UDACITY

Return to Classroom

Investigate a Dataset

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

You've got your brain in gear 🥳 Congratulations! 🥇🎉

✅ Your project meets all the requirements. So proud of you and your captivating analysis. I hope you enjoyed the journey as much I enjoyed reviewing your project. 👏

The visualizations are neat, you followed the correct steps in documenting the cleaning process and draw an accurate conclusion. Keep the great work.

I wanted to express my utmost gratitude for all the consistency you have done so far, and your analytical skills will drive you to move forward to the right direction. 😊

Code Functionality

- All code is functional and produces no errors when run.

- The code given is sufficient to reproduce the results described.

👏✅ Your source code runs smoothly.

Project: Investigate a Dataset

Selected dataset: No-show appointments

```
# Importing all necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

```
df = pd.read_csv('./data/noshowappointments-kagglev2-may-2016.csv')
df.head(5)
```

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	0	0	0
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	0	0	0
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	0	0	0
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	0	0	0	0
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	1	1	0	0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64
 2   Gender          110527 non-null  object
 3   ScheduledDay    110527 non-null  object
 4   AppointmentDay  110527 non-null  object
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hipertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handcap         110527 non-null  int64
 12  SMS_received    110527 non-null  int64
 13  No-show         110527 non-null  object
```

```
df.shape
```

```
(110527, 14)
```

The dataset includes **14** columns and **110527** data rows.

In the next cells, we will identify the data format and types (Quantitative vs. Categorical) of each column to help us plan and determine the best data

For more reference:

- 10 common mistakes Python programmers make (and how to fix them)
- Jupyter Notebook Tutorial: Introduction, Setup, and Walkthrough

- The project uses NumPy arrays and Pandas Series and DataFrames where appropriate rather than Python lists and dictionaries.
- Where possible, vectorized operations and built-in functions are used instead of loops.

👏✅ Great job by using different python libraries in your submission.

Selected dataset: No-show appointments

```
# Importing all necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

For more reference:

- An introduction to seaborn
- Most Popular Python Packages in 2021
- The Ultimate NumPy Tutorial for Data Science Beginners
- Matplotlib Tutorial with Exercises - 1

- The code makes use of at least 1 function to avoid repetitive code.
- The code contains good comments and meaningful variable names, making it easy to read.

Masterpiece 👏

✅ The code makes use of at least 1 function to avoid repetitive code.

✅ The code contains good comments and meaningful variable names, making it easy to read.

0]:
```
# Create a funtion to generate a bar plot the frequency table
# generated by the Show_NoShow_by_Group function.

def Show_No_Show_bar_plot(df, bygroup):
    df_by_Group = pd.crosstab(df[bygroup], df.Status, normalize = 'index')
    df_by_Group = np.round((df_by_Group * 100), decimals=2)
    ax = df_by_Group.plot.bar(figsize=(10,5));
    vals = ax.get_yticks()
    ax.set_yticklabels(['{:3.0f}%'.format(x) for x in vals]);
    ax.set_xticklabels(df_by_Group.index,rotation = 0, fontsize = 15);
    ax.set_title('\nShowUp vs. No ShowUp (%) (by ' + df_by_Group.index.name + ')\n', fontsize = 15)
    ax.set_xlabel(df_by_Group.index.name, fontsize = 12)
    ax.set_ylabel('(%)', fontsize = 12)
    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)
    rects = ax.patches

    # Add Data Labels

    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2,
                height + 2,
                str(height)+'%',
                ha='center',
                va='bottom',
                fontsize = 12)
    return df_by_Group
```

1]:
```
# An updated version of Show_No_Show_bar_plot with different font formatting
# to match the number of column categories

def Show_No_Show_bar_plot_V2(df, bygroup):
    df_by_Group = pd.crosstab(df[bygroup], df.Status, normalize = 'index')
    df_by_Group = np.round((df_by_Group * 100), decimals=2)
    ax = df_by_Group.plot.bar(stacked = True, figsize=(20,10));
    vals = ax.get_yticks()
    ax.set_yticklabels(['{:3.0f}%'.format(x) for x in vals]);
    ax.set_xticklabels(df_by_Group.index,rotation = 45, fontsize = 12);
    ax.set_title('ShowUp vs. No ShowUp (%) (by ' + df_by_Group.index.name + ')\n', fontsize = 15
    ax.set_xlabel(df_by_Group.index.name, fontsize = 12)
    ax.set_ylabel('(%)', fontsize = 12)
    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)
    rects = ax.patches

