UDACITY

Return to Classroom

Communicate Data Findings

REVIEW

HISTORY

Requires Changes

2 specifications require changes

Hi Udacity learner,
Thane Murphy

- Great job submitting the project. You have done outstanding work as most of the requirements of the project are fulfilled. There are some small changes that are to be taken care of before you pass this project. Most of the changes are very small and will not take much of your time.
- I have explained below each of the rubrics what needs to be done. I have also provided you with few links for extra sources.

ADDITIONAL LINKS TO READ IN FREE TIME

- 7 Fundamental Steps to Complete a Data Analytics Project
- A Comprehensive Guide to Data Exploration
- Must Read 26 Data Analyst Interview Questions & Answers

- I hope this review will help you in completing the project.
- If you have any doubt related to this project you can use the Knowledge platform

- Keep up the hard work and keep learning.

Code Quality

1. Code runs without error. Note: a few warnings are okay.
2. Code is documented with enough relevant comments that a person scanning the code can follow what the code is doing.
3. Indentation uses 4 spaces (not tabs) and is consistent throughout the project.
4. Variable names are concise and meaningful.
5. Functions and loops are used as needed to reduce repetitive code.

- All the code runs without any errors. Good job.
- Comments and docstrings are used to document the code functionality.
- Functions are used to avoid code repetition.

```
def assign_day_of_week(df, datetime_col='start_time'):
    # Ensure 'start_time' is in datetime format
    df[datetime_col] = pd.to_datetime(df[datetime_col])

    # Create a new column for day of the week from 'start_time'
    df['day_of_week'] = df[datetime_col].dt.day_name()

    # Define the correct order for days of the week
    days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

    # Convert 'day_of_week' to a categorical type with the specified order
    df['day_of_week'] = pd.Categorical(df['day_of_week'], categories=days_order, ordered=True)
```

```
def categorize_time_of_day(df, datetime_col='start_time'):
    def get_time_category(dt):
        if 5 <= dt.hour < 12:
            return 'Morning'
        elif 12 <= dt.hour < 17:
            return 'Afternoon'
        else:  # Covering evening and night, including from 5 PM to before 5 AM
            return 'Evening'

    # Apply the function to each row based on 'start_time'
    df['time_of_day'] = df[datetime_col].apply(get_time_category)
```

```
def categorize_time_of_day(df, datetime_col='start_time'):
    def get_time_category(dt):
        if 5 <= dt.hour < 12:
            return 'Morning'
        elif 12 <= dt.hour < 17:
            return 'Afternoon'
        else:  # Covering evening and night
            return 'Evening'

    # Apply the function to each row based on the specified datetime column
    df['time_of_day'] = df[datetime_col].apply(get_time_category)
    return df  # Make sure to return the DataFrame

# Now call the function and pass the DataFrame
fordgobike_df = categorize_time_of_day(fordgobike_df)
```

```
def determine_day_type(df):
    # Apply a function to determine if the day is a weekday or weekend
    # 0 = Monday, 6 = Sunday
    df['day_type'] = df['start_time'].dt.dayofweek.apply(lambda x: 'Weekday' if x < 5 else 'Weekend')

# Apply the function to the dataframe
determine_day_type(fordgobike_df)

# Create the FacetGrid with 'time_of_day', 'user_type', and 'day_type'
g = sb.FacetGrid(data=fordgobike_df, col='time_of_day', row='user_type', hue='day_type', height=4, aspect=1.5, palette='Set1', legend_out=True)

# Map the scatterplot to the grid
g.map(sb.scatterplot, 'start_hour', 'duration_sec', alpha=0.5)

# Set axis labels and titles
g.set_axis_labels('Hour of Day', 'Duration (sec)')
g.set_titles(col_template="{col_name}", row_template="{row_name}")

# Add a legend
g.add_legend()

# Show the plot
plt.show()
```

ADDITIONAL LINKS

- If you want to learn more about python programming you can follow this link
- 15 Python tips and tricks to master Data Science and Machine Learning

- The ultimate guide to writing better Python code
- Ten Good Coding Practices for Data Scientists
- Six steps to more professional data science code
- Good coding practices - Describing your code

Exploratory Data Analysis

Student has provided seven (7) exploratory data visualizations distributed over univariate, bivariate, and multivariate plots to explore many relationships in the data set.

For each exploration category, complete the required chart(s) and choose one additional visualization type.

