Team Metro
Jordan Machita (jm8ux)
Pantea Ferdosian (pf5aq)
Jae Yoon Sung (js2yp)
Kaleb Shikur (kms7cu)

CS 5010: Semester Project

Introduction

The purpose of this project was to analyze traffic data to discover patterns in traffic volume based on a variety of factors. By doing so, we hoped to mimic the kinds of analysis that we presume a GPS may conduct in order to determine estimated travel time between two locations. We understand that most navigation systems likely use real-time data such as current accidents and number of drivers on the road to estimate in the moment travel time. However, advanced systems now allow users to input a future date and time to estimate the duration of the trip if they were to embark at that future point. This feat relies on extrapolation because real-time data does not exist for future dates and times. Thus, our goal was to find relevant patterns in historical data that may be used to predict future trip durations. We set out to analyze a number of possible factors that may influence the amount of traffic, including time of day, time of year, holidays, and weather conditions.

The Data

We obtained a dataset from the UC Irvine Machine Learning Repository that related to metropolitan traffic volume for a highway in Minnesota that connects Minneapolis with St. Paul. The dataset measures the number of cars that passed a specific station in the middle of the highway heading westbound from St. Paul to Minneapolis. In addition to the hourly traffic volume, the dataset captures the temperature, weather conditions, date and time, and holiday. The dataset ranges in date from October 2, 2012 to September 30, 2018. Our goal was to use this data to identify patterns related to traffic volume, but to do so we first needed to clean and process the data.

As our data was originally stored as a csv file, all of the values were stored as strings. Thus, we stripped the whitespace from the strings and type casted where appropriate. While the holiday and weather descriptions remained as strings, we converted temperature, inches of rain, and inches of snow to floating point values and cloud coverage and traffic volume to integer values. Additionally, the dataset had the date and time stored in one string but with inconsistent lengths. For example, a singular digit month, day, or hour did not have a zero in front of the single digit to make it the same length string as a double-digit month, day, or hour. To resolve this, we looped through the data and added the appropriate zeros as well extending the year from a two-digit format to a four-digit format. The result was a string with the consistent format of "dd/mm/yyyy hh:00" that has 16 characters. The reason we wanted to make the format of the date and time consistent was to extract specific values from the string and store them in new columns. With the same format, we could easily iterate over the data to do so. We appended four columns to the dataset each with an integer value: month, day, year, and hour. This would allow us to more easily manipulate and visualize the data.

Another issue we discovered was that the holiday was only marked at the zero hour of the day and all other hours of the holiday were classified as "None" like all of the ordinary days. We resolved this by looping through the dataset and setting the holiday of every hour equal to the holiday of the zero hour for that day. Additionally, we noticed that for some unique date and times there were up to six instances of the same day and hour with all of the same values except the two string fields for weather descriptions. To avoid having replicated data points, we concatenated all of the descriptions into one string with semicolons as separators and deleted the excess data points, which resulted in 7,500 deleted rows. Lastly, we converted the temperature from Kelvin to Fahrenheit; however, there were a handful of missing values in temperature (less than 20 out of 40,000). Rather than discarding these points, as all of their other fields had correct values, we updated the temperature to be the same as the prior hour, recognizing that this assumption would allow us to keep the data without skewing the temperatures too much as the prior hour's temperature would likely be in the range of possible temperatures.

We would also like to note that there were some instances of missing data points where one hour is not included in the day. We did not see this as problematic as we would mostly be viewing the data in terms of averages and aggregates so a handful of missing hours throughout the six years is not likely to generate bias in our inferences. Once the data was sanitized, we stored it in a "pandas" data frame for ease of use as we would then be able to import the data frame from the file and work directly with it. In addition, we exported the cleaned data into a new csv file for storage and reference.

Data Distribution

Before we move forward to our experimental design and results, we first look at the histograms of each variable displayed in Figure 1. These graphs enable us to see how the data is distributed for each variable and whether the numerical data are continuous, discrete, or categorical. As we see, the discrete values are Hour, Day, Month, Year, and Clouds_All. Also, Temperature, Traffic_Volume, Rain_1h, and Snow_1h are continuous variables. However, the plots for Rain_1h and Snow_1h are not the best visuals due to outliers skewing the scale of the x-axis. It is most useful for us to visualize the temperature, traffic volume, and cloud coverage in this way as their dispersion may prove useful later in our analyses.

In Figure 2, we turn the histogram of traffic volume from Figure 1 into an enlarged frequency distribution so that we can more clearly visualize the distribution of the data. As we can see, there are small peaks of traffic volumes with the largest being near zero. Since the data is hourly, we expected to see many hours of small traffic volume due to the decreased traffic during the night as we will further discuss in the results section of this report.