    # Add Data Labels

    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2,
                height + 2,
                str(height)+'%',
                ha='center',
                va='bottom',
                fontsize = 12)
    return df_by_Group
```

12]:
```
# Create a funtion to calculate the plot the group proportions based on one variable by number of appointments

def PropByVar(df, variable):
    df_pie = df[variable].value_counts()
    ax = df_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12);
    ax.set_title(variable + ' (%) (Per appointment)\n', fontsize = 15);
    return np.round(df_pie/df.shape[0]*100,2)
```

13]:
```
# Create a funtion to calculate the plot the group proportions based on one variable

def NumOfPatients(df, variable):
    PatID_Count = pd.pivot_table(df, index=variable, values='PatientID',aggfunc = lambda x: len(x.unique()))
    ax = PatID_Count.plot.pie(figsize=(10,10), autopct='%1.2f%%', subplots=True, fontsize = 12, legend = False);
    plt.title(variable + ' (%) (Per patient)\n', fontsize = 15);
    return np.round(PatID_Count/sum(PatID_Count['PatientID'])*100,2)
```

14]:
```
# Create a function to plot the noshow-up % for two variables (i.e. gender and hypertention)

def NoShowBy2Vars(df,var1, var2):
    Freq_df = pd.crosstab(df[var1], columns = df[var2], normalize = 'index')
    Freq_df = np.round(Freq_df * 100,2)
    ax = Freq_df.plot.barh(stacked = True,figsize=(10,5));
    ax.set_title('\nNo ShowUp (%) (by ' + str(var1) + ' & '+ str(var2) + ')\n', fontsize = 15);
    ax.set_ylabel(Freq_df.index.name, fontsize = 12)
    ax.set_xlabel('(%)', fontsize = 12)
    vals = ax.get_xticks()
    ax.set_xticklabels(['{:3.0f}%'.format(x) for x in vals])
    ax.tick_params(axis='both', which='major', labelsize=12)
    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)
    return Freq_df
```

15]:
```
def Recurring_Patient_prct(df,var1,var2):

    # Pivot table to calcuate the patientID count
    PatID_Count = pd.pivot_table(df, index=var1, columns=var2, values='PatientID',aggfunc = lambda x: len(x.unique()

    # Pivot table to calcuate the AppointmentID count
    AptID_Count = pd.pivot_table(df, index=var1, columns=var2, values='AppointmentID',aggfunc='count')

    # divide the two tables above to calculate the percentage and return the resulting table.
    Div_chck = np.round((1 - PatID_Count/AptID_Count)*100,2)

    ax = Div_chck.plot.bar(figsize=(10,5));
    vals = ax.get_yticks()
    ax.set_yticklabels(['{:3.0f}%'.format(x) for x in vals]);
    ax.set_xticklabels(Div_chck.index,rotation = 0, fontsize = 15);
    ax.set_title('\nRecurring Patients (%)\n', fontsize = 15)
    ax.set_xlabel(Div_chck.index.name, fontsize = 12)
    ax.set_ylabel('(%)', fontsize = 12)
    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)
    rects = ax.patches

    # Add Data Labels

    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2,
                height + 2,
                str(height)+'%',
                ha='center',
                va='bottom',
                fontsize = 12)

