Part 1

A. Research question: What causes readmissions for patients in certain states of the country?
B. The data set consists of 50 variables:

Variable	Description	Data Type	Example
CaseOrder	Unique identifier for each case	Int64; Quantitative	CaseOrder =8
Customer_id	Unique customer identifier	Int64; Quantitative	Customer_id= T490287
Interaction, UID	Unique identifier for each patient's transactions	Object; Qualitative	Interaction= 542acef9-365b-4cf4-a92e-21bdf28830d4 UID= 11242d7514ece3946f167db79410254b
City	Patient city	Object; Qualitative	City = Casselberry
State	Patient state	Object; Qualitative	State= FL
County	Patient county	Object; Qualitative	County= Seminole
Zip	Patient zip code	Int64; Qualitative	Zip code= 30153
Lat, Lng	Latitude & longitude of patient's address	Float64; Qualitative	Lat= 34.3496 Lng= -86.72508
Population	Population within a mile radius of patient	Int64; Quantitative	Population= 2951
Area	Area type (rural, urban, suburban)	Object; Qualitative	Area= urban
TimeZone	Time zone of the patient based on address provided during sign up	Object; Qualitative	TimeZone= America/Chicago
Job	Job of the patient	Object; Qualitative	Job= Chief Executive Officer
Children	Patient's number of children	Float64; Quantitative	Children= 2
Age	Patient's age	Float64; Quantitative	Age= 86
Education	Highest earned level of education of the patient	Object; Qualitative	Education= Bachelor's Degree
Employment	Patient employment status	Object; Qualitative	Employment= Retired
Income	Patient annual income	Float64; Quantitative	Income= 50017.99
Marital	Patient martial status	Object; Qualitative	Marital= Married
Gender	Patient Gender	Object; Qualitative	Gender= Female
ReAdmis	If the patient was readmitted within a month of release (yes, no)	Object; Qualitative	ReAdmis= No

Jasmine Cooper
D206
Data Cleaning
July 1, 2023
Western Governor's University

VitD_levels	Patient's vitamin D levels (ng/mL)	Float64; Quantitative	VitD_levels = 14.7918355
Doc_visits	Number of times the primary physician visited the patient during the first hospital visit	Int64; Quantitative	Doc_visits=5
Full_meals_eaten	Number of full meals the patient ate while hospitalized	Int64; Quantitative	Full_meals_eaten= 0
VitD_supp	Number of times that vitamin D supplements were given to the patient	Int64; Quantitative	VitD_supp = 0
Soft_drink	If the patient drank 3 or more sodas per day	Object; Qualitative	Soft_drink= Yes
Initial_admin	Reason for patient initial admission (emergency admission, elective admission, observation)	Object; Qualitative	Initial_admin= Observation Admission
Complication_risk	Patient level of complication risk (high, medium, low)	Object; Qualitative	Complication_risk= Medium
HighBlood	Patient high blood pressure status	Object; Qualitative	HighBlood= Yes
Stroke	If the patient had a stroke	Object; Qualitative	Stroke= No
Overweight	If the patient is overweight	Float64; Qualitative	Overweight= 0
Arthritis	If the patient has arthritis	Object; Qualitative	Arthritis= Yes
Diabetes	If the patient has diabetes	Object; Qualitative	Diabetes= No
Hyperlipidemia	If the patient has hyperlipidemia	Object; Qualitative	Hyperlipidemia= No
BackPain	If the patient has chronic back pain	Object; Qualitative	Backpain = No
Anxiety	If the patient has an anxiety disorder	Float64; Qualitative	Anxiety= 1
Allergic_rhinitis	If the patient has allergic rhinitis	Object; Qualitative	Allergic_rhinitis= No
Reflux_esophagitis	If the patient has reflux esophagitis	Object; Qualitative	Reflux_esophagitis= No
Asthma	If the patient has asthma	Object; Qualitative	Asthma= Yes
Services	Primary service patient received (blood work,	Object; Qualitative	Services= Blood Work