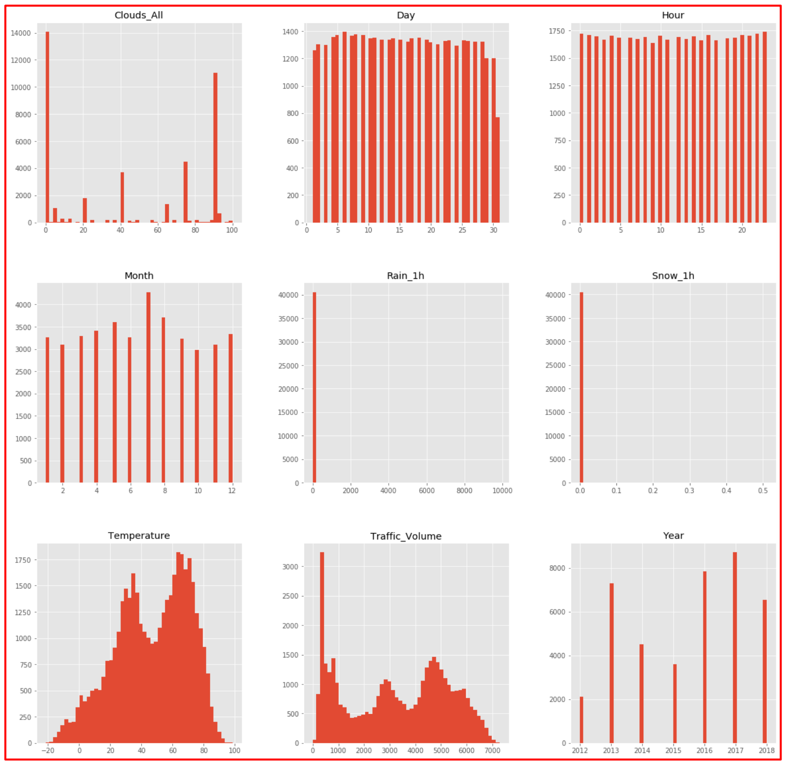


Figure 1: Variables Histograms

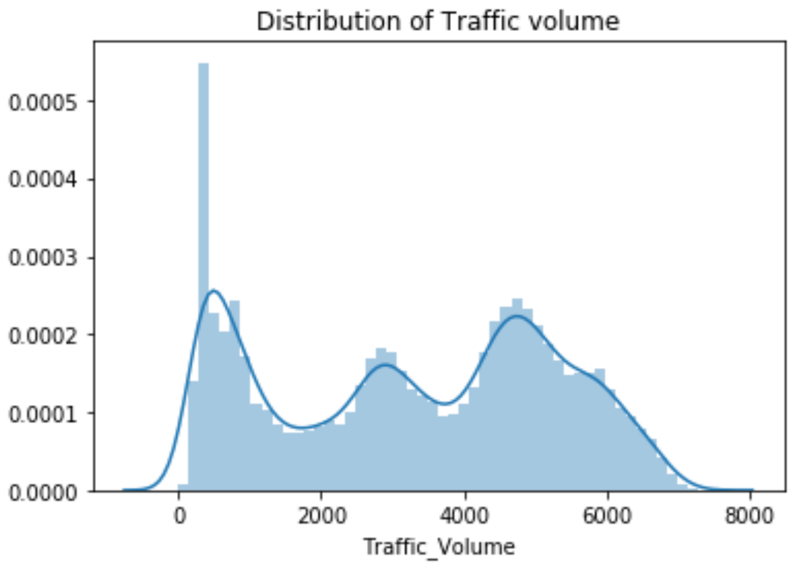


Figure 2: Distribution of Traffic Volume

Experimental Design

Once we decided on the dataset and performed our initial cleaning and calculations prior to storing the data in a “pandas” data frame, we identified various ways in which we could generate insight and information from the data. Our initial interest was in the date and time variables. As we all understand the frustration of rush hour, we decided that viewing the traffic volume by hour of the day may be a good place to start. The results fed our curiosity as we wondered if the month or even the day of the month played a role as well. This led us to question if holidays would prove to be special cases for traffic patterns or if they would follow the same general trends. As our data was collected over six years, we also questioned if traffic trends would change over time. While we quickly became consumed with the various analyses of how date, time, and holiday affect traffic volume, we also began to wonder how temperature and weather patterns fit into the story.

With an extensive list of questions, we set out to employ various plotting and filtering techniques to begin generating answers. Due to the large quantity of data captured, we felt confident that calculating and plotting the means of various attributes would be able to help us turn the data into information, which we could then assess and transform into knowledge. We used various filtering techniques to visualize the information we sought. We also began to use the pandas Groupby function, which has enhanced functionality to aid in answering many of our questions. While we did not discover the functionality of the Groupby function until after we had conducted extensive analyses, we recognize that it could certainly have made our approach more efficient. Nevertheless, we wanted to at least discover some of this functionality, so we began to implement this approach to start to answer some of our questions that we had previously neglected to resolve. We are confident that more extensive use of the Groupby function would allow us to produce additional analyses and generate more insight into the various factors that affect traffic patterns.

Beyond the Original Specifications

We decided to make a few of our plots dynamic by accepting user input that affects the data used to generate the plot. We did this in the form of a dropdown menu that presents several options from which to choose. This was achieved using the “ipywidgets” library. We recognized a few circumstances that were the most logical to implement such user input. For example, when generating a plot to illustrate traffic volume by hour for a particular month, we allowed the user to choose which month the plot displayed. Our dropdown menu contained the full names of each of the twelve months from which the user could select one. This allowed for flexibility in changing the month as well as quick and easy comparison between months. The result of this can be seen when we discuss Figure 4. We followed a similar approach for generating a plot of the traffic volume for each day of a particular month, which is chosen by the user as demonstrated in Figure 8. The third instance where we chose to include this form of user interaction was when plotting the average hourly traffic volume for a particular holiday compared to the traffic pattern for an ordinary day in Figure 12. The data regarding the ordinary day would always be plotted and the user would choose which holiday they would like to see in the comparison from a dropdown list containing the eleven holidays for which we have data.

Results

Hourly Traffic Patterns

As described in our experimental design, one of the first questions we asked was: How does the traffic vary by hour of the day? In order to assess the relationship between hour and traffic volume, we generated a plot that shows the mean traffic volume for each hour across the entire dataset as seen in Figure 3. We can clearly see that the traffic volume increases rapidly from 5am to 7am before falling slightly, which indicates the end of morning rush hour. The traffic volume then peaks again around 4pm, indicating evening rush hour, before falling off as the night approaches. We expected to see this type of trend. However, it is interesting to note that the evening rush hour is busier than the morning rush hour and there is also a small peak around lunchtime.

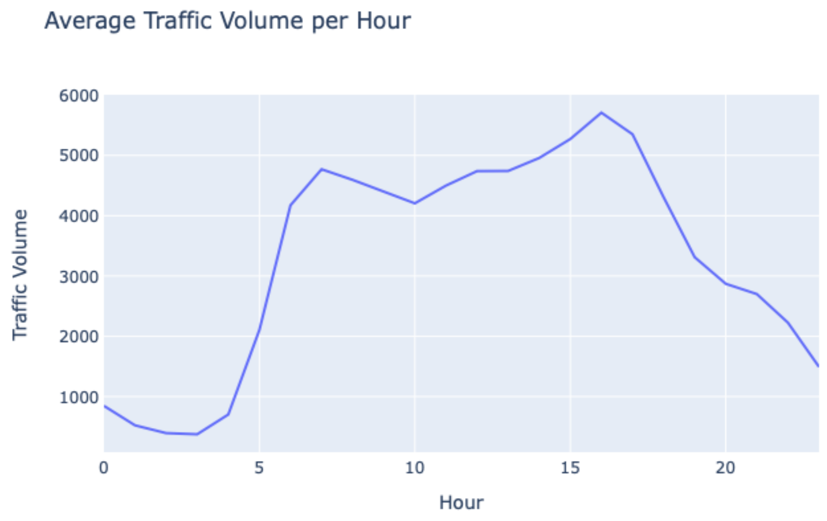


Figure 3: Average Traffic Volume per Hour

We can further analyze this relationship between traffic volume and hour by filtering for more specific criteria. Figure 4 shows an example of this where we look only at the hourly traffic volume for a particular month, in this case May. While a GPS would certainly benefit from the overall daily traffic patterns, filtering by month is likely to enhance the accuracy of the extrapolation.