    return Div_chck
```

For more references:

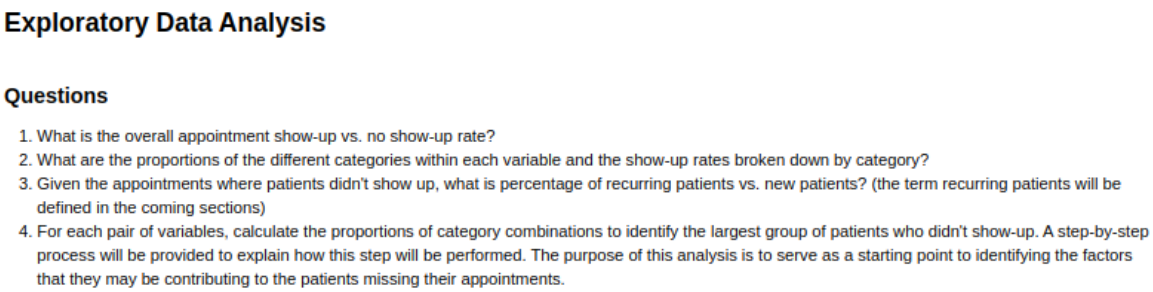
- Clinging to memory: how Python function calls can increase memory use
- Python Functions | Python Tutorial for Absolute Beginners #1
- Python Functions Tutorial
- Defining Your Own Python Function

Quality of Analysis

The project clearly states one or more questions, then addresses those questions in the rest of the analysis.

Bravo! 👏

✅ You did a fabulous job by stating the questions firstly then one-by-one you addressed them.



Exploratory Data Analysis

Questions

1. What is the overall appointment show-up vs. no show-up rate?
2. What are the proportions of the different categories within each variable and the show-up rates broken down by category?
3. Given the appointments where patients didn't show up, what is percentage of recurring patients vs. new patients? (the term recurring patients will be defined in the coming sections)
4. For each pair of variables, calculate the proportions of category combinations to identify the largest group of patients who didn't show-up. A step-by-step process will be provided to explain how this step will be performed. The purpose of this analysis is to serve as a starting point to identifying the factors that they may be contributing to the patients missing their appointments.

For more reference:

- Step 1: Identifying the focal issue with 'Problem Tree Analysis' technique
- Python Coding - Asking User's Questions

Data Wrangling Phase

The project documents any changes that were made to clean the data, such as merging multiple files, handling missing values, etc.

You meet the requirements 👏 Great job!

✅ The changes in the data cleaning section is well documented!

```
df.DayofWeek.cat.reorder_categories(['Monday', 'Tuesday','Wednesday','Thursday','Friday','Saturday'],
```

```
# Define a new column to include the Age Groups

AgeGroupLabels = [ "{0} - {1}".format(i, i + 19) for i in range(0, 120, 20)]
df['AgeGroup'] = pd.cut(df.Age, range(0, 121, 20), right=False, labels=AgeGroupLabels)
df['AgeGroup'] = df.AgeGroup.astype('category')
df.AgeGroup.cat.categories
```

```
Index(['0 - 19', '20 - 39', '40 - 59', '60 - 79', '80 - 99', '100 - 119'], dtype='object')
```

The cell above shows 6 Age groups (in years) ranging from 0 - 119 with an internal of 20 years

```
# Define a new column to include the Waiting Duration Groups

WtngDurtnGroupLabels = [ "{0} - {1}".format(i, i + 29) for i in range(0, 180, 30)]
df['WaitingDurationGroups'] = pd.cut(df.WaitingDuration, range(0, 181, 30), right=False, labels=WtngDurtnGroupLabels
```

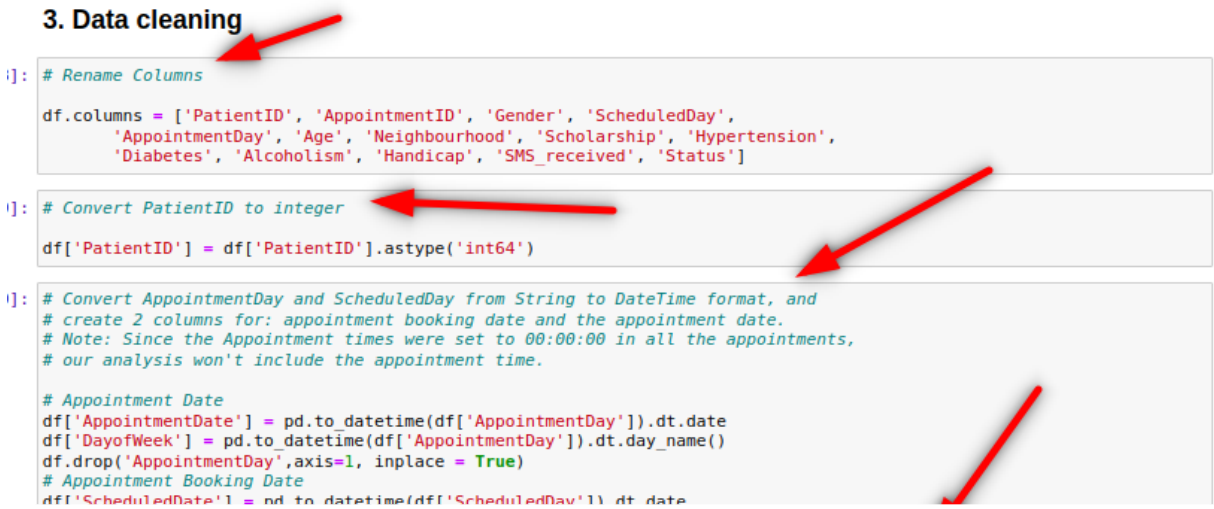
```
# Exclude the 5 records with the negative waiting duration, the new dataset size is 110527 - 5 = 110522

