Lora Milam

Western Governors University

D207 Data Cleaning

17 February 2022

D207 Performance Assessment

1 Research Question

Lora Milam Masters Data Analytics (9/03/2021) Program Mentor:d207@wgu.edu

1.1 Question

The question at hand would be whether a patient's timely admission can be correlated to a patient's likelihood of readmission.

1.2 Benefit from Analysis

Hospitals and medical personnel can benefit from further analysis of this relationship to see if the importance of a timely admission could have a direct or inverse correlation to readmission rates. If there is a relationship between the two factors then respective personnel would be able to address the situations accordingly. Future findings could provide valuable information on how to reduce readmission rates.

1.3 Variables

The data set is 10,000 raw medical patient records. The target variable is whether or not, within a month of release, each customer has been readmitted to the hospital. The title of the column is "ReAdmis".

The predictor variables that are provided in the dataset may have a correlation with the probability of the patient being readmitted due to previous ailments or problems from past admissions. These predictor variables include patient medical conditions (high blood pressure, stroke, obesity, arthritis, diabetes, etc.), patient information (service while hospitalized, days in hospital, type of initial admission, etc.), patient demographics (gender,age, job, education level, etc.). These predictor variables can be seen in the table below:

Variable	**Description**
Unnamed:	- integer index

CaseOrder:	- integer index - correlated to original order of raw data
Customer_id:	- character string object - unique to patient
Interaction:	- character string object - unique to patient transactions, procedures and admissions
UID:	- character string object - unique to the transactions, procedures and admissions of a patient
City:	- character string object - the city of residence of the patient
State:	- character string object - the state of residence of the patient
County:	- character string object - the county of residence of the patient
Zip:	- integer - the zip code corresponding to the residence of the patient's
Lat:	- continuous numeric (floating numeric) - GPS coordinates indicating the latitude corresponding to the patient's residence
Lng:	- continuous numeric (floating numeric) - GPS coordinates indicating the longitude corresponding to the patient's residence
Population:	- integer

	- the population that is within a mile radius of patient's resident
Area:	- nominal categorical - character string object - the area type corresponding to the patient's residence - based on unofficial census data - the unique values are ['Emergency Admission', 'Elective Admission', 'Observation Admission'] In [12]: `df['Area'].unique()` Out[12]: `array(['Suburban', 'Urban', 'Rural'], dtype=object)`
Timezone:	- nominal categorical - character string object - the time zone corresponding to the patient's residence - the unique values are ['America/Chicago', 'America/New_York', 'America/Los_Angeles', 'America/Indiana/Indianapolis', 'America/Detroit', 'America/Denver', 'America/Nome', 'America/Anchorage', 'America/Phoenix', 'America/Boise', 'America/Puerto_Rico', 'America/Yakutat', 'Pacific/Honolulu', 'America/Menominee', 'America/Kentucky/Louisville', 'America/Indiana/Vincennes', 'America/Toronto', 'America/Indiana/Marengo', 'America/Indiana/Winamac', 'America/Indiana/Tell_City', 'America/Sitka', 'America/Indiana/Knox', 'America/North_Dakota/New_Salem', 'America/Indiana/Vevay', 'America/Adak', 'America/North_Dakota/Beulah'] In [13]: `df['Timezone'].unique()` Out[13]: `array(['America/Chicago', 'America/New_York', 'America/Los_Angeles', 'America/Indiana/Indianapolis', 'America/Detroit', 'America/Denver', 'America/Nome', 'America/Anchorage', 'America/Phoenix', 'America/Boise', 'America/Puerto_Rico', 'America/Yakutat', 'Pacific/Honolulu', 'America/Menominee', 'America/Kentucky/Louisville', 'America/Indiana/Vincennes', 'America/Toronto', 'America/Indiana/Marengo', 'America/Indiana/Winamac', 'America/Indiana/Tell_City', 'America/Sitka', 'America/Indiana/Knox', 'America/North_Dakota/New_Salem', 'America/Indiana/Vevay', 'America/Adak', 'America/North_Dakota/Beulah'], dtype=object)`
Job:	- nominal categorical - character string object

