Volvo Truck Analytics

Ioannis Batsios, William Downs, Wahab Ehsan, James Polk, and Christopher Thacker

General Overview

- Bill:
  - APU Predictions by Regression
- Wahab:
  - Row/column removal
  - Outlier Detection
  - Oil Temperature Hypothesis Testing
  - Time for Ideal Oil Temperature Depending on Outside Temperature
- Ioannis:
  - External Temperature Effects on Engine Components
- James:
  - Time Series Analysis on CPU and Related Components
- Christopher:
  - Column Renaming
  - GPS Speed VS. Wheel-Based Speed
  - Truck 2 GPS Speed Corrections

Row/Column removal and Basic Statistics - Wahab

- Clean rows with more than certain amount of NaN Types.
- Delete column if no value given for any row.
- Basic Statistics:
  - Function divideByDay finds average for the whole day for the attribute given.

```
08/05/2019    77.935006
08/06/2019    73.576752
08/07/2019    76.170885
08/08/2019    74.289625
08/09/2019     3.091957
08/10/2019     2.655035
08/12/2019     2.966259
Name: Speed (km/hr), dtype: float64
```

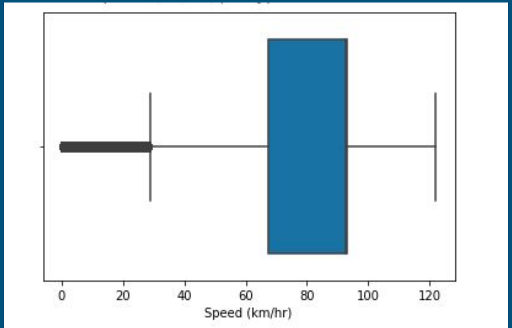
Truck 1

```
03/11/2019    25.192689
03/12/2019    19.167827
03/13/2019    25.641917
Name: Speed (km/hr), dtype: float64
```

Truck 2

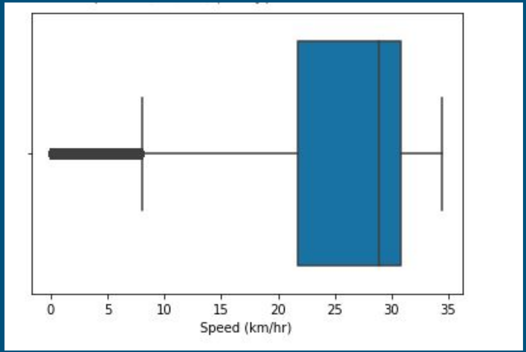
Outlier Detection - Wahab

Both trucks had several outliers making the data seem scattered.
Made function to show boxplot and then had it remove outliers.



Truck 1

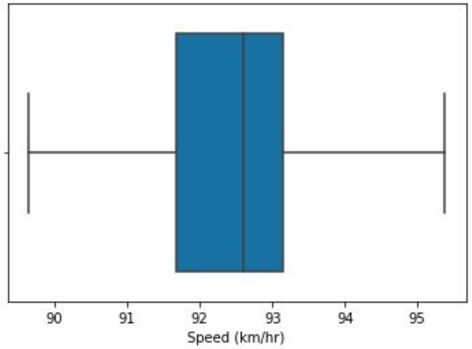
Truck 1
Before Outlier Deletion
Min: 0.0
Quartile 1: 67.22
Median: 92.60
Quartile 3: 93.15
Max: 122.05



Truck 2

Truck 2
Before Outlier Deletion
Min: 0.0
Quartile 1: 21.71
Median: 28.91
Quartile 3: 31.87
Max: 34.47

```
After Outlier Deletion: [28.70599937438965, 91.67400360107422, 92.5999984741211, 93.15560150146484, 122.04680633544922]
Time (DateTime)
08/05/2019    88.711232
08/06/2019    86.577039
08/07/2019    86.438914
08/08/2019    84.607654
Name: Speed (km/hr), dtype: float64
```