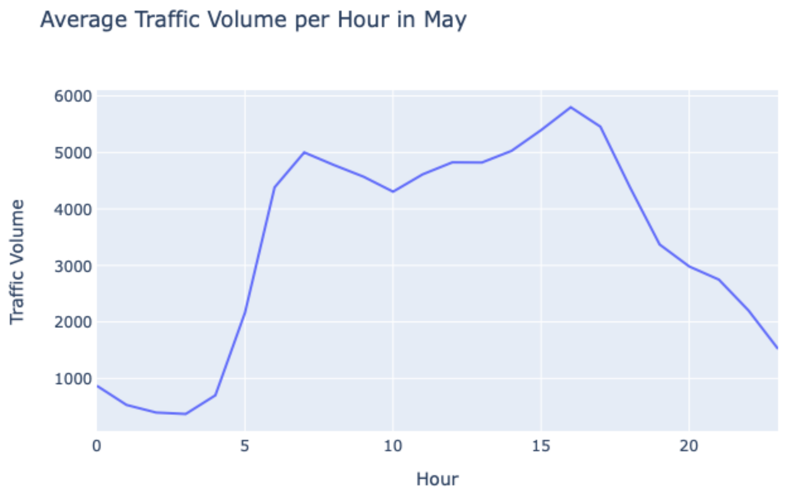


Figure 4: Average Traffic Volume per Hour in May

Another key piece of information is how the daily traffic pattern changes over time. Is rush hour this year worse than it was last year? It may be tempting to feel that traffic is getting worse over time, but by analyzing the data we can see that this is not actually the case at least for this particular location in Minnesota. Figure 5 illustrates that not only have the trends in hourly traffic volume remained virtually unchanged over the six-year period, but also that the actual traffic volume has not changed significantly. This suggests that traffic has actually stayed constant for this area from 2012 to 2018 even if it may not feel like it for individual drivers.

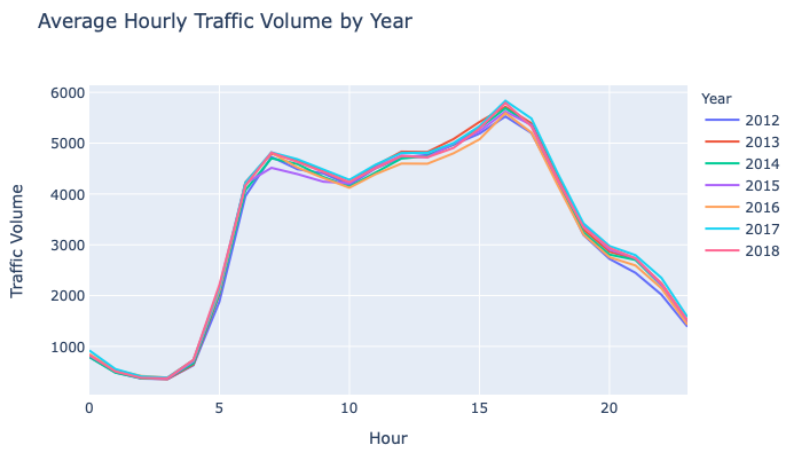


Figure 5: Average Hourly Traffic Volume by Year

Traffic Trends by Date

In addition to understanding daily traffic trends, we also wanted to assess any monthly or seasonal changes in traffic patterns. We assumed that time of year would also play into the calculations performed by a GPS. We first looked at the mean daily traffic volume by month in Figure 6. This revealed that the winter months tended to have lower daily traffic volumes. An interesting exception was the valley that can be seen in the graph at July. The other summer months tend to have even higher traffic volumes than spring, fall, and winter but July is more consistent with February and November. This may suggest that locals are taking their longer summer vacations in July so they may be out of the area for weeks at a time, resulting in lower daily traffic throughout the month of July. However, this is just one potential explanation.

While Figure 6 pertains to the average daily traffic volume per month, we also look at the hourly data by month in Figure 7. On an hourly basis, the trend appears to be much less pronounced but is still present when looking at the means in the boxplot.

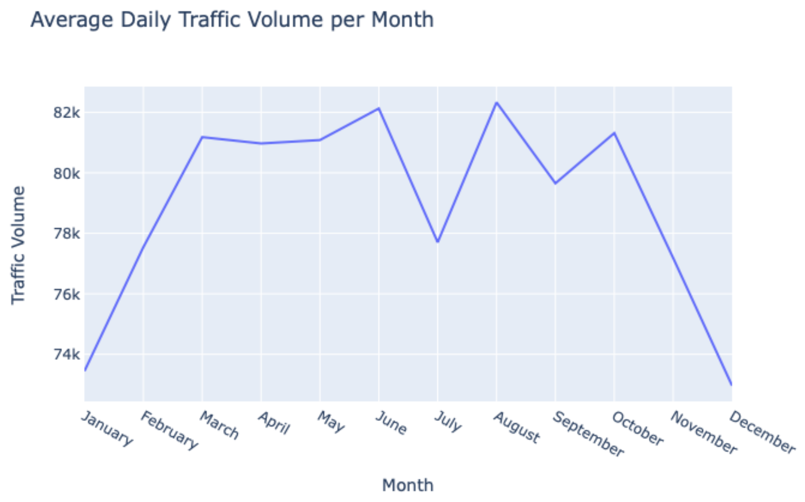


Figure 6: Average Daily Traffic Volume per Month

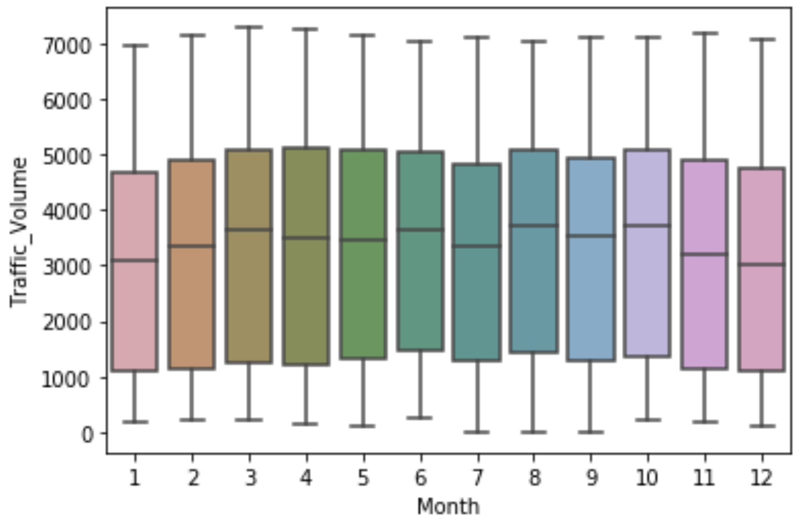


Figure 7: Boxplot of Hourly Traffic Volume by Month

We also chose to analyze the daily traffic patterns based on the time of the month to see if this revealed any useful insight. Figure 8 displays the average hourly traffic volume per day throughout October across all six years. This does not appear to provide any useful information likely because the distribution of weekday versus weekend traffic affects the data when we filter it by day of the month. A similar phenomenon occurs for the remainder of the months, indicating that this does not provide meaningful insight.