	intravenous, CT scan, MRI)		
Initial_days	Number of days the patient stayed in the hospital during the initial visit	Float64; Quantitative	Initial_days = 9.058210335
TotalCharge	Amount charged to the patient daily	Float64; Quantitative	TotalCharge= 2774.08992
Additional_charges	Average amount charged to the patient for miscellaneous medical services	Float64; Quantitative	Additional_charges= 6930.572138
Item1	Survey responses for timely admission	Int64; Quantitative	Item1= 5
Item2	Survey responses for timely treatment	Int64; Quantitative	Item2=4
Item3	Survey responses timely visits	Int64; Quantitative	Item3= 5
Item4	Survey responses reliability	Int64; Quantitative	Item4= 2
Item5	Survey responses for options available	Int64; Quantitative	Item5= 4
Item6	Survey responses for hours of treatment	Int64; Quantitative	Item6= 3
Item7	Survey responses for patient services regarding staff	Integer; Quantitative	Item7= 3
Item8	Survey responses for active listening from doctor	Integer; Quantitative	Item8= 1

Note: These are the original data types prior to any changes and/or data cleaning

Part 2

C. Data Cleaning Plan

1. These are the techniques and specific steps that I completed to investigate quality issues.
    i. Read CSV file into Jupyter notebook using Python pandas and convert the CSV file into a Data Frame.
    ii. Use the methods, 'describe()', 'info()', and 'dtypes()' to observe the summary statistics, data types and nullity, and solely the data types of the data frame respectively.
    iii. Identify if there is duplicate information using the 'duplicated()' method.
    iv. Convert columns from data type object to category using the 'astype()' method and use the 'dtypes()' method to check conversion is completed properly:
        - Area, Employment, Martial, Gender, Initial_admin, Complication_risk, Services, ReAdmis, Soft_drink, HighBlood, Stroke, Arthritis, Diabetes, Hyperlipidemia, BackPain,

Allergic_rhinitis, Reflux_esophagitis, Asthma, Overweight, Anxiety

v. Remap columns Overweight and Anxiety from 0 and 1 to no and yes respectively.
vi. Visualize missing data across the entire data frame using missingno and matplotlib library packages, and 'bar()' and 'show()' methods.
vii. Investigate missing data using methods 'isna()' and 'sum()'. Also, create complete and missing data frames to analyze summary statistics using the 'describe()' method to analyze missingness and potential outliers across all quantitative variables. Lastly complete percentage missing calculations and heatmap visualization to identify if missingness occurs in certain subsets of variables.
viii. Determine missingness type(s) if applicable.

2. The above methods were used to assess the quality of the data for several reasons as listed below, respectively:
    i. Data was read and converted into a data frame to support Python functions, libraries, and methods.
    ii. These methods were used to provide an overall idea of the data set.
    iii. This method was used to determine if duplicate information was included in the data set to decide whether duplicated information would affect the overall data quality.
    iv. Converting columns from data type object to category was necessary because there are only a few different values to be selected from each variable and it helps to save memory. Also, it ensures these variables are treated as categorical variables in other Python libraries and visualizations. Lastly, converting columns from float64 data type to category was necessary because numbers that aren't meant to be represented as numerical data, rather binary data, can skew overall summary statistics.
    v. Remapping column options was necessary to create data consistency and prevent skewing summary statistics.
    vi. Creating a bar chart allows me to visualize all missing data across numerical and non-numerical variables and organize missing data in ascending order.
    vii. The first 2 methods used together allow me to quickly see which variables have missing values, and how many missing values per variable are present. Isolating complete and missing values, by creating 2 separate data frames, allows me to evaluate summary statistics, and determine if variables that present missingness are related (e.g., are income values missing due to employment status?). Lastly, calculating missing percentages per variable helps to analyze the amount of missingness across the dataset. This can also help us determine how to clean data and/or impute missing values.
    viii. Determining missingness types helps us determine how to impute missing values and/or clean the entire data set.
3. Python is the selected programming language because the data set did not require detailed visualizations over periods of time, and it did not require intense statistical technique to clean data. Also, since our main goal is to solely clean the data set, Python is a much more general, robust program that does not require specific libraries and syntax, which makes it easier and more time effective to achieve our goal. Furthermore,

pandas, numpy, missingno, and matplotlib libraries were all used to transform the data properly and investigate missing values. In comparison to R, I would have had to use specific syntax for each library installed, and this could have unnecessarily increased the complexity of my investigation. Lastly, I used the fancy impute package to conduct the K-Nearest Neighbor imputation technique for numerical missing values since those variables with missing values seemed to be missing completely at random, and there was little to no variation between percentage missingness. Similarly, I used sklearn to utilize the Simple Imputation mode technique to impute non numerical missing values since those variables with missing values did not seem to be related based on the heat map visualization.

