Effects of Data Imputation – Boston Housing Data

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study #10
03/19/2019

Abstract

Many real-world datasets may contain missing values due to various reasons. Missing data is one of the most common problem we deal with during exploratory analysis and data cleaning. In this case study, we tried to create three patterns of missingness, namely Missing completely at random (MCAR), Missing at random (MAR)and Missing not at random (MNAR). We used Boston housing dataset which contains information about different houses in Boston. There are 506 samples and 13 feature variables in this dataset. We created baseline multiple regression model to predict the value of prices of the house using the given features. Next, we created the many new datasets from the original to simulate MCAR, MAR and MNAR patterns. We used different imputation techniques like Linear Regression, imputation using Mean values, imputation using constant values. Finally, we compared the loss and goodness of fit of the imputation implementation models to our baseline model.

Introduction

The most important decision in any analysis is the selection, aggregation, and/or creation of the proper dataset. Dataset come in all shapes, sizes, completeness, and quality. There's no 'ideal' dataset available for all situations. Because of this, many analysis datasets are built up from already existing datasets and are sometimes augmented with new data gathered specifically for the research topic at hand.

Over the last few decades, the digitalization of society has brought about the proliferation of data gathering opportunities in all aspects of society. From real-time locational tracking, to website traffic analysis, to individual's health/educational records, to shopping habits, to financial transactions are all examples of data that is constantly being gathered and analyzed by individuals/corporations, and governments, often without the subject's knowledge.

Because most data are gathered passively and not actively reviewed, no quality control, there is ample room for errors, omissions, as well as bias can find their way into datasets.

Before Data Scientists (DS) uses datasets in their analyses, DS should thoroughly review datasets gathered from third parties and audit their own data gathering policies/procedures, to ensure the data they are using for the analysis is of the upmost quality, thoroughness, and as free from bias as possible.

There are no perfect datasets, all have flaws, be it collection bias, missing observations/variables, poor quality observation, outdated information, etcetera. A DS must be able to analyze datasets and utilize datasets that are of the highest possible quality while being complete enough to provide a 'full picture' of analysis' objectives.

One of the most common imperfections present in datasets are missing and/or incomplete observations. The reasons behind them are numerous, ranging from non-responses (not answering question regarding household income), to faulty data collection (faulty sensors), to non-applicable situation (sex of spouse when subject is not married) among others. Missing data does not have to be a problem, unless the missing data itself has an inherent meaning. Non-random missing data needs to be identified and handled properly, if not, bias may be introduced into the dataset. Take the situation of a questionnaire that asked a respondent for the age of their spouse. If there is no response to this question, there are two possible scenarios, either the respondent is not married, or they neglected to answer the question. The two scenarios would require different methods for handling the missing data. How missing observation data is handled; elimination of entire observation, imputation of missing values, and/or removal of variable from model can have an impact on the overall analysis.

Many statistical analysis techniques require a complete dataset (no missing variable values in observations). For the analysis, non-complete observations are withheld from the analysis, reducing the number of observations and reducing the power of the results. Depending on the amount of data missing, this may be a logical choice for dealing with missing data.

For datasets that have a substantial number of missing data points, simply eliminating the entire observation may not be option, especially if the number of complete observations is small in comparison to the number in-complete observations or if the missing observations are not due to chance (i.e. are part of a response pattern).

Data imputation is the process of replacing missing data with appropriate substituted values. The notion of 'appropriate values' can be a topic of contention. What values to impute, if any, must be looked at in the context of the dataset itself. Imputation methods tend to put values that will least likely eschew the results of the analysis, many times this is interpreted as the mean, median values of the variable. Even these 'safe' imputation values can lead to unintended consequences, reduced variation in variable, i.e. reduced standard deviations and confidence intervals.

In this study, we will be discussing the topic of data imputation and how imputation of data can affect the outcome of an analysis. For data, we will be utilizing the Boston Housing Data Set that is part of Scikit-Learn library [1] in python. Table 1 describes the dataset [2]. We will explore the effects of imputation of various percentages of observations for a single variable as well as multiple variables. The case study will also look at the effect of imputation to randomly selected observations as well as non-randomly selected observations on analysis results. For all cases, we will fit a linear regression model

Table -1 Boston house prices Data Set Characteristics	
Number of Instances: 506	
Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.	
Attribute Information (in order):	
CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	nitric oxides concentration (parts per 10 million)

RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIS	weighted distances to five Boston employment centers
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per $10,000
PTRATIO	pupil-teacher ratio by town
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT	% lower status of the population
MEDV	Median value of owner-occupied homes in $1000's
Missing Attribute Values: None	
Creator: Harrison, D. and Rubinfeld, D.L.	
This is a copy of UCI ML housing dataset.	
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/	
This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.	
The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic, prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics...', Wiley, 1980. N.B. Various transformations are used in the table on pages 244-261 of the latter.	
The Boston house-price data has been used in many machine learning papers that address regression problems	

Methods

Using Scikit-Learn library, we obtained Boston Housing Dataset and developed a Linear model as a baseline. We will use this baseline to compare results against imputed models below. We will use the following parameters for comparison:

A. Loss; Defined as Mean Squared Error (MSE)
B. Goodness-of-Fit; Defined as R-squared (R^2)

Machine Learning algorithms are designed to minimize or maximize an objective function. The group of functions that are being minimized are called loss functions. Loss functions measure how well a prediction model does in terms of being able to predict an expected outcome.