Truck 1

<u>After Outlier Deletion</u>

Min: 28.71

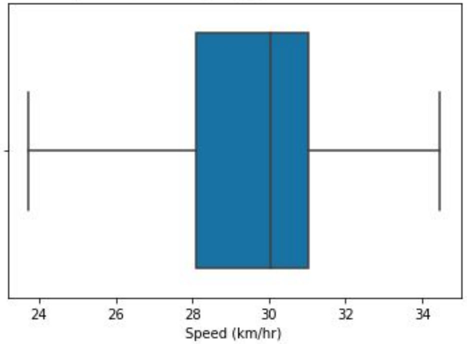
Quartile 1: 91.67

Median: 92.60

Quartile 3: 93.16

Max: 122.05

```
After Outlier Deletion: [8.025333616532958, 28.088665008769745, 30.043555365908862, 31.020999484840303, 34.467777676328375]
Time (DateTime)
03/11/2019    29.578152
03/12/2019    28.254743
03/13/2019    28.565267
Name: Speed (km/hr), dtype: float64
```



Truck 2

<u>After Outlier Deletion</u>

Min: 8.03

Quartile 1: 28.10

Median: 30.04

Quartile 3: 31.02

Max: 34.47

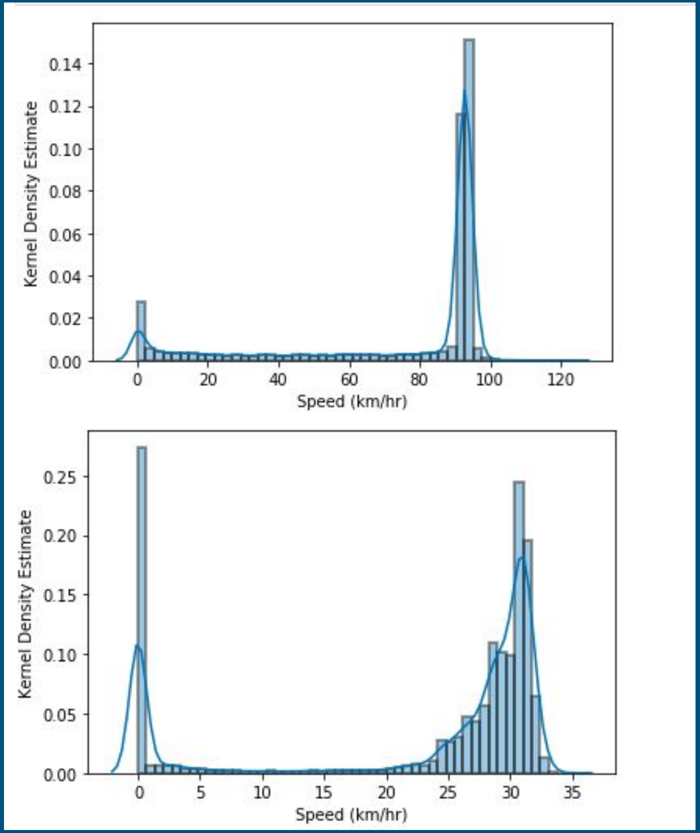
Determining the Estimator - Wahab

Decided with Kernel Density Estimation (KDE).

Non-parametric estimator

There is no assumption for underlying distribution of variables.

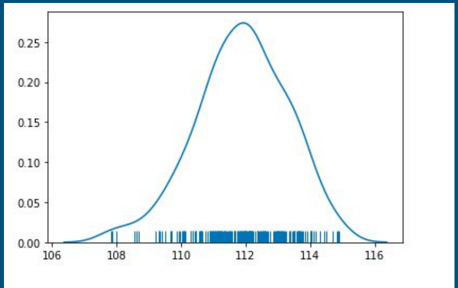
Large Bandwidth since the data is mainly parsed around.