	- the occupation of the patient or insurance holder
Children:	- integer - the amount of children within patient’s household
Age:	- integer - the patient’s age

Education:	- nominal categorical
	- character string object
	- the highest earned degree held by the patient
	- the unique values are ['Some College, Less than 1 Year', 'Some College, 1 or More Years, No Degree', 'GED or Alternative Credential', 'Regular High School Diploma', "Bachelor's Degree", "Master's Degree", 'Nursery School to 8th Grade', '9th Grade to 12th Grade, No Diploma', 'Doctorate Degree', "Associate's Degree", 'Professional School Degree', 'No Schooling Completed']
	`In [15]: df['Education'].unique()`
	`Out[15]: array(['Some College, Less than 1 Year',`
	`'Some College, 1 or More Years, No Degree',`
	`'GED or Alternative Credential', 'Regular High School Diploma',`
	`"Bachelor's Degree", "Master's Degree",`
	`'Nursery School to 8th Grade',`
	`'9th Grade to 12th Grade, No Diploma', 'Doctorate Degree',`
	`"Associate's Degree", 'Professional School Degree',`
	`'No Schooling Completed'], dtype=object)`

Employment:	- categorical
	- character string object
	- the employment status currently being held by the patient
	- the unique values are ['Full Time', 'Retired', 'Unemployed', 'Student', 'Part Time']
	`In [16]: df['Employment'].unique()`
	`Out[16]: array(['Full Time', 'Retired', 'Unemployed', 'Student', 'Part Time'],`
	`dtype=object)`

Income:	- numeric value

	- the patient's or insurance holder's annual income

Marital:	- nominal categorical
	- character string object
	- the marital status of the patient or insurance holder
	- the unique values are ['Divorced', 'Married', 'Widowed', 'Never Married', 'Separated']
	`In [17]: df['Marital'].unique()`
	`Out[17]: array(['Divorced', 'Married', 'Widowed', 'Never Married', 'Separated'], dtype=object)`

Gender:	- nominal categorical
	- character string object
	- the gender of the patient
	- the unique values are ['Male', 'Female', 'Prefer not to answer']
	`In [18]: df['Gender'].unique()`
	`Out[18]: array(['Male', 'Female', 'Prefer not to answer'], dtype=object)`

ReAdmins	- binary categorical - character string object - whether or not, within a month of release, each customer has been readmitted to the hospital - **target variable**
VitD_levels:	- continuous numeric (floating numeric) - value of the vitamin D levels of the patient - measured in ng/mL
Doc_visits:	- integer - the number of times during the initial hospitalization that the primary physician visited the patient

Full_meals_eaten:	- integer - number of full meals eaten Note: It counts as zero if the patient only eats a partial meal
Soft_drink:	- binary categorical - character string object - whether or not a patient on a daily basis drinks three or more sodas - the unique values are [Yes, No]
VitD_supp:	- integer - the number of times that vitamin D supplements were administered to patient
Initial_admin:	- nominal categorical - character string object - the reason why the patient was initially admitted into the hospital In [19]: `df['Initial_admin'].unique()` Out[19]: `array(['Emergency Admission', 'Elective Admission', 'Observation Admission'], dtype=object)`
HighBlood:	- binary categorical - character string object - whether or not the patient has high blood pressure - the unique values are [Yes, No]
Stroke:	- binary categorical - character string object - whether or not the patient has had a stroke - the unique values are [Yes, No]
Complication_risk:	- ordinal categorical

