Deepak Lukulapu

DataEng S24: Data Transformation In-Class Assignment

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?

Answer:

Some potential problems with data transformation include:

- Loss of information: Transformation processes can sometimes lead to the loss of certain details or nuances present in the original data.
- Increased computational overhead: Complex transformations may require significant computational resources, potentially leading to performance issues.
- Interpretability challenges: Highly transformed data might be difficult to interpret or understand, making it harder for analysts to derive meaningful insights.

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

Answer:

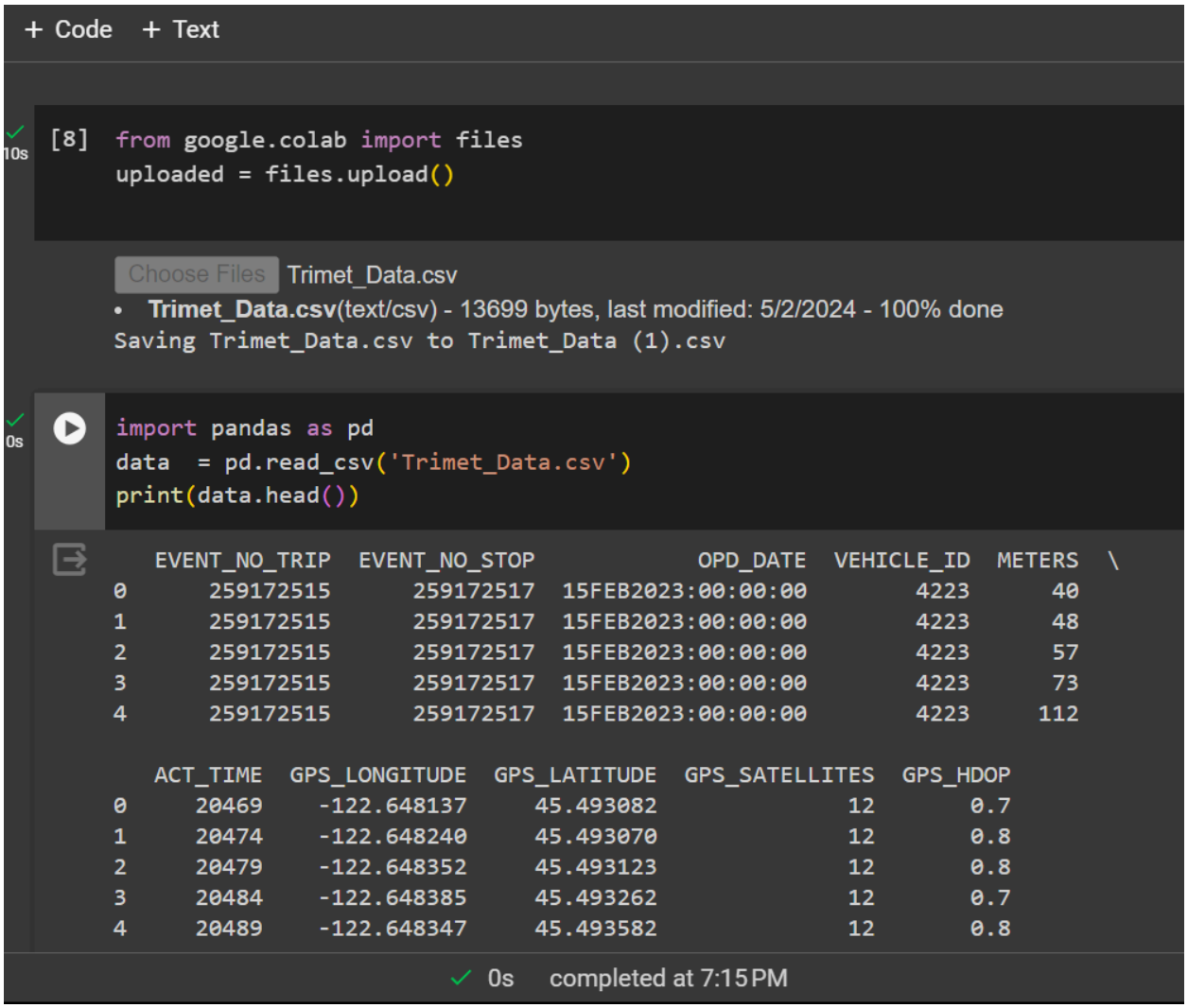
Data transformation should typically occur **before** data validation in the data pipeline. By transforming the data first, we ensure that it is in a consistent and usable format before performing validation checks. This helps in identifying and addressing any issues with the transformed data early in the pipeline, facilitating smoother validation processes downstream.

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.



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.