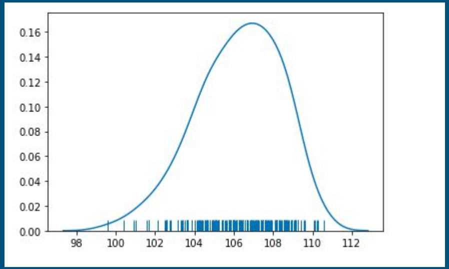


Truck 1

Truck 2

Oil Temperature - Wahab





Ho: Temperature of the oil in both Trucks will remain the same.

Ha: Temperature of the oil in both Trucks will differ from each other.

Using Two Sample T-test with confidence interval of 95%.

```
Ttest_indResult(statistic=-30.33791875326352, pvalue=3.3950875287898653e-102)
```

Reject the Null hypothesis. P-value less than 0.05. Therefore, there is difference in the Oil Temperature between trucks.

Problems and Possible Solutions - Wahab

- Made function to find time between rows.
- Found average oil temperature for each ‘session’ for ideal temperature.
- Filtered and ignored data with sessions less than 300 seconds and temperature difference of less than 30 degrees C.
- Was able to find average outside temperature using divide by day function.

```
Time (DateTime)
08/05/2019    29.521704
08/06/2019    28.527456
08/07/2019    28.066670
08/08/2019    25.790046
08/09/2019    27.893517
08/10/2019    19.213840
08/12/2019    22.412655
Name: Outside Air Temperature (C), dtype: float64
```

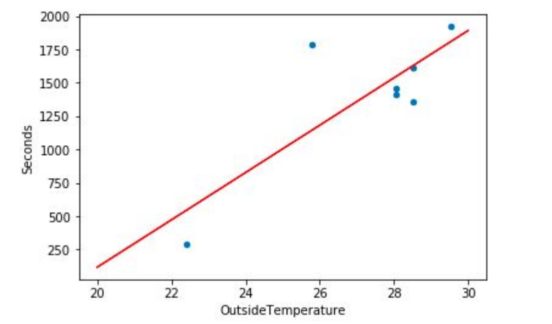
Prediction using Linear Regression - Wahab

- Using statsmodels, was able to figure out the coefficients.
- Seconds = (177.3577 * Outside Temperature) - 3429.54977
- Using 20 degrees, 25 degrees, 30 degrees Celsius, was able to get the following predictions:

```
0      117.604418
1     1004.392964
2     1891.181511
dtype: float64
```

20° C
25° C
30° C

- Plot of Least Square Line =

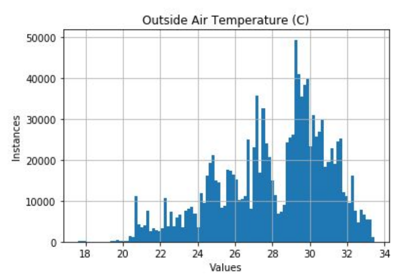


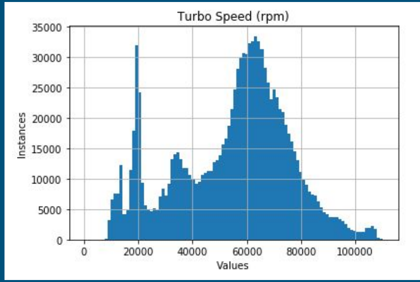
Temperature Analysis - Ioannis

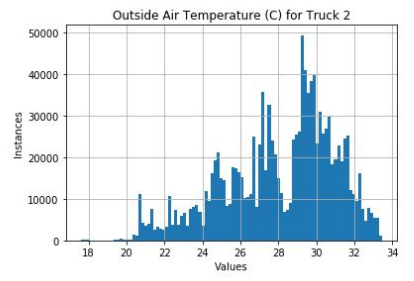
- To determine if temperature affects engine performance.
- Per research - Turbospeed is the main engine part affected by external air temperature.

