|  |
| --- |
| D208 |
| Linear Regression Modeling |
| Task 1 |

|  |
| --- |
| Shantel Johnson  3-25-2023 |

Contents

[Section A 3](#_Toc130665641)

[Part 1: Research Question 3](#_Toc130665642)

[This information could prove practical for the allocation of government funds to areas with higher incidences of factors that increase the additional charges billed. 3](#_Toc130665643)

[Part 2: Goals 3](#_Toc130665644)

[Section B 3](#_Toc130665645)

[Part 1: Assumptions of Linear Regression 3](#_Toc130665646)

[Part 2: Python for Data Analysis 3](#_Toc130665647)

[Part 3: Justification of Methods 3](#_Toc130665648)

[Section C 4](#_Toc130665649)

[Part 1: Data Cleaning 4](#_Toc130665650)

[Duplication 4](#_Toc130665651)

[Missing Values 4](#_Toc130665652)

[Outliers 4](#_Toc130665653)

[Part 2: Data Descriptions 4](#_Toc130665654)

[Part 3: Visualization 4](#_Toc130665655)

[Part 4: Data Transformation 4](#_Toc130665656)

[Ordinal Encoding 4](#_Toc130665657)

[Nominal Encoding 5](#_Toc130665658)

[Part 5: Output 5](#_Toc130665659)

[Section D 5](#_Toc130665660)

[Part 1: Initial Model 5](#_Toc130665661)

[Part 2: Feature Selection 5](#_Toc130665662)

[Part 3: Reduced Model 5](#_Toc130665663)

[Section E 5](#_Toc130665664)

[Part 1: Model Comparison 5](#_Toc130665665)

[Part 2: Output 6](#_Toc130665666)

[Part 3: Code 6](#_Toc130665667)

[Section F 6](#_Toc130665668)

[Part 1: Results 6](#_Toc130665669)

[Regression Equation 6](#_Toc130665670)

[Interpretation 6](#_Toc130665671)

[Statistical Significance 6](#_Toc130665672)

[Practical Significance 7](#_Toc130665673)

[Limitations 7](#_Toc130665674)

[Part 2: Next Steps 7](#_Toc130665675)

[Section G 7](#_Toc130665676)

[Part 1: Data Preparation Code 7](#_Toc130665677)

[Data Cleaning 10](#_Toc130665678)

[Issue Handling 13](#_Toc130665679)

[Data Descriptions 13](#_Toc130665680)

[Visualizations 14](#_Toc130665681)

[Data Transformation 41](#_Toc130665682)

[Part 2: Linear Regression Modeling Code 45](#_Toc130665683)

[Linear Regression Model 1: All Features 45](#_Toc130665684)

[Linear Regression Model 2: Reduced Features 48](#_Toc130665685)

[Section H 53](#_Toc130665686)

[Section I 53](#_Toc130665687)

[Part 1: Web Sources 53](#_Toc130665688)

[Part 2: References 54](#_Toc130665689)

# Section A

## Part 1: Research Question

The dataset for this project comprises observations on patient demographics, length of hospital stay, and associated charges. As the project's data analyst, my responsibility is to answer the research question: Which demographic factors influence the additional charges billed during a patient's hospital stay?

## This information could prove practical for the allocation of government funds to areas with higher incidences of factors that increase the additional charges billed.

## Part 2: Goals

This project aims to identify factors (i.e., variables) in the dataset that contribute to the additional charges that are billed and establish the correlation between them. To achieve these goals, the dataset requires preparation for statistical analysis. Specifically, I will use linear regression models to analyze the data.

# Section B

## Part 1: Assumptions of Linear Regression

Four assumptions to be aware of when using a linear regression model are:

* Predictive purpose: With the exception of time series forecasting, linear regression models should not be used to extrapolate outside of the range of data.
* Multicollinearity: Multicollinearity can lead to dubious regression coefficients. As such, predictor variables should be removed from the model until multicollinearity is gone.
* Outliers: Outliers can distort the results of regression modeling. For this reason, predictor variables should have normal distributions.
* Heteroskedasticity: Heteroskedasticity (the lack of constant residual variance across the range of predicted values) may suggest that the model is incomplete and requires more predictor variables (Bruce, Bruce, & Gedeck, 2019).

## Part 2: Python for Data Analysis

For linear regression in particular, Python is useful because 1) of the availability of packages that are used for data preprocessing and exploratory data analysis (pandas and scipy) and 2) of the availability of packages that are used for building linear regression models (scikit-learn and statsmodels).

## Part 3: Justification of Methods

Linear regression quantifies the nature of the relationship between one or more predictor variables and the target variable (Bruce, Bruce, & Gedeck, 2019). As defined in Section A, Part 1 the research question seeks to understand what patient demographic factors (the predictor variables) influence additional charges (the target variable). Based on this, we can see that linear regression is an appropriate technique for analyzing the research question.

# Section C

## Part 1: Data Cleaning

Data cleaning was performed to identify and remediate the common data quality issues of duplication, missing values, and outliers. Refer to Section G, Part 1 to view the associated data cleaning code.

### Duplication

The pandas.DataFrame.duplicated() function was applied on the dataset to return duplicated rows. No issues were detected after using this method.