	- character string object - the level of complication risk - the unique values are [High, Medium, Low]
Overweight:	- binary categorical - integer - whether or not the patient is overweight, as determined by their BMI elements: age, gender, and height - the unique values are [1,0]
Arthritis:	- binary categorical - character string object - whether or not the patient has arthritis - the unique values are [Yes, No]
Diabetes:	- binary categorical - character string object - whether or not the patient has diabetes - the unique values are [Yes, No]
Hyperlipidemia:	- binary categorical - character string object - whether or not the patient has hyperlipidemia - the unique values are [Yes, No]
BackPain:	- binary categorical - character string object - whether or not the patient has chronic backpain - the unique values are [Yes, No]

Anxiety:	- binary categorical - integer - whether or not the patient has an anxiety disorder - the unique values are [1,0]
Allergic_rhinitis:	- binary categorical - character string object - whether or not the patient has allergic rhinitis - the unique values are [Yes, No]
Reflux_esophagitis:	- binary categorical - character string object - whether or not the patient has reflux esophagitis - the unique values are [Yes, No]
Asthma:	- binary categorical - character string object - whether or not the patient has asthma - the unique values are [Yes, No]

Services:	- nominal categorical
	
	- character string object
	
	- the primary service the patient received while hospitalized
	
	- the unique values are ['Blood Work', 'Intravenous', 'CT Scan', 'MRI']
	
	`In [20]: df['Services'].unique()`
	`Out[20]: array(['Blood Work', 'Intravenous', 'CT Scan', 'MRI'], dtype=object)`

Initial_days:	- numeric value - number of days,during the initial visit, the patient stayed in the hospital

TotalCharge:	- numeric value - patient’s average daily charges,during the initial visit, for typical (not specialized) treatments and services
Additional_charges:	- numeric value - patient’s average charges,during the initial visit, for additional treatments and services
Item1:	- integer value - from most important to least important, the level of importance of timely admission - the unique values are [1:8]
Item2:	- integer value - from most important to least important, the level of importance of timely treatment - the unique values are [1:8]
Item3:	- integer value - from most important to least important, the level of importance of timely visits - the unique values are [1:8]
Item4:	- integer value - from most important to least important, the level of importance of reliability - the unique values are [1:8]
Item5:	- integer value - from most important to least important, the level of importance of options
Item6:	- integer value - from most important to least important, the level of importance of hours of

	treatment - the unique values are [1:8]
Item7:	- integer value - from most important to least important, the level of importance of courteous staff - the unique values are [1:8]
Item8:	- integer value - from doctor from most important to least important, the level of importance of evidence of active listening - the unique values are [1:8]

2 Data Analysis

2.1 Justification of Approach

With a certain amount of confidence, hospitals and medical personnel will benefit from knowing which patients have a higher risk of being readmitted because this will allow for a better correlation of what characteristics, in this case the importance of a timely admission to the patient, are associated with readmittance from past patients and which services should be improved. Hospitals and medical personnel can utilize this information to better accommodate individuals so as to lower readmittance rates.

2.2 Justification of Tools

As mentioned in previous sections, I will be utilizing Python's many capabilities to better analyze the database of medical patient records. Python3 is the latest iteration of the programming language Python, as provided within Jupyter Notebooks. Python is a high level, general purpose language that utilizes a variety of packages to tailor data. Two major packages I will be using to filter the data are Pandas and Numpy. Pandas is a powerful open source data analysis tool built on top of the Python programming language. Numpy provides a multidimensional array object for various math operations. Other packages include Matplotlib.pyplot for plotting data, seaborn for visualization, Sci-Py for linear algebra transformations(specifically chi-square test in this case), and PyLab, statsmodels.api and statistics for statistical analysis.

Even though there are other methods that can be used to address this problem, I find Python3 and Jupyter Notebooks to be a convenient and intuitive way to visualize and draw conclusions from databases.

2.3 Code

The Chi-square technique will be used.