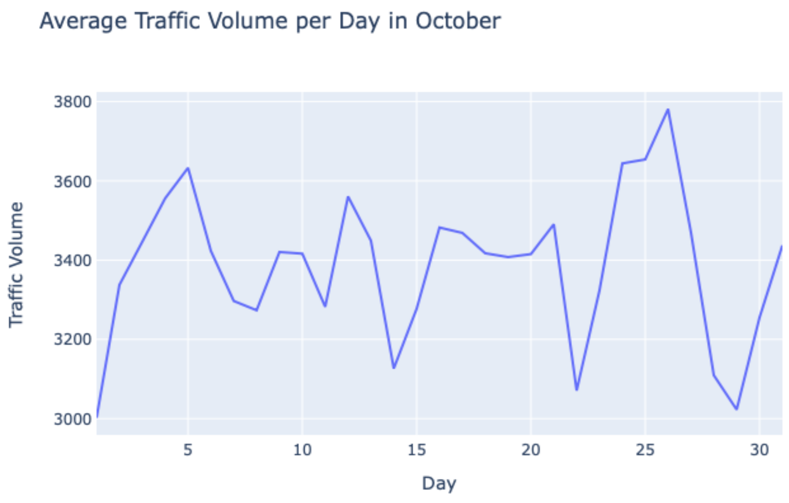


Figure 8: Average Traffic Volume per Day in October

Holiday Traffic Trends

We also recognize that holidays may follow abnormal traffic patterns. In order to assess if this is the case, we wanted to first look at the daily traffic volume by holiday. We can see this in Figure 9, which shows the average daily traffic volume for each of the eleven tracked holidays as well as "None" that represents non-holidays. This plot shows that some holidays, such as Christmas, Thanksgiving, and New Year's, have significantly lower traffic than the average day. While this may be expected, we can also see that Memorial Day, Independence Day, and Labor Day have lower traffic, which is a bit more surprising. This set of holidays tends to be linked to beach or lake trips, which led us to expect increased traffic. There are also a handful of holidays with slightly above average traffic as well as the State Fair, which has significantly higher traffic, indicating that locals are making day trips to the fair. In Figure 10, we can see the same holiday traffic trends on an average hourly level as opposed to the total daily traffic volume in Figure 9.

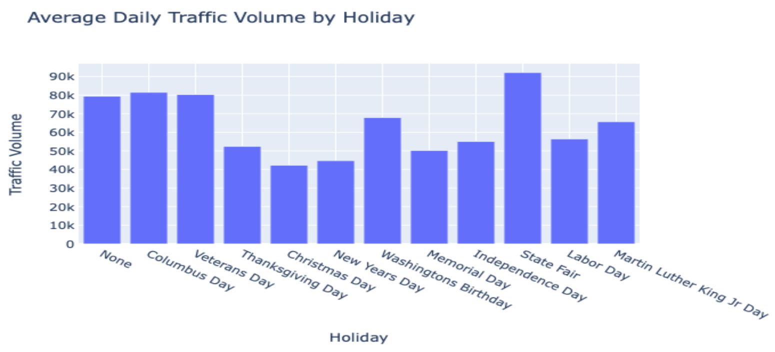


Figure 9: Average Daily Traffic Volume by Holiday

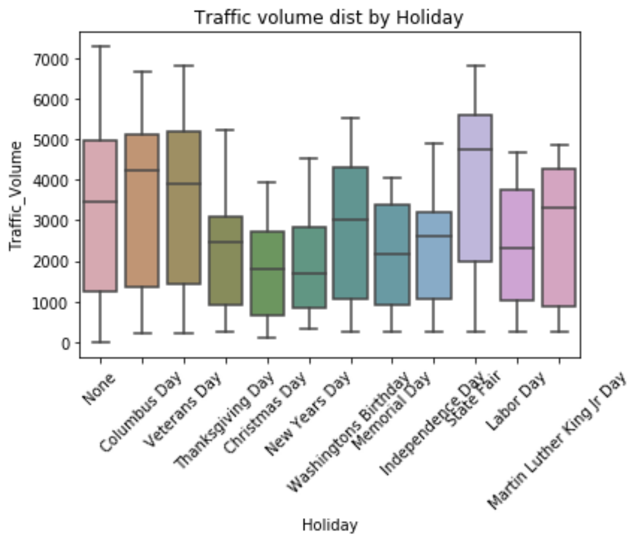


Figure 10: Boxplot of Hourly Traffic Volume by Holiday

Since we have confirmed that holidays do have an effect on traffic patterns, we wanted to dig deeper into the hour by hour impact that they have. Figure 11 shows the mean hourly traffic volume by holiday. We can see that for a significant number of holidays, there does not appear to be a morning rush hour, but they each still show a peak in the late afternoon even if it is much smaller than normal. In order to generate a clearer comparison between a singular holiday and the average ordinary day, we have plotted a version of Figure 11 as Figure 12 that includes only these two instances. We have demonstrated this comparison in Figure 12 by using Christmas Day as our holiday of interest. As we can see, this makes for a much easier comparison as only two lines are visible instead of twelve.