### Missing Values

The pandas.DataFrame.isnull.sum() function was applied on the dataset to return the number of missing values found in each variable. No issues were detected after using this method.

### Outliers

The scipy.stats.zscore() function was applied on the dataset to return the number of outliers found in each numeric variable. The following data issues were identified:

|  |  |
| --- | --- |
| **Variable** | **Number of Outliers** |
| Population | 218 |
| Children | 202 |
| Income | 143 |
| VitD\_levels | 24 |
| Doc\_visits | 8 |
| Full\_meals\_eaten | 33 |
| vitD\_supp | 70 |

As noted in Section B, Part 1, outliers impact the results of regression modeling. For this reason, records containing outliers were dropped from the dataset using the pandas.DataFrame.drop() function.

## Part 2: Data Descriptions

Summary statistics were calculated for the dependent and all independent variables. Refer to Section G, Part 1 for more information.

## Part 3: Visualization

Distributions of the dependent and independent variables were depicted using univariate and bivariate plots. To view these visualizations, refer to Section G, Part 1.

## Part 4: Data Transformation

Data transformation was performed on categorical variables in preparation for linear regression modeling (refer to Section G, Part 1).

### Ordinal Encoding

Statistical models should be presented with a numeric encoding of ordered categories that represent linear ordering (Kuhn & Johnson, 2019). If a categorical variable consisted strictly of binary values (e.g., yes/no, true false), then these variables were transformed using the pandas.DataFrame.replace() function. Otherwise, they were transformed into numeric values using the sklearn.preprocessing.OrdinalEncoder() function.

### Nominal Encoding

According to Kuhn and Johnson, the most basic approach for re-expressing categorical variables as numeric data is to create C-1 indicator variables, where C represents the possible values of the predictor. (2019). Categorical variables of the nominal type were transformed into numeric values using the pandas.get\_dummies() function.

## Part 5: Output

To view the output of the data preparation tasks performed in Section C, refer to the attached file, medical\_prepared\_data.csv.

# Section D

## Part 1: Initial Model

An initial linear regression model was constructed using the independent variables identified in Section C, Part 2. Refer to Section G, Part 2 to view the initial model.

## Part 2: Feature Selection

Feature selection is important in predictive modeling because the predictive performance of the model often decreases as the number of uninformative predictors increases. In addition, removing predictors can reduce the cost of data acquisition and can increase throughput of the software used to make predictions (Kuhn & Johnson, 2019). For this reason, the number of predictor variables was reduced using feature selection.

The sklearn.feature\_selection.SequentialFeatureSelection() function was used to reduce the number of predictor variables included in the model. The sequential feature selector (SFS) uses forward selection to form a feature subset in a greedy fashion (Pedregosa, et. al, 2011). The SFS falls under the wrapper methodology for feature selection. According to Kuhn and Johnson, wrapper methods use iterative search procedures that repeatedly supply predictor subsets to the model (2019).

## Part 3: Reduced Model

A reduced linear regression model was constructed using a subset of the independent variables. Refer to Section G, Part 2 to view the reduced model.

# Section E

## Part 1: Model Comparison

The following table compares the initial and reduced linear regression models:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Predictor Variables** | **R-Squared** | **RMSE** |
| Initial | 32 | 0.939 | 1619.68 |
| Reduced | 2 | 0.936 | 1662.42 |

While the initial model includes significantly more predictor variables than the reduced model, both models have approximately the same R-Squared value of about 0.94. This means that the 32 predictor variables in the initial model explain the same proportion of variance as the 2 predictor variables in the reduced model. The initial model also has a root mean squared error (RSMSE) that is only slightly lower than that of the reduced model.

## Part 2: Output

Residual plots were generated for the reduced model. To view these, along with other calculations such as the residual standard error, refer to Section G, Part 2.

## Part 3: Code

To review the executable code used to build the initial and reduced linear regression models, refer to the attached file, ‘D208 Task1 Model.csv’.

# Section F

## Part 1: Results

### Regression Equation

The regression equation for the reduced linear regression model is given below:

### Interpretation

For each predictor variable, the coefficient sign indicates whether the target variable increases or decreases as the predictor variable increases. Both Age and HighBlood have positive coefficients, meaning that as they increase, Additional\_charges increases. The coefficient values represent the average change in the target variable given a one-unit increase in the independent variable. As age increases by one unit, Additional\_charges increases by 225.59 units (dollars). HighBlood has a binary value of either 0 or 1 (corresponding with no or yes, respectively). If a patient has high blood pressure, Additional\_charges increases by 8645.61 units; otherwise, Additional\_charges increase by 0 units.

For example, to predict the additional charges for a 26-year-old patient without high blood pressure, plug the variables into the above regression equation:

Based on this result, $3191.62 in additional charges will be billed to the patient.

### Statistical Significance

According to the summary output (refer to Section G, Part 2) the reduced linear regression model has a Prob(F-statistic) of 0, indicating that the model is statistically significant.

### Practical Significance

The reduced model has practical significance for two reasons. First, it identified and quantified the patient demographic factors that influence the additional charges billed, namely age and high blood pressure. Second, the geographic distribution of these factors has been studied by public entities such as the US Centers for Disease Control and the US Census Bureau, which means that the relevant data is publicly available for further investigation, as summarized in Part 2 of this section.