2.3.1 Calculations

In [1]:
```
# Standard data science imports
import numpy as np
import pandas as pd
from pandas import DataFrame

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Statistics packages
import pylab
import statsmodels.api as sm
import statistics
from scipy import stats

# Import chisquare from SciPy.stats
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
```

In [2]:
```
# Load data set into Pandas dataframe
df = pd.read_csv('medical_clean.csv')
df.columns
```

Out[2]:
```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       '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', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
       'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

In [3]:
```
# Rename survey columns to more identifiable names
df.rename(columns =
    {'Item1':'Survey_TimelyAdmin',
     'Item2':'Survey_TimelyTreatment',
     'Item3':'Survey_TimelyVisits',
     'Item4':'Survey_Reliability',
     'Item5':'Survey_Options',
     'Item6':'Survey_HoursTreatment',
     'Item7':'Survey_CourteousStaff',
     'Item8':'Survey_ActiveListening'}, inplace=True)
```

In [4]:
```
contingency = pd.crosstab(df['ReAdmis'], df['Survey_TimelyAdmin'])
contingency
```

Out[4]:

Survey_TimelyAdmin	1	2	3	4	5	6	7	8
ReAdmis								
No	134	823	2137	2184	891	155	7	0
Yes	79	492	1267	1271	486	70	3	1

In [5]:

```
contingency_pct = pd.crosstab(df['ReAdmis'], df['Survey_TimelyAdmin'], normalize='index')
contingency_pct
```

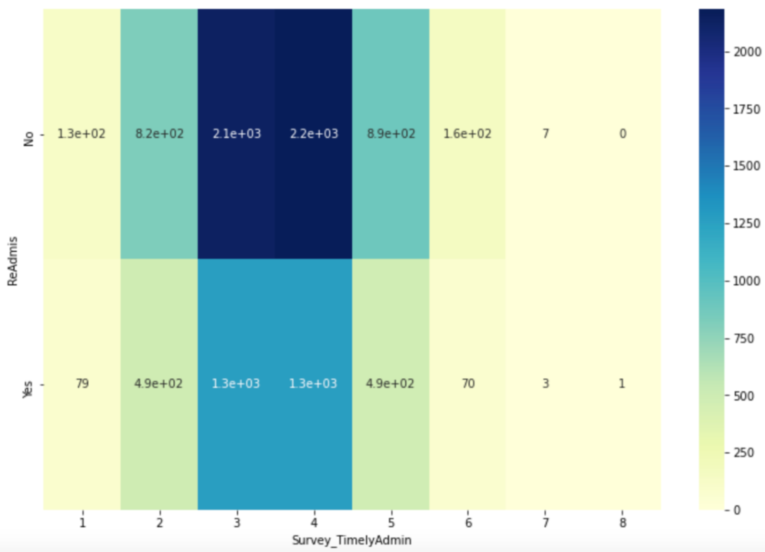
Out[5]:

Survey_TimelyAdmin	1	2	3	4	5	6	7	8
ReAdmis								
No	0.021166	0.129995	0.337545	0.344969	0.140736	0.024483	0.001106	0.000000
Yes	0.021532	0.134096	0.345326	0.346416	0.132461	0.019079	0.000818	0.000273

In [6]:

```
plt.figure(figsize=(12,8))
sns.heatmap(contingency, annot=True, cmap="YlGnBu")
```

Out[6]: <AxesSubplot:xlabel='Survey_TimelyAdmin', ylabel='ReAdmis'>



2.3.2 Output

In [7]:

```
# Chi-square test of independence
c, p, dof, expected = chi2_contingency(contingency)
print('p-value = ' + str(p))
```

