/Values** |
| CRIM | per capita crime rate by town | 0.01 to 88.98 |
| ZN | proportion of residential land zoned for lots over 25,000 sq.ft. | 0 to 100 |
| INDUS | proportion of non-retail business acres per town | 0.46 to 27.74 |
| CHAS | Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) | 0 to 1 |
| NOX | nitric oxides concentration (parts per 10 million) | 0.39 to 0.87 |
| RM | average number of rooms per dwelling | 3.56 to 8.78 |
| AGE | proportion of owner-occupied units built prior to 1940 | 2.9 to 100 |
| DIS | weighted distances to five Boston employment centres | 1.13 to 12.13 |
| RAD | index of accessibility to radial highways | 1 to 24 |
| TAX | full-value property-tax rate per $10,000 | 187 to 711 |
| PTRATIO | pupil-teacher ratio by town | 12.6 to 22 |
| B | 1000(Bk - 0.63) ^2 where Bk is the proportion of blacks by town | 0.32 to 396.9 |
| LSTAT | % lower status of the population | 1.73 to 37.97 |
| MEDV | Median value of owner-occupied homes in $1000's | 5 to 50 |

**Perform a fit with the imputed data from MCAR**

The goal of this case study is to see how missing data, and imputing that missing data affects a predictive model. To start a baseline linear regression model was fitted predicting the price of houses in Boston. From here a few different techniques were attempted to simulate missing data. The first involved taking a random column and removing at random 1, 5, 10, 20, 33, and 50% of the values in that column. Each time the missing values were imputed with the mean of the column. A new linear regression was fit for each of the modified data sets.

**Perform a fit with the imputed data from MAR**

The next technique involved removing and imputing values of a specific column based on the value of another column. For all the records that had a CRIM value of over 3.467, 10, 20 and 30% of those were chosen at random and had their corresponding TAX and PTRATIO values removed. These missing values in the TAX and PTRATIO were then imputed with the mean of the respective column. Each of these three modified datasets were the used to fit a linear regression model predicting the price of a home.

**Perform a fit with the imputed data from MNAR**

The last technique used was again removing and imputing values in a column based on another column’s values but this time the values were not chosen at random. If a record had a value for age that was in the 25th percentile than that records RM value was removed. Instead of imputing the missing RM values with the mean of the RM column the missing values were imputed with the 25th percentile value for RM. This resulted in the youngest 25% of people having their RM values replaced with the 25th percentile value for RM.

**Compare performance between imputation with mean and MCMC method**

For MCMC method, the MNAR missing data are imputed through repeating the imputation step and the posterior step. Specifically, we use PROC MI method in SAS to accomplish the MNAR missing data imputation, and then compare the performance with mean imputation as mentioned above.

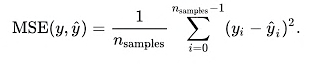
## **Results**

**Linear regressor baseline to fit the Boston Housing dataset**

In linear regression, the **R2 coefficient of determination** is a statistical measure of how well the regression predictions approximate the real data points. An R2 of 1 indicates that the regression predictions perfectly fit the data.

**Root mean squared error** (**MSE)** and **Mean Absolute Error** (**MAE**) are two of the most common metrics used to measure accuracy for continuous variables. Both express average model prediction error in units of the variable of interest. The range of MSE and MAE is from 0 to infinity. Both are indifferent to the direction of errors. They are negatively-oriented scores, which means lower values are better.

MSE is a risk metric corresponding to the expected value of the squared error or loss. It is computed with prediction and actual observation. If *i*is the predicted value of the *i*-th sample, and *yi* is the corresponding true value, then the mean squared error (MSE) estimated over *nsamples* is defined as

