D206 PA code - Doug Haunsperger

Do initial package import and data read

In [ ]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('medical_raw_data.csv')

#view first 5 rows
df.head(5)
```

Show variable names, non-null counts using `info()`

In [ ]:
```
df.info()
```

Check for duplicate data

First, check for duplicate rows

In [ ]:
```
df.duplicated().value_counts()
```

Check to see if there are any duplicate `Customer_id` values, perhaps with different data entered

In [ ]:
```
df.duplicated(subset = ['Customer_id'], keep = False).value_counts()
```

Now we notice in the `head()` above that the first two columns are suspiciously the same. Let's check to see if the column is completely duplicated:

In [ ]:
```
df[df['Unnamed: 0'] == df['CaseOrder']].shape
```

In [ ]:
```
df[df['Unnamed: 0'] != df['CaseOrder']].shape
```

Check for missing values

Null values already shown in `info()` cell [2] above. Here we will visualize them. In the data treatment section, we will try to determine if the missing values are MAR/MCAR/MNAR.
Code to install package in Jupyter environment taken from Jake VanderPlas (2017).

In [ ]:
```
# Install package if needed for environment
# !{sys.executable} -m pip install missingno
import missingno as msno
import matplotlib.pyplot as plt
msno.matrix(df, labels=True) # matrix function turns off labels by default when # columns > 50
# See https://github.com/ResidentMario/missingno/issues/93 (BiLogur, 2019)
plt.show()
```

We see we have missing values in the columns `Children`, `Age`, `Income`, `Soft_drink`, `Overweight`, `Anxiety`, and `Initial_days`.
`Children`, `Age`, `Income`, and `Soft_drink` have the most missing values; each have on the order of 25% of the observations missing.

Check for outliers

In [ ]:
```
import seaborn
# Choose only the quantitative columns
quant_cols=['Population', 'Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'VitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges']

# Ref: https://stackoverflow.com/questions/16392921/make-more-than-one-chart-in-same-ipython-notebook-cell (Kassies, 2013)
for col in quant_cols:
    fig, axs = plt.subplots(1,2, figsize=(15,5))
    seaborn.histplot(df[col], ax=axs[0])
    plt.title(col)
    seaborn.boxplot(df[col], orient='h', ax=axs[1])
    plt.title(col)
    plt.show()
```

Re-expressing Categorical Variables

Per the PA Guide, I am ignoring the index/ID/location variables.

In [ ]:
```
cat_cols = ['Education', 'ReAdmis', 'Soft_drink', 'HighBlood', 'Stroke', 'Complication_risk', 'Arthritis', 'Diabetes', 'Hyperlipidemia',
            'BackPain', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Overweight', 'Anxiety', 'Item1', 'Item2', 'Item3', 'Item4',
            'Item5', 'Item6', 'Item7', 'Item8']
for col in cat_cols:
    print(df[col].unique())
```

Set up dictionaries based on the above unique values in the data set. Code adapted from Larose & Larose (2019).

In [ ]:
```
dict_edu = {"No Schooling Completed": 0, "Nursery School to 8th Grade": 8, "9th Grade to 12th Grade, No Diploma": 10, "Regular High School Diploma": 12,
            "GED or Alternative Credential": 12, "Professional School Degree": 12, "Some College, Less than 1 Year": 12, "Some College, 1 or More Years, No Degree": 13,
            "Associate's Degree": 14, "Bachelor's Degree": 16, "Master's Degree": 18, "Doctorate Degree": 20}
dict_compl = {"Low": 1, "Medium": 2, "High": 3}
dict_yn = {"Yes": 1, "No": 0}
replace_dict = {'Education': dict_edu, 'ReAdmis': dict_yn, 'Soft_drink': dict_yn, 'HighBlood': dict_yn, 'Stroke': dict_yn, 'Complication_risk':dict_compl, 'Arthritis': dict_yn,
                'Diabetes': dict_yn, 'Hyperlipidemia': dict_yn, 'BackPain': dict_yn, 'Allergic_rhinitis': dict_yn, 'Reflux_esophagitis': dict_yn, 'Asthma': dict_yn}
# Overweight, Anxiety, and Item[1-8] are already expressed ordinally
# Make copy of original df
df_clean = df.copy(deep = True)
df_clean.replace(replace_dict, inplace = True)
```

Re-check unique vals to make sure all categoricals have been re-expressed

In [ ]:
```
for col in cat_cols:
    print(df_clean[col].unique())
```

Data Cleaning

Treating Duplicates

Drop the duplicated column

In [ ]:
```
df_clean = df_clean.drop(columns=['Unnamed: 0'])
df_clean.tail(5)
```

In [ ]:
```
df['Age'].unique()
```

In [ ]:
```

```