df = df[df['AppointmentDate'] >= df['ScheduledDate']]
df.shape
```

```
(110522, 16)
```

```
# Exclude the record with the negative Age value

df = df[df.Age != -1]
df.shape
```

```
(110521, 16)
```

```
# Update the values in the status, gender, scholarship, hypertension, diabetes, alcoholism, SMS_received columns
# to make them more user friendly

df['Status'] = df.Status.astype('category')
df.Status.cat.rename_categories(['Show','NoShow'], inplace = True)

df['Gender'] = df.Gender.astype('category')
df.Gender.cat.rename_categories(['Female','Male'], inplace = True)

df['Scholarship'] = df.Scholarship.astype('category')
df.Scholarship.cat.rename_categories(['No Scholarship','Scholarship'], inplace = True)

df['Hypertension'] = df.Hypertension.astype('category')
df.Hypertension.cat.rename_categories(['No Hypertension','Hypertension'], inplace = True)

df['Diabetes'] = df.Diabetes.astype('category')
```



3. Data cleaning

```
# Rename Columns

df.columns = ['PatientID', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hypertension',
       'Diabetes', 'Alcoholism', 'Handicap', 'SMS_received', 'Status']
```

```
# Convert PatientID to integer

df['PatientID'] = df['PatientID'].astype('int64')
```

```
# Convert AppointmentDay and ScheduledDay from String to DateTime format, and
# create 2 columns for: appointment booking date and the appointment date.
# Note: Since the Appointment times were set to 00:00:00 in all the appointments,
# our analysis won't include the appointment time.

