

Data 100/200C Project 2A, Part 2 – EDA, Visualization, Feature Engineering

Junsheng Pei

TOTAL POINTS

37 / 37

QUESTION 1

3 2 pts

1.1 3c 2 / 2

✓ + 2 pts long trips tend to end at la guardia more than short trips do. short trips usually centralized around manhattan (something along the lines of this)

+ 0 pts Correct

+ 1 pts Partial Credit for identification of long/short trip patterns.

missing 1 of the responses

+ 0 pts (Vis 2) No credit

2.3 4c 4 / 4

✓ + 4 pts Response is thorough and clearly supported by the visualizations created in 4a

+ 2 pts (Partial) Response is lacking depth or not clearly supported by the visualizations created in 4a

+ 0 pts Otherwise

2.4 4d 2 / 2

✓ + 2 pts Plot resembles solution

+ 0 pts No credit

+ 1 pts Partially correct plot

2.5 4e 2 / 2

✓ + 2 pts Addresses that the weekends have more trips late at night and explains a violin plot clearly / what it visualizes

+ 1 pts Missing one of the required points above / stated something that is true but not one of the above

+ 0 pts Otherwise

QUESTION 2

4 16 pts

2.1 4a 4 / 4

✓ + 2 pts (Vis 1) Properly labeled, well visualized plots relating to temporal features

+ 0 pts (Vis 1) Plot is not labelled/visualization is unclear

+ 0 pts (Vis 1) No credit

✓ + 2 pts (Vis 2) Properly labeled, well visualized plots relating to temporal features

+ 0 pts (Vis 2) Plot is not labelled/visualization is unclear

+ 0 pts (Vis 2) No credit

2.2 4b 4 / 4

✓ + 4 pts All questions answered thoroughly

+ 2 pts (Vis 1) Each of the 3 questions is answered thoroughly

+ 1 pts (Vis 1) Responses are not thoughtful / missing 1 of the responses

+ 0 pts (Vis 1) No credit

+ 2 pts (Vis 2) Each of the 3 questions is answered thoroughly

+ 1 pts (Vis 2) Responses are not thoughtful /

QUESTION 3

5 6 pts

3.1 5a 2 / 2

✓ + 2 pts Conditions on Vender ID and creates a properly labeled plot comparing the 2 vendor ids

+ 0 pts Otherwise

3.2 5b 2 / 2

✓ + 2 pts Response states how conditioning on vender ID influences (or doesn't) the duration and should be supported by graph

+ 0 pts Otherwise

3.3 5c 2 / 2

✓ + 2 pts Indicates whether vendor id use useful or

not and supports argument with proof from the graph

+ 0 pts Otherwise

QUESTION 4

6 11 pts

4.1 6a 3 / 3

✓ + 3 pts Discusses bearing and orientation of manhattan in relation to the rides within Manhattan, notices degrees are 180 degrees apart indicating that taxi rides head up or down manhattan

+ 0 pts Otherwise

4.2 6b 3 / 3

✓ + 3 pts Creates a plot that compares haversine and manhattan distance that is properly labeled, uses the log distribution

+ 0 pts Otherwise

4.3 6c 3 / 3

✓ + 3 pts Backs up why they chose this graph and states if it helps to use manhattan / haversine distance as a feature in predicting trip duration, must correspond to the graph

+ 0 pts Otherwise

4.4 6e 2 / 2

✓ + 2 pts Discusses the duration and manhattan distance outliers

+ 0 pts Otherwise

QUESTION 5

7 2 pts

5.1 7b 2 / 2

✓ + 2 pts Describes how the clustering visualization helps gain insight on speed, comments on whether spatial clustering would be useful in reducing model error

+ 1 pts One of the two responses

+ 0 pts Otherwise

```
In [14]: 1 q3a_answer = r"""
2 The large circle is Manhattan
3 The square is New York Central Park
4 The small circle is LaGuardia Airport.
5 """
6
7 # YOUR CODE HERE
8 #raise NotImplementedError()
9
10 print(q3a_answer)
```

The large circle is Manhattan
The square is New York Central Park
The small circle is LaGuardia Airport.

Question 3b

In each scatter plot above, why are there no points contained within the small rectangular region (towards the top left between the blue points)? Could this be an error/mistake in our data?

```
In [15]: 1 q3b_answer = r"""
2 It's not an error.
3 The rectangular region in New York Central Park. You can't drive inside the park.
4 """
5
6
7 # YOUR CODE HERE
8 #raise NotImplementedError()
9
10 print(q3b_answer)
```

It's not an error.
The rectangular region in New York Central Park. You can't drive inside the park.

Question 3c

What observations/conclusions do you make based on the scatter plots above? In particular, how are trip duration and pickup/dropoff location related?

```
In [16]: 1 q3c_answer = r"""
2
3 Most of pickup and dropoff locations for short duration rides are in Manhattan, and very few of them are outside
4 Manhattan. However, more pickup locations and dropoff location for long duration rides are outside Manhattan,
5 especially for dropoff locations.
6
7 So we can conclude that People New York mainly live in Manhattan.
8 """
9
10 # YOUR CODE HERE
11 #raise NotImplementedError()
12
13 print(q3c_answer)
```

Most of pickup and dropoff locations for short duration rides are in Manhattan, and very few of them are outside Manhattan. However, more pickup locations and dropoff location for long duration rides are outside Manhattan, especially for dropoff locations.

So we can conclude that People New York mainly live in Manhattan.

This confirms that the trips are localized in NYC, with a very strong concentration in Manhattan **and** on the way to LaGuardia Airport. This might give you ideas of relevant features for feature engineering.

Another way to visualize ride coordinates is using a **heat map** (this also helps us avoid overplotting). The following plots count the number of trips for NYC neighborhoods and areas, plotting with the `geopandas` package and these [shapefiles](https://geo.nyu.edu/catalog/nyu_2451_36743) (https://geo.nyu.edu/catalog/nyu_2451_36743) (do not mind the values on the colorbar). If you are curious about how to create the figures below, feel free to check out `geopandas` (<http://geopandas.org/>).

1.1 3c 2 / 2

✓ + 2 pts long trips tend to end at la guardia more than short trips do. short trips usually centralized around manhattan (something along the lines of this)

+ 0 pts Correct

+ 1 pts Partial Credit for identification of long/short trip patterns.

We can utilize the `start_timestamp` column to design a lot of interesting features.

We implement the following temporal (related to time) features using the `add_time_columns` function below.

- month derived from `start_timestamp`.
- `week_of_year` derived from `start_timestamp`.
- `day_of_month` derived from `start_timestamp`.
- `day_of_week` derived from `start_timestamp`.
- hour derived from `start_timestamp`.
- `week_hour` derived from `start_timestamp`.

Note 1: You can use the `dt` attribute of the `start_timestamp` column to convert the entry into a `DateTime` object.

Note 2: We set `df.is_copy = False` to explicitly write back to the original dataframe, `df`, that is being passed into the `add_time_columns` function. Otherwise pandas will complain.

```
In [17]: 1 def add_time_columns(df):
2     """
3         Add temporal features to df
4     """
5     df.is_copy = False
6     df.loc[:, 'month'] = df['tpep_pickup_datetime'].dt.month
7     df.loc[:, 'week_of_year'] = df['tpep_pickup_datetime'].dt.weekofyear
8     df.loc[:, 'day_of_month'] = df['tpep_pickup_datetime'].dt.day
9     df.loc[:, 'day_of_week'] = df['tpep_pickup_datetime'].dt.dayofweek
10    df.loc[:, 'hour'] = df['tpep_pickup_datetime'].dt.hour
11    df.loc[:, 'week_hour'] = df['tpep_pickup_datetime'].dt.weekday * 24 + df['hour']
12
13    # No real need to return here, but we harmonize with remove_outliers for later pipelinezation
14    return df
```

```
In [18]: 1 # Note that we are applying this transformation to train_df, short_rides and long_rides
2 train_df = add_time_columns(train_df)
3 short_rides = add_time_columns(short_rides)
4 long_rides = add_time_columns(long_rides)
```

/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:4388: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a future version.
object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:4389: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a future version.
return object.__setattr__(self, name, value)

```
In [19]: 1 train_df[['month', 'week_of_year', 'day_of_month', 'day_of_week', 'hour', 'week_hour']].head()
```

```
Out[19]:
month week_of_year day_of_month day_of_week hour week_hour
16434 1 3 21 3 17 89
21929 1 4 29 4 23 119
3370 1 1 5 1 18 42
21975 1 4 30 5 0 120
13758 1 3 18 0 13 13
```

Your `train_df.head()` should look like this, although the ordering of the data in `id` might be different:

	month	week_of_year	day_of_month	day_of_week	hour	week_hour
758948	5	19	11	2	18	66
1254646	5	21	26	3	21	93
22560	1	2	12	1	7	31
1552894	2	6	9	1	1	25
1464545	4	14	5	1	1	25

```
In [20]: 1 time_columns = ['month',
2                 'week_of_year',
3                 'day_of_month',
4                 'day_of_week',
5                 'hour',
6                 'week_hour']
7
8
9 # Check columns were created
10 assert all(column in train_df.columns for column in time_columns)
11
12 # Check type
13 assert train_df[time_columns].dtypes.nunique() == 1
14
15 assert train_df[time_columns].dtypes.nunique() == 1
```

Visualizing Temporal Features

Question 4a

Let us now use the features we created to plot some histograms and visualize patterns in our dataset. We will analyze the distribution of the number of taxi rides across months and days of the week. This can help us visualize and understand patterns and trends within our data.