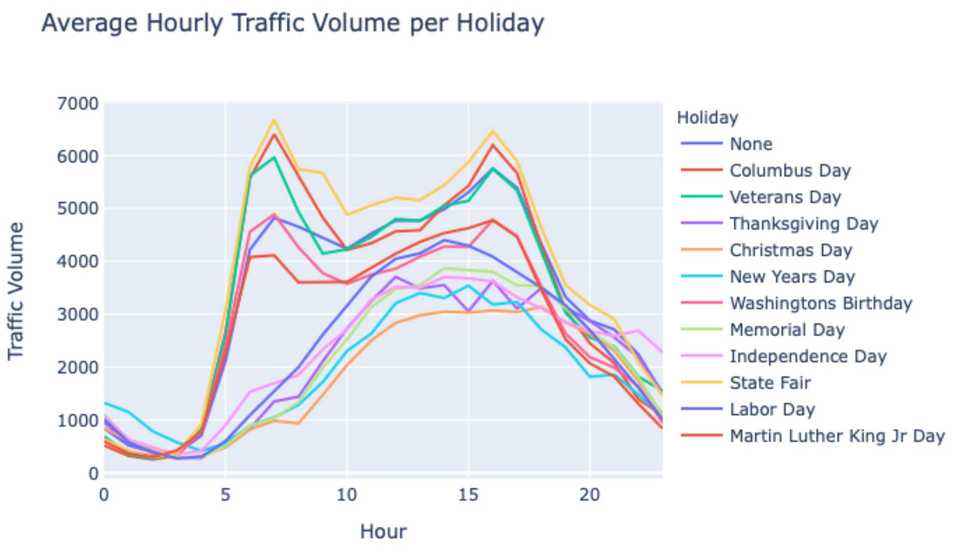


Figure 11: Average Hourly Traffic Volume per Holiday

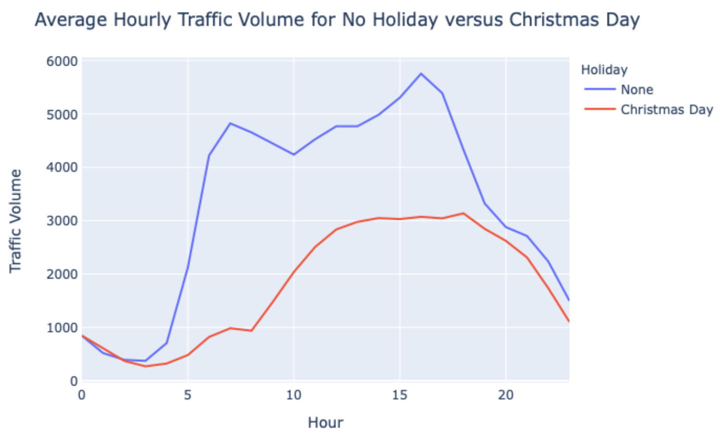


Figure 12: Average Hourly Traffic Volume for No Holiday versus Christmas Day

Traffic Pattern by Temperature

In addition to date, time, and holiday, the temperature and weather may have an effect on traffic. By analyzing the trends of past data, a GPS could then extrapolate by pulling the predicted weather information from an appropriate source to account for those effects on traffic and trip duration. To begin our analysis on temperature, we first plotted the average temperature per month to visualize the trends. It is not surprising to see in Figure 13 that the average temperature was much higher in the summer than the winter, but it was still useful to plot. We can also see the numerical values for mean, maximum, and minimum temperature per month in Figure 14. Also included in Figure 14 are the same statistics but for traffic volume by month. However, as the average hourly temperature changes throughout the months, we do not see any relation to the change in hourly traffic volume.

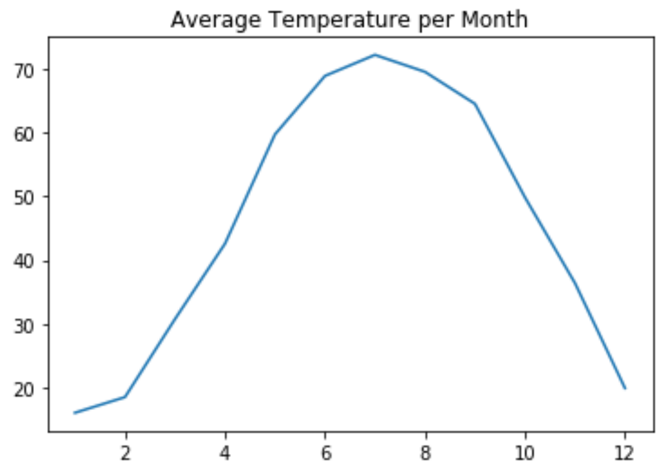
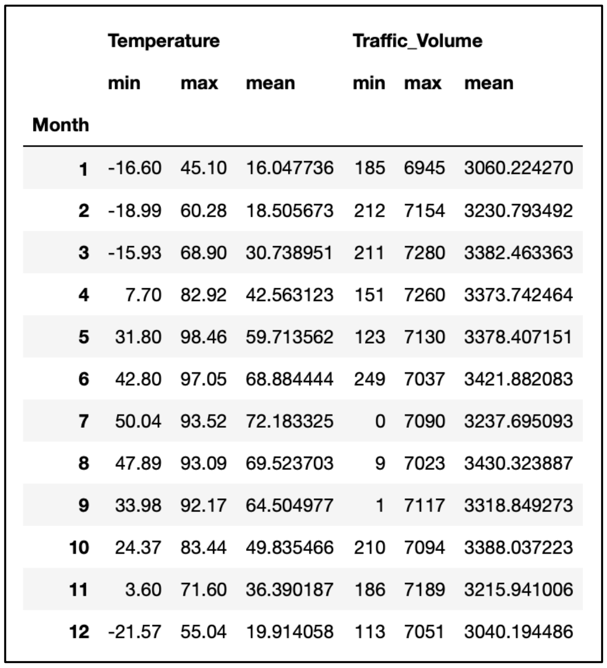


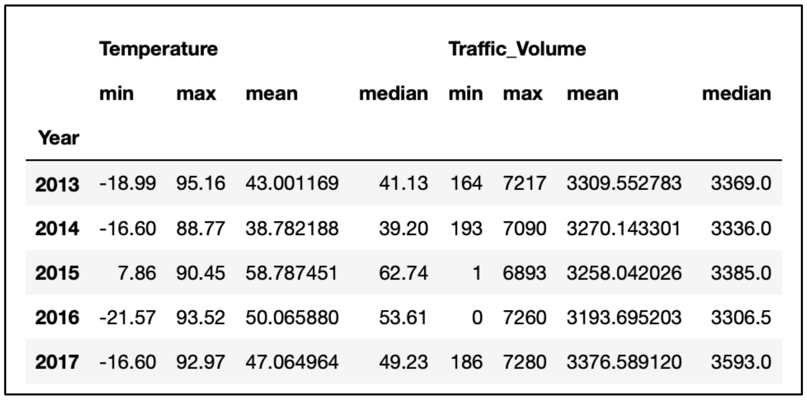
Figure 13: Average Temperature per Month



	Temperature			Traffic_Volume		
	min	max	mean	min	max	mean
Month						
1	-16.60	45.10	16.047736	185	6945	3060.224270
2	-18.99	60.28	18.505673	212	7154	3230.793492
3	-15.93	68.90	30.738951	211	7280	3382.463363
4	7.70	82.92	42.563123	151	7260	3373.742464
5	31.80	98.46	59.713562	123	7130	3378.407151
6	42.80	97.05	68.884444	249	7037	3421.882083
7	50.04	93.52	72.183325	0	7090	3237.695093
8	47.89	93.09	69.523703	9	7023	3430.323887
9	33.98	92.17	64.504977	1	7117	3318.849273
10	24.37	83.44	49.835466	210	7094	3388.037223
11	3.60	71.60	36.390187	186	7189	3215.941006
12	-21.57	55.04	19.914058	113	7051	3040.194486

Figure 14: Temperature and Traffic Volume Statistics by Month

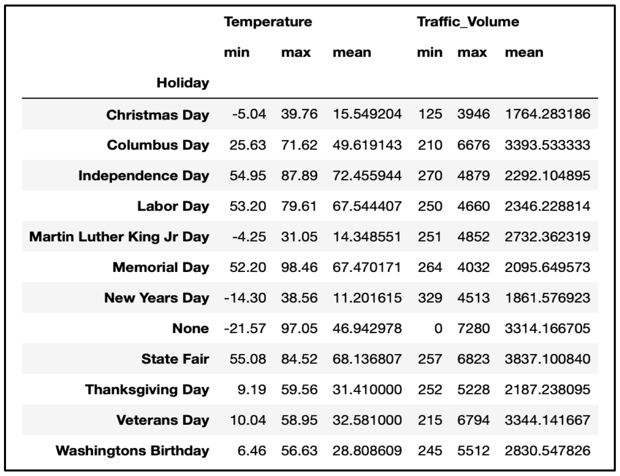
Similarly, we can view this relationship by year rather than month as an attempt to discover something useful but once again, we cannot make any meaningful conclusions. Nevertheless, this data, displayed in Figure 15, is still interesting to view. In order to avoid a biased comparison, we removed the years 2012 and 2018 from this table because they did not contain a full calendar year's worth of data.