Univariate Exploration

1. Histogram - required
2. Bar Chart
3. Count Plot

Bivariate Exploration

1. Scatterplots - required
2. Box Plots - required
3. Clustered Bar Chart
4. Heatmap

Multivariate Exploration

1. Facet Plot - required
2. Plot Matrix
3. Scatterplot with multiple encodings

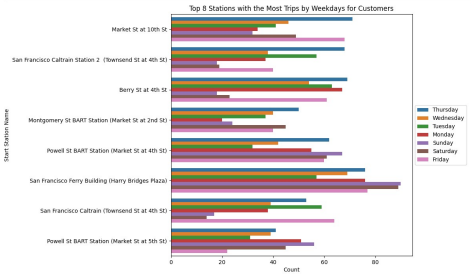
Plots should include relevant annotations, such as; lines denoting the average value of a distribution or the values in each cell of a cluster plot. Appropriate scales, axis labels, axis limits, and encodings should be used.

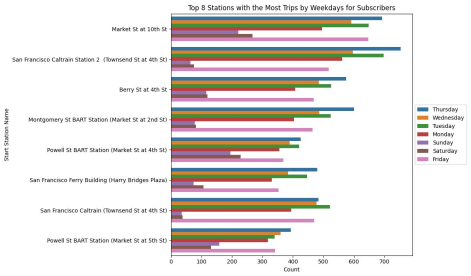
> Tip: Do not overplot or incorrectly plot ordinal data.
> Tip: Be willing to investigate unexpected relationships and don't be afraid of finding a dead-end in your exploration.

CHANGES REQUIRED

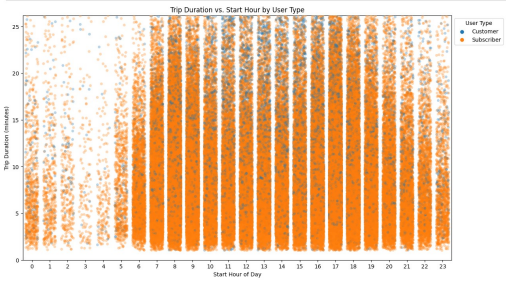
Change-1

- Some of the plots are wrongly placed that is bivariate plots are included in the multivariate section and vice-versa.





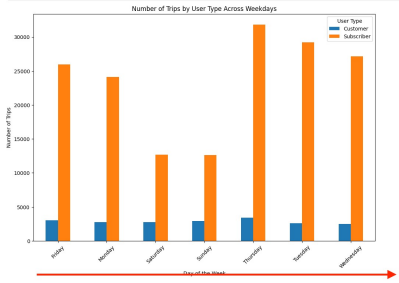
These are bivariate plots as count is not considered as a separate variable. move them to the bivariate section.



This is a multivariate plot move this to the multivariate section

Change-2

- Put the days of the week in order as weekday is an ordinal variable and they have a defined order
- Go through these knowledge questions to know how to get them right.
  1. Knowledge question 1
  2. Knowledge question 2



NOTE
Do this for all the plots where Weekdays are used

Weekdays should be correctly ordered

Change-3

- Since the scatterplot included in the bivariate section is multivariate include another bivariate scatterplot in the bivariate section.

ADDITIONAL LINKS

> The Python Graph Gallery
> How to use Python Seaborn for Exploratory Data Analysis
> Univariate, Bivariate, and Multivariate

- Fundamentals of Data Visualization
- How to avoid overplotting with python

Questions and observations are placed regularly throughout the report.

- For each exploration category (univariate, bivariate, multivariate) state the assumptions and questions the charts should answer.
- After each plot or set of related plots, describe what you found.

> Tip: Use the ""Question-Visualization-Observations"" framework throughout the exploration.
>
> Tip: For the Part I notebook, use *File > Download as... > HTML or PDF* menu option to generate the HTML/PDF.

- great job placing the question in between your exploration. it really helps in going through the flow of the exploration. I was able to follow along, and understand what was going on in your head while you were trying to answer those questions!
- You have added observation after the plots which really helped.

Good job framing questions at the end of the section and answering them based on findings

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

Seems to be subscribers taking interest in bikes and riding more than customers.No transformations needed.

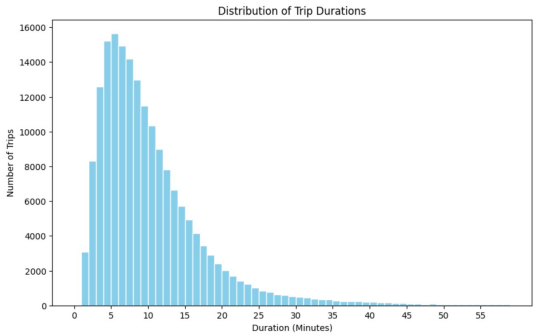
Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

Unusual distributions would be the higher and lower points of the trip start and end times and the amount difference between the two categories of subscribers and customers. Nothing that needs to be changed as of now for this data. But for future data we need to identify based on date and time the day of the week, and the time of day each trip was taken.