This is an open ended question. Create 2 plots that visualize temporal information from our dataset. At least one of them must visualize the hour of each day. Aside from that you can use any column from `time_columns`.

You can use the same column multiple times, but if the plots are redundant you will not receive full credit. This will be graded based on how informative each plot is and how "good" the visualization is (remember what good/bad visualizations look like for different kinds of data).

Visualization 1

```
In [21]: 1 train_df['tpep_pickup_datetime'].min().weekday()
```

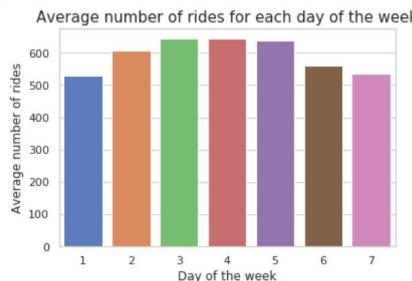
```
Out[21]: 4
```

```
In [22]: 1 len(train_df['day_of_month'].unique())
2
```

```
Out[22]: 31
```

```
In [23]: 1 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
2 number_of_weekday = [0]*7
3 for i in range(31):
4     number_of_weekday[(i+week_of_firstday) % 7] +=1
```

```
In [24]: 1 # Visualization 1
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 # We want to know the average number of rides for each day of the week, for January we have 31 days, 3 days of
5 # the day have 5 days, 4 have 4 days. 31 = 5*3 + 4*4
6 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
7 number_of_weekday = [0]*7
8 for i in range(31):
9     number_of_weekday[(i+week_of_firstday) % 7] +=1
10 train_df['tpep_pickup_datetime'].unique()
11 train_df_groupby_week = train_df.groupby("day_of_week")["day_of_week"].count() / number_of_weekday
12 sns.barplot(x =np.array(train_df_groupby_week.index)+1, y = train_df_groupby_week)
13 plt.xlabel("Day of the week")
14 plt.ylabel('Average number of rides')
15 plt.title("Average number of rides for each day of the week", size = 15)
16 plt.show()
```



Visualization 2

```
In [25]: 1 # Visualization 2
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 train_df_groupby_hour = train_df.groupby('hour')['hour'].count()
5 plt.plot(train_df_groupby_hour, '-.')
6 plt.xlabel("Hour of the day")
7 plt.ylabel('Number of the rides')
8 plt.title('Number of rides for each hour of the day', size =15)
9 plt.show()
```



Question 4b

Briefly explain for each plot

1. What feature you're visualization
2. Why you chose this feature
3. Why you chose this visualization method

```
In [26]: 1 q4b_answer = r"""
2
3 In plot one:
4 1. I want to visualize the average number of riders in each of the week.
5 2. I choose this feature, because we want to show whether different days of the week affect the number of rides,
6 especially the differences between weekdays and weekends.
7 3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day
8 clearly.
9
10 In plot two:
11 1. I want to visualize the total number of riders in each hour of a day.
12 2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most
13 people taking taxis and when there are least people taking taxis
14 3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.
15 """
16
17 # YOUR CODE HERE
18 #raise NotImplementedError()
19
20 print(q4b_answer)
```

In plot one:

1. I want to visualize the average number of riders in each of the week.
2. I choose this feature, because we want to show whether different days of the week affect the number of rides, especially the differences between weekdays and weekends.
3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day clearly.

In plot two:

1. I want to visualize the total number of riders in each hour of a day.
2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most people taking taxis and when there are least people taking taxis
3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.

Question 4c

From the various plots above, what conclusions can you draw about the temporal aspects of our data? How does this relate to duration?

2.1 4a 4 / 4

✓ + 2 pts (Vis 1) Properly labeled, well visualized plots relating to temporal features

+ 0 pts (Vis 1) Plot is not labelled/visualization is unclear

+ 0 pts (Vis 1) No credit

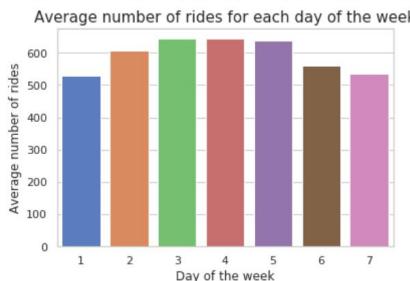
✓ + 2 pts (Vis 2) Properly labeled, well visualized plots relating to temporal features

+ 0 pts (Vis 2) Plot is not labelled/visualization is unclear

+ 0 pts (Vis 2) No credit

```
In [23]: 1 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
2 number_of_weekday = [0]*7
3 for i in range(31):
4     number_of_weekday[(i+week_of_firstday) % 7] +=1
```

```
In [24]: 1 # Visualization 1
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 # We want to know the average number of rides for each day of the week, for January we have 31 days, 3 days of
5 # the day have 5 days, 4 have 4 days. 31 = 5*3 + 4*4
6 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
7 number_of_weekday = [0]*7
8 for i in range(31):
9     number_of_weekday[(i+week_of_firstday) % 7] +=1
10 train_df['tpep_pickup_datetime'].unique()
11 train_df_groupby_week = train_df.groupby("day_of_week")["day_of_week"].count() / number_of_weekday
12 sns.barplot(x =np.array(train_df_groupby_week.index)+1, y = train_df_groupby_week)
13 plt.xlabel("Day of the week")
14 plt.ylabel('Average number of rides')
15 plt.title("Average number of rides for each day of the week", size = 15)
16 plt.show()
```



Visualization 2

```
In [25]: 1 # Visualization 2
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 train_df_groupby_hour = train_df.groupby('hour')['hour'].count()
5 plt.plot(train_df_groupby_hour, '-.')
6 plt.xlabel("Hour of the day")
7 plt.ylabel('Number of the rides')
8 plt.title('Number of rides for each hour of the day', size =15)
9 plt.show()
```



Question 4b

Briefly explain for each plot

1. What feature you're visualization
2. Why you chose this feature
3. Why you chose this visualization method

```
In [26]: 1 q4b_answer = r"""
2
3 In plot one:
4 1. I want to visualize the average number of riders in each of the week.
5 2. I choose this feature, because we want to show whether different days of the week affect the number of rides,
6 especially the differences between weekdays and weekends.
7 3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day
8 clearly.
9
10 In plot two:
11 1. I want to visualize the total number of riders in each hour of a day.
12 2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most
13 people taking taxis and when there are least people taking taxis
14 3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.
15 """
16
17 # YOUR CODE HERE
18 #raise NotImplementedError()
19
20 print(q4b_answer)
```

In plot one:

1. I want to visualize the average number of riders in each of the week.
2. I choose this feature, because we want to show whether different days of the week affect the number of rides, especially the differences between weekdays and weekends.
3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day clearly.

In plot two:

1. I want to visualize the total number of riders in each hour of a day.
2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most people taking taxis and when there are least people taking taxis
3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.

Question 4c

From the various plots above, what conclusions can you draw about the temporal aspects of our data? How does this relate to duration?