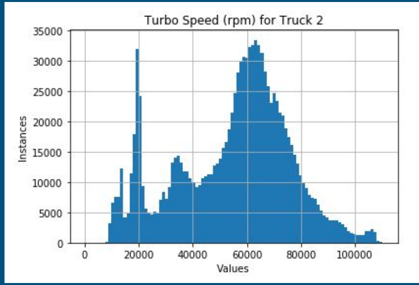
- Hypothesis
  - Null hypothesis: Mean of Turbospeed = Mean of Air Temperature
  - Alternative hypothesis: Mean of Turbospeed ≠ Mean of Air Temperature

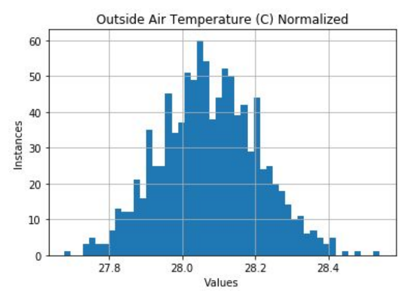
Temperature Analysis - Ioannis

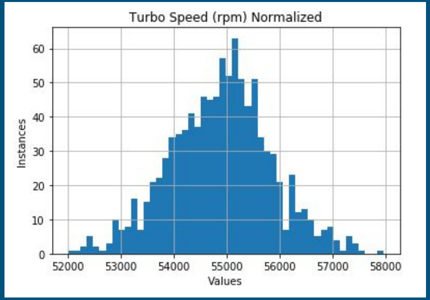


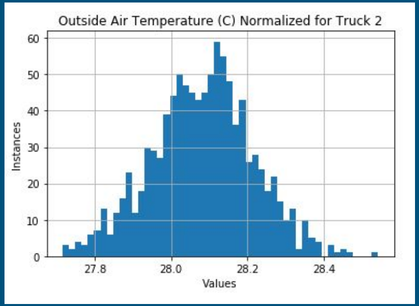


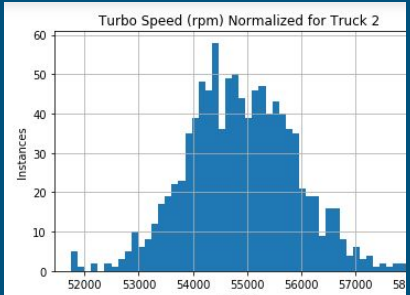












Temperature Analysis - Ioannis

- Using Pearson to test for correlation coefficient and p-value for testing non-correlation I found:

```
In [11]: stats.pearsonr(turbo_points, air_points)
Out[11]: (-0.01443574086022436, 0.6484252039220186)
```

```
In [19]: stats.pearsonr(turbo_points2, air_points2)
Out[19]: (-0.05282380444583l8, 0.09501599378699249)
```

- P-values are greater than .05, therefore we accept null hypothesis that the mean of TurboSpeed and Ambient Air are equal to each other.

Temperature Analysis - Ioannis

Since I knew that Ambient Air Temperature can affect engine components, I wanted to use that knowledge to create a multiple regression line.

- Reasoning:
  - If we were wanted to improve a specific component, we'd know the temperature that is needed to optimize the component.
    - These types of tests are often used to pass medicine and food to consumers that may or may not improve a person's health dependent on certain conditions. Why not engine parts?

Temperature Analysis - Ioannis

First task was to keep only columns I needed:

```
truck1_temp_only.head()
```

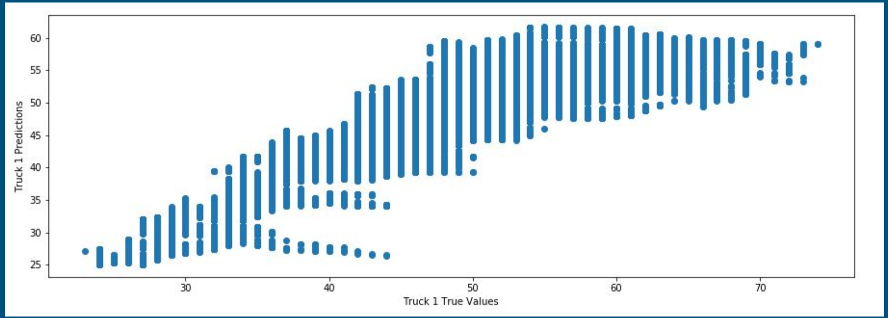
	EngineCoolantTemp_stat	EngineOilTemperature	EngIntakeManifold1Temp	TransmissionOilTemp	AmbientAirTemperature_V
16	31.0	26.65625	27.0	27.84375	21.34375
17	31.0	26.65625	27.0	27.84375	21.34375
18	31.0	26.65625	27.0	27.84375	21.34375
19	31.0	26.65625	27.0	27.84375	21.34375
20	31.0	26.65625	27.0	27.84375	21.34375