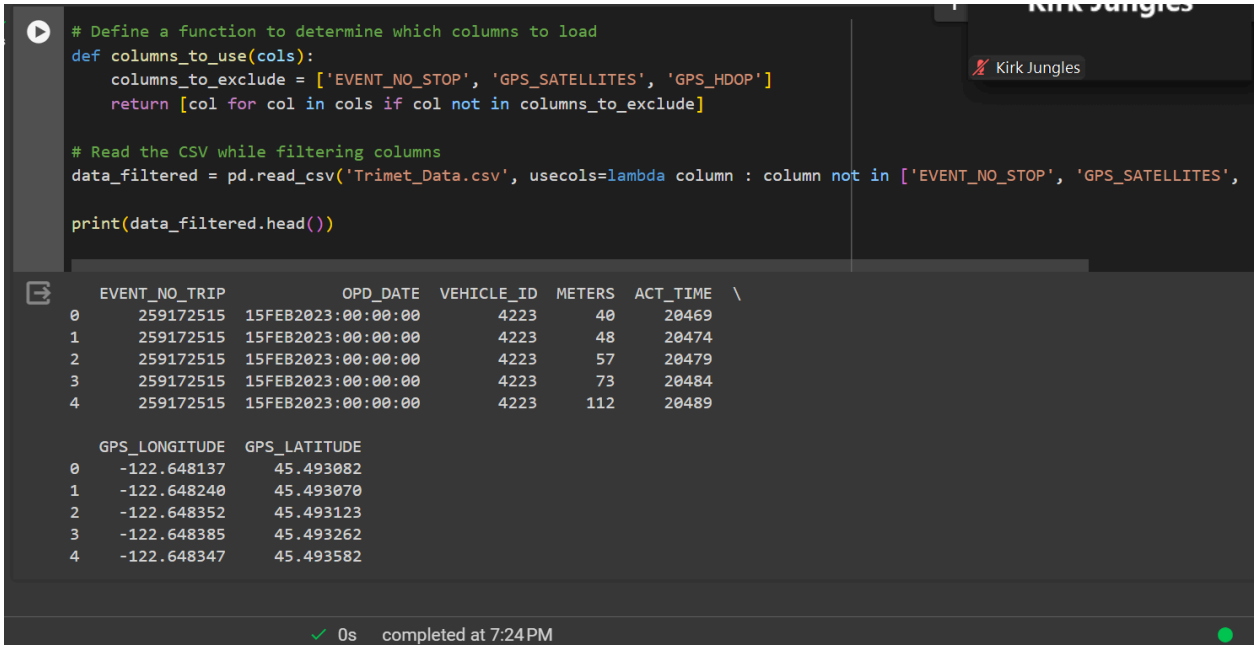
Use pandas.DataFrame.drop() to filter the EVENT_NO_STOP column.

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

```
#Dropping Unnecessary columns
#New Data
data_cleaned = data.drop(columns=['EVENT_NO_STOP', 'GPS_SATELLITES', 'GPS_HDOP'])
print(data_cleaned.head())
```

```
   EVENT_NO_TRIP           OPD_DATE  VEHICLE_ID  METERS  ACT_TIME  \
0      259172515  15FEB2023:00:00:00        4223      40     20469
1      259172515  15FEB2023:00:00:00        4223      48     20474
2      259172515  15FEB2023:00:00:00        4223      57     20479
3      259172515  15FEB2023:00:00:00        4223      73     20484
4      259172515  15FEB2023:00:00:00        4223     112     20489

   GPS_LONGITUDE  GPS_LATITUDE
0    -122.648137     45.493082
1    -122.648240     45.493070
2    -122.648352     45.493123
3    -122.648385     45.493262
4    -122.648347     45.493582
```

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



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

Answer:

Using usecols instead of drop() can be beneficial for several reasons:

1. Memory Efficiency: By specifying usecols, we avoid loading data into memory that we know we won't use. This is particularly useful with large datasets, reducing memory footprint and potentially speeding up loading times.
2. Performance: Processing time is reduced as there are fewer data to manipulate post-load.
3. Simplicity: It simplifies our data processing pipeline by eliminating the need for additional steps to remove unwanted columns after loading the data.

By filtering columns at the point of reading the file, we streamline our data handling process, making it both faster and less resource-intensive.

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

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.