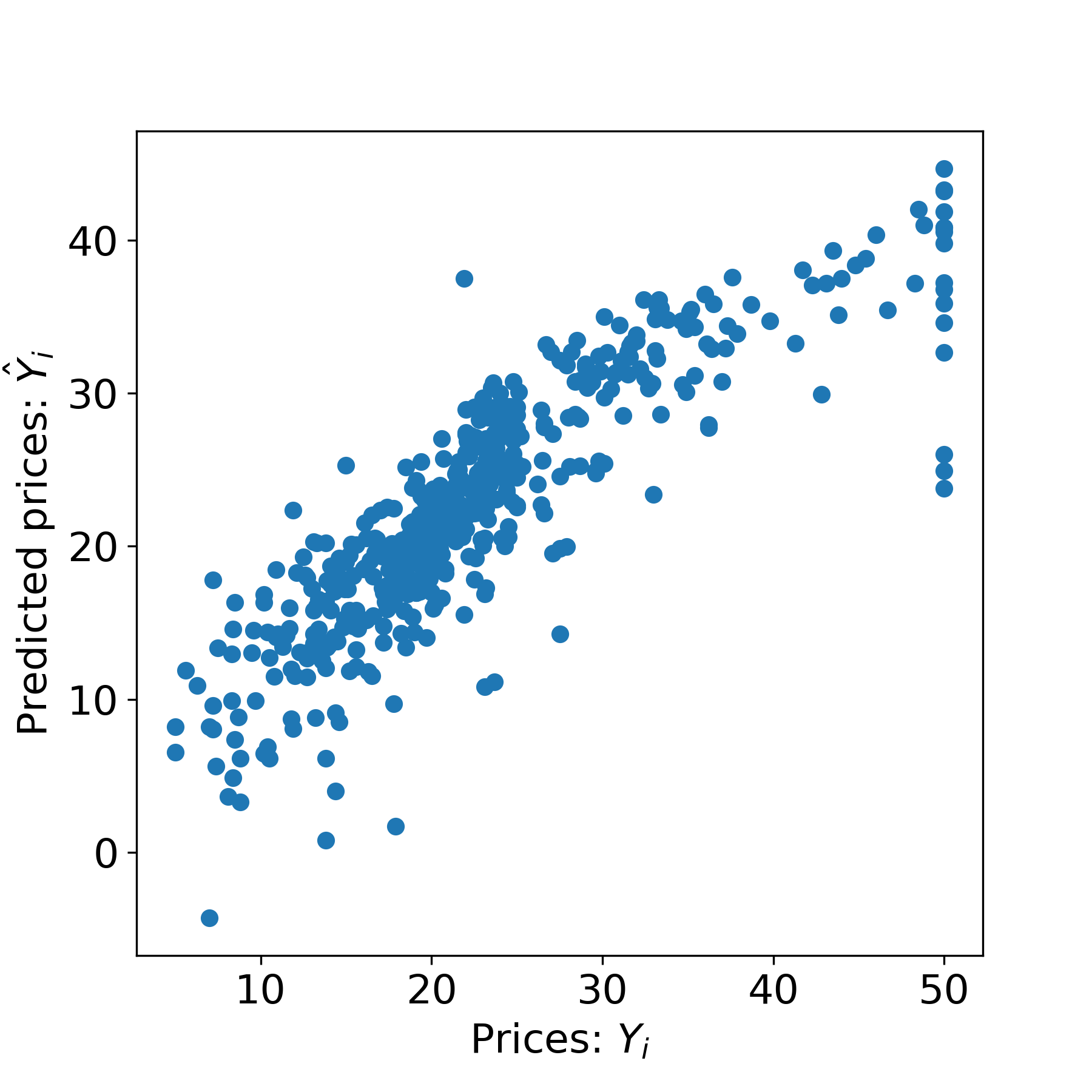


MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is robust to outliers. The loss is calculated by taking the median of all absolute differences between the prediction and actual observation where all individual differences have equal weight. If *i*is the predicted value of the *i*-th sample, and *yi* is the corresponding true value, then the median absolute error (MedAE) estimated over *nsamples* is defined as



There are no missing values corresponding to any observations in the dataset. As shown in **Figure 1**, the baseline value for R2 is 0.7406, MSE baseline score is 21.8948, and MAE baseline score is 3.2709.

**Figure 1**: **Prices vs Predicted prices**



Next, we will discuss the fit performance in three types of common missing data mechanisms (MCAR, MAR and MANR).

**Perform a fit with MCAR imputed data**

The present values in single column “NOX” are replaced with NANs and imputed with mean at varied imputation rates (1, 5 10, 20, 33, and 50%), then calculate the R2, MSE, and MAE scores in linear models.

**Figures 2 and 3** shows the scatterplots with actual and predict values (**Figure 2**), R2 (**Figure 3A**), MAE (**Figure 3B**) and MSE (**Figure 3C**)scores for each of the linear regression models with a certain percent of values in a random column imputed with the mean. It should be noted the same column was selected every time. Not surprising as the higher percent of values was imputed with the mean the worse the model got in terms of all three evaluation statistics. Each statistic followed the same pattern as well, improving early then gradually getting worse.

From the baseline to 1% imputed and then 1% imputed to 5% imputed the model actually got better. One would not generally think that by replacing a randomly selected 5% of your data with the mean that it would improve the model. This could be explained though by which 5% of the data is getting replaced. If there are outliers in the 5% than it would make sense that the model actually increased in performance because it is removing observations that would have caused an issue if the remained in the dataset. This also indicates that the researcher can drop missing data if the number of the cases is less than 5% of the sample.