### Limitations

The reason for hospital admission has been omitted from the dataset. It is possible that this factor, if included in the model as a predictor variable, could increase the R-Squared value and reduce the RMSE value, thus leading to improved predictions of additional charges billed.

## Part 2: Next Steps

As a next step, I would first recommend adding the reason for hospital admission as a feature in the dataset. From there, we can re-perform the SFS indicated in Section D, Part 2 to determine whether reason for hospital admission improves the model. Once that step is complete, my second recommendation is to perform a follow-up analysis to identify geographic locations (city, state, or zip code) of older-aged populations that display increased rates of high blood pressure. This follow-up analysis would be pivotal to allocating government funds to areas of need, as indicated in Section A, Part 1.

# Section G

## Part 1: Data Preparation Code

In [1]:

# read data into pandas DataFrame

# import pandas for data manipulation

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

import pandas as pd

df = pd.read\_csv('medical dataset/medical\_clean.csv')

In [2]:

# reduce df to relevant features

df = df[['Population',

'Children',

'Age',

'Income',

'Marital',

'Gender',

'VitD\_levels',

'Doc\_visits',

'Full\_meals\_eaten',

'vitD\_supp',

'Soft\_drink',

'Initial\_admin',

'HighBlood',

'Stroke',

'Complication\_risk',

'Overweight',

'Arthritis',

'Diabetes',

'Hyperlipidemia',

'BackPain',

'Anxiety',

'Allergic\_rhinitis',

'Reflux\_esophagitis',

'Asthma',

'Services',

'Additional\_charges'

]]

In [3]:

# get first 3 rows of data

df.head(3).transpose()

Out[3]:

|  | **0** | **1** | **2** |
| --- | --- | --- | --- |
| **Population** | 2951 | 11303 | 17125 |
| **Children** | 1 | 3 | 3 |
| **Age** | 53 | 51 | 53 |
| **Income** | 86575.93 | 46805.99 | 14370.14 |
| **Marital** | Divorced | Married | Widowed |
| **Gender** | Male | Female | Female |
| **VitD\_levels** | 19.141466 | 18.940352 | 18.057507 |
| **Doc\_visits** | 6 | 4 | 4 |
| **Full\_meals\_eaten** | 0 | 2 | 1 |
| **vitD\_supp** | 0 | 1 | 0 |
| **Soft\_drink** | No | No | No |
| **Initial\_admin** | Emergency Admission | Emergency Admission | Elective Admission |
| **HighBlood** | Yes | Yes | Yes |
| **Stroke** | No | No | No |
| **Complication\_risk** | Medium | High | Medium |
| **Overweight** | No | Yes | Yes |
| **Arthritis** | Yes | No | No |
| **Diabetes** | Yes | No | Yes |
| **Hyperlipidemia** | No | No | No |
| **BackPain** | Yes | No | No |
| **Anxiety** | Yes | No | No |
| **Allergic\_rhinitis** | Yes | No | No |
| **Reflux\_esophagitis** | No | Yes | No |
| **Asthma** | Yes | No | No |
| **Services** | Blood Work | Intravenous | Blood Work |
| **Additional\_charges** | 17939.40342 | 17612.99812 | 17505.19246 |

In [4]:

# get information on df

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 26 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Population 10000 non-null int64

1 Children 10000 non-null int64

2 Age 10000 non-null int64

3 Income 10000 non-null float64

4 Marital 10000 non-null object

5 Gender 10000 non-null object

6 VitD\_levels 10000 non-null float64

7 Doc\_visits 10000 non-null int64

8 Full\_meals\_eaten 10000 non-null int64

9 vitD\_supp 10000 non-null int64

10 Soft\_drink 10000 non-null object

11 Initial\_admin 10000 non-null object

12 HighBlood 10000 non-null object

13 Stroke 10000 non-null object

14 Complication\_risk 10000 non-null object

15 Overweight 10000 non-null object

16 Arthritis 10000 non-null object

17 Diabetes 10000 non-null object

18 Hyperlipidemia 10000 non-null object

19 BackPain 10000 non-null object

20 Anxiety 10000 non-null object

21 Allergic\_rhinitis 10000 non-null object

22 Reflux\_esophagitis 10000 non-null object

23 Asthma 10000 non-null object

24 Services 10000 non-null object

25 Additional\_charges 10000 non-null float64

dtypes: float64(3), int64(6), object(17)

memory usage: 2.0+ MB

In [5]:

# get number of unique values for each feature

df.nunique()

Out[5]:

Population 5951

Children 11

Age 72

Income 9993

Marital 5

Gender 3

VitD\_levels 9976

Doc\_visits 9

Full\_meals\_eaten 8

vitD\_supp 6

Soft\_drink 2

Initial\_admin 3

HighBlood 2

Stroke 2

Complication\_risk 3

Overweight 2

Arthritis 2

Diabetes 2

Hyperlipidemia 2

BackPain 2

Anxiety 2

Allergic\_rhinitis 2

Reflux\_esophagitis 2

Asthma 2

Services 4

Additional\_charges 9418

dtype: int64

### Data Cleaning[¶](" \l "Data-Cleaning)

#### Issue Identification[¶](#Issue-Identification)

##### Duplicates[¶](#Duplicates)

In [6]:

# check duplication across entire row

df.duplicated().sum()

Out[6]:

0

##### Missing Values[¶](#Missing-Values)

In [7]:

# check for missing values

df.isnull().sum()

Out[7]:

Population 0

Children 0

Age 0

Income 0

Marital 0

Gender 0

VitD\_levels 0

Doc\_visits 0

Full\_meals\_eaten 0

vitD\_supp 0

Soft\_drink 0

Initial\_admin 0

HighBlood 0

Stroke 0

Complication\_risk 0

Overweight 0

Arthritis 0

Diabetes 0

Hyperlipidemia 0

BackPain 0

Anxiety 0

Allergic\_rhinitis 0

Reflux\_esophagitis 0

Asthma 0

Services 0

Additional\_charges 0

dtype: int64

##### Outliers[¶](#Outliers)

In [8]:

# create new DataFrame with numeric features

df\_num = df.select\_dtypes(include='number')

df\_num.head()

Out[8]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Additional\_charges** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2951 | 1 | 53 | 86575.93 | 19.141466 | 6 | 0 | 0 | 17939.403420 |
| **1** | 11303 | 3 | 51 | 46805.99 | 18.940352 | 4 | 2 | 1 | 17612.998120 |
| **2** | 17125 | 3 | 53 | 14370.14 | 18.057507 | 4 | 1 | 0 | 17505.192460 |
| **3** | 2162 | 0 | 78 | 39741.49 | 16.576858 | 4 | 1 | 0 | 12993.437350 |
| **4** | 5287 | 1 | 22 | 1209.56 | 17.439069 | 5 | 0 | 2 | 3716.525786 |

In [9]:

# import scipy to calculate z-scores

# Title: scipy/scipy: Scipy

# Author: Gommers, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7655153

from scipy import stats

# replace values in df\_num with z-scores

df\_num = stats.zscore(df\_num)

df\_num.head()

Out[9]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Additional\_charges** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -0.473168 | -0.507129 | -0.024795 | 1.615914 | 0.583603 | 0.944647 | -0.993387 | -0.634713 | 0.765005 |
| **1** | 0.090242 | 0.417277 | -0.121706 | 0.221443 | 0.483901 | -0.967981 | 0.990609 | 0.956445 | 0.715114 |
| **2** | 0.482983 | 0.417277 | -0.024795 | -0.915870 | 0.046227 | -0.967981 | -0.001389 | -0.634713 | 0.698635 |
| **3** | -0.526393 | -0.969332 | 1.186592 | -0.026263 | -0.687811 | -0.967981 | -0.001389 | -0.634713 | 0.009004 |
| **4** | -0.315586 | -0.507129 | -1.526914 | -1.377325 | -0.260366 | -0.011667 | -0.993387 | 2.547602 | -1.408991 |

In [10]:

# return number of outliers found in each column of df\_num

df\_num[df\_num.abs() >= 3].count()

Out[10]:

Population 218

Children 202

Age 0

Income 143

VitD\_levels 24

Doc\_visits 8

Full\_meals\_eaten 33

vitD\_supp 70

Additional\_charges 0

dtype: int64

### Issue Handling[¶](" \l "Issue-Handling)

#### Outliers[¶](#Outliers)

In [11]:

# drop records containing outliers

for col in df\_num.columns:

df.drop(df\_num.loc[df\_num[col].abs() >= 3].index, inplace=True, errors='ignore')

df.shape

Out[11]:

(9331, 26)

### Data Descriptions[¶](" \l "Data-Descriptions)

In [12]:

# get summary statistics for numeric features

df.describe()

Out[12]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Additional\_charges** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 |
| **mean** | 8701.586754 | 1.942343 | 53.510342 | 38895.729734 | 17.961006 | 5.014146 | 0.989176 | 0.377880 | 12924.292040 |
| **std** | 12063.299403 | 1.898241 | 20.623584 | 25378.151394 | 1.987202 | 1.039526 | 0.978105 | 0.585608 | 6547.313250 |
| **min** | 0.000000 | 0.000000 | 18.000000 | 154.080000 | 12.038470 | 2.000000 | 0.000000 | 0.000000 | 3125.703000 |
| **25%** | 683.000000 | 0.000000 | 36.000000 | 19414.450000 | 16.639135 | 4.000000 | 0.000000 | 0.000000 | 7965.541909 |
| **50%** | 2640.000000 | 1.000000 | 53.000000 | 33261.940000 | 17.944850 | 5.000000 | 1.000000 | 0.000000 | 11553.164690 |
| **75%** | 12463.000000 | 3.000000 | 71.000000 | 53213.445000 | 19.331745 | 6.000000 | 2.000000 | 1.000000 | 15614.084690 |
| **max** | 54413.000000 | 8.000000 | 89.000000 | 125977.370000 | 23.941426 | 8.000000 | 4.000000 | 2.000000 | 30566.070000 |

In [13]:

# get summary statistics for categorical features

df.describe(exclude='number')