Mean Squared Error (MSE) is one of the most commonly used regression loss functions and is the primary loss function that we will be using in our analysis. MSE measures the average squared difference between the estimated values and what is estimated. The smaller the mean squared error, the closer you are to finding the line of best fit. MSE is measured in units that are the square of the target variable. MSE can be influenced by outliers, so care should be exercised when using and interpreting MES.

R-squared (R^2) it the proportion of variation in the outcome Y explained by the covariates X. It is a statistical measure of how close the data are to a fitted regression line. R^2 is always between zero and one. An R^2 score of zero indicates that the model explains none of the variability of the response data around its mean. R^2 of 1 indicates that model explains all the variability of the response data around its mean. It is also important to note that before accessing a goodness of fit, like R^2, you should evaluate residual plots. Residual plots can expose bias within a model more effectively than the numeric output of R^2. Thus, if your residuals look good, R^2 can be a meaningful measure of goodness-of-fit.

Analysis

Baseline Model – Boston Housing Dataset

The first task is to measure the baseline results of our dataset and compare our findings to the datasets when we alter the imputations. Table 2 illustrates the Boston Housing Dataset baseline findings using MSE and R-squared, using a Linear regression model.

Table -2 Boston Housing Dataset - Baseline

Model	MSE	R-squared
Baseline – 0% Imputation	21.894	0.741

Imputation Model – 1 Variable, Random

To demonstrate how imputation of missing observation variables affect analysis results, we will replace a certain percentage of values in the Boston dataset with imputed values. We will repeat this process for all selected imputation percentage values chosen. In each case, we will fit a linear model to the imputed dataset and compare the loss and goodness of fit relative to our baseline.
Process:

Percent of values Imputed: 1,5,10,20,33, and 50%
Observations: Chosen at Random
Variable Imputed: ‘LSTAT’ (see below)
Imputation value: Mean of variable

LSTAT is defined as the percent of lower status of the population and will be the attribute that we focus on for our analysis. LSTAT was chosen because of the large t-statistic in the initial analysis.

Figure 1 indicates that as we impute ten-percent of the dataset we begin to see a steady increase in MSE, a less desirable result. When the twenty-percent of the dataset is imputed, MSE takes on a somewhat of an exponential shape.

Like our MSE chart, Figure 2, shows a decline in R^2 with the imputation of five-percent of the dataset. Note that the level of velocity of the decline appears to be less, as compared to MSE chart.

Figure 1 - MSE with imputation of 1,5,10,20,33, and 50% of dataset

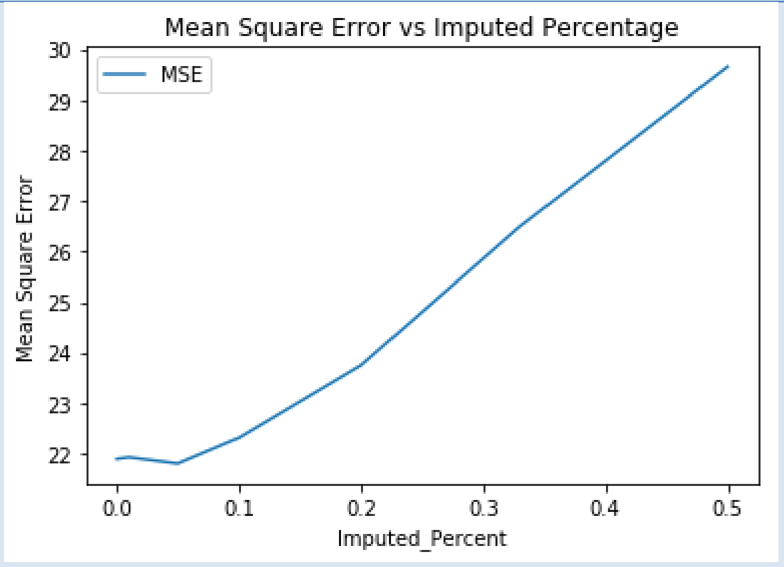


Figure 2 – R² with imputation of 1,5,10,20,33, and 50% of dataset

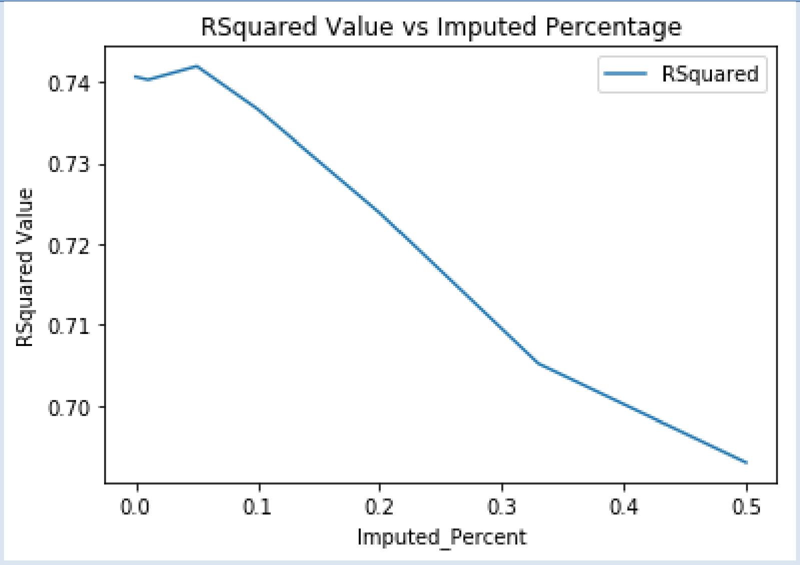


Table 3 provides the results of the analysis in table format

Table -3 Boston Housing Dataset – Imputation / Mean

Imputed_ Percentage (in Decimal)	R^2	Adj R^2	BIC	MSE
0	0.740643	0.73379	3084.78	21.89483
0.01	0.740284	0.733422	3085.479	21.92669
0.05	0.741960	0.735141	3082.204	21.80364

0.10	0.736645	0.729686	3092.52	22.31236
0.20	0.723877	0.716581	3116.476	23.75005
0.33	0.705228	0.697439	3149.546	26.49279
0.50	0.693047	0.684937	3170.034	29.65312

Table 4 and Figures 3 and 4 provides descriptive statistics for LSAT and box plots at various imputation percentages. From the figures, it is apparent that imputation of a variable, using a constant value (mean/median/other) can have a substantial effect on the distribution/variability of that variable. Excessive imputation of variable values can lead to decreased variability and falsely smaller confidence intervals.