2.2 4b 4 / 4

✓ + 4 pts All questions answered thoroughly

+ 2 pts (Vis 1) Each of the 3 questions is answered thoroughly

+ 1 pts (Vis 1) Responses are not thoughtful / missing 1 of the responses

+ 0 pts (Vis 1) No credit

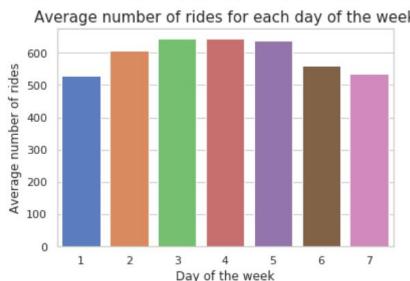
+ 2 pts (Vis 2) Each of the 3 questions is answered thoroughly

+ 1 pts (Vis 2) Responses are not thoughtful / missing 1 of the responses

+ 0 pts (Vis 2) No credit

```
In [23]: 1 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
2 number_of_weekday = [0]*7
3 for i in range(31):
4     number_of_weekday[(i+week_of_firstday) % 7] +=1
```

```
In [24]: 1 # Visualization 1
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 # We want to know the average number of rides for each day of the week, for January we have 31 days, 3 days of
5 # the day have 5 days, 4 have 4 days. 31 = 5*3 + 4*4
6 week_of_firstday = train_df['tpep_pickup_datetime'].min().weekday()
7 number_of_weekday = [0]*7
8 for i in range(31):
9     number_of_weekday[(i+week_of_firstday) % 7] +=1
10 train_df['tpep_pickup_datetime'].unique()
11 train_df_groupby_week = train_df.groupby("day_of_week")["day_of_week"].count() / number_of_weekday
12 sns.barplot(x =np.array(train_df_groupby_week.index)+1, y = train_df_groupby_week)
13 plt.xlabel("Day of the week")
14 plt.ylabel('Average number of rides')
15 plt.title("Average number of rides for each day of the week", size = 15)
16 plt.show()
```



Visualization 2

```
In [25]: 1 # Visualization 2
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4 train_df_groupby_hour = train_df.groupby('hour')['hour'].count()
5 plt.plot(train_df_groupby_hour, '-.')
6 plt.xlabel("Hour of the day")
7 plt.ylabel('Number of the rides')
8 plt.title('Number of rides for each hour of the day', size =15)
9 plt.show()
```



Question 4b

Briefly explain for each plot

1. What feature you're visualization
2. Why you chose this feature
3. Why you chose this visualization method

```
In [26]: 1 q4b_answer = r"""
2
3 In plot one:
4 1. I want to visualize the average number of riders in each of the week.
5 2. I choose this feature, because we want to show whether different days of the week affect the number of rides,
6 especially the differences between weekdays and weekends.
7 3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day
8 clearly.
9
10 In plot two:
11 1. I want to visualize the total number of riders in each hour of a day.
12 2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most
13 people taking taxis and when there are least people taking taxis
14 3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.
15 """
16
17 # YOUR CODE HERE
18 #raise NotImplementedError()
19
20 print(q4b_answer)
```

In plot one:

1. I want to visualize the average number of riders in each of the week.
2. I choose this feature, because we want to show whether different days of the week affect the number of rides, especially the differences between weekdays and weekends.
3. I use bar plot because there are only 7 days in a week, and we can show the average number of each day clearly.

In plot two:

1. I want to visualize the total number of riders in each hour of a day.
2. I choose this feature because I want to see the trend of taking a taxi across a day like when there are most people taking taxis and when there are least people taking taxis
3. I use line plot because it's easy to see the changes of the numbers of rides in 24 hours.

Question 4c

From the various plots above, what conclusions can you draw about the temporal aspects of our data? How does this relate to duration?

```
In [27]: 1 q4c_answer = r"""
2
3 In plot one:
4     There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday,
5     Wednesday and Thursdays.
6
7 In plot two:
8     There most people who will take taxis at 8pm, and least people at 4-5am
9
10 It's hard to relate to duration because we only count data for pickup time.
11
12 """
13
14 # YOUR CODE HERE
15 #raise NotImplementedError()
16
17 print(q4c_answer)
```

In plot one:
 There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday, Wednesday and Thursdays.

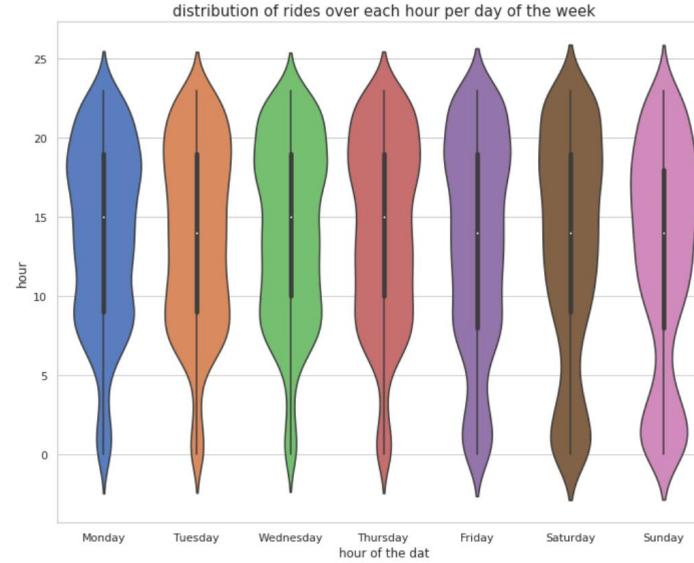
In plot two:
 There most people who will take taxis at 8pm, and least people at 4-5am

It's hard to relate to duration because we only count data for pickup time.

Question 4d

Previously, we have analyzed the temporal features hour and day_of_week independently, but these features may in fact have a relationship between each other. Determining the extent to their relationship may be useful in helping us create new features in our model. Create a violin plot that displays distribution of rides over each hour per day of the week.

```
In [28]: 1 fig, axes = plt.subplots(1, 1, figsize=(10, 8))
2 days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
3
4
5 # YOUR CODE HERE
6
7 sns.violinplot(x = train_df['day_of_week'], y = train_df['hour'])
8 plt.xticks(range(7), days_of_week)
9 plt.xlabel("day of the week")
10 plt.xlabel('hour of the dat')
11 plt.title('distribution of rides over each hour per day of the week',size = 15)
12 #raise NotImplementedError()
13 plt.tight_layout()
14 plt.show()
```



Question 4e

Do you notice anything interesting about your visualization? How would you explain this plot to a lay person? What are the features/patterns of interest?

```
In [29]: 1 q4e_answer = r"""
2 The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more
3 similar to each other. The shapes of Friday to Sunday are more similar to each other.
4
5 I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour
6 one day of the week. The wider the part of violin is, the more people will take taxis at this time.
7
8 All the violins are narrowest at around 4 am, and widest at around 8pm.
9 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.
10
11 """
12
13 # YOUR CODE HERE
14 #raise NotImplementedError()
15
16 print(q4e_answer)
```

The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more similar to each other. The shapes of Friday to Sunday are more similar to each other.

I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour one day of the week. The wider the part of violin is, the more people will take taxis at this time.

All the violins are narrowest at around 4 am, and widest at around 8pm.
 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.

5: Vendors

Recall that in Part 1, we found that there are only two unique vendors represented in the dataset. We may wonder if the vendor feature can be useful when trying to understand taxi ride duration.

2.3 4c 4 / 4

✓ + 4 pts Response is thorough and clearly supported by the visualizations created in 4a

+ 2 pts (Partial) Response is lacking depth or not clearly supported by the visualizations created in 4a

+ 0 pts Otherwise

```
In [27]: 1 q4c_answer = r"""
2
3 In plot one:
4     There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday,
5     Wednesday and Thursdays.
6
7 In plot two:
8     There most people who will take taxis at 8pm, and least people at 4-5am
9
10 It's hard to relate to duration because we only count data for pickup time.
11
12 """
13
14 # YOUR CODE HERE
15 #raise NotImplementedError()
16
17 print(q4c_answer)
```

In plot one:
 There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday, Wednesday and Thursdays.