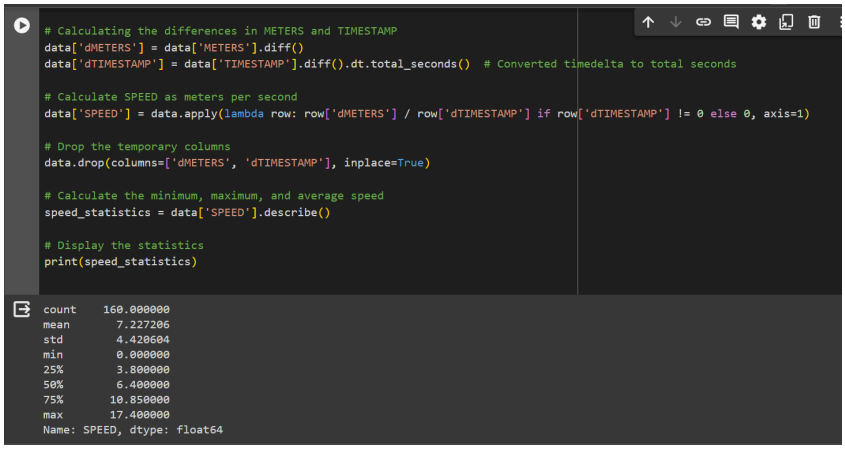
```
print(data.head(10))
```

```
   EVENT_NO_TRIP  EVENT_NO_STOP  VEHICLE_ID  METERS  GPS_LONGITUDE  \
0      259172515      259172517        4223      40    -122.648137
1      259172515      259172517        4223      48    -122.648240
2      259172515      259172517        4223      57    -122.648352
3      259172515      259172517        4223      73    -122.648385
4      259172515      259172517        4223     112    -122.648347
5      259172515      259172517        4223     159    -122.648357
6      259172515      259172517        4223     215    -122.648383
7      259172515      259172517        4223     272    -122.648375
8      259172515      259172517        4223     330    -122.648330
9      259172515      259172517        4223     391    -122.648213

   GPS_LATITUDE  GPS_SATELLITES  GPS_HDOP           TIMESTAMP
0     45.493082              12       0.7 2023-02-15 05:41:09
1     45.493070              12       0.8 2023-02-15 05:41:14
2     45.493123              12       0.8 2023-02-15 05:41:19
3     45.493262              12       0.7 2023-02-15 05:41:24
4     45.493582              12       0.8 2023-02-15 05:41:29
5     45.494003              11       0.8 2023-02-15 05:41:34
6     45.494510              12       0.7 2023-02-15 05:41:39
7     45.495023              12       0.7 2023-02-15 05:41:44
8     45.495555              12       0.7 2023-02-15 05:41:49
9     45.496103              12       0.7 2023-02-15 05:41:54
```

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.

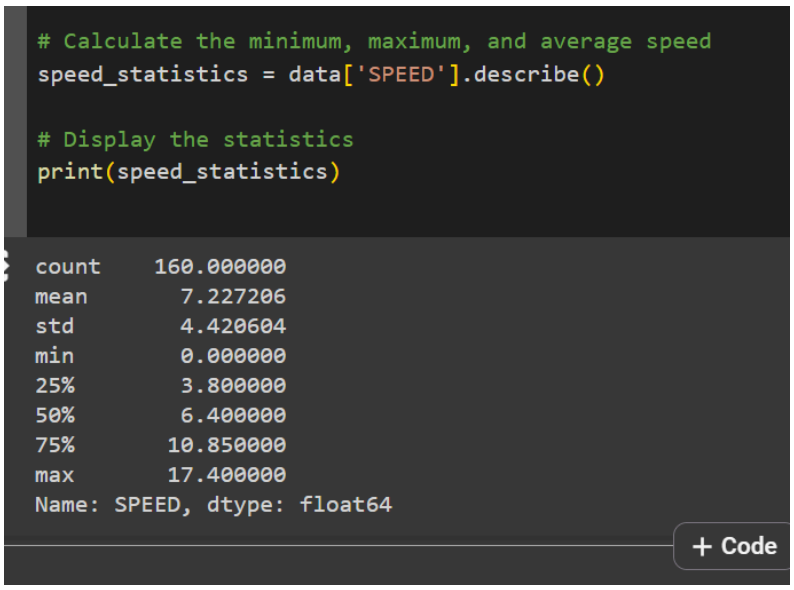
Utilize the pandas.DataFrame.diff() method for this calculation. diff() allows you to calculate the difference between a cell value and the preceding row's value for that same column. Use diff() to create a new dMETERS column and then again to create a new dTIMESTAMP column. Then use apply() (with a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



```
# Calculating the differences in METERS and TIMESTAMP
data['dMETERS'] = data['METERS'].diff()
data['dTIMESTAMP'] = data['TIMESTAMP'].diff().dt.total_seconds()  # Converted timedelta to total seconds

# Calculate SPEED as meters per second
data['SPEED'] = data.apply(lambda row: row['dMETERS'] / row['dTIMESTAMP'] if row['dTIMESTAMP'] != 0 else 0, axis=1)

# Drop the temporary columns
data.drop(columns=['dMETERS', 'dTIMESTAMP'], inplace=True)

# Calculate the minimum, maximum, and average speed
speed_statistics = data['SPEED'].describe()

# Display the statistics
print(speed_statistics)
```

```
count    160.000000
mean       7.227206
std        4.420604
min        0.000000
25%        3.800000
50%        6.400000
75%       10.850000
max       17.400000
Name: SPEED, dtype: float64
```

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)



```
# Calculate the minimum, maximum, and average speed
speed_statistics = data['SPEED'].describe()

# Display the statistics
print(speed_statistics)
```

```
count    160.000000
mean       7.227206
std        4.420604
min        0.000000
25%        3.800000
50%        6.400000
75%       10.850000
max       17.400000
Name: SPEED, dtype: float64
```

+ Code

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?