4. Code:

Detect missing values:

```
medical_raw_data.isna().sum()

!pip install missingno
import missingno as msno
import matplotlib.pyplot as plt
msno.bar(medical_raw_data)
plt.show()

column_order = medical_raw_data.isnull().sum().sort_values().index
msno.bar(medical_raw_data[column_order])
plt.show()
```

*change variable names as needed based on code to detect outliers (e.g., population = 0 does not make sense and prompts the idea these records probably are missing values instead)

```
medical_raw_data.Population[medical_raw_data.Population == 0]
```

*change variable names as needed

-determine if missing variable 1 values are related to missing variable 2 and missing variable 3

```
missing_income = medical_raw_data[medical_raw_data['Income'].isnull()]
grouped = missing_income.groupby(['Employment', pd.cut(missing_income['Age'],
bins=[0, 18, 30, 50, np.inf])])
missing_count = grouped.size()
percentage_missing = missing_count / medical_raw_data.groupby(['Employment',
pd.cut(medical_raw_data['Age'], bins=[0, 18, 30, 50, np.inf])]).size() * 100
print(percentage_missing)

msno.heatmap(medical_raw_data)
```

Detect outliers:

*change variable names as needed
missing= medical_raw_data[medical_raw_data['Income'].isna()]
complete= medical_raw_data[~medical_raw_data['Income'].isna()]
pd.set_option('display.max_columns', None)
missing.describe()
complete.describe()

Detect duplicates:
duplicates = medical_raw_data.duplicated()
medical_raw_data[duplicates]

Convert columns (change variable names as needed):
medical_raw_data["Area"] = medical_raw_data["Area"].astype('category')

Remapping (change variable names as needed):
mapping = {0: 'no', 1:'yes'}
medical_raw_data['Anxiety'] = medical_raw_data['Anxiety'].map(mapping)

(“Handling Missing Values”, n.d.)
(“Analyze the amount of missingness”, n.d.)
(“Uniqueness constraints”, n.d.)
(“Importing flat files using pandas”, n.d.)
(“Data type constraints”, n.d.)
(“Completeness”, n.d.)
(“Handling Missing Values”, n.d.)
(“Finding patterns in missing data”, n.d.)