Figure 3 - Grid Distribution/Histogram of Imputed Variable vs different Percentage of Imputation

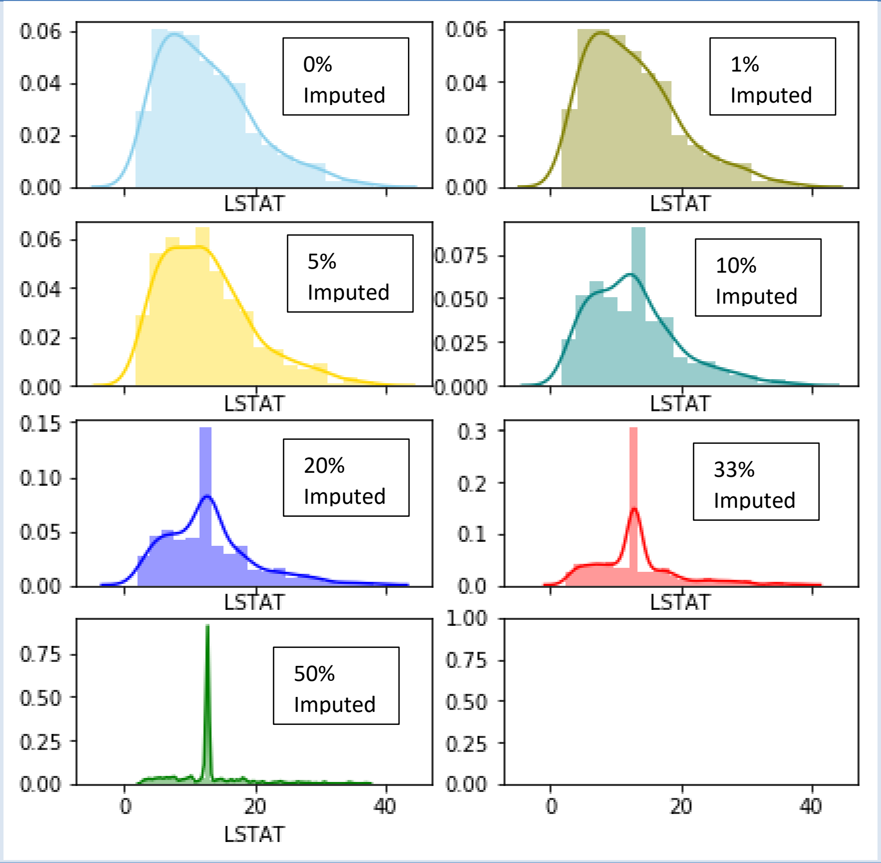


Figure 4 – Boxplot - Imputed Variable vs different Percentage of Imputation

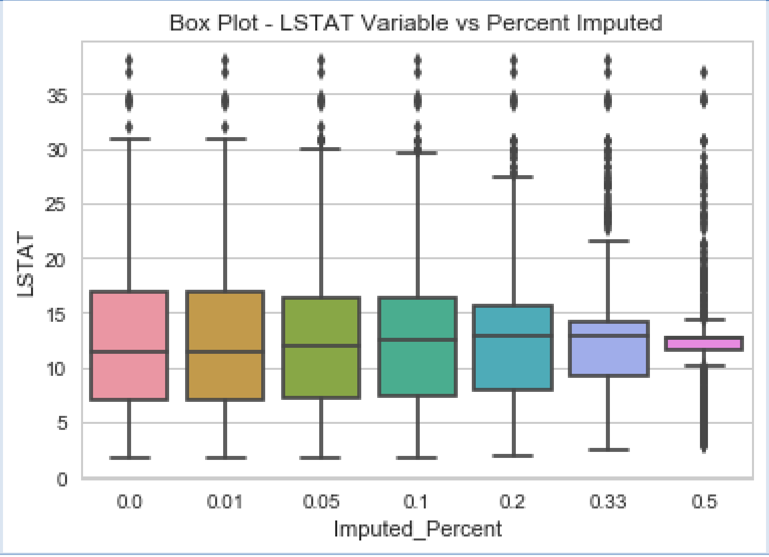


Table -4 Boston Housing Dataset – 'LSTAT' Variable							
Imputed_ Percent	25%	50%	75%	max	mean	min	std
0	6.95	11.36	16.955	37.97	12.65306	1.73	7.141062
0.01	7.0375	11.43	16.86	37.97	12.63653	1.73	7.113619
0.05	7.205	11.975	16.4625	37.97	12.64208	1.73	6.993045
0.10	7.44	12.465	16.2975	37.97	12.67395	1.73	6.800013
0.20	7.89	12.83343	15.7075	37.97	12.83343	1.92	6.530622
0.33	9.2825	12.89209	14.18	37.97	12.89209	2.47	6.029979
0.50	11.6975	12.75107	12.75107	36.98	12.75107	2.87	5.07996

Imputation Model – 2 Variable Random, when controlled for a 3rd variable

In this section, we will impute values for two variables in the Boston dataset at random while controlling for a 3rd variable. We will repeat this process for all selected imputation percentage values chosen. In each case, we will fit a linear model to the imputed dataset and compare the loss and goodness of fit relative to our baseline.