# Appointment Date
df['AppointmentDate'] = pd.to_datetime(df['AppointmentDay']).dt.date
df['DayofWeek'] = pd.to_datetime(df['AppointmentDay']).dt.day_name()
df.drop('AppointmentDay',axis=1, inplace = True)
# Appointment Booking Date
df['ScheduledDate'] = pd.to_datetime(df['ScheduledDay']).dt.date
```

Additional source:

- Data Cleaning Techniques in Python: the Ultimate Guide
- Handling Missing Data
- Pythonic Data Cleaning With Pandas and NumPy

Exploration Phase

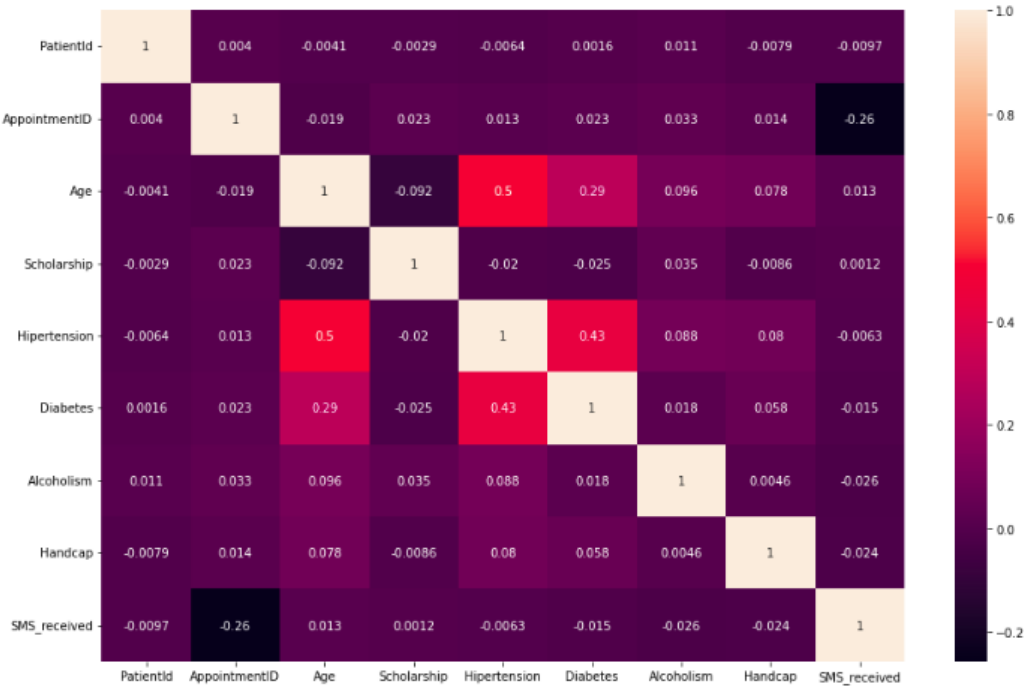
- The project investigates the stated question(s) from multiple angles.
- The project explores at least three variables in relation to the primary question. This can be an exploratory relationship between three variables of interest, or looking at how two independent variables relate to a single dependent variable of interest.
- The project performs both single-variable (1d) and multiple-variable (2d) explorations.

Faultless 👏

✅ The project investigates both single-variable (1d) and multiple-variable (2d) explorations.

```
]: fig, ax = plt.subplots(figsize=(15,10))
   sns.heatmap(df.corr(), ax=ax, annot=True);
```

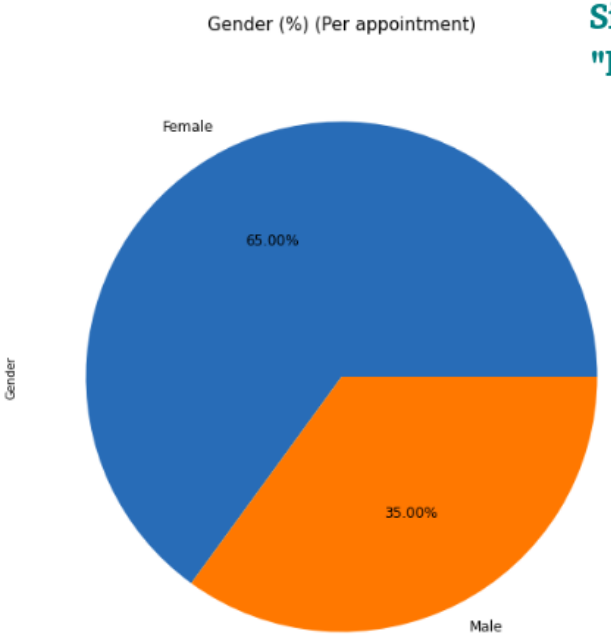
Multiple variable exploration "Heatmap"



Q2.1: GENDER