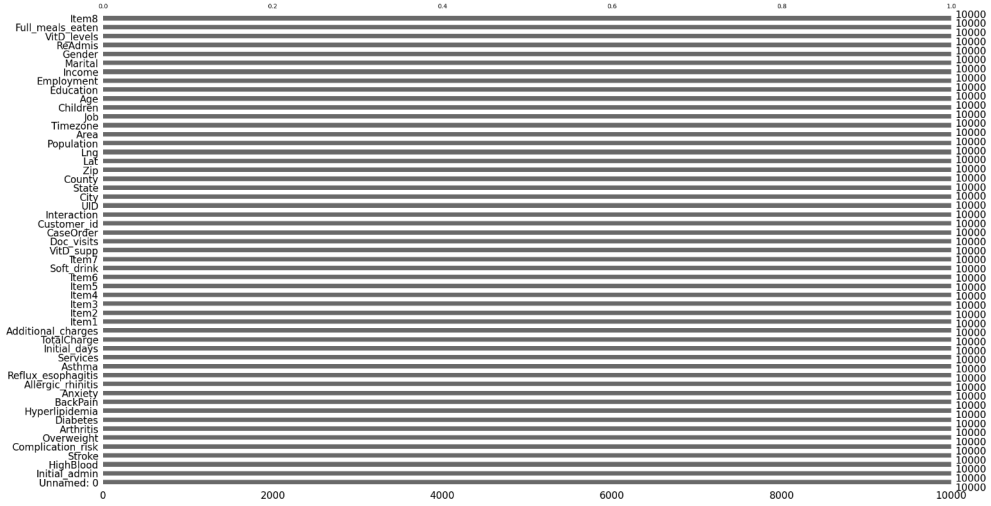
Part 3

D. Data Cleaning
    1. Upon completion of my investigation, I did not find any duplicated information, but I did find one outlier that potentially could skew statistical information. The outlier I discovered was the Population variable. While analyzing the summary statistics for the missing data, I noticed that the minimum for the Population variable was consistently zero. Since population size is not typically zero, these records are more than likely missing values. Lastly, I found that the following variables had missing values, and these were the number of values missing in each variable:
        i. Children- 2588 (numerical)
        ii. Soft_drink- 2467 (non-numerical)

iii. Income- 2464 (numerical)
iv. Age- 2414 (numerical)
v. Initial_days- 1056 (numerical)
vi. Anxiety- 984 (non-numerical)
vii. Overweight- 982 (numerical)

2. To treat the population outlier, I replaced all population records with values equal to zero with NaN values utilizing the numpy library and the 'nan' and 'isnan' methods. I did this because a population size equal to zero is highly unlikely, statistical summaries could be skewed, and changing from zero to NaN allows Python to recognize missing values properly before imputation. The missing Population values were imputed using the K-Nearest Neighbor technique along with the other numerical values. The numerical missing values seem to be missing at random because the missing values could probably be predicted by other observed values in the data set. For examples, missing income values could be relative to age and/or employment status (e.g., if a patient is a student they are likely to be within the age range 18-22, and not list their income value since it might not be applicable). To treat numerical missing values, I decided to use the K-Nearest Neighbor imputation technique because as previously mentioned missing values can likely be predicted based on similar observed variables and values. The non-numerical variables' values seem to be missing completely at random because there does not seem to be systematic relationship between neither observed nor unobserved values that create missingness. For example, whether a patient consumed 3 or more soft drinks in a day cannot necessarily be predicted by other variables in the data set. Likewise, whether a patient is overweight and/or has anxiety could be based on a wide variety of variables and can significantly vary from patient to patient. To impute non-numerical variables' values, I decided to use the Simple Imputation mode technique, and used frequency to fill in missing data.
3. The overall outcome of cleaning the data step is a data set with no missing values. As mentioned in D2, converting the records where the population was equal to zero to NaN helped to correctly identify that missing variable's values prior to imputation. The K-Nearest Neighbor Imputation technique was implemented to fill numerical missing variables's values, and the Simple Imputation technique was used to impute non-numerical variables' values. Below is a bar chart that portrays no missing values in the data set.

```
[86]: #data clean
import matplotlib.pyplot as plt
column_order = medical_raw_data.isnull().sum().sort_values().index
msno.bar(medical_raw_data[column_order])
plt.show()
```



4. Code:

Identify records with population equal to zero and convert to nan:

```
medical_raw_data.Population[medical_raw_data.Population == 0]
medical_raw_data.Population[medical_raw_data.Population == 0].count()
medical_raw_data.loc[medical_raw_data.Population == 0, 'Population'] = np.nan
medical_raw_data.Population[np.isnan(medical_raw_data.Population)]
```

Use the fancy impute package to use the KNN technique for numerical variables:

```
pip install fancyimpute
from fancyimpute import KNN

columns_to_impute = ['Children', 'Income', 'Age', 'Initial_days', 'Population']

knn_imputer = KNN()

columns_to_impute_indices = [medical_raw_data.columns.get_loc(col) for col in
columns_to_impute]

imputed_values = knn_imputer.fit_transform(medical_raw_data.iloc[:,
columns_to_impute_indices])

medical_raw_data.iloc[:, columns_to_impute_indices] = imputed_values

print(medical_raw_data.head())
```

Use sklearn package to use the Simple Imputation technique:

```
from sklearn.impute import SimpleImputer

cat_variables = ['Soft_drink', 'Anxiety', 'Overweight']

imputer = SimpleImputer(strategy='most_frequent')

medical_raw_data[cat_variables] =
imputer.fit_transform(medical_raw_data[cat_variables])
```

("Mean, median, & mode imputations", n.d.)
("Imputing using fancyimpute", n.d.)