Process:

Percent of values Imputed:	10,20,30%
Observations:	Chosen at Random
Variable Imputed:	'LSTAT', 'PTRATIO'
Controlled Variable:	'B'
Controlled Variable Value:	375.377 (25 percentile of Variable B)
Imputation value:	0

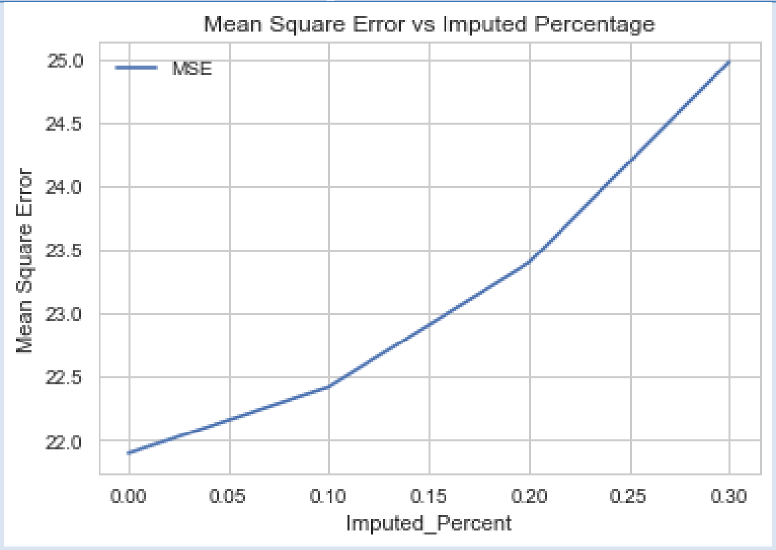
When we perform this analysis, we see a large difference between R-squared and a substantial difference in MSE. For problem this imputation case study, R^2 is at approximately 67 percent when

30 percent of the data is imputed (Table 5 & Fig 5,6). For the single variable imputation study, the R2 is at 67 percent with 50 percent of the data imputed. The difference is most likely due to the value of the imputed observations (0 for this example and 'mean' for the previous example). The difference in MSE is so large that it would have to be measured in multiples to make meaningful comparison. This highlights the effect the choice of the value imputed for missing observations.

Table -5 Boston Housing Dataset – 2 Var Imputed – Controlled for a 3rd variable

Imputed_ Percentage (in Decimal)	R^2	Adj R^2	BIC	MSE
0	0.740643	0.73379	3084.78	21.89483
0.10	0.689204	0.680991	3176.331	49.60313
0.20	0.673673	0.66505	3201.005	85.04977
0.30	0.66994	0.661219	3206.761	123.4882

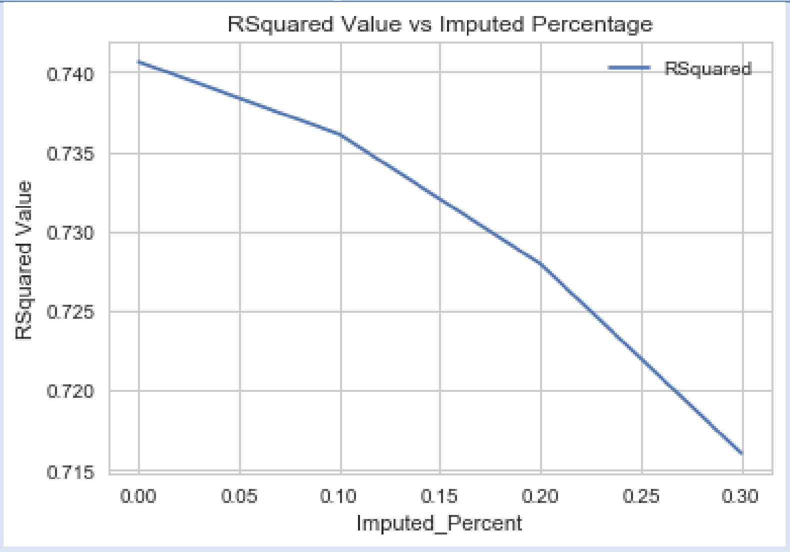
Figure 5 – MSE with imputation of 2-Variables while controlling for 3rd variable; Percent imputation: 10,20,30 of dataset



The MSE chart above (Fig 5) shows a substantial increase of MSE as the percentage of the imputation of the dataset increases. This increase accelerates at a higher velocity than the second problem.

With respect to the R-squared (Fig 6), the difference between problem two and problem three is not nearly as vast, particularly when it is compared to differences in the MSE for the respected problems.

Figure 6 – R^2 with imputation of 2-Variables while controlling for 3rd variable; Percent imputation: 10,20,30 of dataset



Imputation Model – 1 Variable, Not-Random

In this section, we will impute values for a single variable in the Boston dataset with a non-random pattern, to observe how imputation of missing observation variables affect analysis results, we will replace a certain percentage of values in the Boston dataset with imputed values. We will fit a linear model to the imputed dataset and compare the loss and goodness of fit relative to our baseline.