As the missing percentage increased from 5% on though the model got drastically worse each time. This could be explained by the missing percentage. Imputing 10% of your data is a large amount. By imputing such a large amount, it could cause the model to struggle to differentiate between observations because so many observations have the same value, the mean.

|  |  |
| --- | --- |
| **Figure 2: Prices vs predicted prices with varied imputation rates in “NOX” column** | |
| **A: 1%** | **B: 5%** |
| **C: 10%** | **D: 20%** |
| **E: 33%** | **F: 50%** |

**Table2: Compare R2, MSE and MAE scores at varied MCAR data missing percentages**

|  |  |  |  |
| --- | --- | --- | --- |
| **Missing Percent** | **R-squared Value** | **MSE Scores** | **MAE Scores** |
| 0% | 0.7406 | 21.8978 | 3.2729 |
| 1% | 0.7412 | 21.8501 | 3.2701 |
| 5% | 0.7423 | 21.7517 | 3.2644 |
| 10% | 0.7371 | 22.1939 | 3.2775 |
| 20% | 0.7333 | 22.5145 | 3.2893 |
| 33% | 0.7304 | 22.7629 | 3.3049 |
| 50% | 0.7292 | 22.8594 | 3.3264 |

**Figure 3: R2, MSE and MAE scores in varied MCAR data missing percentages**

|  |
| --- |
| **A** |
| **B** |
| **C** |

**Perform a fit with MAR imputed data**

Take 2 different columns (“TAX” and “PTRATIO”) and create MAR data when controlled for a third variable (“CRIM”). If Variable CRIM >3.467, then variables “TAX” and “PTRATIO” are randomly missing. Make runs with 10%, 20% and 30% missing data imputed via mean, then repeat the fit and compare to the baseline.

**Figure 4: Prices vs predicted prices with varied MAR imputation rates**

(MAR columns 9: “TAX” and 10: “PTRATIO” based on CRIM >3.467)

|  |  |
| --- | --- |
| **A: 0%** | **B: 10%** |
| **C: 20%** | **D: 30%** |

**Table 3: Compare R2, MSE and MAE scores at varied MAR data missing percentages**

(MAR columns 9: “TAX” and 10: “PTRATIO” based on CRIM >3.467)

|  |  |  |  |
| --- | --- | --- | --- |
| **Missing Percent** | **R-squared Value** | **MSE Scores** | **MAE Scores** |
| 0% | 0.7406 | 21.89 | 3.27 |
| 10% | 0.7282 | 22.94 | 3.32 |
| 20% | 0.7255 | 23.16 | 3.34 |
| 30% | 0.7324 | 22.58 | 3.30 |

Using this technique provided similar results as the previous strategy. As the missing percentage increased the evaluation statistics became less favorable. The largest jump that was seen was between the 0% and 10% missing models. The jump between 10% and 20% was very minimal.

From 20% to 30% the model actual improved which is extremely interesting. 30% is a huge amount of data to impute with the mean and two columns were being imputed not just one. This is a lot of data that is being alerted and is very surprising that the model was able to identify records better with even more data being similar.

**Perform a fit with MNAR imputed data**

Create MNAR pattern in which 25% of the data is missing for a single column “RM”. Impute data with mean and fit the results and compare to a baseline.

**Figure 5: Prices vs predicted prices with imputing bottom 25th percentile (MNAR)**

|  |  |
| --- | --- |
| **A: 0%** | **B: 25%** |

**Table 4: Compare R2, MSE and MAE scores with imputing bottom 25th percentile (MNAR)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Missing Percent** | **R-squared Value** | **MSE Scores** | **MAE Scores** |
| 0% | 0.7406 | 21.8948 | 3.2709 |
| 25% | 0.6934 | 25.8856 | 3.6428 |

**Compare performance between imputation with mean or MCMC**

Monte Carlo Markov Chain (MCMC), a Bayesian simulation method, is used for multiple imputation. In a Bayesian modelling framework, missing data are accommodated simply by treating them as unknown model parameters.

MCMC provides for a posterior distribution based on markov chain random walks given the variability and uncertainty in the dataset, often via initial estimates of a covariance matrix and other critical parameters. This distribution stabilizes when there is no autocorrelation and is used to sample from in order to provide for imputed values.

**Table 5: Compare R2, MSE and MAE scores between imputation with mean or MCMC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Missing Percent** | **R-squared Value** | **MSE Scores** | **MAE Scores** |
| 0% | 0.7406 | 21.8948 | 3.2709 |
| 25% (Mean) | 0.6934 | 25.8856 | 3.6428 |
| 25% (MCMC) | 0.7071 | 24.7260 | 3.5329 |

Use this MCMC method, and the missing data from MANR pattern as shown in **Figure 5** and **Table 4,** we compare the difference in performance between imputation via ‘guess’ (mean/median, etc) and MCMC. The results are shown in **Table 5**, indicating that MCMC method performs slightly better than single imputation with mean.

