DataEng S24: Data Transformation In-Class Assignment

Chase Verbout

Submit: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code. Submit the in-class activity submission form by Friday at 10:00 pm.

A. [MUST] Initial Discussion Questions

Discuss the following questions among your working group members at the beginning of the week and place your own response into this space. If desired, also include responses from your group members.

1. In the lecture we mentioned the benefits of Data Transformation, but can you think of any problems that might arise with Data Transformation?

What if there are more errors after the Transformation that were uncaught? This falls in with the re-iteration that we did last week. Or what if we perform the wrong transformation? This might fall into more than one round of validations being needed again. In some cases the performance of extrapolation or interpolation may be seen as data manipulation and it could be necessary to divulge a practice like that when using the data for critical issues.

2. Should data transformation occur before data validation in your data pipeline or after?

I think that data transformation should occur after validation. With validation we can ensure that the data is formatted in the way we expect it to be for the transformations to be applied correctly. When transforming the data it would be nice to be able to assume that we do not need to be on the lookout for any errors. Though I believe that an argument could be made for validations before and after. Namely because of the issues I mentioned above in the first question. We could introduce new errors or find ourselves not catching all errors.

B. [MUST] Small Sample of TriMet data

Here is sample data for one trip of one TriMet bus on one day (February 15, 2023): bc_trip259172515_230215.csv It's in .csv format not json format, but otherwise, the data is a typical subset of the data that you are using for your class project.

We recommend that you use google Colab or a Jupyter notebook for this assignment, though any python environment should suffice.

Use the pandas.read_csv() method to read the data into a DataFrame.

```
[2] import pandas as pd
    url = 'https://drive.google.com/uc?export=download&id=1G4MUbdUFeACgjGcg6dcwykBZn21zg1jO'
    df = pd.read_csv(url)
    df.head()
```

	EVENT_NO_TRIP	EVENT_NO_STOP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE	GPS_SATELLITES	GPS_HDOP
0	259172515	259172517	15FEB2023:00:00:00	4223	40	20469	-122.648137	45.493082	12	0.7
1	259172515	259172517	15FEB2023:00:00:00	4223	48	20474	-122.648240	45.493070	12	0.8
2	259172515	259172517	15FEB2023:00:00:00	4223	57	20479	-122.648352	45.493123	12	0.8
3	259172515	259172517	15FEB2023:00:00:00	4223	73	20484	-122.648385	45.493262	12	0.7
4	259172515	259172517	15FEB2023:00:00:00	4223	112	20489	-122.648347	45.493582	12	0.8

C. [MUST] Filtering

Some of the columns in our TriMet data are not generally useful for our class project. For example, our contact at TriMet told us that the EVENT_NO_STOP column is not used and can be safely eliminated for any type of analysis of the data.

Use pandas.DataFrame.drop() to filter the EVENT_NO_STOP column.

```
[3] df2 = df.drop('EVENT_NO_STOP', axis=1)
    df2.head()
```

	EVENT_NO_TRIP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE	GPS_SATELLITES	GPS_HDOP
0	259172515	15FEB2023:00:00:00	4223	40	20469	-122.648137	45.493082	12	0.7
1	259172515	15FEB2023:00:00:00	4223	48	20474	-122.648240	45.493070	12	0.8
2	259172515	15FEB2023:00:00:00	4223	57	20479	-122.648352	45.493123	12	0.8
3	259172515	15FEB2023:00:00:00	4223	73	20484	-122.648385	45.493262	12	0.7
4	259172515	15FEB2023:00:00:00	4223	112	20489	-122.648347	45.493582	12	0.8

Next steps: Generate code with df2 View recommended plots

For this in-class assignment we won't need the GPS_SATELLITES or GPS_HDOP columns, so drop them as well.

```
[4] df3 = df2.drop(['GPS_SATELLITES', 'GPS_HDOP'], axis=1)
    df3.head()
```

	EVENT_NO_TRIP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE
0	259172515	15FEB2023:00:00:00	4223	40	20469	-122.648137	45.493082
1	259172515	15FEB2023:00:00:00	4223	48	20474	-122.648240	45.493070
2	259172515	15FEB2023:00:00:00	4223	57	20479	-122.648352	45.493123
3	259172515	15FEB2023:00:00:00	4223	73	20484	-122.648385	45.493262
4	259172515	15FEB2023:00:00:00	4223	112	20489	-122.648347	45.493582