	Temperature				Traffic_Volume			
	min	max	mean	median	min	max	mean	median
Year								
2013	-18.99	95.16	43.001169	41.13	164	7217	3309.552783	3369.0
2014	-16.60	88.77	38.782188	39.20	193	7090	3270.143301	3336.0
2015	7.86	90.45	58.787451	62.74	1	6893	3258.042026	3385.0
2016	-21.57	93.52	50.065880	53.61	0	7260	3193.695203	3306.5
2017	-16.60	92.97	47.064964	49.23	186	7280	3376.589120	3593.0

Figure 15: Temperature and Traffic Volume Statistics by Year

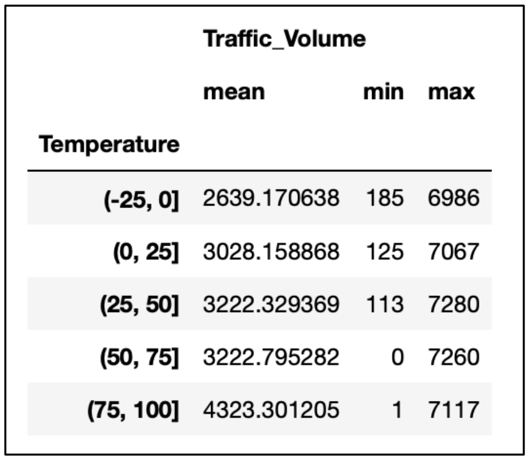
We display these summary statistics yet again in Figure 16, but this time by holiday. This is less of an attempt to assert a causal relationship and more to garner other insight from the data as it is unlikely that the trends in traffic data are influenced more by temperature than by the holiday. Figure 16 gives us useful insight into the general temperature variations across holidays. An interesting note is that Memorial Day actually contained the hottest single hour out of the entire dataset despite there being significantly more non-holidays.



	Temperature			Traffic_Volume		
	min	max	mean	min	max	mean
Holiday						
Christmas Day	-5.04	39.76	15.549204	125	3946	1764.283186
Columbus Day	25.63	71.62	49.619143	210	6676	3393.533333
Independence Day	54.95	87.89	72.455944	270	4879	2292.104895
Labor Day	53.20	79.61	67.544407	250	4660	2346.228814
Martin Luther King Jr Day	-4.25	31.05	14.348551	251	4852	2732.362319
Memorial Day	52.20	98.46	67.470171	264	4032	2095.649573
New Years Day	-14.30	38.56	11.201615	329	4513	1861.576923
None	-21.57	97.05	46.942978	0	7280	3314.166705
State Fair	55.08	84.52	68.136807	257	6823	3837.100840
Thanksgiving Day	9.19	59.56	31.410000	252	5228	2187.238095
Veterans Day	10.04	58.95	32.581000	215	6794	3344.141667
Washingtons Birthday	6.46	56.63	28.808609	245	5512	2830.547826

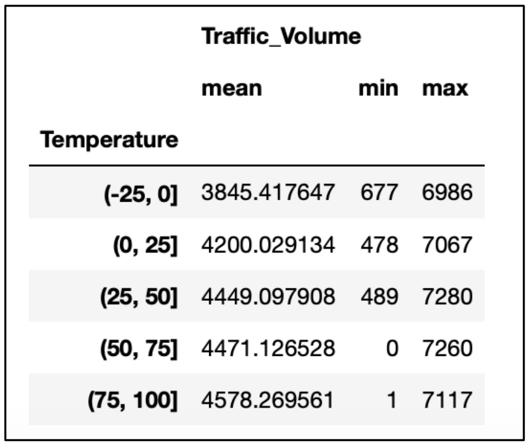
Figure 16: Temperature and Traffic Volume Statistics by Holiday

Since we were unable to draw any meaningful conclusion when looking at the temperature and traffic data by month or year, we next wanted to solely compare temperature with traffic volume. To do this, we grouped the temperatures into bins of 25 degrees Fahrenheit prior to calculating the mean, minimum, and maximum for hourly traffic volume as can be seen in Figure 17. We immediately noticed that warmer temperatures appear to be strongly correlated with increased traffic volume. However, this may be biased due to the fact that the temperature tends to drop at night and we previously saw that traffic volume is significantly lower at night. In an attempt to account for this bias, we show the same statistics in Figure 18, but we calculate them only based on the hours from 7am to 9pm. We can see that the trend of traffic volume increasing with temperature still holds true but is much less drastic than when we included the nighttime hours.



	Traffic_Volume		
	mean	min	max
Temperature			
(-25, 0]	2639.170638	185	6986
(0, 25]	3028.158868	125	7067
(25, 50]	3222.329369	113	7280
(50, 75]	3222.795282	0	7260
(75, 100]	4323.301205	1	7117

Figure 17: Hourly Traffic Volume Statistics by Temperature Grouping



	Traffic_Volume		
	mean	min	max
Temperature			
(-25, 0]	3845.417647	677	6986
(0, 25]	4200.029134	478	7067
(25, 50]	4449.097908	489	7280
(50, 75]	4471.126528	0	7260
(75, 100]	4578.269561	1	7117

Figure 18: Hourly Traffic Volume Statistics by Temperature Grouping from 7am to 9pm

Trends in Weather

Lastly, we began to analyze how weather patterns not related to temperature may play a role in determining traffic volume. While we do not dig into the full extent of this relationship in our project, we at least plot the average hourly rain and snowfall in inches by month. We can see in Figure 19 that the amount of rain is much greater in the summer months than the remainder of the year with a particularly noticeable peak in July. Earlier in the report, we noticed a pronounced decrease in traffic volume for July compared to the other summer months, so this may indicate a potential relationship that could be investigated further.

Conversely, Figure 20 illustrates that the amount of snowfall peaks in the winter months as to be expected. This corresponds to the decrease in traffic volume during the winter that we stated previously. These relationships would need to be further tested before concluding with certainty whether the weather or time of year contributed more significantly to the traffic patterns. Nevertheless, this interpretation could provide a GPS with enough information to infer lower traffic volumes during periods of snowfall or rain.