Replacing the missed value with the arithmetic mean of the observed values for the variable in question preserves the mean of the variable. However, this method decreases its variance and its covariance with other variables. This can lead to biased regression coefficients and invalid inferences.

Comparatively, MCMC method replaces missed data with predicted values obtained from a statistical learning model, typically a regression model. Using the available data, regress each variable with missing data on the other variables in the data set. Then use the regression model to generate predicted values for the missing data in the regressed variable. This method can perform better than unconditional mean imputation as mentioned above. However, the imputed values form MCMC method still tend to be less variable than the real data because they lack residual variation. Additionally, there are still uncertainty in obtaining the imputed values with the estimates of the regression coefficients.

## **Conclusions**

# In this case study, we use Boston Housing Dataset to perform a fit with the imputed data from these types (MCAR, MAR or MNAR) of missing data, compare the loss and goodness of fit to the baseline, and use the MCMC multiple imputation method to compare the performance difference with simple mean imputation.

## If the missing data are produced with MCAR or MAR mechanism, the missing data are basically ignorable as compared the loss and goodness of fit to the baseline values. However, for MNAR data, the way to obtain an unbiased estimate of the parameters is to model the missing data. MCMC method is one of popular models to replace missed data with predicted values obtained from a regression model. Our results show this method performs better than unconditional mean imputation.

Better results translate to better decisions. The decisions to impute or not and which imputation method is to implement will impact on the drawn significantly. The imputation method for handling missing data must maintain the existing characteristics of the data at hand and provide the most appropriate solutions.

## **Future Work**

For this case study, there are several research approaches to be implemented in the future for improving the imputation accuracy.

(1) Investigate quality of imputation methods for datasets with more than 50% of missing data;

(2) Use and improve imputation of missing covariates for larger datasets [5];

(3) Can imputation result be improved if multiple imputation results are combined by different imputation algorithms (e.g. tuning parameter λ for the LASSO and adaptive LASSO methods)?

(4) Investigate if more complex imputation methods (decision trees, k-NN, and SVM) perform better than mean imputation and MCMC methods;

(5) Develop methods for data imputation diagnostics and provide more guidance on handling missing data.

## **Ethical considerations**

The imputation of missing data in sensitive or personal datasets should be approached with caution. Imputing data sets implies an assumption about the data or the individuals that the data originated from. While this is a useful tool for handling missing data in order to deploy Machine Learning techniques, it should be made clear that this original assumption was made. The researcher should be sure to inform those using the model or the results that an assumption was made in the data, otherwise unknown bias or factors in the data could be unknowingly taken as truth. There is always a degree of uncertainty in models, and data scientists should be careful to be transparent about imputation methods used that may make models more or less uncertain.

## **References**

1. Dong and Peng, Principle missing data methods for reserachers. Springgerplus. 2013; 2:222. doi: [10.1186/2193-1801-2-222](https://dx.doi.org/10.1186%2F2193-1801-2-222)
2. Pederson et al., Missing data and multiple imputation in clinical epidemiological research. Clin Epidemiol. 2017; 9:157-166. doi: [10.2147/CLEP.S129785](https://dx.doi.org/10.2147%2FCLEP.S129785)
3. The Boston Housing Dataset was obtained from the StatLib archive (<http://lib.stat.cmu.edu/datasets/boston>),
4. published by Harrison, D. and Rubinfeld, D.L. Hedonic prices and the demand for clean air. J. Environ. Economics & Management, vol.5, 81-102, 1978.
5. Dandan Xu. Sequential BART for imputation of missing covariates. [Biostatistics](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4915613/). 2016 Jul; 17(3): 589-602 ttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4915613/

## **Appendix**

## Working codes for case study 5 (Units 9 and 10)

## Missing Data Assignment

# Step 1:

# Using Sklearn get the Boston Housing dataset.

# Fit a linear regressor to the data as a baseline. There is no need to do Cross-Validation. We are exploring the change in results

get\_ipython().run\_line\_magic('matplotlib', 'inline')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import sklearn

import seaborn as sns

# disable chained assignments

pd.options.mode.chained\_assignment = None

from sklearn.datasets import load\_boston

#load dataset

boston = load\_boston()

# Get metadata on Boston data set

print(boston.DESCR)

bos = pd.DataFrame(boston.data)

bos.columns = boston.feature\_names

bos.head()

bos.dropna().corr(method='spearman')