Process:

Percent of values Imputed: 25%
Observations: Not Random (Selected every 4th observation)
Variable Imputed: 'LSTAT'
Imputation value: Mean of variable

With a twenty-five percent missing not at random pattern, there is not a material change in MSE or R-squared (Table 6). Looking back at the first analysis (Table 3), where we imputed the same variable on a random basis (20%-R^2-0.7238 and 33%-R^2-0.7052) and compare it the non-random imputation (25% - R^2-0.7148), the presence of a pattern in the imputation does not appear to affect the results substantially.

Table -6 Boston Housing Dataset – 1 Var Imputed – Non_Random

Imputed_ Percentage (in Decimal)	R^2	Adj R^2	BIC	MSE
0	0.7406	0.73379	3084.78	21.89483
0.25	0.7148	0.70727	3132.83	25.25198

Future Work

Every dataset is different, so it is impossible to predict the effects of incomplete datasets on an analysis. However; it is important to highlight some of the possible effects of working with incomplete datasets. We look to expand the case study in the future to address a few additional areas that were not covered in this study.

Utilization of Alternative Imputation Methods

We have seen that, depending on the percentage of imputed variables, it can have a substantial effect on the statistical distribution of a variable (Fig. 3 & 4). We would like to investigate alternatives to using imputations of the typical mean/median.

One option would be to use Markov Chain Monte Carlo (MCMC) method for imputation. MCMC imputes multiple values for missing observations using probability density, thus attempting to preserve the original distribution of the variable(s) before imputation of missing observations. This has the potential to mitigate some of the negative effects of reduced variability (standard deviation) inherent in imputed variables.

Effect of Goodness of Fit (R^2) vs Importance (T-Statistics) for Imputed Variables

Investigation into the effect of imputation of incomplete observation on goodness of fit (R^2) of analysis results. There is undoubtedly an effect on R^2 using imputed datasets as we have demonstrated in this study. We would like to take this one step further and analyze how the affect changes relative to the T-statistic, investigating the assumption that the more influential (statistically significant) the imputed variable is, the greater the impact on the R2.

Discussions

When working with imputed datasets, there are always consequences. Some good; increased power, utilization of more observations, and some are bad; eschewed results. In some ways, results based on incomplete datasets (imputed or not-imputed) can be interpreted as data manipulation. How a Data Scientist uses the results of analyses based on incomplete datasets and to what degree he/she discloses these manipulations are not often discussed. The end users/consumers of the analyses are often unaware of these dataset manipulations and what impact it may or may not have on the outcome of the analyses. As demonstrated in this case study, substantial variability in results can occur due to imputations and/or elimination of observations. A non-ethical group and/or Data Scientist could use this to manipulate analysis results in their favor, 'coaxing' the results to reinforce their opinions and agendas (i.e. political poll results, corporate sales figures, etc.).

Dataset manipulation should be part of the discussion for all analyses and should be disclosed to all end users, general public, so that they can make an informed decision. The difficulty with disclosure is 2-fold. One, a typical consumer does not have the technical knowledge to understand the ramifications of data manipulation and how it may or may not affect the overall results. Two, the typical consumers' attention spans are not long enough to consider how these acts might affect the results. Society has devolved into an instant gratification society; pretty pictures/graphs and headlines/soundbites. There is often little thought of what goes into to these headlines/pictures. A non-ethical group and/or Data Scientist can use this fact against the consumer, manipulating the consumer's opinion to what the group/scientist wants.

Ethics

There are numerous ethical dilemmas a data scientist must contend with when it comes to dealing with incomplete datasets; can we use incomplete observations in the analysis, should we remove incomplete observations, should we impute missing observation variable and if so, what values should we impute. Unfortunately, there is no one correct response. Decisions regarding datasets must be taken in context of the dataset itself and the intended use of the resulting analysis.

Below are scenarios that may occur if incomplete datasets are not handled properly.

Manipulation of Results-Removal of Observations

Removal of incomplete observations may lead to under representation of classes in a population, especially if the methods of gathering data (i.e. surveys, etc.) are poorly designed and lend themselves to incomplete datasets.
Removal of observations that are not within 'an acceptable range' can also manipulate the results. This can be considered 'cherry-picking' the data that will be included in the analysis. Data Scientist can 'massage' the dataset using various techniques, to obtain the results that they desire and not what the data is portraying.

Manipulation of Results-Imputation of Values

If dataset have incomplete observations and data imputation is selected as the method of handling these observations, the proper selection of the imputed values must be investigated thoroughly, to ensure that unintentional bias is not imparted into the dataset. The use of typical mean/median imputation or some other more advanced technique should be looked at for imputation. One option would be to conduct the analysis using different imputation techniques and observe how the overall analysis changes due to the imputations. If results are little changed, then the technique most appropriate can be used. If the results are substantially changed, then a more in-depth review of the imputation effects should be investigated.

Conclusion

Our research analyzes the Boston Housing dataset and the effects of data imputation. We began by discussing the important decision of the selection, aggregation and creation of a dataset prior to analysis. Most data are gathered passively, not actively reviewed and are susceptible to errors that would cause a material misstatement in the analysis. We will explore the effects of imputation of various percentages of observations for a single variable as well as multiple variables.

Data imputation is the process of replacing missing data with appropriate substituted values. It is paramount for good analysis that the imputation not lead to any unintended consequences that would alter; for example, the standard deviation of confidence intervals. The case study will look at the effect of imputation to randomly selected observations as well as non-randomly selected observations on analysis results. For all cases, we will fit a linear regression model

In order to compare the results of our datasets we begin with a baseline reading of loss and goodness of fit of the Boston Housing dataset. We are measuring loss as Mean Squared Error (MSE) and goodness of fit as R-squared (R^2). These two parameters are widely accepted to measure linear regression problems, we also felt compelled to discuss their respective strengths and weaknesses.