Question 1: What is the distribution of trip durations under 1 hour?

Here we identify the lower end of the data showing the amount of trips taken under 1 hour.

```
## Histogram of trip durations
plt.figure(figsize=(10, 6))
plt.hist(fordgobike_df['duration_sec'], bins=np.arange(0, 3600, 60), color='skyblue', edgecolor='white')
plt.title('Distribution of Trip Durations')
plt.xlabel('Duration (Minutes)')
plt.ylabel('Number of Trips')
plt.xticks(np.arange(0, 3600, 300), labels=np.arange(0, 60, 5))
plt.show()
```



Great job using Question-Visualization-Observations framework

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

With this information we see several people had trips under 5 minutes which could indicate that the people either had issues or the distance between the two points were really short. No transformations needed.

Explanatory Data Analysis

The explanatory analysis follows a logical flow and is organized in the following manner:

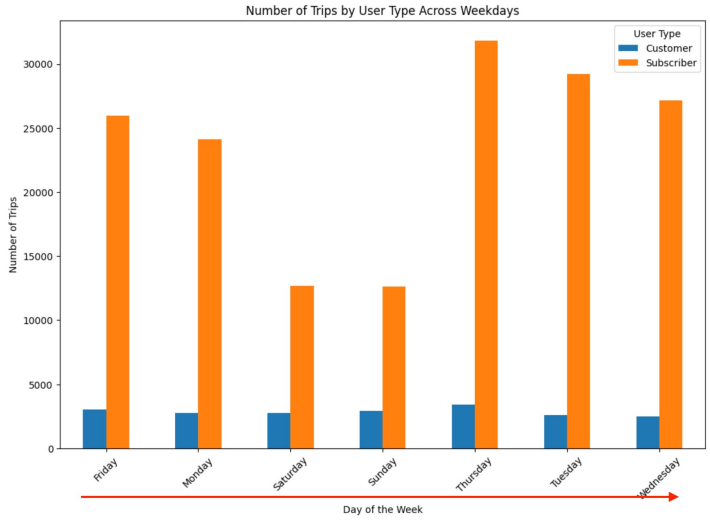
1. Student presents a clear overarching question or theme that was used to guide their data analysis.
2. Student provides some context for the dataset analyzed.
3. Student presents their findings and key insights.
4. Visualizations from their exploratory analysis are used to support findings.
5. Visualizations should be polished with clear axis, labels, and annotations.

- Student presents a clear overarching question or theme that was used to guide their data analysis. ✅
- Student provides some context for the dataset analyzed. ✅
- Student presents their findings and key insights. ✅
- Visualizations from their exploratory analysis are used to support findings. ✅

CHANGES REQUIRED

Change-1

- As suggested in earlier **Weekdays** should be correctly ordered.



Weekdays should be correctly ordered

Change-2

(Visualization 1)

> Write a comment about your visualization here. Make sure your visualizations are polished!
>
> **Rubric Tip**: Provide at least 3 **polished** visualizations to convey key insights. The total number of visualizations shoould be less than 50% of the number of visualizations in the exploratory analysis. Generally 3-5 visualizations suffice.
>
> **Rubric Tip**: Each visualization in the slideshow is associated with **descriptive comments** that accurately depict their purpose and your observation.
>
> **Rubric Tip**: All plots shown are appropriate, meaning the plot type, encodings, and transformations are suitable to the underlying data.
>
> **Rubric Tip**: All plots in the slideshow are polished, meaning all plots have a title, labeled x/y axes (with units), x/y ticks, and legends.

NOT REQUIRED

Submission

If you are using classroom workspace, you can choose from the following two ways of submission:

1. **Submit from the workspace**. Make sure you have removed the example project from the /home/workspace directory. You must submit the following files:
   - Part_I_notebook.ipynb
   - Part_I_notebook.html or pdf
   - Part_II_notebook.ipynb
   - Part_II_notebook.html or pdf
2. **Submit a zip file on the last page of this project lesson**. In this case, open the Jupyter terminal and run the command below to generate a ZIP file.

NOT REQUIRED

```
zip -r my_project.zip .
```

The command abobve will ZIP every file present in your /home/workspace directory. Next, you can download the zip to your local, and follow the instructions on the last page of this project lesson.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

START