```
p-value = 0.44711691481022053
```

2.3.2 Justification

In this analysis, readmission to a hospital is the target variable. “ReAdis” is a binomial, categorical dependent variable. Therefore, the chi-square test shall be used because it is a non-parametric test for this “yes/no” target variable.

3 Univariate Statistics

Two continuous variables:

1. TotalCharge
2. Additional_charges

Two categorical (ordinal) variables:

1. Item1 - relabeled “Survey_TimelyAdmin”
2. Item7 - relabeled “Survey_CourteousStaff”

In [8]:
```
df.describe()
```

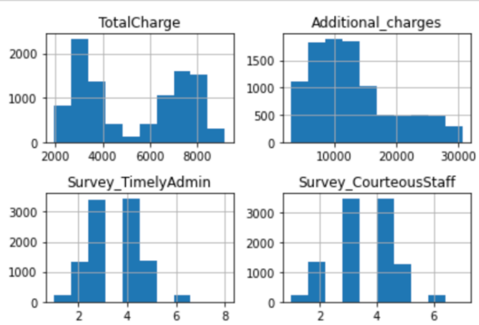
Out[8]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	Doc_visits	...	Tot
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	1000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	53.511700	40490.495160	17.964262	5.012200	...	531
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	20.638538	28521.153293	2.017231	1.045734	...	218
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	18.000000	154.080000	9.806483	1.000000	...	193
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	36.000000	19598.775000	16.626439	4.000000	...	317
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33768.420000	17.951122	5.000000	...	521
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54296.402500	19.347963	6.000000	...	745
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.100000	26.394449	9.000000	...	918

8 rows × 23 columns

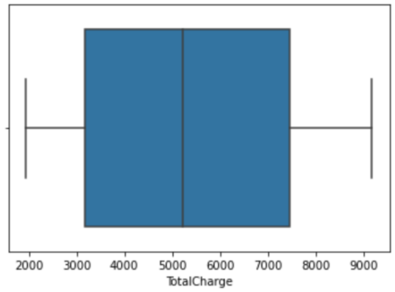
3.1 Visual of Findings

In [9]:
```
# Histograms of continous and categorical variables
df[['TotalCharge','Additional_charges','Survey_TimelyAdmin','Survey_CourteousStaff']].hist()
plt.tight_layout()
```



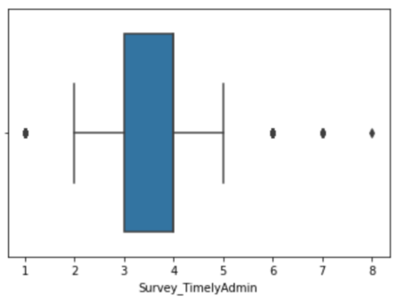
In [10]:
```
# Seaborn boxplots of continous and categorical variables
sns.boxplot('TotalCharge', data = df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



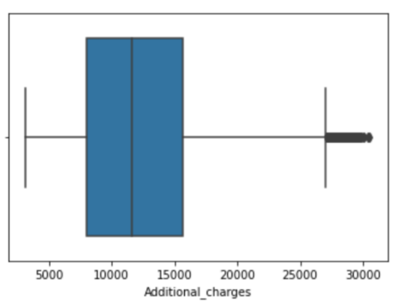
In [12]:
```
sns.boxplot('Survey_TimelyAdmin', data = df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



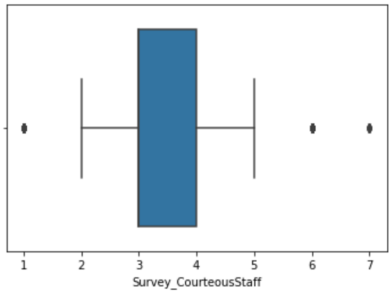
In [11]:
```
sns.boxplot('Additional_charges', data = df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [13]:
```
sns.boxplot('Survey_CourteousStaff', data = df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



4 Bivariate Statistics

Two continuous variables:

1. TotalCharge
2. Additional_charges

Two categorical variables:

1. Item1 - relabeled “Survey_TimelyAdmin”
2. Item7 - relabeled “Survey_CourteousStaff”

4.1 Visual of Findings

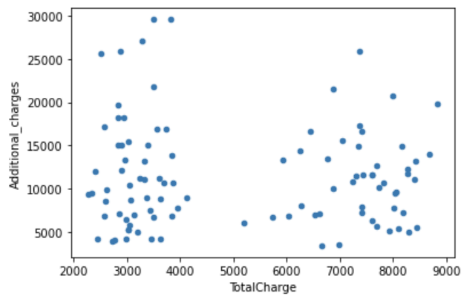
In [14]:
```
# Dataframe for heatmap for bivariate analysis of correlation
readmis_bivariate = df[['TotalCharge','Additional_charges','Survey_TimelyAdmin','Survey_CourteousStaff']]
```

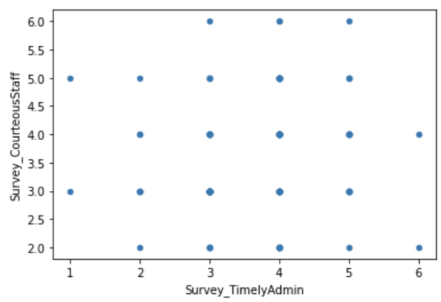
In [15]:
```
sns.heatmap(readmis_bivariate.corr(),annot=True)
plt.show()
```