Next steps: Generate code with df3 View recommended plots

Next, start over and this time try filtering these same columns using the usecols parameter of the read_csv() method.

```
url = 'https://drive.google.com/uc?export=download&id=1G4MUbdUFeACgjGcg6dcwykBZn21zg1jO'
df4 = pd.read_csv(url, usecols=['EVENT_NO_TRIP', 'OPD_DATE', 'VEHICLE_ID', 'METERS', 'ACT_TIME', 'GPS_LONGITUDE', 'GPS_LATITUDE'])
df4.head()
```

	EVENT_NO_TRIP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE
0	259172515	15FEB2023:00:00:00	4223	40	20469	-122.648137	45.493082
1	259172515	15FEB2023:00:00:00	4223	48	20474	-122.648240	45.493070
2	259172515	15FEB2023:00:00:00	4223	57	20479	-122.648352	45.493123
3	259172515	15FEB2023:00:00:00	4223	73	20484	-122.648385	45.493262
4	259172515	15FEB2023:00:00:00	4223	112	20489	-122.648347	45.493582

Why might we want to filter columns this way instead of using drop()?

If memory is an issue then only loading in the data you need from the start could lessen the amount used storing all of it initially.

D. [MUST] Decoding

Notice that the timestamp for each bread crumb record is encoded in an odd way that might make analysis difficult. The breadcrumb timestamps are represented by two columns, OPD_DATE and ACT_TIME. OPD_DATE merely represents the date on which the bus ran, and it should be constant, unchanging for all breadcrumb records for a single day. The ACT_TIME field indicates an offset, specifically the number of seconds elapsed since midnight on that day.

We're not sure why TriMet represents the breadcrumb timestamps this way. We do know that this encoding of the timestamps makes automated analysis difficult. So your job is to decode

TriMet's representation and create a new "TIMESTAMP" column containing a pandas.Timestamp value for each breadcrumb.

Suggestions:

- Use DataFrame.apply() to apply a function to all rows of your DataFrame
- The applied function should input the two to-be-decoded columns, then it should:
    - create a datetime value from the OPD_DATE input using datetime.strptime()
    - create a timedelta value from the ACT_TIME
    - add the timedelta value to the datetime value to produce the resulting timestamp

```
[6] def create_pd_datetime(row):
      return pd.to_datetime(row['OPD_DATE'], format="%d%b%Y:%H:%M:%S") + pd.to_timedelta(row['ACT_TIME'], unit='s')

df4['TIMESTAMP'] = df4.apply(create_pd_datetime, axis=1)
df4.head()
```

	EVENT_NO_TRIP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE	TIMESTAMP
0	259172515	15FEB2023:00:00:00	4223	40	20469	-122.648137	45.493082	2023-02-15 05:41:09
1	259172515	15FEB2023:00:00:00	4223	48	20474	-122.648240	45.493070	2023-02-15 05:41:14
2	259172515	15FEB2023:00:00:00	4223	57	20479	-122.648352	45.493123	2023-02-15 05:41:19
3	259172515	15FEB2023:00:00:00	4223	73	20484	-122.648385	45.493262	2023-02-15 05:41:24
4	259172515	15FEB2023:00:00:00	4223	112	20489	-122.648347	45.493582	2023-02-15 05:41:29

E. [MUST] More Filtering

Now that you have decoded the timestamp you no longer need the OPD_DATE and ACT_TIME columns. Delete them from the DataFrame.

```
[8] df5 = df4.drop(['OPD_DATE', 'ACT_TIME'], axis=1)
    df5.head()
```

	EVENT_NO_TRIP	VEHICLE_ID	METERS	GPS_LONGITUDE	GPS_LATITUDE	TIMESTAMP
0	259172515	4223	40	-122.648137	45.493082	2023-02-15 05:41:09
1	259172515	4223	48	-122.648240	45.493070	2023-02-15 05:41:14
2	259172515	4223	57	-122.648352	45.493123	2023-02-15 05:41:19
3	259172515	4223	73	-122.648385	45.493262	2023-02-15 05:41:24
4	259172515	4223	112	-122.648347	45.493582	2023-02-15 05:41:29

F. [MUST] Enhance

Create a new column, called SPEED, that is a calculation of meters traveled per second. Calculate SPEED for each breadcrumb using the breadcrumb's METERS and TIMESTAMP

values along with the METERS and TIMESTAMP values for the immediately preceding breadcrumb record.