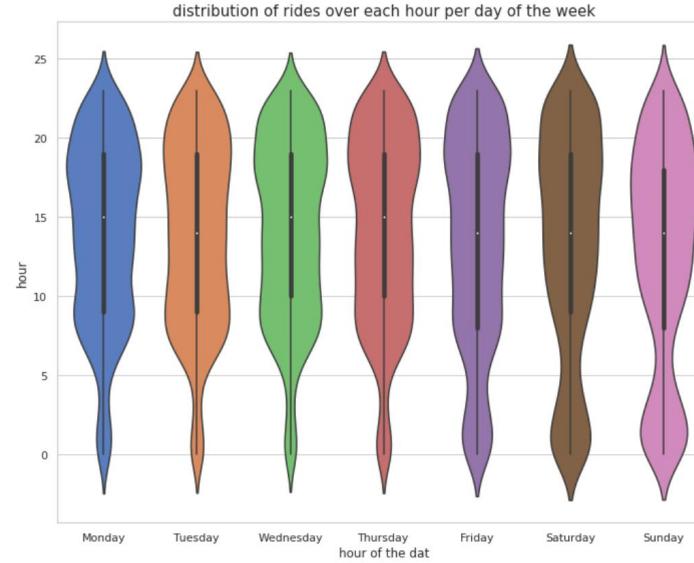
In plot two:
 There most people who will take taxis at 8pm, and least people at 4-5am

It's hard to relate to duration because we only count data for pickup time.

Question 4d

Previously, we have analyzed the temporal features hour and day_of_week independently, but these features may in fact have a relationship between each other. Determining the extent to their relationship may be useful in helping us create new features in our model. Create a violin plot that displays distribution of rides over each hour per day of the week.

```
In [28]: 1 fig, axes = plt.subplots(1, 1, figsize=(10, 8))
2 days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
3
4
5 # YOUR CODE HERE
6
7 sns.violinplot(x = train_df['day_of_week'], y = train_df['hour'])
8 plt.xticks(range(7), days_of_week)
9 plt.xlabel("day of the week")
10 plt.xlabel('hour of the dat')
11 plt.title('distribution of rides over each hour per day of the week',size = 15)
12 #raise NotImplementedError()
13 plt.tight_layout()
14 plt.show()
```



Question 4e

Do you notice anything interesting about your visualization? How would you explain this plot to a lay person? What are the features/patterns of interest?

```
In [29]: 1 q4e_answer = r"""
2 The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more
3 similar to each other. The shapes of Friday to Sunday are more similar to each other.
4
5 I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour
6 one day of the week. The wider the part of violin is, the more people will take taxis at this time.
7
8 All the violins are narrowest at around 4 am, and widest at around 8pm.
9 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.
10
11 """
12
13 # YOUR CODE HERE
14 #raise NotImplementedError()
15
16 print(q4e_answer)
```

The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more similar to each other. The shapes of Friday to Sunday are more similar to each other.

I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour one day of the week. The wider the part of violin is, the more people will take taxis at this time.

All the violins are narrowest at around 4 am, and widest at around 8pm.
 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.

5: Vendors

Recall that in Part 1, we found that there are only two unique vendors represented in the dataset. We may wonder if the vendor feature can be useful when trying to understand taxi ride duration.

2.4 4d 2 / 2

✓ + 2 pts Plot resembles solution

+ 0 pts No credit

+ 1 pts Partially correct plot

```
In [27]: 1 q4c_answer = r"""
2
3 In plot one:
4     There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday,
5     Wednesday and Thursdays.
6
7 In plot two:
8     There most people who will take taxis at 8pm, and least people at 4-5am
9
10 It's hard to relate to duration because we only count data for pickup time.
11
12 """
13
14 # YOUR CODE HERE
15 #raise NotImplementedError()
16
17 print(q4c_answer)
```

In plot one:
 There are more people that will take taxis on weekdays than weekends. And people take most taxis on Tuesday, Wednesday and Thursdays.

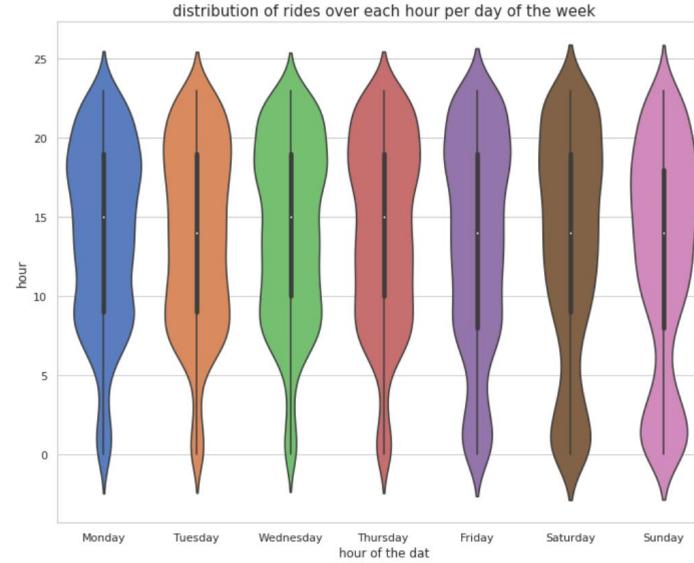
In plot two:
 There most people who will take taxis at 8pm, and least people at 4-5am

It's hard to relate to duration because we only count data for pickup time.

Question 4d

Previously, we have analyzed the temporal features hour and day_of_week independently, but these features may in fact have a relationship between each other. Determining the extent to their relationship may be useful in helping us create new features in our model. Create a violin plot that displays distribution of rides over each hour per day of the week.

```
In [28]: 1 fig, axes = plt.subplots(1, 1, figsize=(10, 8))
2 days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
3
4
5 # YOUR CODE HERE
6
7 sns.violinplot(x = train_df['day_of_week'], y = train_df['hour'])
8 plt.xticks(range(7), days_of_week)
9 plt.xlabel("day of the week")
10 plt.xlabel('hour of the dat')
11 plt.title('distribution of rides over each hour per day of the week', size = 15)
12 #raise NotImplementedError()
13 plt.tight_layout()
14 plt.show()
```



Question 4e

Do you notice anything interesting about your visualization? How would you explain this plot to a lay person? What are the features/patterns of interest?

```
In [29]: 1 q4e_answer = r"""
2 The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more
3 similar to each other. The shapes of Friday to Sunday are more similar to each other.
4
5 I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour
6 one day of the week. The wider the part of violin is, the more people will take taxis at this time.
7
8 All the violins are narrowest at around 4 am, and widest at around 8pm.
9 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.
10
11 """
12
13 # YOUR CODE HERE
14 #raise NotImplementedError()
15
16 print(q4e_answer)
```

The shapes of the distributions for 7 days in the week are similar. And the shapes of Monday to Thursday are more similar to each other. The shapes of Friday to Sunday are more similar to each other.

I will tell the lay person that each violin correspond to a distribution distribution of rides over each hour one day of the week. The wider the part of violin is, the more people will take taxis at this time.

All the violins are narrowest at around 4 am, and widest at around 8pm.
 For Friday, Saturday and Sunday, the violins are more wider from 0am to 5 am.

5: Vendors

Recall that in Part 1, we found that there are only two unique vendors represented in the dataset. We may wonder if the vendor feature can be useful when trying to understand taxi ride duration.

2.5 4e 2 / 2

✓ + 2 pts Addresses that the weekends have more trips late at night and explains a violin plot clearly / what it visualizes

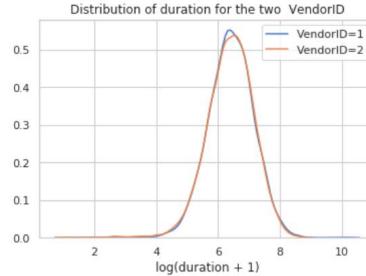
+ 1 pts Missing one of the required points above / stated something that is true but not one of the above

+ 0 pts Otherwise

Question 5a

Visualize the VendorID feature. Create at least one plot that gives insight as to whether this feature would be useful or not in our model.

```
In [30]: 1 # Visualization
2
3 # YOUR CODE HERE
4 sns.distplot(np.log(train_df[train_df['VendorID'] == 2]['duration']+1),hist=False,label= 'VendorID=1')
5 sns.distplot(np.log(train_df[train_df['VendorID'] == 1]['duration']+1),hist=False,label= 'VendorID=2')
6 plt.xlabel('log(duration + 1)')
7 plt.title("Distribution of duration for the two VendorID")
8 plt.show()
9 #raise NotImplementedError()
```

**Question 5b**

Justify why you chose this visualization method and how it helps determine whether `vendor_id` is useful in our model or not.

```
In [31]: 1 q5b_answer = r"""
2
3 I chose KDE distribution to visualize because we care about the proportion of each duration to see the
4 distribution. And we don't care about the actual number for each duration, so we won't use histogram
5
6 From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine
7 that vendor_id is not useful in our model.
8 """
9
10
11 # YOUR CODE HERE
12 #raise NotImplementedError()
13
14 print(q5b_answer)
```

I chose KDE distribution to visualize because we care about the proportion of each duration to see the distribution. And we don't care about the actual number for each duration, so we won't use histogram

From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine that vendor_id is not useful in our model.