```
]: PropByVar(df, 'Gender')
]: Female    65.0
   Male      35.0
   Name: Gender, dtype: float64
```

Single variable exploration "Pie plot"



Here's a great guide on the difference between the Univariate, Bivariate:

- Univariate plotting
- Multivariate plotting

Understanding Data with Visualization

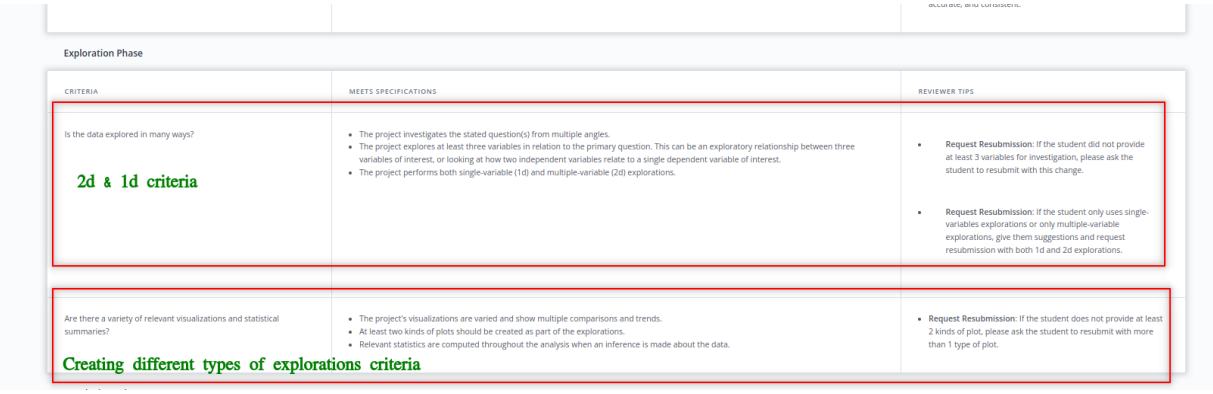
Introduction to Data Visualization in Python

A Beginner's Guide to matplotlib for Data Visualization and Exploration in Python

Examine the differences between univariate and bivariate data.

Univariate Data	Bivariate Data
• involving a **single variable**	• involving **two variables**
• does not deal with causes or relationships	• deals with causes or relationships
• the major purpose of univariate analysis is to describe	• the major purpose of bivariate analysis is to explain
• central tendency - mean, mode, median • dispersion - range, variance, max, min, quartiles, standard deviation. • frequency distributions • bar graph, histogram, pie chart, line graph, box-and-whisker plot	• analysis of two variables simultaneously • correlations • comparisons, relationships, causes, explanations • tables where one variable is contingent on the values of the other variable. • independent and dependent variables
Sample question: How many of the students in the freshman class are female?	**Sample question:** Is there a relationship between the number of females in Computer Programming and their scores in Mathematics?

👉 Udacity criteria



Exploration Phase

CRITERIA	MEETS SPECIFICATIONS	REVIEWER TIPS
Is the data explored in many ways? 2d & 1d criteria	• The project investigates the stated question(s) from multiple angles. • The project explores at least three variables in relation to the primary question. This can be an exploratory relationship between three variables of interest, or looking at how two independent variables relate to a single dependent variable of interest. • The project performs both single-variable (1d) and multiple-variable (2d) explorations.	• **Request Resubmission**: If the student did not provide at least 3 variables for investigation, please ask the student to resubmit with this change. • **Request Resubmission**: If the student only uses single-variables explorations or only multiple-variable explorations, give them suggestions and request resubmission with both 1d and 2d explorations.
Are there a variety of relevant visualizations and statistical summaries? Creating different types of explorations criteria	• The project's visualizations are varied and show multiple comparisons and trends. • At least two kinds of plots should be created as part of the explorations. • Relevant statistics are computed throughout the analysis when an inference is made about the data.	• **Request Resubmission**: If the student does not provide at least 2 kinds of plot, please ask the student to resubmit with more than 1 type of plot.

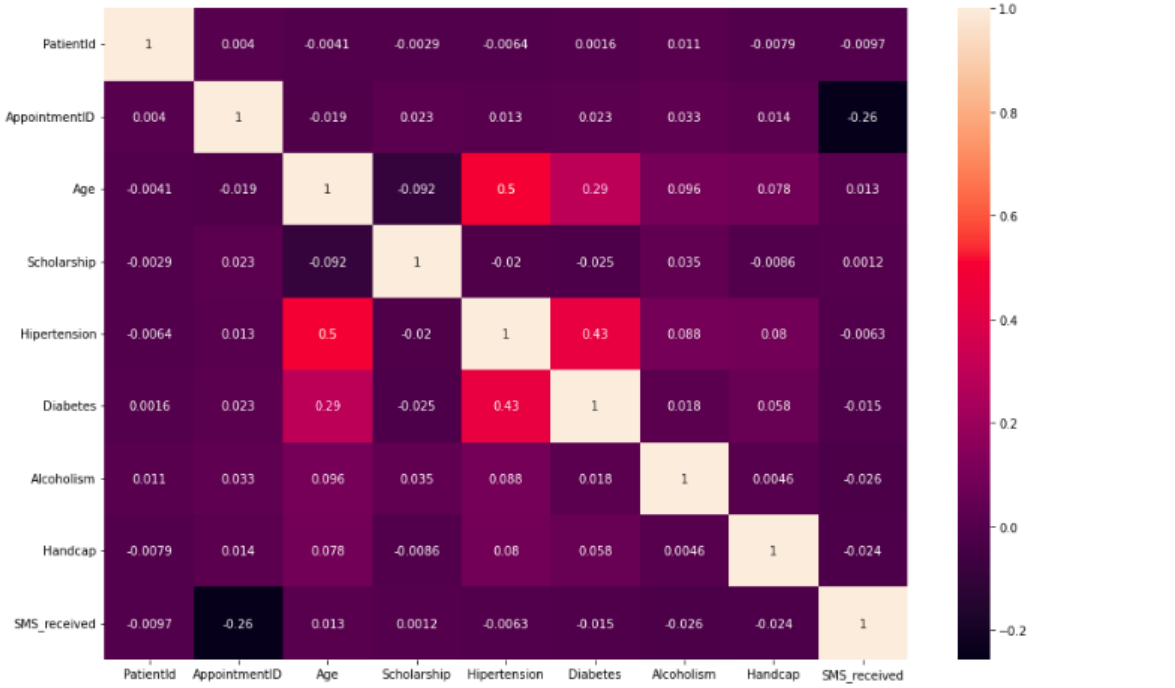
- The project's visualizations are varied and show multiple comparisons and trends.
- At least two kinds of plots should be created as part of the explorations.
- Relevant statistics are computed throughout the analysis when an inference is made about the data.

Marvelous job 👏

✅ Such a great job in including at least two kinds of visualizations.