```
In [16]: # Scatter plot of continuous variables TotalCharge and Additional_charges
readmis_bivariate[readmis_bivariate['TotalCharge'] < 10000].sample(100).
    plot.scatter(x = 'TotalCharge',y = 'Additional_charges')

# Scatter plot of categorical variables Survey_TimelyAdmin and Survey_CourteousStaff
readmis_bivariate[readmis_bivariate['Survey_TimelyAdmin'] < 10].sample(100).
    plot.scatter(x = 'Survey_TimelyAdmin',y = 'Survey_CourteousStaff')
```

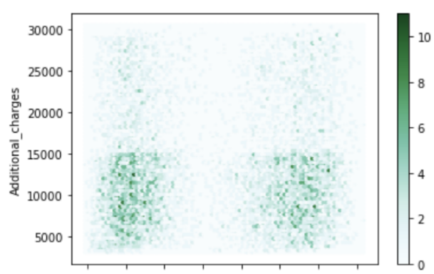
```
Out[16]: <AxesSubplot:xlabel='Survey_TimelyAdmin', ylabel='Survey_CourteousStaff'>
```






```
In [17]: # Heatmap for continous variables TotalCharge and Additional_charges
readmis_bivariate[readmis_bivariate['TotalCharge'] < 10000].plot.hexbin(x = 'TotalCharge', y = 'Additional_charges')
```

```
Out[17]: <AxesSubplot:xlabel='TotalCharge', ylabel='Additional_charges'>
```



5 Summary

5.1 Results of Analysis

Considering the p-value from the chi-square test, p-value = 0.44711691481022053, it can be determined that the null hypothesis cannot be rejected given that the standard level of alpha is .05. There is no clear relationship between the survey responses and whether or not this causes a patient to be readmitted to a hospital.

5.2 Limitations of Analysis

There are two main limitations to the chi-square test. The chi-square test is very sensitive to sample size. Even the most simplistic relationships can appear to be statistically significant with a large enough sample size. This means that when using the chi-square test, it should be noted that something "statistically significant" doesn't necessarily mean "meaningful." Secondly, the chi-square can only tell whether two variables are related to one another. It can not necessarily tell whether one variable has any causal effect on the other. In order to establish causality, a more detailed analysis would be required.

5.3 Recommended Course of Action

With the p-value so high, there will need to be additional investigations. Considering that a chi-square test only evaluates linearity, the next step to take would be to see if the relationship is non-linear. Perhaps there is also a need for more and better data to analyze this relationship further.

6 Supporting Documents

6.1 Video

This can be found within the attached file ‘Panopto Recording’.

6.2 Sources

Gagner, David. (2022). D207 Exploratory Data Analysis . Salt Lake City ; Western Governors University.

Western Governors University. (n.d.). D207 D208 D209 Medical Data Considerations and Dictionary. Salt Lake City.