#add predictor to df

bos['PRICE'] = boston.target

bos.head()

from numpy.polynomial.polynomial import polyfit

from sklearn.linear\_model import LinearRegression

#fit a linear regressor model

lm = LinearRegression()

X = bos.iloc[:,:-1] #iloc gives the positional locations, so 'all rows' and 'all but the last column'

Y\_true = bos.iloc[:,-1] # Gives us all rows in the last column

lm.fit(X,Y\_true)

Y\_pred = lm.predict(X)

#plot predicted vs actual

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('LRModel.png', dpi=300)

# What is the loss and what are the goodness of fit parameters? This will be our baseline for comparison

#calculate baseline r2

rSquared\_Base = lm.score(X,Y\_true) # r squared score

rSquared\_Base

# Compare these baseline mse and mean absolute error to the new ones

# when we impute data

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

mseBase = (mean\_squared\_error(Y\_pred,Y\_true))

print (mseBase)

maeBase = (mean\_absolute\_error(Y\_pred,Y\_true))

print (maeBase)

# Step 2: (repeated)

# For select between 1, 5 10, 20, 33, and 50% of your data on a single column (Completely at random), replace the present value with a NAN and then perform an imputation of that value.

#

# In. each case perform a fit with the imputed data and compare the loss and goodness of fit to your baseline.

# 1%

import random

random.random()

# MCAR column 4

limit = .01

for i in range(X.shape[0]):

miss\_1 = random.random()

# miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

#if miss\_2 < limit:

# X.iloc[i,7] = np.nan

#X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

#X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

Y\_pred = lm.predict(X)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('mcar1.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_1 = lm.score(X,Y\_true)

rSquared\_1

mse1Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse1Pct)

mae1Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae1Pct)

# 5%

# MCAR columns 4 and 7

limit = .05

for i in range(X.shape[0]):

miss\_1 = random.random()

#miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

#if miss\_2 < limit:

# X.iloc[i,7] = np.nan

#X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

#X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

Y\_pred = lm.predict(X)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('mcar5.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_5 = lm.score(X,Y\_true)

rSquared\_5

mse5Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse5Pct)

mae5Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae5Pct)

# 10%

# MCAR columns 4 and 7

limit = .10

for i in range(X.shape[0]):

miss\_1 = random.random()

#miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

#if miss\_2 < limit:

# X.iloc[i,7] = np.nan

X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('mcar10.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_10 = lm.score(X,Y\_true)

rSquared\_10

mse10Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse10Pct)

mae10Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae10Pct)

# 20%

# MCAR columns 4 and 7

limit = .20

for i in range(X.shape[0]):

miss\_1 = random.random()

#miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

#if miss\_2 < limit:

# X.iloc[i,7] = np.nan

#X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

#X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('mcar20.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_20 = lm.score(X,Y\_true)

rSquared\_20

mse20Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse20Pct)

mae20Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae20Pct)

# 33%

# MCAR columns 4 and 7

limit = .33

for i in range(X.shape[0]):

miss\_1 = random.random()

#miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

# if miss\_2 < limit:

# X.iloc[i,7] = np.nan

#X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

#X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

# plt.show()

plt.savefig('mcar33.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_33 = lm.score(X,Y\_true)

rSquared\_33

mse33Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse33Pct)

mae33Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae33Pct)

# 50%

# MCAR columns 4 and 7

limit = .50

for i in range(X.shape[0]):

miss\_1 = random.random()

#miss\_2 = random.random()

if miss\_1 < limit:

X.iloc[i,4] = np.nan

#if miss\_2 < limit:

# X.iloc[i,7] = np.nan

#X.shape

# Number of NA's in column 4 (NOX)

X.loc[X.iloc[:,4].isna(),'NOX'].shape

# Substitute NA's with the mean

sub = X['NOX'].mean()

#X

X.loc[X.iloc[:,4].isna(),'NOX'] = sub

# Assign sub the mean value of column 7

#sub = X['DIS'].mean()

#X.loc[X.iloc[:,7].isna(),'DIS'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('mcar50.png', dpi=300)

# r squared with filled in value for the mean

rSquared\_50 = lm.score(X,Y\_true)

rSquared\_50

mse50Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse50Pct)

mae50Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae50Pct)

# Change in r squared with increasing missing data

print("r squared values")

print("0% missing data (Baseline)", rSquared\_Base)

print("1% missing Data", rSquared\_1)

print("5% missing Data", rSquared\_5)

print("10% missing Data",rSquared\_10)

print("20% missing Data",rSquared\_20)

print("33% missing Data",rSquared\_33)

print("50% missing Data",rSquared\_50)