```
]: fig, ax = plt.subplots(figsize=(15,10))
   sns.heatmap(df.corr(), ax=ax, annot=True);
```

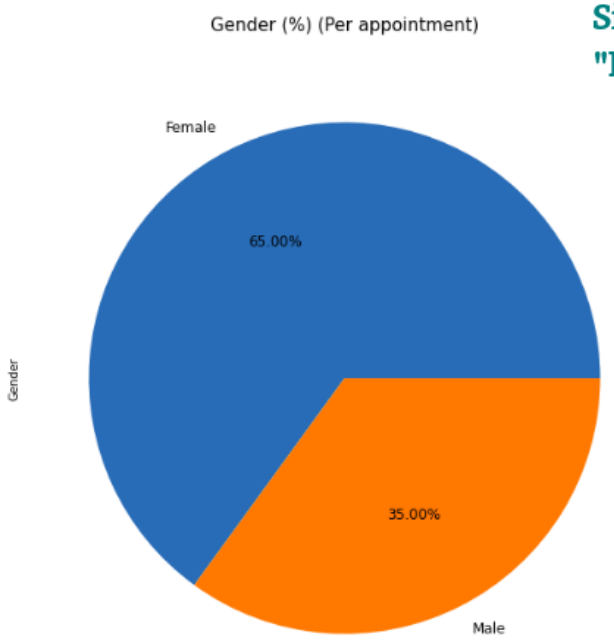
Multiple variable exploration "Heatmap"



Q2.1: GENDER