Utilize the [pandas.DataFrame.diff()] method for this calculation. diff() allows you to calculate the difference between a cell value and the preceding row's value for that same column. Use diff() to create a new dMETERS column and then again to create a new dTIMESTAMP column. Then use apply() (with a lambda function) to calculate SPEED = dMETERS / dTIMESTAMP. Finally, drop the unneeded dMETERS And dTIMESTAMP columns.

```
df5['dMETERS'] = df5['METERS'].diff()
df5['dTIMESTAMP'] = df5['TIMESTAMP'].diff()
df5.head()
```

	EVENT_NO_TRIP	VEHICLE_ID	METERS	GPS_LONGITUDE	GPS_LATITUDE	TIMESTAMP	dMETERS	dTIMESTAMP
0	259172515	4223	40	-122.648137	45.493082	2023-02-15 05:41:09	NaN	NaT
1	259172515	4223	48	-122.648240	45.493070	2023-02-15 05:41:14	8.0	0 days 00:00:05
2	259172515	4223	57	-122.648352	45.493123	2023-02-15 05:41:19	9.0	0 days 00:00:05
3	259172515	4223	73	-122.648385	45.493262	2023-02-15 05:41:24	16.0	0 days 00:00:05
4	259172515	4223	112	-122.648347	45.493582	2023-02-15 05:41:29	39.0	0 days 00:00:05

Next steps: Generate code with df5 View recommended plots

```
df5['dMETERS'].fillna(0)
df5['dTIMESTAMP'].fillna(pd.Timedelta(seconds=0))
df5['dTIMESTAMP_seconds'] = df5['dTIMESTAMP'].dt.total_seconds()
df5['SPEED'] = df5.apply(lambda row: row['dMETERS'] / row['dTIMESTAMP_seconds'] if row['dTIMESTAMP_seconds'] != 0 else 0, axis=1)

df5.head()
```

	EVENT_NO_TRIP	VEHICLE_ID	METERS	GPS_LONGITUDE	GPS_LATITUDE	TIMESTAMP	dMETERS	dTIMESTAMP	dTIMESTAMP_seconds	SPEED
0	259172515	4223	40	-122.648137	45.493082	2023-02-15 05:41:09	0.0	0 days 00:00:00	0.0	0.0
1	259172515	4223	48	-122.648240	45.493070	2023-02-15 05:41:14	8.0	0 days 00:00:05	5.0	1.6
2	259172515	4223	57	-122.648352	45.493123	2023-02-15 05:41:19	9.0	0 days 00:00:05	5.0	1.8
3	259172515	4223	73	-122.648385	45.493262	2023-02-15 05:41:24	16.0	0 days 00:00:05	5.0	3.2
4	259172515	4223	112	-122.648347	45.493582	2023-02-15 05:41:29	39.0	0 days 00:00:05	5.0	7.8

Question: What is the minimum, maximum and average speed for this bus on this trip? (Suggestion: use the Dataframe.describe() method to find these statistics)

```
df5['SPEED'].describe()
```

```
count    161.000000
mean       7.182316
std        4.443425
min        0.000000
25%        3.800000
50%        6.400000
75%       10.800000
max       17.400000
Name: SPEED, dtype: float64
```

G. [SHOULD] Larger Data Set

Here is breadcrumb data for the same bus TriMet for the entire day (February 15, 2023):
bc_veh4223_230215.csv

Do the same transformations (parts C through F) for this larger data set. Be careful, you might need to treat each trip separately. For example, you might need to find all of the unique values for the EVENT_NO_TRIP column and then do the transformations separately on each trip.

Questions:
What was the maximum speed for vehicle #4223 on February 15, 2023?

Where and when did this maximum speed occur?

What was the median speed for this vehicle on this day?

H. [ASPIRE] Full Data Set

Here is breadcrumb data for all TriMet vehicles for the entire day (February 15, 2023):
bc_230215.csv

Do the same transformations (parts C through F) for the entire data set. Again, beware that simple transformations developed in parts C through F probably will need to be modified for the full data set which contains interleaved breadcrumbs from many vehicles.

Questions:
What was the maximum speed for any vehicle on February 15, 2023?

Where and when did this maximum speed occur?

Which vehicle had the fastest mean speed for any single trip on this day? Which vehicle and which trip achieved this fastest average speed?