# Change in MSE with increasing missing data

print("MCAR MSE scores compared to baseline")

print("0% missing data (Baseline)", mseBase)

print("1% missing Data", mse1Pct)

print("5% missing Data", mse5Pct)

print("10% missing data", mse10Pct)

print("20% missing Data",mse20Pct)

print("33% missing Data",mse33Pct)

print("50% missing Data",mse50Pct)

# Change in MAE with increasing missing data

print("MCAR Mean Absolute Error scores compared to baseline")

print("0% missing data (Baseline)", maeBase)

print("1% missing Data", mae1Pct)

print("5% missing Data", mae5Pct)

print("10% missing data", mae10Pct)

print("20% missing Data",mae20Pct)

print("33% missing Data",mae33Pct)

print("50% missing Data",mae50Pct)

results=pd.DataFrame()

results["Missing Percent"]=["0%","1%","5%","10%","20%","33%","50%"]

results["R-squared Value"]=[0.7406077428649428,0.7411720289240704,0.7423381569139282,0.7371002976714791,0.733302316393921,0.7303593574373863,0.7292162781155465]

results["MSE Scores"]=[21.897779217687496,21.85014243903083,21.751698431705748,22.19387618416177,22.514500077902657,22.762943366799135,22.859441615800144]

results["MAE Scores"]=[3.2729446379969342,3.2700766912445407,3.264413684584658,3.2774589781104932,3.2893156252721663,3.304949028207812,3.3263520482511675]

results

results.to\_excel('s2results.xlsx')

import matplotlib.pyplot as plt

plt.rc('figure', figsize=(12, 6))

plt.bar(np.arange(len(results["R-squared Value"])),results["R-squared Value"])

plt.xticks(np.arange(len(results["R-squared Value"])),results["Missing Percent"])

plt.ylim(.7,.76)

plt.xlabel("Missing Percentage")

plt.ylabel("R-squared Value")

# plt.title("R-squared Value by Missing Percentage")

plt.rc('xtick', labelsize=24)

plt.rc('ytick', labelsize=24)

plt.rc('axes', labelsize=24)

plt.savefig('rsv.png', dpi=300)

plt.rc('figure', figsize=(12, 6))

plt.bar(np.arange(len(results["R-squared Value"])),results["MAE Scores"])

plt.xticks(np.arange(len(results["R-squared Value"])),results["Missing Percent"])

plt.ylim(3,3.4)

plt.xlabel("Missing Percentage")

plt.ylabel("MAE Score")

# plt.title("MAE Score by Missing Percentage")

plt.rc('xtick', labelsize=24)

plt.rc('ytick', labelsize=24)

plt.rc('axes', labelsize=24)

plt.savefig('mae.png', dpi=300)

plt.rc('figure', figsize=(12, 6))

plt.bar(np.arange(len(results["R-squared Value"])),results["MSE Scores"])

plt.xticks(np.arange(len(results["R-squared Value"])),results["Missing Percent"])

plt.ylim(21,23)

plt.xlabel("Missing Percentage")

plt.ylabel("MSE Score")

# plt.title("MSE Score by Missing Percentage")

plt.rc('xtick', labelsize=24)

plt.rc('ytick', labelsize=24)

plt.rc('axes', labelsize=24)

plt.savefig('mse.png', dpi=300)

# Step 3: Take 2 different columns and create data “Missing at Random” when controlled for a third variable (i.e if Variable Z is > 30, than Variables X, Y are randomly missing). Make runs with 10%, 20% and 30% missing data imputed via your best guess. Repeat your fit and comparisons to the baseline.

# 10%

# MAR columns 9 (TAX) and 10 (PTRATIO) based on CRIM >3.467

limit = .10

for i in range(X.shape[0]):

miss\_1 = random.random()

miss\_2 = random.random()

if miss\_1 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,9] = np.nan

if miss\_2 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,10] = np.nan

X.shape

# impute missing data in the TAX and PTRATIO columns based on the crime rate (CRIM) being greater than 3.647

# Number of NA's in column 9 (NOX)

X.loc[X.iloc[:,9].isna(),'TAX'].shape

# Substitute NA's with the mean

sub = X['TAX'].mean()

X.loc[X.iloc[:,9].isna(),'TAX'] = sub

# Assign sub the mean value of column 10

sub = X['PTRATIO'].mean()

X.loc[X.iloc[:,10].isna(),'PTRATIO'] = sub

# Verify the NaN's are filled in correctly

X

lm.fit(X,Y\_true)