Question 5c

From the plot above, do you think vendor_id will help us understand duration? Why or why not?

```
In [32]: 1 q5c_answer = r"""
2
3 I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution
4 of duration.
5 """
6
7
8 # YOUR CODE HERE
9 #raise NotImplementedError()
10
11 print(q5c_answer)
```

I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution of duration.

6: Distance features

We can also use the coordinates information to compute distance features. This will allow us to compute speed related features.

We will compute the [haversine](https://en.wikipedia.org/wiki/Haversine_formula) (https://en.wikipedia.org/wiki/Haversine_formula) distance, the [manhattan](https://en.wikipedia.org/wiki/Taxicab_geometry) (https://en.wikipedia.org/wiki/Taxicab_geometry) distance and the [bearing](http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) (http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) angle.

3.1.5a 2 / 2

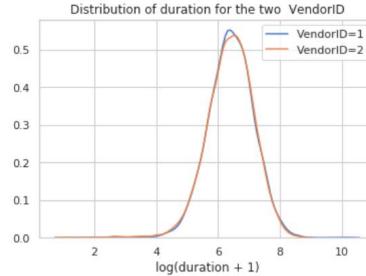
✓ + 2 pts Conditions on Vender ID and creates a properly labeled plot comparing the 2 vendor ids

+ 0 pts Otherwise

Question 5a

Visualize the VendorID feature. Create at least one plot that gives insight as to whether this feature would be useful or not in our model.

```
In [30]: 1 # Visualization
2
3 # YOUR CODE HERE
4 sns.distplot(np.log(train_df[train_df['VendorID'] == 2]['duration']+1),hist=False,label= 'VendorID=1')
5 sns.distplot(np.log(train_df[train_df['VendorID'] == 1]['duration']+1),hist=False,label= 'VendorID=2')
6 plt.xlabel('log(duration + 1)')
7 plt.title("Distribution of duration for the two VendorID")
8 plt.show()
9 #raise NotImplementedError()
```

**Question 5b**

Justify why you chose this visualization method and how it helps determine whether vendor_id is useful in our model or not.

```
In [31]: 1 q5b_answer = r"""
2
3 I chose KDE distribution to visualize because we care about the proportion of each duration to see the
4 distribution. And we don't care about the actual number for each duration, so we won't use histogram
5
6 From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine
7 that vendor_id is not useful in our model.
8 """
9
10
11 # YOUR CODE HERE
12 #raise NotImplementedError()
13
14 print(q5b_answer)
```

I chose KDE distribution to visualize because we care about the proportion of each duration to see the distribution. And we don't care about the actual number for each duration, so we won't use histogram

From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine that vendor_id is not useful in our model.

Question 5c

From the plot above, do you think vendor_id will help us understand duration? Why or why not?

```
In [32]: 1 q5c_answer = r"""
2
3 I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution
4 of duration.
5 """
6
7
8 # YOUR CODE HERE
9 #raise NotImplementedError()
10
11 print(q5c_answer)
```

I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution of duration.

6: Distance features

We can also use the coordinates information to compute distance features. This will allow us to compute speed related features.

We will compute the [haversine](https://en.wikipedia.org/wiki/Haversine_formula) (https://en.wikipedia.org/wiki/Haversine_formula) distance, the [manhattan](https://en.wikipedia.org/wiki/Taxicab_geometry) (https://en.wikipedia.org/wiki/Taxicab_geometry) distance and the [bearing](http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) (http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) angle.

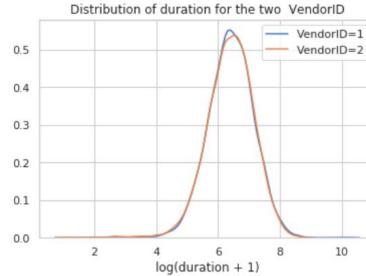
3.2 5b 2 / 2

- ✓ + 2 pts Response states how conditioning on vendor ID influences (or doesn't) the duration and should be supported by graph
- + 0 pts Otherwise

Question 5a

Visualize the VendorID feature. Create at least one plot that gives insight as to whether this feature would be useful or not in our model.

```
In [30]: 1 # Visualization
2
3 # YOUR CODE HERE
4 sns.distplot(np.log(train_df[train_df['VendorID'] == 2]['duration']+1),hist=False,label= 'VendorID=1')
5 sns.distplot(np.log(train_df[train_df['VendorID'] == 1]['duration']+1),hist=False,label= 'VendorID=2')
6 plt.xlabel('log(duration + 1)')
7 plt.title("Distribution of duration for the two VendorID")
8 plt.show()
9 #raise NotImplementedError()
```

**Question 5b**

Justify why you chose this visualization method and how it helps determine whether `vendor_id` is useful in our model or not.

```
In [31]: 1 q5b_answer = r"""
2
3 I chose KDE distribution to visualize because we care about the proportion of each duration to see the
4 distribution. And we don't care about the actual number for each duration, so we won't use histogram
5
6 From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine
7 that vendor_id is not useful in our model.
8 """
9
10
11 # YOUR CODE HERE
12 #raise NotImplementedError()
13
14 print(q5b_answer)
```

I chose KDE distribution to visualize because we care about the proportion of each duration to see the distribution. And we don't care about the actual number for each duration, so we won't use histogram

From the distribution plot of the two VendorIDs, we can see the plots are mostly overlapped. So we can determine that vendor_id is not useful in our model.

Question 5c

From the plot above, do you think vendor_id will help us understand duration? Why or why not?

```
In [32]: 1 q5c_answer = r"""
2
3 I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution
4 of duration.
5 """
6
7
8 # YOUR CODE HERE
9 #raise NotImplementedError()
10
11 print(q5c_answer)
```

I don't think vendor_if will help us understand duration. Because both the two vendor ids have the distribution of duration.

6: Distance features

We can also use the coordinates information to compute distance features. This will allow us to compute speed related features.

We will compute the [haversine](https://en.wikipedia.org/wiki/Haversine_formula) (https://en.wikipedia.org/wiki/Haversine_formula) distance, the [manhattan](https://en.wikipedia.org/wiki/Taxicab_geometry) (https://en.wikipedia.org/wiki/Taxicab_geometry) distance and the [bearing](http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) (http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) angle.

3.3 5C 2 / 2

✓ + 2 pts Indicates whether vendor id use useful or not and supports argument with proof from the graph