The three models below were compared to our baseline model, which produced an MSE of 21.894 and an R^2 of 0.741

- Imputation Model – 1 Variable, Random
- Imputation Model – 2 Variable Random, when controlled for a 3rd variable
- Imputation Model – 1 Variable, Not-Random

The first model shows that an imputation of a variable using a constant can have a substantial effect on the distribution/variability of a variable. The second model shows a decreasing R^2 with an increasing imputation. While the MSE is parabolic and increases in the order of multiples. The third model, which has twenty five-percent of the data not in a random pattern, does not have a material change to MSE or R^2.

This research clearly shows that missing data and the imputation of random data can have a material impact of your results. The prudent data scientist will take the time to determine what is missing in their datasets and take steps to measure the integrity of the data that they are analyzing. Failure to adhere to this policy could produce inaccurate results and lead to loss of time and money.

References

[1] https://scikit-learn.org/stable/
[2] https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

Appendix A

```
#  MSDS 7333 - Quantifying the World - Case Study #10
#  Imputation
#  Team Members:
#         Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
#  Date: 03/19/2019

import os
import numpy as np
import pandas as pd
from numpy.random import randn
pd.options.display.max_rows = 12
np.set_printoptions(precision=5, suppress=True)
import matplotlib.pyplot as plt
from sklearn import datasets

################################################
#--------   Problem 1   ----------------------
# Using sklearn, get Boston Housing Dataset
# Fit Linear regressor to the data as a baseline. (No need to cross validate)
# Determine Loss and Goodness-of-Fit of model

# https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9
# https://scikit-learn.org/stable/modules/linear_model.html
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html
# https://scikit-learn.org/stable/auto_examples/plot_missing_values.html#sphx-glr-auto-examples-plot-missing-
values-py
# https://bigdata-madesimple.com/how-to-run-linear-regression-in-python-scikit-learn/
# https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
bostonDS = datasets.load_boston()
print(bostonDS.data.shape)
type(bostonDS)
print (bostonDS.DESCR) # Prints characteristics of dataset
bostonDS.feature_names
bostonDS.target

# define the data/predictors as the pre-set feature names
boston = pd.DataFrame(bostonDS.data, columns=bostonDS.feature_names)

# Put the target (housing value -- MEDV) in another DataFrame
target = pd.DataFrame(bostonDS.target, columns=["MEDV"])

# Exploring predictors dataset
print('Dataframe Size:',boston.shape) # (506, 13)
print('Dataframe Null values:',boston.isnull().values.ravel().sum()) # No null values
print ('Number of unique classes:\n',boston.nunique())
```

```
# Base Line - Full Dataset
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
reg = LinearRegression().fit(boston, target) #Fitting linearRegression line to dataset
baselineR2 = reg.score(boston, target) #
print('Baseline R2:',baselineR2) # 0.7406426641094094
reg.coef_ # Regression coefficients:    array([[ -0.10801,   0.04642,   0.02056,   2.68673, -17.76661,   3.80987,
                                        #0.00069,  -1.47557,   0.30605,  -0.01233,  -0.95275,   0.00931,
                                        #-0.52476]])

# Predicting MEDV using LinearRession model
target_pred = reg.predict(boston)

#Calculating Mean_Squared_error
   #Note: This MSE does not include an intercept value in the model
mse = mean_squared_error(target, target_pred)
print('Baseline Mean Square Error(MSE) with intercept:',mse) #21.894831181729206

# https://becominghuman.ai/stats-models-vs-sklearn-for-linear-regression-f19df95ad99b

### Secondary way of calculating LinearModel ###
### Used as a sanity check
import statsmodels.api as sm
#X = boston
X = sm.add_constant(boston) #This model does include an intercept value
regOLS = sm.OLS(target,X).fit()
regOLS.summary()
regOLS._results.rsquared # 0.7406426641094095
regOLS._results.mse_total # 84.58672359409856; Not sure where getting this figure

sm_pred = regOLS.predict(X) # Predicting MEDV using LinearRession model
smMSE = mean_squared_error(target, sm_pred)
print('Mean Square Error with intercept:',smMSE) #24.166099330126492


##################################################
#--------   Problem 2   ----------------------
# Select between 1,5,10,20,33, and 50% of dataset on a single column completely
# random.
# Replace the present value with NaN
# 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.
```

```
######################## Creation of Prob 2 Function #############
########################     get_results()      #############
def get_results(dataset,target,percent,imp_var,strategy):
    # Set percentage of values randomly to NA
    if percent != 0:
        np.random.seed(42)
        indexer = np.sort(np.random.permutation(len(dataset))[len(dataset)-(int(len(dataset)*percent)):])
        dataset_imp = dataset.copy()
        # Chose LSTAT varialbe to randomly set to NaN. Chose LSTAT becuase of the
        # large t-statistic in the original analysis
        dataset_imp[imp_var][indexer] = np.nan