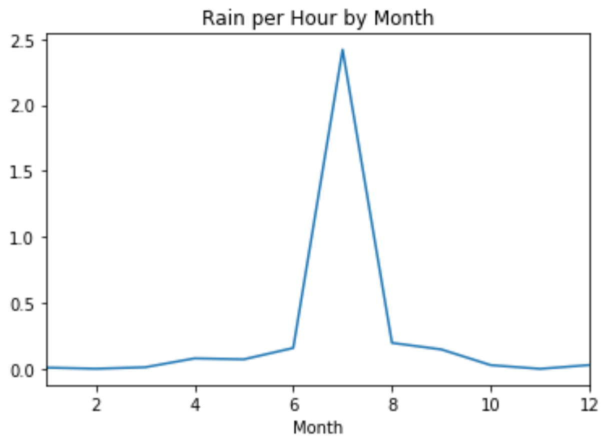


Figure 19: Average Rain (in inches) per Hour by Month

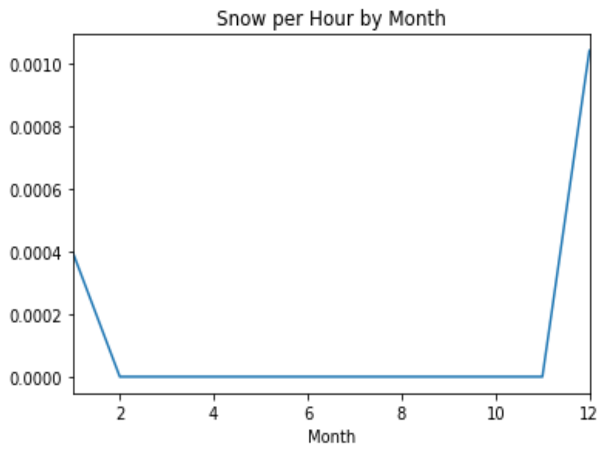


Figure 20: Average Snowfall (in inches) per Hour by Month

Multiple Linear Regression

Now that we have a general idea of how some of the different variables in the data set may impact traffic volume, the next step would be to model this relationship to attempt to predict the expected traffic volume based on changes in the other variables. While the full extent of modeling was not within the scope of our project, we decided to at least run and test one of many possible multiple linear regression models. We ran a model using the average monthly data and chose temperature, cloud coverage, snowfall, day, and average number of holidays as our predictors. Given more time, we would further refine our choice of predictors and model selection techniques; however, the goal of this approach was just to see if a model with a variety of predictors had any merit. In Figure 21, we plot the hourly traffic volume that our model predicted against the actual volume for a testing set of the data that was not used in model fitting. As we can see, our model does appear to have some predictive ability; however, there are many more steps we would take to discover the best possible model for our data.

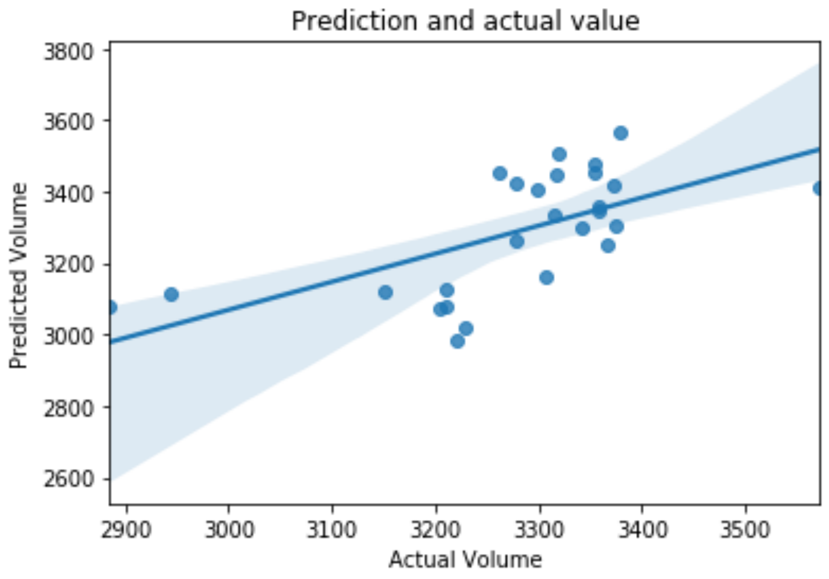


Figure 21: Multiple Linear Regression Result

Testing

For our project, we performed several unit tests to check multiple aspects of our data cleaning. We understand the importance of test-driven development, so we built out our unit tests as we generated the corresponding code. We ran these tests through the various steps of our data sanitization process to ensure that any additional code was not affecting what we had already accomplished and tested. We performed our tests for each variable mostly based on black box testing, where we knew the input and expected output of our code. The testing process was significantly helpful in our case as we discovered new issues while conducting the data cleaning. Our unit tests not only assured us that the cleaning appeared to be successful but also helped us identify additional measures we could undertake and alerted us of potential missing data. In the following subsections, we provide a detailed explanation of our unit tests.

Data Type Tests

The first set of tests we conducted checked the type of data in each column. This helped us discover that all of the data was stored as strings, so in our cleaning we type casted to integers and floats where appropriate. However, after the appropriate type casting, our unit tests were still not passing, which alerted us to an interesting phenomenon whereby all of our integer variables were being stored as “numpy.int64” despite casting to int. We were unable to resolve this and presumed that this discrepancy was generated when we stored the data into a pandas dataframe. Thus, we altered our unit tests to assess whether the integers were of type “numpy.int64” rather than of type int.

The types for each variable are as follows:

- Holiday – string
- Temperature – float
- Rain_h1 – float
- Snow_1h – float
- Clouds_All – integer
- Weather_Main – string
- Weather_Description – string
- Date_Time – string
- Traffic_Volume – integer

For the purposes of this project to better analyze the dataset, we decided to extract specifics from the date and time string and make new, separate columns with integer values in our cleaned data set:

- Month – integer
- Day – integer
- Year – integer
- Hour – integer

Thus, we also tested the types of these column values in this set of unit tests. This resulted in a total of thirteen unit tests because our cleaned dataframe contained thirteen columns. Each of these thirteen unit tests were stored and conducted under the same class of unit tests called "DataTypesTestCase" in our unit test python file.

Date and Time Format Test

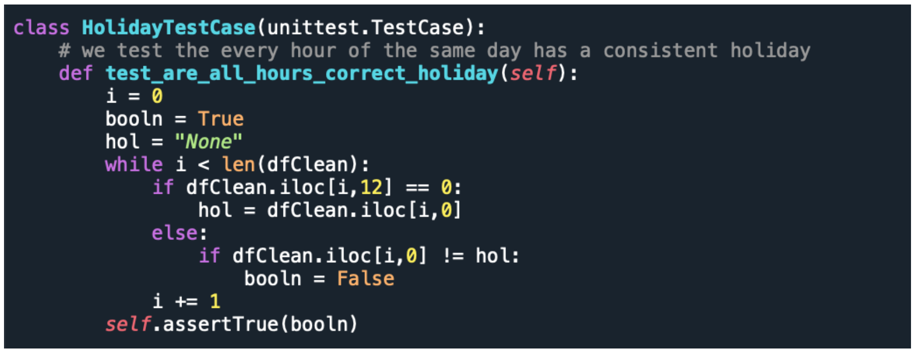
When we first encountered the data set, we noticed that the Date_Time column data was not formatted as a consistent length. For instance, the format was "X/XX/XX HH:00" for months January–September and "XX/XX/XX HH:00" for October-December. During the cleaning, we changed this format to "XX/XX/20XX HH:00" for all data points regardless if month, day, or hour only had a single digit. Therefore, for this section, we tested to see if the Date_Time column matched the formatting that we expected after cleaning and had a consistent string length. It was important in our test to check every data point instead of randomly choosing one because some values, such as 11/23/12 19:00, would already be mostly what we expected (with the exception of a four digit year) with or without the cleaning process. To ensure the test would catch if a single data point did not match as expected, we looped through the dataframe and used a boolean value that we initialized as True but would become False if any point did not match. This enabled us to use the self.assertTrue function rather than using the self.assertEqual function as we might if we were testing a singular point. The value of this became apparent when we later removed data points as a single point test might then not match up correctly as indices change. This resulted in four different tests, one for each of month, day, year, and hour, which are contained within the "DateTimeFormatTestCase" class.