Out[13]:

|  | **Marital** | **Gender** | **Soft\_drink** | **Initial\_admin** | **HighBlood** | **Stroke** | **Complication\_risk** | **Overweight** | **Arthritis** | **Diabetes** | **Hyperlipidemia** | **BackPain** | **Anxiety** | **Allergic\_rhinitis** | **Reflux\_esophagitis** | **Asthma** | **Services** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 |
| **unique** | 5 | 3 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 |
| **top** | Widowed | Female | No | Emergency Admission | No | No | Medium | Yes | No | No | No | No | No | No | No | No | Blood Work |
| **freq** | 1921 | 4677 | 6923 | 4724 | 5525 | 7466 | 4224 | 6629 | 5992 | 6774 | 6195 | 5515 | 6315 | 5654 | 5514 | 6623 | 4917 |

### Visualizations[¶](" \l "Visualizations)

#### Univariate Plots[¶](#Univariate-Plots)

In [14]:

# import matplotlib and seaborn for visualizations

# Title: matplotlib/matplotlib

# Author: Caswell, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7697899

# Title: seaborn: statistical data visualization

# Author: Michael Waskom

# Date: 2021

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.4645478

import matplotlib.pyplot as plt

import seaborn as sns

In [15]:

# create new DataFrames numerical and categorical data

numerical = df.select\_dtypes(include='number')

categorical = df.select\_dtypes(exclude='number')

In [16]:

# create all barcharts

for col in categorical.columns:

plt.figure()

sns.barplot(data=categorical, x=categorical[col].value\_counts(), y=categorical[col].value\_counts().index)

plt.title(col)

plt.show()

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated

In [17]:

# create all histograms

for col in numerical.columns:

plt.figure()

sns.histplot(data=numerical[col])

plt.title(col)

plt.show()

Chart, histogram

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

#### Bivariate Plots[¶](#Bivariate-Plots)

In [18]:

# create all catplots

for col in categorical.columns:

plt.figure()

sns.catplot(data=df, x='Additional\_charges', y=col, kind='box')

plt.show()

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

In [19]:

# drop dependent variable (Initial\_days) from the numerical dataframe

numerical.drop(columns='Additional\_charges', inplace=True)

In [20]:

# create all scatterplots

for col in numerical.columns:

plt.figure()

sns.scatterplot(data=numerical, x=col, y=df['Additional\_charges'])

plt.show()

Chart, scatter chart

Description automatically generated

Table

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart

Description automatically generated with medium confidence

Chart

Description automatically generated

Chart

Description automatically generated

### Data Transformation[¶](" \l "Data-Transformation)

In [21]:

categorical.nunique()

Out[21]:

Marital 5

Gender 3

Soft\_drink 2

Initial\_admin 3

HighBlood 2

Stroke 2

Complication\_risk 3

Overweight 2

Arthritis 2

Diabetes 2

Hyperlipidemia 2

BackPain 2

Anxiety 2

Allergic\_rhinitis 2

Reflux\_esophagitis 2

Asthma 2

Services 4

dtype: int64

In [22]:

# encode ordinal data

# start with binary values

df.replace(to\_replace='No', value=0, inplace=True)

df.replace(to\_replace='Yes', value=1, inplace=True)

In [23]:

# encode remaining ordinal data

# import scikit-learn to perform label encoding

# Title: scikit-learn/scikit-learn

# Author: Grisel, et al.

# Date: 2022

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.6543413

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder(categories=[['Low', 'Medium', 'High']])

df['Complication\_risk'] = oe.fit\_transform(df[['Complication\_risk']])

In [24]:

# perform dummy variable encoding nominal data

df = pd.get\_dummies(df, columns=['Marital', 'Gender', 'Initial\_admin', 'Services'], drop\_first=True)

df.head(5).transpose()

Out[24]:

|  | **0** | **1** | **2** | **3** | **4** |
| --- | --- | --- | --- | --- | --- |
| **Population** | 2951.000000 | 11303.000000 | 17125.000000 | 2162.000000 | 5287.000000 |
| **Children** | 1.000000 | 3.000000 | 3.000000 | 0.000000 | 1.000000 |
| **Age** | 53.000000 | 51.000000 | 53.000000 | 78.000000 | 22.000000 |
| **Income** | 86575.930000 | 46805.990000 | 14370.140000 | 39741.490000 | 1209.560000 |
| **VitD\_levels** | 19.141466 | 18.940352 | 18.057507 | 16.576858 | 17.439069 |
| **Doc\_visits** | 6.000000 | 4.000000 | 4.000000 | 4.000000 | 5.000000 |
| **Full\_meals\_eaten** | 0.000000 | 2.000000 | 1.000000 | 1.000000 | 0.000000 |
| **vitD\_supp** | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 2.000000 |
| **Soft\_drink** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **HighBlood** | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Stroke** | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Complication\_risk** | 1.000000 | 2.000000 | 1.000000 | 1.000000 | 0.000000 |
| **Overweight** | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Arthritis** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Diabetes** | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Hyperlipidemia** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **BackPain** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Anxiety** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Allergic\_rhinitis** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **Reflux\_esophagitis** | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Asthma** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Additional\_charges** | 17939.403420 | 17612.998120 | 17505.192460 | 12993.437350 | 3716.525786 |
| **Marital\_Married** | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Marital\_Never Married** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Marital\_Separated** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Marital\_Widowed** | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 |
| **Gender\_Male** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Gender\_Nonbinary** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Initial\_admin\_Emergency Admission** | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Initial\_admin\_Observation Admission** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Services\_CT Scan** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **Services\_Intravenous** | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Services\_MRI** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |

In [25]:

df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 9331 entries, 0 to 9999

Data columns (total 33 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Population 9331 non-null int64

1 Children 9331 non-null int64

2 Age 9331 non-null int64

3 Income 9331 non-null float64

4 VitD\_levels 9331 non-null float64

5 Doc\_visits 9331 non-null int64

6 Full\_meals\_eaten 9331 non-null int64

7 vitD\_supp 9331 non-null int64

8 Soft\_drink 9331 non-null int64

9 HighBlood 9331 non-null int64

10 Stroke 9331 non-null int64

11 Complication\_risk 9331 non-null float64

12 Overweight 9331 non-null int64

13 Arthritis 9331 non-null int64

14 Diabetes 9331 non-null int64

15 Hyperlipidemia 9331 non-null int64

16 BackPain 9331 non-null int64

17 Anxiety 9331 non-null int64

18 Allergic\_rhinitis 9331 non-null int64

19 Reflux\_esophagitis 9331 non-null int64

20 Asthma 9331 non-null int64

21 Additional\_charges 9331 non-null float64

22 Marital\_Married 9331 non-null uint8

23 Marital\_Never Married 9331 non-null uint8

24 Marital\_Separated 9331 non-null uint8

25 Marital\_Widowed 9331 non-null uint8

26 Gender\_Male 9331 non-null uint8

27 Gender\_Nonbinary 9331 non-null uint8

28 Initial\_admin\_Emergency Admission 9331 non-null uint8

29 Initial\_admin\_Observation Admission 9331 non-null uint8

30 Services\_CT Scan 9331 non-null uint8

31 Services\_Intravenous 9331 non-null uint8

32 Services\_MRI 9331 non-null uint8

dtypes: float64(4), int64(18), uint8(11)

memory usage: 1.7 MB

In [26]:

# save prepared data to csv

df.to\_csv('medical\_prepared\_data.csv', header=True)

In [ ]:

## Part 2: Linear Regression Modeling Code

### Linear Regression Model 1: All Features[¶](" \l "Linear-Regression-Model-1:-All-Features)

In [1]:

# read csv into DataFrame

# import pandas for data manipulation

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

import pandas as pd

df = pd.read\_csv('medical\_prepared\_data.csv', index\_col=0)

In [2]:

# install statsmodels

# !pip install statsmodels

In [3]:

# import statsmodels to perform linear regression

import statsmodels.api as sm

# Title: statsmodels/statsmodels

# Author: Seabold, et al.

# Date: 2017

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.275519

# define the variables

x = df.drop(columns='Additional\_charges').assign(const=1)

y = df['Additional\_charges']

# perform the regression and fitting the model

model1 = sm.OLS(y, x).fit()

model1.summary()

Out[3]:

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | Additional\_charges | **R-squared:** | 0.939 |
| **Model:** | OLS | **Adj. R-squared:** | 0.939 |
| **Method:** | Least Squares | **F-statistic:** | 4457. |
| **Date:** | Sat, 25 Mar 2023 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 19:00:55 | **Log-Likelihood:** | -82196. |
| **No. Observations:** | 9331 | **AIC:** | 1.645e+05 |
| **Df Residuals:** | 9298 | **BIC:** | 1.647e+05 |
| **Df Model:** | 32 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **Population** | 0.0006 | 0.001 | 0.414 | 0.679 | -0.002 | 0.003 |
| **Children** | 18.4695 | 8.863 | 2.084 | 0.037 | 1.095 | 35.844 |
| **Age** | 225.5985 | 0.816 | 276.420 | 0.000 | 223.999 | 227.198 |
| **Income** | 0.0002 | 0.001 | 0.256 | 0.798 | -0.001 | 0.001 |
| **VitD\_levels** | -12.9507 | 8.472 | -1.529 | 0.126 | -29.558 | 3.656 |
| **Doc\_visits** | -11.1326 | 16.190 | -0.688 | 0.492 | -42.868 | 20.603 |
| **Full\_meals\_eaten** | 22.9622 | 17.219 | 1.334 | 0.182 | -10.790 | 56.715 |
| **vitD\_supp** | 9.6960 | 28.743 | 0.337 | 0.736 | -46.647 | 66.039 |
| **Soft\_drink** | 13.0740 | 38.464 | 0.340 | 0.734 | -62.324 | 88.472 |
| **HighBlood** | 8630.6336 | 34.242 | 252.047 | 0.000 | 8563.511 | 8697.756 |
| **Stroke** | 357.1042 | 42.045 | 8.493 | 0.000 | 274.687 | 439.522 |
| **Complication\_risk** | 272.9873 | 23.054 | 11.841 | 0.000 | 227.797 | 318.178 |
| **Overweight** | 58.1587 | 37.094 | 1.568 | 0.117 | -14.553 | 130.871 |
| **Arthritis** | -64.1156 | 35.101 | -1.827 | 0.068 | -132.922 | 4.691 |
| **Diabetes** | 47.2134 | 37.734 | 1.251 | 0.211 | -26.753 | 121.179 |
| **Hyperlipidemia** | -4.7020 | 35.630 | -0.132 | 0.895 | -74.544 | 65.140 |
| **BackPain** | -23.2197 | 34.275 | -0.677 | 0.498 | -90.406 | 43.966 |
| **Anxiety** | 25.7995 | 35.949 | 0.718 | 0.473 | -44.669 | 96.268 |
| **Allergic\_rhinitis** | 2.4874 | 34.425 | 0.072 | 0.942 | -64.994 | 69.969 |
| **Reflux\_esophagitis** | 26.4276 | 34.214 | 0.772 | 0.440 | -40.640 | 93.495 |
| **Asthma** | 62.1308 | 37.058 | 1.677 | 0.094 | -10.512 | 134.773 |
| **Marital\_Married** | 46.2187 | 53.396 | 0.866 | 0.387 | -58.450 | 150.887 |
| **Marital\_Never Married** | 44.6879 | 53.778 | 0.831 | 0.406 | -60.729 | 150.105 |
| **Marital\_Separated** | 8.3781 | 53.603 | 0.156 | 0.876 | -96.696 | 113.452 |
| **Marital\_Widowed** | 15.5964 | 53.210 | 0.293 | 0.769 | -88.707 | 119.900 |
| **Gender\_Male** | 156.7706 | 34.020 | 4.608 | 0.000 | 90.084 | 223.457 |
| **Gender\_Nonbinary** | 28.9380 | 118.203 | 0.245 | 0.807 | -202.765 | 260.641 |
| **Initial\_admin\_Emergency Admission** | 476.7859 | 41.151 | 11.586 | 0.000 | 396.120 | 557.451 |
| **Initial\_admin\_Observation Admission** | -84.5127 | 47.930 | -1.763 | 0.078 | -178.466 | 9.441 |
| **Services\_CT Scan** | -36.8501 | 53.486 | -0.689 | 0.491 | -141.693 | 67.993 |
| **Services\_Intravenous** | 22.1080 | 37.964 | 0.582 | 0.560 | -52.309 | 96.525 |
| **Services\_MRI** | 130.8031 | 89.071 | 1.469 | 0.142 | -43.796 | 305.402 |
| **const** | -3219.4100 | 195.981 | -16.427 | 0.000 | -3603.576 | -2835.244 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 1472.775 | **Durbin-Watson:** | 2.007 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 336.224 |
| **Skew:** | -0.024 | **Prob(JB):** | 9.77e-74 |
| **Kurtosis:** | 2.071 | **Cond. No.** | 5.53e+05 |

Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The condition number is large, 5.53e+05. This might indicate that there are  
strong multicollinearity or other numerical problems.

#### Evaluation[¶](#Evaluation)

In [4]:

# evaluate model

from statsmodels.tools.eval\_measures import rmse

# generate predictions

y\_pred = model1.predict(x)

# calculate RMSE

RMSE1 = rmse(y, y\_pred)

RMSE1

Out[4]:

1619.679345691746

In [5]:

# calculate VIF to detect multicollinearity

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["feature"] = x.columns

# calculate VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x.values, i)

for i in range(len(x.columns))]

vif\_data

Out[5]:

|  | **feature** | **VIF** |
| --- | --- | --- |
| **0** | Population | 1.002979 |
| **1** | Children | 1.003222 |
| **2** | Age | 1.004033 |
| **3** | Income | 1.002950 |
| **4** | VitD\_levels | 1.004465 |
| **5** | Doc\_visits | 1.003769 |
| **6** | Full\_meals\_eaten | 1.005218 |
| **7** | vitD\_supp | 1.004086 |
| **8** | Soft\_drink | 1.004003 |
| **9** | HighBlood | 1.003691 |
| **10** | Stroke | 1.002005 |
| **11** | Complication\_risk | 1.003213 |
| **12** | Overweight | 1.003256 |
| **13** | Arthritis | 1.003490 |
| **14** | Diabetes | 1.003936 |
| **15** | Hyperlipidemia | 1.003970 |
| **16** | BackPain | 1.006412 |
| **17** | Anxiety | 1.001974 |
| **18** | Allergic\_rhinitis | 1.002954 |
| **19** | Reflux\_esophagitis | 1.002961 |
| **20** | Asthma | 1.002656 |
| **21** | Marital\_Married | 1.636135 |
| **22** | Marital\_Never Married | 1.624056 |
| **23** | Marital\_Separated | 1.624048 |
| **24** | Marital\_Widowed | 1.640628 |
| **25** | Gender\_Male | 1.023461 |
| **26** | Gender\_Nonbinary | 1.023424 |
| **27** | Initial\_admin\_Emergency Admission | 1.500275 |
| **28** | Initial\_admin\_Observation Admission | 1.501183 |
| **29** | Services\_CT Scan | 1.084115 |
| **30** | Services\_Intravenous | 1.098494 |
| **31** | Services\_MRI | 1.034672 |
| **32** | const | 136.132107 |

### Linear Regression Model 2: Reduced Features[¶](" \l "Linear-Regression-Model-2:-Reduced-Feat)

In [6]:

# import packages from scikit-learn for feature reduction

from sklearn.linear\_model import LinearRegression

from sklearn.feature\_selection import SequentialFeatureSelector

# Title: scikit-learn/scikit-learn

# Author: Grisel, et al.

# Date: 2022

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.6543413

# re-define independent variables

x = df.drop(columns='Additional\_charges')