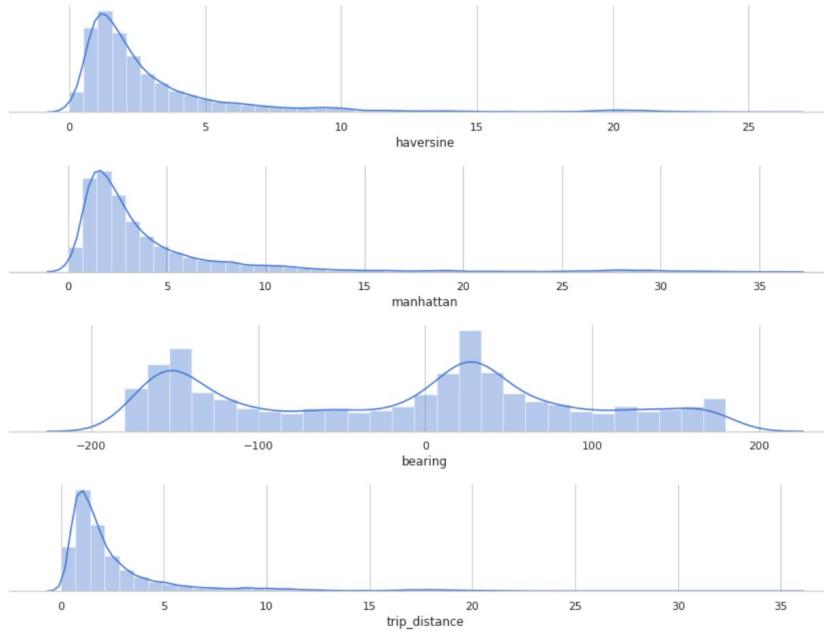
+ 0 pts Otherwise

```
In [33]: 1 # These functions are implemented for you
2 def haversine(lat1, lng1, lat2, lng2):
3     """
4         Compute haversine distance
5
6         The haversine formula determines the great-circle distance between two points
7         on a sphere given their longitudes and latitudes. Important in navigation, it
8         is a special case of a more general formula in spherical trigonometry,
9         the law of haversines, that relates the sides and angles of spherical triangles.
10        """
11    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
12    average_earth_radius = 6371
13    lat = lat2 - lat1
14    lng = lng2 - lng1
15    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng * 0.5) ** 2
16    h = 2 * average_earth_radius * np.arcsin(np.sqrt(d))
17    return h
18
19 def manhattan_distance(lat1, lng1, lat2, lng2):
20     """
21         Computes Manhattan distance
22
23         The name alludes to the grid layout of most streets on the island of Manhattan,
24         which causes the shortest path a car could take between two intersections in the borough
25         to have length equal to the intersections' distance in taxicab geometry.
26        """
27    a = haversine(lat1, lng1, lat1, lng2)
28    b = haversine(lat1, lng1, lat2, lng1)
29    return a + b
30
31 def bearing(lat1, lng1, lat2, lng2):
32     """
33         Compute the bearing, or angle, from (lat1, lng1) to (lat2, lng2).
34         A bearing of 0 refers to a NORTH orientation.
35        """
36    lng_delta_rad = np.radians(lng2 - lng1)
37    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
38    y = np.sin(lng_delta_rad) * np.cos(lat2)
39    x = np.cos(lat1) * np.sin(lat2) - np.sin(lat1) * np.cos(lat2) * np.cos(lng_delta_rad)
40    return np.degrees(np.arctan2(y, x))
```

```
In [34]: 1 def add_distance_columns(df):
2     df.loc[:, 'manhattan'] = manhattan_distance(lat1=df['pickup_latitude'],
3                                                 lng1=df['pickup_longitude'],
4                                                 lat2=df['dropoff_latitude'],
5                                                 lng2=df['dropoff_longitude'])
6
7     df.loc[:, 'bearing'] = bearing(lat1=df['pickup_latitude'],
8                                     lng1=df['pickup_longitude'],
9                                     lat2=df['dropoff_latitude'],
10                                    lng2=df['dropoff_longitude'])
11
12     df.loc[:, 'haversine'] = haversine(lat1=df['pickup_latitude'],
13                                       lng1=df['pickup_longitude'],
14                                       lat2=df['dropoff_latitude'],
15                                       lng2=df['dropoff_longitude'])
16
17     return df
```

```
In [35]: 1 train_df = add_distance_columns(train_df)
2 short_rides = add_distance_columns(short_rides)
3 long_rides = add_distance_columns(long_rides)
```

```
In [36]: 1 fig, axes = plt.subplots(4, 1, figsize=(12, 9))
2 sns.distplot(train_df['haversine'], ax=axes[0], xlabel='haversine');
3 sns.distplot(train_df['manhattan'], ax=axes[1], xlabel='manhattan');
4 sns.distplot(train_df['bearing'], ax=axes[2], xlabel='bearing');
5 sns.distplot(train_df['trip_distance'], ax=axes[3], xlabel='trip_distance');
6
7 sns.despine(left=True);
8 plt.setp(axes, yticks[]);
9 plt.tight_layout();
```



Question 6a

The `bearing` direction is angle, the initial direction of the trip.

The bearing direction has two prominent peaks around 30 and -150 degrees.

Can you relate these peaks to the orientation of Manhattan? What do you notice about these angles?

Hint: This [wikipedia article](https://en.wikipedia.org/wiki/Commissioners%27_Plan_of_1811) (https://en.wikipedia.org/wiki/Commissioners%27_Plan_of_1811) has the answer, although it may take some digging. Alternatively, try to look at a map of Manhattan.

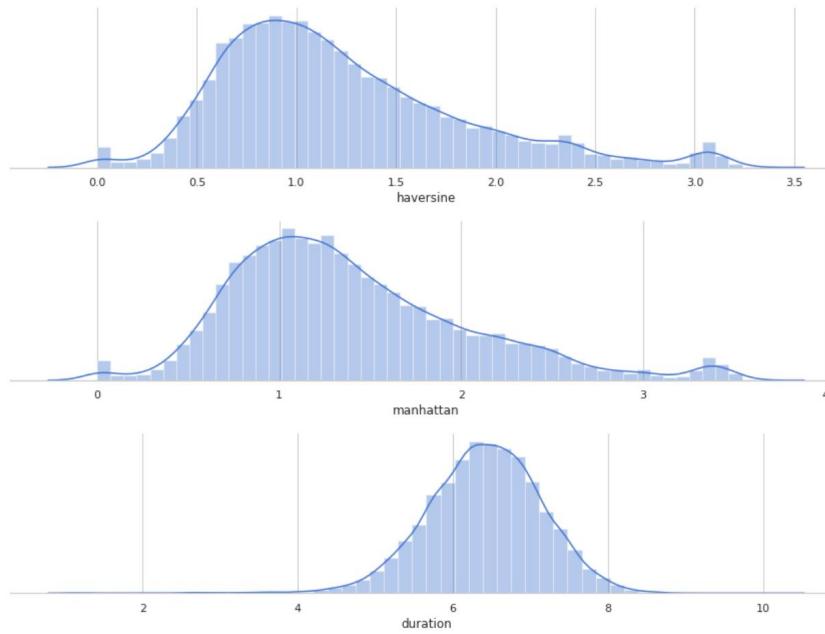
```
In [37]: 1 q6a_answer = r"""
2
3 As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues
4 also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of
5 Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.
6
7 """
8
9 # YOUR CODE HERE
10 #raise NotImplementedError()
11
12 print(q6a_answer)
```

As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.

Question 6b

For haversine and manhattan distances, it is probably more helpful to look at the log distribution. We are also curious about whether these distance features can help us understand duration. Create at least one plot that compares haversine and manhattan distances and gives insight as to whether this would be a useful feature in our model.

```
In [38]: 1 # Visualization
2 # YOUR CODE HERE
3 fig, axes = plt.subplots(3, 1, figsize=(12, 9))
4 sns.distplot(np.log(train_df['haversine']+1), ax=axes[0], xlabel='haversine');
5 sns.distplot(np.log(train_df['manhattan']+1), ax=axes[1], xlabel='manhattan');
6 sns.distplot(np.log(train_df['duration']+1), ax=axes[2], xlabel='duration');
7
8 sns.despine(left=True);
9 plt.setp(axes, yticks[]);
10 plt.tight_layout();
11 #raise NotImplementedError()
```



Question 6c

Justify why you chose this visualization method and how it helps inform you about using manhattan/haversine distance as a feature for predicting trip duration.

```
In [39]: 1 q6c_answer = r"""
2 I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing
3 the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those
4 distributions more clearly.
5 From the above plot, we can see that the distribution of haversine and that of manhattan distances are
6 pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either
7 one of harversine and manhattan distances for predicting trip duration
8
9 """
10
11
12 # YOUR CODE HERE
13 #raise NotImplementedError()
14
15 print(q6c_answer)
```

I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those distributions more clearly.
From the above plot, we can see that the distribution of haversine and that of manhattan distances are pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either one of harversine and manhattan distances for predicting trip duration

Question 6d

Fill in the code below to plot a scatter plot of manhattan distance vs duration.