        # https://scikit-
learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html#sklearn.impute.SimpleImputer
        from sklearn.impute import SimpleImputer
        imp_model = SimpleImputer(missing_values=np.nan, strategy=strategy)
        imp_model.fit(dataset_imp)

        imp_model = pd.DataFrame(imp_model.transform(dataset_imp),columns=dataset.columns,
index=dataset.index)
    else:
        imp_model = dataset.copy()

    X = sm.add_constant(imp_model)
    regOLS_imp = sm.OLS(target,X).fit()
    #regOLS_imp.summary()
    #regOLS_imp._results.rsquared
    target_pred = reg.predict(imp_model)
    mse = mean_squared_error(target, target_pred) # Not sure about MSE values
    return((percent,regOLS_imp._results.rsquared,regOLS_imp._results.rsquared_adj,
            regOLS_imp._results.bic,mse,imp_model))

#----- End of get_results() -------------


###  Input variables for Problem #2 function  ##########

results_boston = pd.DataFrame([])
per_num = [0.0,0.01,0.05,0.10,0.20,0.33,0.50]
imp_var = ['LSTAT'] #Choose from CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT
strategy = 'mean' #Choose from 'mean', 'median', 'constant'
#fill_value = 0 # If you choose 'constant', need to select value to fill NaNs


for i in range(len(per_num)):
    results_boston[i] = np.array(get_results(boston,target,per_num[i],imp_var[0],strategy))
```

```
results_boston = results_boston.T
results_boston.columns=['Imputed_Percent','RSquared','AdjRSquared','BIC','MSE','Model']

print(results_boston.loc[:,results_boston.columns !='Model']) # Prints results of analysis for each percent of
Imputed variables

# Plot of R2 values analysis results vs different Percentage of Imputation
results_boston.plot('Imputed_Percent','RSquared')
plt.ylabel('RSquared Value')
plt.title('RSquared Value vs Imputed Percentage')

# Plot of MSE values analysis results vs different Percentage of Imputation
results_boston.plot('Imputed_Percent','MSE')
plt.ylabel('Mean Square Error')
plt.title('Mean Square Error vs Imputed Percentage')



# Grid Distribution/Histogram of Imputed variable vs different Percentage of Imputation
import seaborn as sns
f, axes = plt.subplots(4, 2, figsize=(7, 7), sharex=True)
sns.distplot( results_boston.Model[0]['LSTAT'] , color="skyblue",ax=axes[0, 0])
sns.distplot( results_boston.Model[1]['LSTAT'] , color="olive", ax=axes[0, 1])
sns.distplot( results_boston.Model[2]['LSTAT'] , color="gold", ax=axes[1, 0])
sns.distplot( results_boston.Model[3]['LSTAT'] , color="teal", ax=axes[1, 1])
sns.distplot( results_boston.Model[4]['LSTAT'] , color="blue", ax=axes[2, 0])
sns.distplot( results_boston.Model[5]['LSTAT'] , color="red", ax=axes[2, 1])
sns.distplot( results_boston.Model[6]['LSTAT'] , color="green", ax=axes[3, 0])


imputed_dataframe = pd.DataFrame([])
plot_result = pd.DataFrame([])
for k in range(0,len(results_boston)):
   imputed_dataframe = results_boston.Model[k]
   imputed_dataframe['Imputed_Percent'] = per_num[k]
   plot_result = plot_result.append(imputed_dataframe, ignore_index=True)

#  Boxplot of Imputed Variable vs different Percentage of Imputation
sns.set(style="whitegrid")
ax = sns.boxplot(x="Imputed_Percent", y="LSTAT", data=plot_result)
ax.set_title('Box Plot - LSTAT Variable vs Percent Imputed')


summaryResult = pd.DataFrame([])
for k in range(0,len(results_boston)):
    tempResults = results_boston.Model[k]['LSTAT'].describe()
    tempResults['Imputed_Percent']=per_num[k]
    summaryResult = summaryResult.append(tempResults, ignore_index=True)

print(summaryResult)



##################################################
#--------   Problem 3   ----------------------

# Take 2 different columns and create data 'Missing at Random' when controlled
# for a third variable. (i.e. if Variable Z is >30, then Variable 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.

# The code below is for reference only.  Was used to figure out how to
# create function get_resultsP3().  Can follow through step-by-step
# to gain understanding of what function does.

Prob3 = boston.copy()

Prob3Desc = Prob3.describe()
Prob3Desc
Prob3Desc.loc['25%'] # Identifying 1st Quartile
Prob3Desc.loc['25%']['B']
SubProb3 = Prob3[Prob3.B > 375.377]
#SubProb3Count = SubProb3.B.count()
#SubProb3['tempIndex'] = pd.Series(range(0,len(SubProb3)),index=SubProb3.index)
#SubProb3.iloc[18]

Prob3SubIndex = Prob3[Prob3.B > 375.377].index
lenIndex = len(Prob3[Prob3.B > 375.377])
SubProb3 = Prob3[Prob3.B > 375.377] # Note Index #18 is the first one missing
Prob3SubIndex[18]
np.random.seed(42)
percent = 0.30
numbVariables = 2
impvar = ['LSTAT','PTRATIO']
Prob3['LSTAT'][4]
Prob3[impvar[0]][4]

indexer1 = np.sort(np.random.permutation(Prob3SubIndex)[lenIndex-(int(lenIndex*percent)):])
Prob3[impvar[0]][indexer1] = np.nan

np.random.seed(39)
```

```
indexer2 = np.sort(np.random.permutation(Prob3SubIndex)[lenIndex-(int(lenIndex*percent)):])
Prob3[impvar[1]][indexer2] = np.nan

# https://scikit-
learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html#sklearn.impute.SimpleImputer
from sklearn.impute import SimpleImputer
imp_meanP3 = SimpleImputer(missing_values=np.nan, strategy='constant',fill_value=0)
imp_meanP3.fit(Prob3)

imp_meanP3 = pd.DataFrame(imp_meanP3.transform(Prob3),columns=Prob3.columns, index=Prob3.index)

X = sm.add_constant(imp_meanP3)