# linear regression estimator to be used in feature selection

reg = LinearRegression().fit(x, y)

In [7]:

# reduce features using SequentialFeatureSelector

sfs = SequentialFeatureSelector(reg, n\_features\_to\_select=2)

sfs.fit(x, y)

Out[7]:

SequentialFeatureSelector(estimator=LinearRegression(), n\_features\_to\_select=2)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.   
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

SequentialFeatureSelector

SequentialFeatureSelector(estimator=LinearRegression(), n\_features\_to\_select=2)

estimator: LinearRegression

LinearRegression()

LinearRegression

LinearRegression()

In [8]:

# return selected features

sfs.get\_feature\_names\_out()

Out[8]:

array(['Age', 'HighBlood'], dtype=object)

In [9]:

# reduce x to selected features

x = df[['Age', 'HighBlood']].assign(const=1)

In [10]:

# re-perform linear regression

# perform the regression and fitting the model

model2 = sm.OLS(y, x).fit()

model2.summary()

Out[10]:

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | Additional\_charges | **R-squared:** | 0.936 |
| **Model:** | OLS | **Adj. R-squared:** | 0.936 |
| **Method:** | Least Squares | **F-statistic:** | 6.767e+04 |
| **Date:** | Sat, 25 Mar 2023 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 19:00:58 | **Log-Likelihood:** | -82439. |
| **No. Observations:** | 9331 | **AIC:** | 1.649e+05 |
| **Df Residuals:** | 9328 | **BIC:** | 1.649e+05 |
| **Df Model:** | 2 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **Age** | 225.5933 | 0.835 | 270.271 | 0.000 | 223.957 | 227.229 |
| **HighBlood** | 8645.6055 | 35.026 | 246.831 | 0.000 | 8576.946 | 8714.265 |
| **const** | -2673.7164 | 49.823 | -53.664 | 0.000 | -2771.380 | -2576.053 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 951.668 | **Durbin-Watson:** | 2.004 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 274.464 |
| **Skew:** | -0.022 | **Prob(JB):** | 2.52e-60 |
| **Kurtosis:** | 2.161 | **Cond. No.** | 171. |

Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

#### Evaluation[¶](#Evaluation)

In [11]:

# re-evaluate model

# generate predictions

y\_pred = model2.predict(x)

# calculate RMSE

RMSE2 = rmse(y, y\_pred)

RMSE2

Out[11]:

1662.4152991826438

In [12]:

# re-create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["feature"] = x.columns

# calculate VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x.values, i)

for i in range(len(x.columns))]

vif\_data

Out[12]:

|  | **feature** | **VIF** |
| --- | --- | --- |
| **0** | Age | 1.000104 |
| **1** | HighBlood | 1.000104 |
| **2** | const | 8.378513 |

In [13]:

# import matplotlib to create visualizations

# Title: matplotlib/matplotlib

# Author: Caswell, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7697899

# Title: seaborn: statistical data visualization

# Author: Michael Waskom

# Date: 2021

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.4645478

import matplotlib.pyplot as plt

import seaborn as sns

# create partial residual plots for each predictor variablesns.scatterplot(x=df['Age'], y=model2.resid)

for col in x.drop(columns='const').columns:

plt.figure()

sns.scatterplot(x=df[col], y=model2.resid)

plt.axhline(0, color='r')

string = 'Residual Plot: ' + str(col)

plt.title(string)

plt.show()

Chart, scatter chart

Description automatically generated

Chart, line chart

Description automatically generated

In [14]:

sns.scatterplot(x=y\_pred, y=model2.resid)

plt.axhline(0, color='r')

plt.title('Residual Plot')

plt.show()

Chart, scatter chart

Description automatically generated

In [15]:

# calculate residual standard error

model2.resid.std(ddof=x.shape[1])

Out[15]:

1662.6826043376636

In [16]:

sns.histplot(model1.resid)

plt.title('Distribution of Residuals')

plt.show()

Chart, histogram

Description automatically generated

In [ ]:

# Section H

To view a walkthrough demonstration of the code referenced in Section G, refer to the following Panopto link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=46afdc08-79f5-46ee-b824-afd1001e4421>

# Section I

## Part 1: Web Sources

Caswell, et al. (2023, March). doi:10.5281/zenodo.7697899

Gommers, et al. (2023). doi:10.5281/zenodo.7655153

Grisel, et al. (2022). doi:10.5281/zenodo.6543413

Seabold, et al. (2017). doi:10.5281/zenodo.275519

The pandas development team. (2023). doi:10.5281/zenodo.7741580

Waskom, M. (2021). doi:10.5281/zenodo.4645478

## Part 2: References

Bruce, P., Bruce, A., & Gedeck, P. (2019). *Practical Statistics for Data Scientists : 50+ Essential Concepts Using R and Python* (2 ed.). O'Reilly Media, Incorporated. Retrieved March 2023

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection : A Practical Approach for Predictive Models.* CRC Press LLC. Retrieved March 2023

Larose, D. T., & Larose, C. D. (2019). *Data Science Using Python and R.* Hoboken: Wiley. Retrieved March 2023

Pedregosa, e. (2011). *scikit-learn*. Retrieved March 2023, from sklearn.feature\_selection.SequentialFeatureSelector: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SequentialFeatureSelector.html