4.1 6a 3 / 3

- ✓ + 3 pts Discusses bearing and orientation of manhattan in relation to the rides within Manhattan, notices degrees are 180 degrees apart indicating that taxi rides head up or down manhattan
- + 0 pts Otherwise

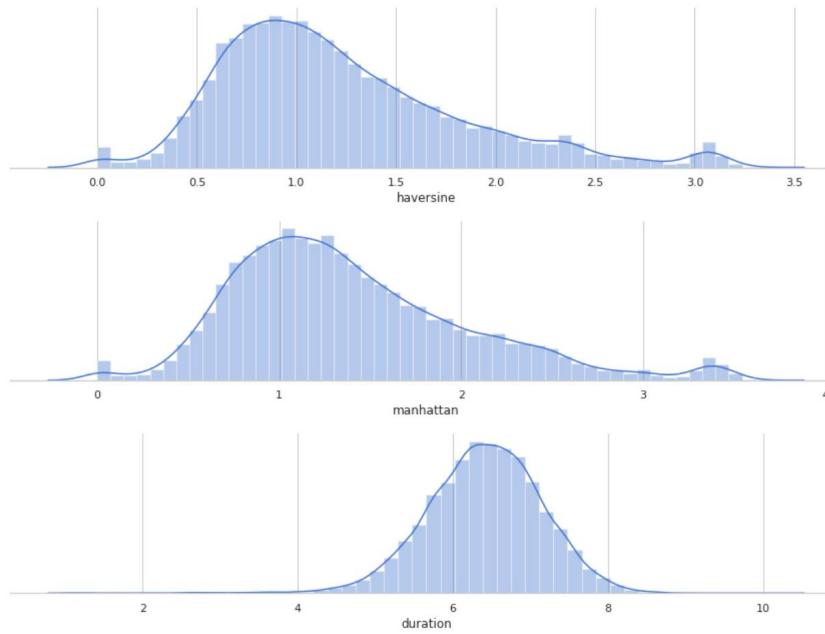
```
In [37]: 1 q6a_answer = r"""
2
3 As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues
4 also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of
5 Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.
6
7 """
8
9 # YOUR CODE HERE
10 #raise NotImplementedError()
11
12 print(q6a_answer)
```

As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.

Question 6b

For haversine and manhattan distances, it is probably more helpful to look at the log distribution. We are also curious about whether these distance features can help us understand duration. Create at least one plot that compares haversine and manhattan distances and gives insight as to whether this would be a useful feature in our model.

```
In [38]: 1 # Visualization
2 # YOUR CODE HERE
3 fig, axes = plt.subplots(3, 1, figsize=(12, 9))
4 sns.distplot(np.log(train_df['haversine']+1), ax=axes[0], xlabel='haversine');
5 sns.distplot(np.log(train_df['manhattan']+1), ax=axes[1], xlabel='manhattan');
6 sns.distplot(np.log(train_df['duration']+1), ax=axes[2], xlabel='duration');
7
8 sns.despine(left=True);
9 plt.setp(axes, yticks[]);
10 plt.tight_layout();
11 #raise NotImplementedError()
```



Question 6c

Justify why you chose this visualization method and how it helps inform you about using manhattan/haversine distance as a feature for predicting trip duration.

```
In [39]: 1 q6c_answer = r"""
2 I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing
3 the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those
4 distributions more clearly.
5 From the above plot, we can see that the distribution of haversine and that of manhattan distances are
6 pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either
7 one of harversine and manhattan distances for predicting trip duration
8
9 """
10
11
12 # YOUR CODE HERE
13 #raise NotImplementedError()
14
15 print(q6c_answer)
```

I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those distributions more clearly.
From the above plot, we can see that the distribution of haversine and that of manhattan distances are pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either one of harversine and manhattan distances for predicting trip duration

Question 6d

Fill in the code below to plot a scatter plot of manhattan distance vs duration.

4.2 6b 3 / 3

- ✓ + 3 pts Creates a plot that compares haversine and manhattan distance that is properly labeled, uses the log distribution
- + 0 pts Otherwise

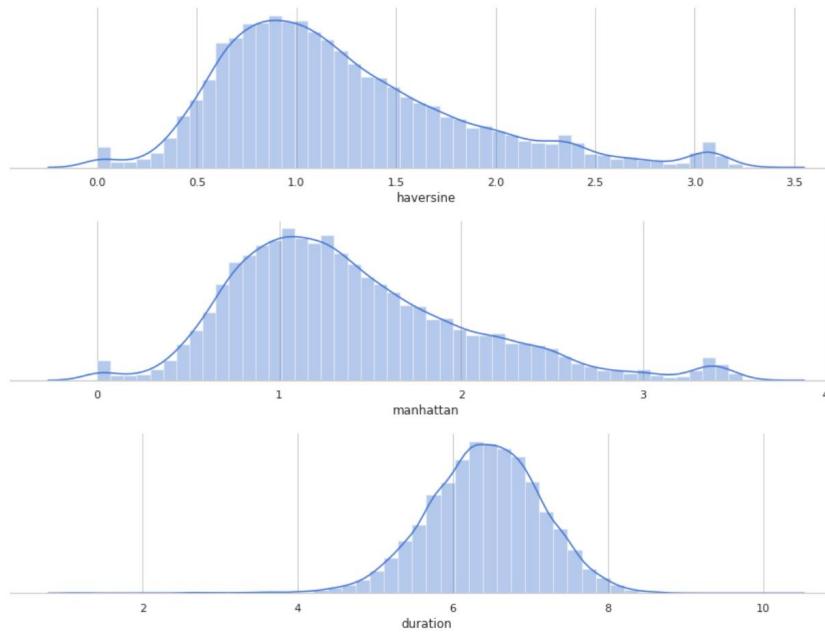
```
In [37]: 1 q6a_answer = r"""
2
3 As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues
4 also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of
5 Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.
6
7 """
8
9 # YOUR CODE HERE
10 #raise NotImplementedError()
11
12 print(q6a_answer)
```

As we can observe Manhattan in the map, the peninsula has an orientation around 30 degrees. And the main avenues also have an orientation around 30 degrees. In fact 30 and -150 degrees correspond to the orientation of Manhattan. So it's most strange that The bearing direction has two prominent peaks around 30 and -150 degrees.

Question 6b

For haversine and manhattan distances, it is probably more helpful to look at the log distribution. We are also curious about whether these distance features can help us understand duration. Create at least one plot that compares haversine and manhattan distances and gives insight as to whether this would be a useful feature in our model.

```
In [38]: 1 # Visualization
2 # YOUR CODE HERE
3 fig, axes = plt.subplots(3, 1, figsize=(12, 9))
4 sns.distplot(np.log(train_df['haversine']+1), ax=axes[0], xlabel='haversine');
5 sns.distplot(np.log(train_df['manhattan']+1), ax=axes[1], xlabel='manhattan');
6 sns.distplot(np.log(train_df['duration']+1), ax=axes[2], xlabel='duration');
7
8 sns.despine(left=True);
9 plt.setp(axes, yticks[]);
10 plt.tight_layout();
11 #raise NotImplementedError()
```



Question 6c

Justify why you chose this visualization method and how it helps inform you about using manhattan/haversine distance as a feature for predicting trip duration.

```
In [39]: 1 q6c_answer = r"""
2 I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing
3 the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those
4 distributions more clearly.
5 From the above plot, we can see that the distribution of haversine and that of manhattan distances are
6 pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either
7 one of harversine and manhattan distances for predicting trip duration
8
9 """
10
11
12 # YOUR CODE HERE
13 #raise NotImplementedError()
14
15 print(q6c_answer)
```

I chose histograms and KDE to visualize our distances and duration, and made a comparison. Instead of viewing the raw data, we show them as log distribution, in this way we can solve the right skew problem and compare those distributions more clearly.
From the above plot, we can see that the distribution of haversine and that of manhattan distances are pretty similar. And They both have an analogous distribution as the distribution of duration. So we can use either one of harversine and manhattan distances for predicting trip duration

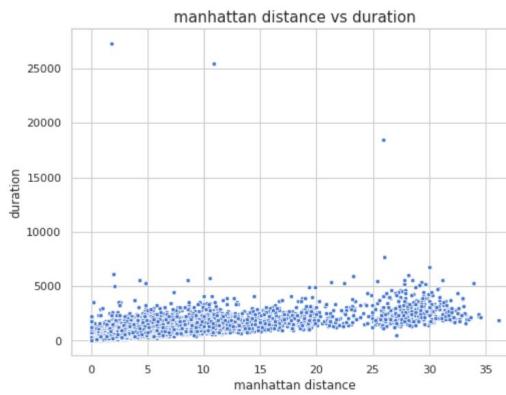
Question 6d

Fill in the code below to plot a scatter plot of manhattan distance vs duration.