Testing the Appended Columns

Because we created four new columns in the cleaned dataset, we tested that each of these, which are Month, Day, Year, and Hour, had the correct value as it should correspond to the original Date/Time column. We had to make sure they appended correctly to the dataset the way that they were intended. We followed a similar procedure of looping through the data and utilizing a boolean variable for these tests. Each new column was tested in its own method but all four were part of the same "AppendColumnsTestCase" class.

Holiday and Hour Test

As we discussed in the Data section, we fixed an issue whereby a holiday only showed the name of the holiday for the zero hour. To ensure that we resolved this correctly, we created a unit test that verifies that every hour of the same day has a consistent holiday. We include a snapshot of this test for reference in Figure 22.



```python
class HolidayTestCase(unittest.TestCase):
    # we test the every hour of the same day has a consistent holiday
    def test_are_all_hours_correct_holiday(self):
        i = 0
        booln = True
        hol = "None"
        while i < len(dfClean):
            if dfClean.iloc[i,12] == 0:
                hol = dfClean.iloc[i,0]
            else:
                if dfClean.iloc[i,0] != hol:
                    booln = False
            i += 1
        self.assertTrue(booln)
```

Figure 22: Holiday Unit Test

Test for Unique Data Points

This test allowed us to verify that no two data points had the exact same date and hour. Originally, this test failed, which alerted us to the issue that we had repeat data points with different weather descriptions. We solved this by concatenating the weather data and deleting the repeated rows. Once we modified our data cleaning accordingly, the test passed and informed us that all of the rows had a unique date and time combination. However, it is worth noting that this test assumes the repeated data would be sequential in the set given that the data is in date and time order. Thus, a data point could be repeated non-consecutively and our test would still pass. We show the code for this test in Figure 23. We would be interested in discovering more advanced functionality to determine true uniqueness rather than the consecutive uniqueness that we test here.

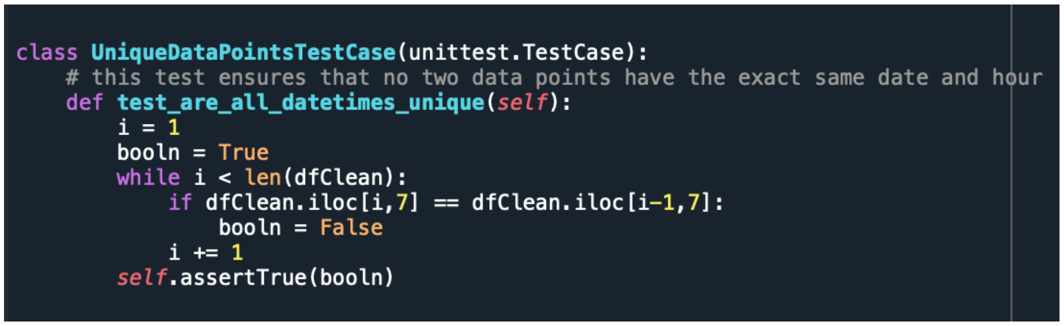

```
class UniqueDataPointsTestCase(unittest.TestCase):
    # this test ensures that no two data points have the exact same date and hour
    def test_are_all_datetimes_unique(self):
        i = 1
        booln = True
        while i < len(dfClean):
            if dfClean.iloc[i,7] == dfClean.iloc[i-1,7]:
                booln = False
            i += 1
        self.assertTrue(booln)
```

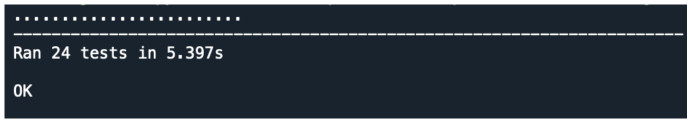
Figure 23: Unique Data Point Unit Test

Test the Conversion for Temperature

The temperatures given in the data set were in units of Kelvin. Therefore, after we converted them to Fahrenheit, we had to make sure that the values associated were appropriate. Due to our decision to overwrite the Kelvin values rather than append a Fahrenheit column, we could not test the actual calculation itself. However, due to such large discrepancies in the numerical values of Kelvin and Fahrenheit, we tested to see if there were any illogical Fahrenheit values that would indicate they may still be in Kelvin. We set our test to fail if the temperature was greater than 120 degrees Fahrenheit or less than negative 50 degrees Fahrenheit. The failure of this test led to the discovery of a handful of data points with temperatures below -450 degrees Fahrenheit. Although the calculation was happening correctly, the Kelvin value was set at zero, which indicated that the data was missing. Once we accounted for this, our test passed, indicating that all temperatures fell between “-50” and “120” degrees Fahrenheit.

Results of Testing

We show the output of running the unit test file with all 24 tests in Figure 24. These tests correspond solely to the data cleaning and sanitization portion of the project. The importance of these tests is that they enabled us to proceed with confidence into the visualization component. While we did not conduct structured unit tests for the visualizations as we did with the data cleaning and manipulation, we tested them by viewing the output of the visuals and ensuring that they corresponded to what we were trying to achieve. This allowed us to contextually ensure that our figures resembled their code prior to interpreting the information they provided.



```
........................
----------------------------------------------------------------------
Ran 24 tests in 5.397s

OK
```

Figure 24: Unit Test Results

Conclusion

When we set out at the start of our project, our goal was to discover various trends in traffic volume and the factors that cause these trends. We did this to demonstrate the kind of analyses a GPS estimator might incorporate to determine an estimated travel duration for some date and time in the future. After extensive cleaning, we were able to visualize much of the data and hone in on observable trends. We drew meaningful conclusions on how hour, date, time, holiday, temperature, and weather all affect traffic patterns in their own way. These analyses could be used in a GPS system to make predictions about future traffic patterns based on what we have learned from the historical data in our dataset. Given the opportunity, we would like to dive deeper into the effects of temperature and weather conditions on traffic volume. While we touched on temperature, rainfall, and snowfall, we did not even begin the analyses on cloud coverage and weather descriptions despite their presence in our dataset. Nevertheless, we were able to reach meaningful conclusions and extract valuable insight from the analyses we did perform.

Works Cited

Hogue, John. "Metro Interstate Traffic Volume Data Set." UCI Machine Learning Repository, University
of California, Irvine, School of Information and Computer Sciences, 2019, https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume#.