5. Extracted CSV file from Python called 'new cleaned data.csv'
6. A major limitation of this data cleaning process was utilizing the Simple Imputation mode technique to impute missing values for non-numerical variables. This is because those variables appeared to be missing completely at random, and frankly would not contribute to answering the research question that I proposed and/or some of the issues stated in the assignment prompt. Furthermore, simple imputation ignores patterns and relationships between values, and treats records as stand-alone entities, which can create some level of bias in the data, however, since variables appeared to be missing completely at random using this technique should not have created many biased issues. Another limitation of this data cleaning process was to impute the numerical values with the default k parameter. This could be considered a limitation because there did not seem to be enough missing numerical variables with high correlation. If the k value was set too small then the imputations might have been too similar, and if the k value was set too high then the imputations might have been unrelated. Therefore, in this case, it seemed best to impute with defaults.
7. These limitations could affect the analysis of the research question if one decided to use non-numerical variables to help answer the research question, and it could potentially skew the output of the analysis if one were to count too many 'yes' or 'no' binary values and use it to draw conclusions. Furthermore, for numerical variables, the limitations could affect grouped information if imputed values are too similar, and lead to inaccurate information about certain groups. For example, if one were to group and draw conclusions based on age or income amount being a potential reason for higher or lower readmission rates.

E. PCA

1. I chose 16 numerical variables for principal component analysis. The output of the principal component loading matrix is below.

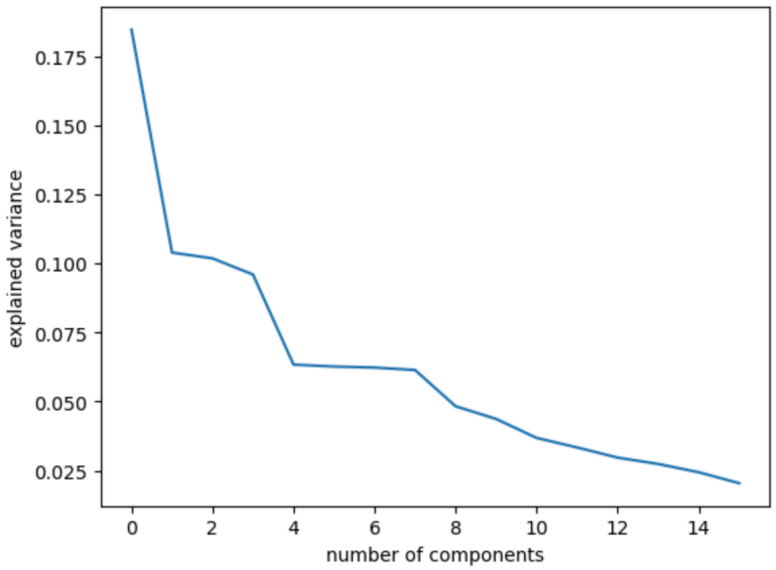
[72]:

	PC1	PC2	PC3	PC4	PC5	PC6
Population	0.008838	-0.003354	0.026083	0.045623	0.320200	0.771391
Children	0.004309	0.000573	0.042664	-0.043642	-0.222391	0.498226
Age	0.000961	0.241814	0.420257	-0.511742	0.011022	-0.012492
Income	-0.006964	-0.017167	-0.058274	0.002629	0.573817	-0.369049
VitD_supp	-0.005095	0.024935	0.042073	0.025371	0.718267	0.106010
Initial_days	-0.019627	0.325116	0.405565	0.476838	-0.018559	-0.029755
TotalCharge	-0.014510	0.344201	0.402480	0.463259	-0.023556	-0.044013
Additional_charges	0.004358	0.237510	0.395629	-0.533597	0.035917	-0.030620
Item1	0.454846	-0.240539	0.171003	0.026663	0.001395	-0.008983
Item2	0.428525	-0.233016	0.175222	0.031719	0.008743	0.006917
Item3	0.395292	-0.236104	0.175158	0.030926	0.005721	-0.033283
Item4	0.152159	0.450540	-0.325881	-0.044094	0.002301	0.043042
Item5	-0.189921	-0.477604	0.333128	0.045881	0.001111	-0.006957
Item6	0.410315	0.134936	-0.094407	-0.001745	-0.001534	0.002050
Item7	0.356597	0.153821	-0.082974	-0.006743	-0.015206	0.019492
Item8	0.312566	0.138386	-0.095533	-0.001825	0.013375	-0.043049