Y\_pred = lm.predict(X)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=18)

plt.rc('ytick', labelsize=18)

plt.rc('axes', labelsize=18)

plt.savefig('s3mcar10.png', dpi=300)

rSquared\_MAR\_10 = lm.score(X,Y\_true)

rSquared\_MAR\_10

mse\_MAR10Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse\_MAR10Pct)

mae\_MAR10Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae\_MAR10Pct)

# 20%

# MAR columns 9 (TAX) and 10 (PTRATIO) based on CRIM >3.467

limit = .20

for i in range(X.shape[0]):

miss\_1 = random.random()

miss\_2 = random.random()

if miss\_1 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,9] = np.nan

if miss\_2 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,10] = np.nan

#X.shape

# Number of NA's in column 9 (NOX)

X.loc[X.iloc[:,9].isna(),'TAX'].shape

# Substitute NA's with the mean

sub = X['TAX'].mean()

#X

X.loc[X.iloc[:,9].isna(),'TAX'] = sub

# Assign sub the mean value of column 10

sub = X['PTRATIO'].mean()

X.loc[X.iloc[:,10].isna(),'PTRATIO'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('s3mcar20.png', dpi=300)

rSquared\_MAR\_20 = lm.score(X,Y\_true)

rSquared\_MAR\_20

mse\_MAR20Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse\_MAR20Pct)

mae\_MAR20Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae\_MAR20Pct)

# 30%

# MAR columns 9 (TAX) and 10 (PTRATIO) based on CRIM >3.467

limit = .30

for i in range(X.shape[0]):

miss\_1 = random.random()

miss\_2 = random.random()

if miss\_1 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,9] = np.nan

if miss\_2 < limit and X.iloc[i,0] > 3.647:

X.iloc[i,10] = np.nan

#X.shape

# Number of NA's in column 9 (NOX)

X.loc[X.iloc[:,9].isna(),'TAX'].shape

# Substitute NA's with the mean

sub = X['TAX'].mean()

#X

X.loc[X.iloc[:,9].isna(),'TAX'] = sub

# Assign sub the mean value of column 10

sub = X['PTRATIO'].mean()

X.loc[X.iloc[:,10].isna(),'PTRATIO'] = sub

#X

from sklearn.linear\_model import LinearRegression

lm.fit(X,Y\_true)

plt.rc('figure', figsize=(6, 6))

Y\_pred = lm.predict(X)

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('s3mcar30.png', dpi=300)

rSquared\_MAR\_30 = lm.score(X,Y\_true)

rSquared\_MAR\_30

mse\_MAR30Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse\_MAR30Pct)

mae\_MAR30Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae\_MAR30Pct)

# look at differences

print("r squared values where crime rate > 3.647 missing at random")

print("0% missing data (Baseline)", rSquared\_Base)

print("10% missing Data",rSquared\_MAR\_10)

print("20% missing Data",rSquared\_MAR\_20)

print("30% missing Data",rSquared\_MAR\_30)

print("MAR MSE scores compared to baseline")

print("0% missing data (Baseline)", mseBase)

print("10% missing data", mse\_MAR10Pct)

print("20% missing Data",mse\_MAR20Pct)

print("30% missing Data",mse\_MAR30Pct)

print("MAR Mean Absolute Error scores compared to baseline")

print("0% missing data (Baseline)", maeBase)

print("10% missing data", mae\_MAR10Pct)

print("20% missing Data",mae\_MAR20Pct)

print("30% missing Data",mae\_MAR30Pct)

step3\_results=pd.DataFrame()

step3\_results["Missing Percent"]=["0%","10%","20%","30%"]

step3\_results["R-squared Value"]=[0.7406,0.7282,0.7255,0.7324]

step3\_results["MSE Score"]=[21.89,22.94,23.16,22.58]

step3\_results["Mae Score"]=[3.27,3.32,3.34,3.30]

step3\_results

step3\_results.to\_excel('s3results.xlsx')

# Step 4: Create a Missing Not at Random pattern in which 25% of the data is missing for a single column. Impute your data, fit the results and compare to a baseline.

# Missing not at random 25 % of data for 1 column

X.loc[X.AGE <= 45.025, 'RM'] = np.nan

X.describe()

# Impute the 25% quantile for RM (5.885500)

# Number of NA's in column

X.loc[X.iloc[:,5].isna(),'RM'].shape

#X.loc[X.iloc[:,9].isna(),'TAX'] = sub

# Assigning the 25% quartile value of RM to NaN's

# Substitute NA's with the mean

sub1 = X['RM'].mean()

sub1

sub2 = sub1 - 5.885500

sub2

sub = sub1 - sub2

sub

# Fill in the NaN's with the sub value of 5.8855

X.loc[X.iloc[:,5].isna(),'RM'] = sub

X

# Fit the regressor

lm.fit(X,Y\_true)