regOLS_impP3 = sm.OLS(target,X).fit()
regOLS_impP3.summary()
regOLS_impP3._results.rsquared



######################## Creation of Prob 3 Function #############
########################     get_resultsP3()       #############

def get_resultsP3(dataset,target,percent,ran_seed,cond_var,cond_var_val,imp_var,strategy,fill_value=None):
  datasetDesc = dataset.describe()

  if percent != 0:
    Prob3SubIndex = dataset[dataset[cond_var] > datasetDesc.loc[cond_var_val][cond_var]].index
    lenIndex = len(dataset[dataset[cond_var] > datasetDesc.loc[cond_var_val][cond_var]])
    from sklearn.impute import SimpleImputer
    for i in range(len(imp_var)):
      np.random.seed(ran_seed[i])
      indexer = np.sort(np.random.permutation(Prob3SubIndex)[lenIndex-(int(lenIndex*percent)):])
      Prob3[imp_var[i]][indexer] = np.nan

      imp_modelP3 = SimpleImputer(missing_values=np.nan, strategy=strategy,fill_value=fill_value)
      imp_modelP3.fit(dataset)

      imp_modelP3 = pd.DataFrame(imp_modelP3.transform(dataset),columns=dataset.columns,
index=dataset.index)
  else:
    imp_modelP3 = dataset.copy()

  X = sm.add_constant(imp_modelP3)
  regOLS_impP3 = sm.OLS(target,X).fit()
  regOLS_impP3.summary()
  regOLS_impP3._results.rsquared
  target_pred = reg.predict(imp_modelP3)
```

```
    mse = mean_squared_error(target, target_pred) # Not sure about MSE values
    return((percent,regOLS_impP3._results.rsquared,regOLS_impP3._results.rsquared_adj,
          regOLS_impP3._results.bic,mse))

#--- End of get_resultsP3 ----------------


### Input variables for Problem #3 function ##########

per_numP3 = [0.0,0.10,0.20, 0.30]  # Percentage of imputed values in dataset
ran_seed = [42,39] #Random seed numbers for permutation
cond_var = 'B'   #Choose from CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT
cond_var_val ='25%'   # Choose from 25%, 50%, 75%, mean,min
imp_var = ['LSTAT','PTRATIO'] #Choose from CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT
strategy = 'constant' #Choose from 'mean', 'median', 'constant'
fill_value = 0 # If you choose 'constant', need to select value to fill NaNs


### Execution of Problem #3 function #####

Prob3 = boston.copy() # Making copy of original dataset
#Prob3.describe()
results_bostonP3 = pd.DataFrame([]) # Initalizing results dataframe
for j in range(len(per_numP3)):
    results_bostonP3[j] =
np.array(get_resultsP3(Prob3,target,per_numP3[j],ran_seed,cond_var,cond_var_val,imp_var,strategy,fill_value))

results_bostonP3 = results_bostonP3.T
results_bostonP3.columns=['Imputed_Percent','RSquared','AdjRSquared','BIC','MSE']

print(results_bostonP3) # Prints results of analysis for each percent of Imputed variables

results_bostonP3.plot('Imputed_Percent','RSquared')
plt.ylabel('RSquared Value')
plt.title('RSquared Value vs Imputed Percentage')

# Plot of MSE values analysis results vs different Percentage of Imputation
results_bostonP3.plot('Imputed_Percent','MSE')
plt.ylabel('Mean Square Error')
plt.title('Mean Square Error vs Imputed Percentage')



#---------  End of Problem #3 ------------------------------------------
```

```
##################################################
#--------   Problem 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.


######################## Creation of Prob 4 Function #############
#########################      get_resultsP4()       #############
def get_resultsP4(dataset,target,percent,imp_var,strategy,fill_value=None):
  # Set 25% percent of values to NA, selecting every 4th entry
  dataset_imp = dataset.copy()
  indexer = dataset_imp.iloc[::4, :].index
  dataset_imp[imp_var][indexer] = np.nan

  from sklearn.impute import SimpleImputer
  imp_model = SimpleImputer(missing_values=np.nan, strategy=strategy,fill_value=fill_value)
  imp_model.fit(dataset_imp)

  imp_model = pd.DataFrame(imp_model.transform(dataset_imp),columns=dataset.columns,
index=dataset.index)

  X = sm.add_constant(imp_model)
  regOLS_imp = sm.OLS(target,X).fit()
  #regOLS_imp.summary()
  #regOLS_imp._results.rsquared
  target_pred = reg.predict(imp_model)
  mse = mean_squared_error(target, target_pred) # Not sure about MSE values
  return((percent,regOLS_imp._results.rsquared,regOLS_imp._results.rsquared_adj,
      regOLS_imp._results.bic,mse))

#--------- End of get_resultsP4() -----------------------------------------


###  Input variables for Problem #4 function  ##########

imp_var = ['LSTAT'] #Choose from CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT
strategy = 'mean' #Choose from 'mean', 'median', 'constant'
fill_value = 0 # If you choose 'constant', need to select value to fill NaNs
per_numP4 = [0.25]  # Percentage of imputed values in dataset; set at 25% for this exercise

###  Execution of Problem #4 function #####

Prob4 = boston.copy() # Making copy of original dataset
```

```
#Prob4.describe()
results_bostonP4 = pd.DataFrame([]) # Initalizing results dataframe
for j in range(len(imp_var)):
   results_bostonP4[j] = np.array(get_resultsP4(Prob4,target,per_numP4[j],imp_var[j],strategy,fill_value))

results_bostonP4 = results_bostonP4.T
results_bostonP4.columns=['PercentImputed','RSquared','AdjRSquared','BIC','MSE']

print(results_bostonP4) # Prints results of analysis for each percent of Imputed variables


#---------  End of Problem #4 -------------------------------------------
```