[72]:

PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16
0.330639	-0.433988	0.006775	-0.023032	-0.005645	-0.022493	-0.026549	0.006334	0.005321	-0.004689
0.829182	0.089794	0.043687	-0.004418	0.015007	-0.015577	-0.003892	0.020574	-0.010940	-0.003024
0.015784	-0.028787	-0.000853	-0.029665	-0.005568	0.026036	-0.099723	-0.553862	0.428128	-0.009107
0.444127	-0.575482	-0.043149	0.007847	-0.006855	0.012535	-0.006114	-0.010745	0.006337	-0.006571
0.052600	0.681661	0.033716	0.029587	0.013735	-0.003211	0.008761	-0.002756	0.008305	0.000426
0.023612	-0.026579	-0.014843	-0.011181	-0.003673	-0.011156	0.037017	-0.426473	-0.557773	-0.068695
0.022817	-0.029153	0.001962	-0.006256	0.007069	0.016902	-0.031195	0.452225	0.540101	0.057313
0.035712	-0.038132	-0.000348	0.013810	0.011850	-0.017755	0.095388	0.542884	-0.439137	0.007444
0.000323	0.006662	-0.096131	-0.075116	-0.010343	0.083754	0.184608	0.043777	0.070952	-0.801337
0.006515	-0.005398	-0.147411	-0.133567	-0.061352	0.093355	0.620499	-0.074017	0.040546	0.534765
0.010907	0.029127	-0.207327	-0.211106	-0.238920	-0.425090	-0.620390	0.044769	-0.076098	0.190370
0.015309	0.041897	-0.364480	-0.364417	-0.387558	0.483247	-0.102366	0.024786	-0.035489	-0.011995
0.002022	-0.004147	0.125620	0.054620	-0.133165	0.698058	-0.291852	0.034373	-0.064274	0.091664
0.006320	-0.005177	-0.050468	0.062609	0.795992	0.271418	-0.270049	-0.000260	-0.049553	0.123789
0.002007	-0.015211	0.040108	0.844721	-0.337870	0.072419	-0.063056	-0.018571	-0.002262	0.050127
0.008589	-0.030977	0.876404	-0.275634	-0.150414	0.039042	-0.035152	-0.000897	-0.021614	0.032342

2. To determine the number of principal components to retain, I applied the elbow rule by observing a sharp change in slope of the explained variance curve. By using the elbow rule, I used the elbow point on the scree plot to determine the number of principal components where additional components would contribute less to the overall variance explained. Observing the scree plot suggests that the elbow point occurs at 8 principal components; meaning the components before the elbow contribute to the variance explained, and the components after the elbow point contribute less to the overall variance. Retaining 8 principal components will capture a significant amount of the variance in the data while reducing dimensionality. Please see scree plot below.



3. The organization would benefit from the use of PCA because it would reduce the number of dimensions analyzed, identify the most important variables in the data set, and enhance the performance of machine learning models. Specifically, when training and modeling the data to create machine learning models that identify what patients are most likely to be readmitted, PCA would help to decrease unnecessary complexity when attempting to draw meaningful conclusions for decision making. As seen in the

loadings output, it can be noted that some variables have strong correlation between itself and the principal component. For example, Initial_days, which is the amount of days a patient stayed in the hospital during their first visit, has a strong correlation to PC4. And VitD_supp, which is the amount of vitamin D supplements the patient received during their initial visit has a strong correlation to PC5. These insights suggest how the variability of each variable can contribute to the patterns observed. Furthermore, it allows an organization to determine a variable's influence on a principal component that can be used to deepen data analysis, and potentially make predictions based on the relationships between these variables.