Y\_pred = lm.predict(X)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('s4mcar25.png', dpi=300)

rSquared\_MNAR\_25 = lm.score(X,Y\_true)

rSquared\_MNAR\_25

mse\_MNAR25Pct = (mean\_squared\_error(Y\_pred,Y\_true))

print(mse\_MNAR25Pct)

mae\_MNAR25Pct = (mean\_absolute\_error(Y\_pred,Y\_true))

print(mae\_MNAR25Pct)

print("MNAR situation where RM is dependant on AGE")

print("0% missing data (Baseline)", rSquared\_Base)

print("25% missing data", rSquared\_MNAR\_25)

print("MNAR MSE scores compared to baseline")

print("0% missing data (Baseline)", mseBase)

print("25% missing data", mse\_MNAR25Pct)

print("MNAR mean absolute error scores compared to baseline")

print("0% missing data (Baseline)", maeBase)

print("25% missing data", mae\_MNAR25Pct)

step4\_results=pd.DataFrame()

step4\_results["Missing Percent"]=["0%","25%"]

step4\_results["R-squared Value"]=[0.7406,0.6934]

step4\_results["MSE Score"]=[21.8948,25.8856]

step4\_results["Mae Score"]=[3.2709,3.6428]

step4\_results

s4 = step4\_results.to\_excel('s4results.xlsx')

# Step 5 (Extra Credit) (10 points): Using the MCMC method, and your data from step 4, What is the difference in performance between imputation via ‘guess’ (mean/median, etc) and MCMC.

X.loc[X.AGE <= 45.025, 'RM'] = np.nan

X['PRICE'] = boston.target

s5 = X.to\_excel('s5.xlsx')

file = r's5.xlsx'

s5 = pd.read\_excel(file)

s5\_imp = r's5\_imp.xlsx'

s5\_imp = pd.read\_excel(s5\_imp)

s5\_imp1 = s5\_imp.loc[s5\_imp.\_Imputation\_==1]

s5\_imp1 = s5\_imp1.drop(labels=['\_Imputation\_','A'], axis=1)

L = s5\_imp1.pop('PRICE').values

M = s5\_imp1.astype(float)

# Fit the regressor

lm.fit(M,Y\_true)

L\_pred = lm.predict(M)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, L\_pred)

plt.xlabel("Prices: $L\_i$")

plt.ylabel("Predicted prices: $\hat{L}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('M1.png', dpi=300)

rSquared\_M1 = lm.score(M,Y\_true)

rSquared\_M1

mse\_M1 = (mean\_squared\_error(L\_pred,Y\_true))

print(mse\_M1)

mae\_M1 = (mean\_absolute\_error(L\_pred,Y\_true))

print(mae\_M1)

print("MNAR situation where RM is dependant on AGE")

print("0% missing data (Baseline)", rSquared\_Base)

print("25% missing data", rSquared\_MNAR\_25)

print("M1\_rSquared", rSquared\_M1)

print("MNAR MSE scores compared to baseline")

print("0% missing data (Baseline)", mseBase)

print("25% missing data", mse\_MNAR25Pct)

print("M1\_MSE", mse\_M1)

print("MNAR MAE scores compared to baseline")

print("0% missing data", maeBase)

print("25% missing data", mae\_MNAR25Pct)

print("M1\_MAE", mae\_M1)

X['RM'] = X.RM.fillna(X.RM.mean())

Y = X.pop('PRICE').values

X= X.astype(float)

# Fit the regressor

lm.fit(X,Y\_true)

L\_pred = lm.predict(X)

plt.rc('figure', figsize=(6, 6))

plt.scatter(Y\_true, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

# plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")

# plt.show()

plt.rc('xtick', labelsize=16)

plt.rc('ytick', labelsize=16)

plt.rc('axes', labelsize=16)

plt.savefig('Y1.png', dpi=300)

rSquared\_X1 = lm.score(X,Y\_true)

rSquared\_X1

**# #### SAS code for MI imputation**

#

# proc import datafile='s5.xlsx'

#

# out = s5 DBMS=xlsx REPLACE; GETNAMES=YES;

#

# run;

#

# ods select misspattern;

#

# proc mi data=s5 nimpute=0;

#

# var rm crim zn indus chas;

#

# run;

#

# proc mi data=s5 out=s5\_impute seed=335467 nimpute=5;

#

# var rm crim zn indus chas ;

#

# run;

#

# proc export data=s5\_impute

#

# outfile='s5\_imp.xlsx' dbms=xlsx

#

# replace;

#

# run;

## The end of case study 5 (Units 9 and 10) ##