```
]: PropByVar(df, 'Gender')
]: Female    65.0
   Male      35.0
   Name: Gender, dtype: float64
```

Single variable exploration "Pie plot"



For more references:

- The Python Graph Gallery
- Python Plotting With Matplotlib (Guide)

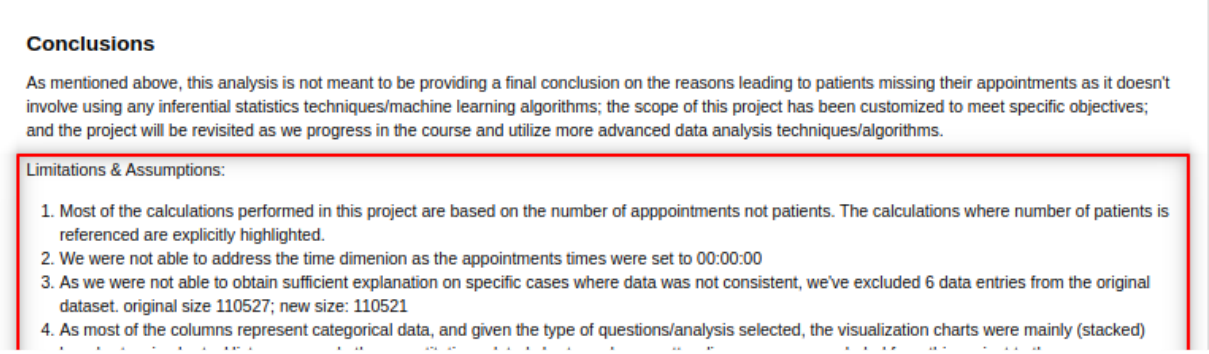
- How to build beautiful plots with Python and Seaborn

Conclusions Phase

- The Conclusions have reflected on the steps taken during the data exploration.
- The Conclusions have summarized the main findings in relation to the question(s) provided at the beginning of the analysis accurately.
- The project has pointed out where additional research can be done or where additional information could be useful.
- The conclusion should have at least 1 limitation explained clearly.
- The analysis does not state or imply that one change causes another based solely on a correlation.

Nicely done 👏

✅ Thank you so much for including the conclusion section & the limitation sub-section at the end of your submission.



Conclusions

As mentioned above, this analysis is not meant to be providing a final conclusion on the reasons leading to patients missing their appointments as it doesn't involve using any inferential statistics techniques/machine learning algorithms; the scope of this project has been customized to meet specific objectives; and the project will be revisited as we progress in the course and utilize more advanced data analysis techniques/algorithms.

Limitations & Assumptions:

1. Most of the calculations performed in this project are based on the number of apppointments not patients. The calculations where number of patients is referenced are explicitly highlighted.
2. We were not able to address the time dimenion as the appointments times were set to 00:00:00
3. As we were not able to obtain sufficient explanation on specific cases where data was not consistent, we've excluded 6 data entries from the original dataset. original size 110527; new size: 110521
4. As most of the columns represent categorical data, and given the type of questions/analysis selected, the visualization charts were mainly (stacked)

Additional sources:

- Python Statistics Fundamentals: How to Describe Your Data
- Conclusion - Learning to Program with Python 3 (basics)
- Limitations of the Study
- The Limitations of the Data in Predictive Analytics

Communication

- The code should have ideally the following sections: Introduction; Questions; Data Wrangling; Exploratory Data Analysis; Conclusions, Limitation.
- Reasoning is provided for each analysis decision, plot, and statistical summary.
- Interpretation of plots and application of statistical tests should be correct and without error.
- Comments are used within the code cells.
- Documented the flow of analysis in the mark-down cells.

Splendid! 👏

✅ Thank you so much for including a reasoning section below every chart, where you have made an optimal analysis with different types of visualizations.

For more references:

- Tips on Interpreting Data Visualizations
- Interpreting Data Visualizations: The basics: Home
- Interpreting Data through Visualization with Python Matplotlib

Visualizations made in the project depict the data in an appropriate manner (i.e., has appropriate labels, scale, legends, and plot type) that allows plots to be readily interpreted.

Magnificent job 👏

✅ You did a wonderful work by making the visualization well interpreted.

TIPS:

- We add the plot title using the plt.title() function. Here's a great documentation on the title function.
- We also add the x-axis label using the plt.xlabel() function. Here's a great documentation on the xlabel function.
- We add the y-axis label using the plt.ylabel() function. Here's a great documentation on the ylabel function.

DOWNLOAD PROJECT

RETURN TO PATH