Part 4: Citations

F. Panopto video recording
G. Citations for Code

DataCamp. (n.d.). Importing flat files using pandas [Video file]. Retrieved from https://campus.datacamp.com/courses/introduction-to-importing-data-in-python/introduction-and-flat-files-1?ex=15

DataCamp. (n.d.). Uniqueness constraints [Video file]. Retrieved from https://app.datacamp.com/learn/courses/cleaning-data-in-python

DataCamp. (n.d.). Categorical Variables [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/text-and-categorical-data-problems?ex=4

DataCamp. (n.d.). Data type constraints [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/common-data-problems-1?ex=1

DataCamp. (n.d.). Completeness [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/advanced-data-problems-3?ex=8

DataCamp. (n.d.). Handling Missing Values [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/the-problem-with-missing-data?ex=5

DataCamp. (n.d.). Analyze the amount of missingness [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/the-problem-with-missing-data?ex=9

DataCamp. (n.d.). Is the data missing at random? [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/does-missingness-have-a-pattern?ex=1

DataCamp. (n.d.). Finding patterns in missing data [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/does-missingness-have-a-pattern?ex=4

DataCamp. (n.d.). Mean, median, & mode imputations [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/imputation-techniques?ex=1

DataCamp. (n.d.). Imputing using fancyimpute [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/advanced-imputation-techniques?ex=1

DataCamp. (n.d.). Principal Component selection [Video file]. Retrieved from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-extraction?ex=13

Lesson7: How to Perform PCA in Python. Retrived from https://cgp-oex.wgu.edu/courses/course-v1:WGUx+OEX0026+v02/courseware/1f468770545f494fa657b4dc0ed3762f/2b5f23c5dad64357b352728993788677/6?activate_block_id=block-v1%3AWGUx%2BOEX0026%2Bv02%2Btype%40vertical%2Bblock%403f0422d47d8b4a3eaec25de3bced8bf8

H. Citations for Content

DataCamp. (n.d.). Importing flat files using pandas [Video file]. Retrieved from https://campus.datacamp.com/courses/introduction-to-importing-data-in-python/introduction-and-flat-files-1?ex=15

DataCamp. (n.d.). Uniqueness constraints [Video file]. Retrieved from https://app.datacamp.com/learn/courses/cleaning-data-in-python

DataCamp. (n.d.). Data type constraints [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/common-data-problems-1?ex=1

DataCamp. (n.d.). Completeness [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/advanced-data-problems-3?ex=8

DataCamp. (n.d.). Handling Missing Values [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/the-problem-with-missing-data?ex=5

DataCamp. (n.d.). Analyze the amount of missingness [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/the-problem-with-missing-data?ex=9

DataCamp. (n.d.). Is the data missing at random? [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/does-missingness-have-a-pattern?ex=1

DataCamp. (n.d.). Finding patterns in missing data [Video file]. Retrieved from

https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/does-missingness-have-a-pattern?ex=4

DataCamp. (n.d.). Mean, median, & mode imputations [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/imputation-techniques?ex=1

DataCamp. (n.d.). Imputing using fancyimpute [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/advanced-imputation-techniques?ex=1

DataCamp. (n.d.). Principal Component Analysis [Video file]. Retrieved from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-extraction?ex=5

DataCamp. (n.d.). PCA Applications [Video file]. Retrieved from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-extraction?ex=9

DataCamp. (n.d.). Principal Component selection [Video file]. Retrieved from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-extraction?ex=13

Lesson7: How to Perform PCA in Python. Retrived from https://cgp-oex.wgu.edu/courses/course-v1:WGUx+OEX0026+v02/courseware/1f468770545f494fa657b4dc0ed3762f/2b5f23c5dad64357b352728993788677/6?activate_block_id=block-v1%3AWGUx%2BOEX0026%2Bv02%2Btype%40vertical%2Bblock%403f0422d47d8b4a3eaec25de3bced8bf8