4.3 6C 3 / 3

- ✓ + 3 pts Backs up why they chose this graph and states if it helps to use manhattan / haversine distance as a feature in predicting trip duration, must correspond to the graph
- + 0 pts Otherwise

```
In [40]: 1 # YOUR CODE HERE
2 plt.figure(figsize=(8,6))
3 sns.scatterplot(train_df['manhattan'], train_df['duration'], s = 20)
4 plt.xlabel('manhattan distance')
5 plt.ylabel('duration')
6 plt.title('manhattan distance vs duration', size = 15)
7 plt.show()
8 #raise NotImplementedError()
```



Question 6

According to the plot above, there are a few outliers in both duration and manhattan distance. Which type of outliers is most likely to be a mistake in our data?

```
In [41]: 1 q6e_answer = r"""
2 The outliers of duration are most likely to be mistake. For some points, The manhattan distances are not larger,
3 however the durations are more than 25000 seconds, almost 8 hours, it is very unusual.
4 """
5
6 # YOUR CODE HERE
7 #raise NotImplementedError()
8
9 print(q6e_answer)
```

The outliers of duration are most likely to be mistake. For some points, The manhattan distances are not larger, however the durations are more than 25000 seconds, almost 8 hours, it is very unusual.

7: Advanced features

You do not need to incorporate these features into your model, although it may help lower your error. You are required to read through this portion and respond to the questions. All of the code is provided, please skim through it and try to understand what each cell is doing.

Clustering

Clustering (https://en.wikipedia.org/wiki/Cluster_analysis) is the task of grouping objects such that members within each group are more similar to each other than members of other groups. Clustering is a powerful tool used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. Recall cluster sampling, which we learned earlier in the semester. We will use a simple clustering method (clustering by spatial locality) to reveal some more advanced features.

Speed features

For `train_df`, we have the `duration` and now some distance information. This is enough for us to compute average speed and try to better understand our data.

For `test_df`, we cannot use `duration` as a feature because it is what we are trying to predict. One clever way to include speed information for modeling would be as follows:

1. Cluster the observations in `train_df` by rounding the latitude and longitudes.
2. Compute the average speed per pickup cluster and dropoff cluster.
3. Match each observation in `test_df` to its pickup cluster and dropoff cluster based off the latitude and longitude, thus assigning the average speed for the pickup and dropoff cluster.
4. We have added speed information as features for `test_df`.

Therefore, we have propagated information computed in the `train_df` into the `test_df` via clustering. This is not something we will do in this notebook, although you can try it for yourself!

Other information that could be added based on clustering (both pickup cluster and dropoff cluster):

- Average of `avg_speed_m` per cluster.
- Average of `duration` per cluster.
- Average of `avg_speed_m` per cluster and hour.
- Average of `duration` per cluster and hour.
- In-cluster flow of trips for 60 min period.
- Out-cluster flow of trips for 60 min period.

```
In [42]: 1 # Calculate average manhattan speed
2 train_df['avg_speed_m'] = 1000 * train_df['manhattan'] / train_df['duration']
3 train_df['avg_speed_m'] = train_df['avg_speed_m'][train_df['avg_speed_m'] < 100]
4 train_df['avg_speed_m'].fillna(train_df['avg_speed_m'].median(), inplace=True)
```

```
In [43]: 1 train_df['avg_speed_m'].describe()
```

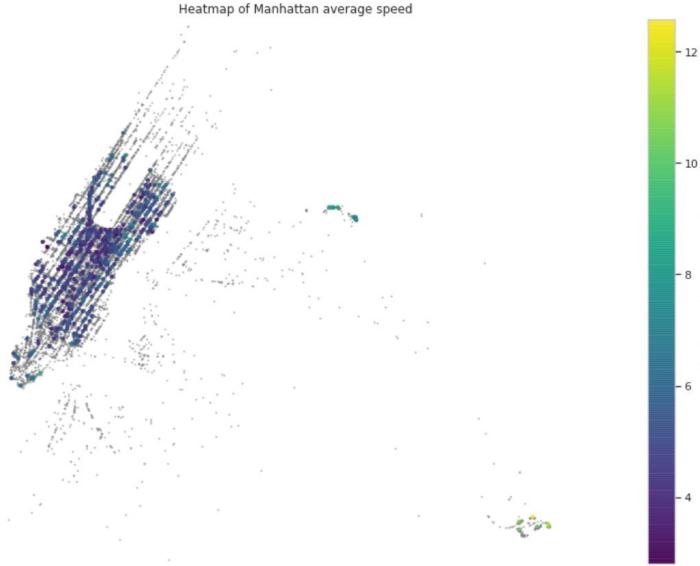
```
Out[43]: count    18354.000000
mean      5.210825
std       2.883174
min       0.000000
25%      3.287328
50%      4.617264
75%      6.413992
max      59.225577
Name: avg_speed_m, dtype: float64
```

4.4 6e 2 / 2

✓ + 2 pts Discusses the duration and manhattan distance outliers

+ 0 pts Otherwise

```
In [47]: 1 # Visualize the average speed per region
2 city_long_border = (-74.03, -73.75)
3 city_lat_border = (40.63, 40.85)
4 fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(14, 10))
5
6 scatter_trips = ax.scatter(train_df['pickup_longitude'].values,
7                             train_df['pickup_latitude'].values,
8                             color='grey', s=1, alpha=0.5)
9
10 scatter_cmap = ax.scatter(coord_stats['start_lng_bin'].values,
11                            coord_stats['start_lat_bin'].values,
12                            c=coord_stats['avg_speed_m'].values,
13                            cmap='viridis', s=10, alpha=0.9)
14
15 cbar = fig.colorbar(scatter_cmap)
16 cbar.set_label("Manhattan average speed")
17 ax.set_xlim(city_long_border)
18 ax.set_ylim(city_lat_border)
19 ax.set_xlabel('Longitude')
20 ax.set_ylabel('Latitude')
21 plt.title('Heatmap of Manhattan average speed')
22 plt.axis('off');
```



Question 7b

In 2-3 sentences, describe how we can use the clustering visualization above to gain insight on the speed. Do you think spatial clustering would be useful in reducing the error of our model?

```
In [48]: 1 q7b_answer = r"""
2
3 1. The more central the pickup points are in Manhattan, the speed will be slower. The farther the pickup points
4 are away from the downtown in Manhattan, the speed will be faster.
5
6 2.The speeds in central Manhattan are very close to each other, and they merge together.
7
8 3.The speeds in the two airports of New York are the highest.
9
10 I think it would be useful in reducing the error, because we can predict the speed by the location, and combined
11 with the distance, we can calculate the duration time. But the improvement will be very limited. Since in the
12 most crowded location in New York, the speeds are very close, and it's easy to make a mistake when finding the \
cluster.
13 """
14
15
16 # YOUR CODE HERE
17 #raise NotImplementedError()
18
19 print(q7b_answer)
```

1. The more central the pickup points are in Manhattan, the speed will be slower. The farther the pickup points are away from the downtown in Manhattan, the speed will be faster.

2. The speeds in central Manhattan are very close to each other, and they merge together.

3. The speeds in the two airports of New York are the highest.

I think it would be useful in reducing the error, because we can predict the speed by the location, and combined with the distance, we can calculate the duration time. But the improvement will be very limited. Since in the most crowded location in New York, the speeds are very close, and it's easy to make a mistake when finding the cluster.

Part 2 Exports

We are not requiring you to export anything from this notebook, but you may find it useful to do so. There is a space below for you to export anything you wish.

```
In [49]: 1 Path("data/part2").mkdir(parents=True, exist_ok=True)
2 data_file = Path("data/part2", "data_part2.hdf") # Path of hdf file
3 ...
```

Out[49]: Ellipsis

Part 2 Conclusions

We now have a good understanding of the taxi data we are working with. Visualizing large amounts of data can be a difficult task. One helpful tool is [datasader](https://github.com/bokeh/datasader) (<https://github.com/bokeh/datasader>), a data rasterization pipeline for automating the process of creating meaningful representations of large amounts of data. Using the [geopandas](http://geopandas.org/) (<http://geopandas.org/>) package also makes working with geospatial data easier. We encourage you to explore these tools if you are interested in learning more about visualization!

Within our taxi data set, we have explored different features and their relationship with ride duration. Now, we are ready to incorporate more data in order to add to our set of features.

Please proceed to part 3 where we will be engineering more features and building our models using a processing pipeline.

5.1 7b 2 / 2

✓ + 2 pts Describes how the clustering visualization helps gain insight on speed, comments on whether spatial clustering would be useful in reducing model error

+ 1 pts One of the two responses

+ 0 pts Otherwise