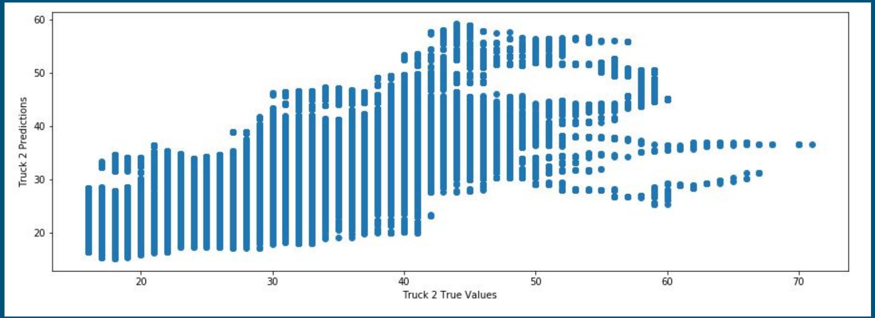
```
truck2_temp_only.head()
```

	EngineCoolantTemp_stat	EngineOilTemperature	EngIntakeManifold1Temp	TransmissionOilTemp	AmbientAirTemperature_V
0	79.0	118.875	33.0	85.0	16.9375
1	79.0	118.875	33.0	85.0	16.9375
2	79.0	118.875	33.0	85.0	16.9375
3	79.0	118.875	33.0	85.0	16.9375
4	79.0	118.875	33.0	85.0	16.9375

Temperature Analysis - Ioannis

Using MatPlotLib to create a scatter plot showing the predicted values based on the actual values





Temperature Analysis - Ioannis

Using SciKit Linear Regression Model tools I created a model and got my scores:

```
model1.score(X1_test, y1_test)
0.7062050718088208
```

```
model.score(X_test,y_test)
0.7341028591090475
```

Temperature Analysis - Ioannis

ANOVA:

OLS Regression Results

Dep. Variable:	y	R-squared:	0.704
Model:	OLS	Adj. R-squared:	0.704
Method:	Least Squares	F-statistic:	7.241e+05
Date:	Wed, 11 Dec 2019	Prob (F-statistic):	0.00
Time:	23:47:21	Log-Likelihood:	-3.0963e+06
No. Observations:	1216197	AIC:	6.193e+06
Df Residuals:	1216192	BIC:	6.193e+06
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-7.8663	0.044	-176.954	0.000	-7.953	-7.779
x1	0.5260	0.001	618.388	0.000	0.524	0.528
x2	-0.5932	0.001	-894.889	0.000	-0.595	-0.592
x3	0.7914	0.001	916.350	0.000	0.790	0.793
x4	0.7143	0.001	699.112	0.000	0.712	0.716

Omnibus:	340260.757	Durbin-Watson:	0.002
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1288210.875
Skew:	1.365	Prob(JB):	0.00
Kurtosis:	7.239	Cond. No.	2.48e+03

OLS Regression Results

Dep. Variable:	y	R-squared:	0.735
Model:	OLS	Adj. R-squared:	0.735
Method:	Least Squares	F-statistic:	1.167e+06
Date:	Wed, 11 Dec 2019	Prob (F-statistic):	0.00
Time:	23:47:21	Log-Likelihood:	-4.8943e+06
No. Observations:	1686308	AIC:	9.789e+06
Df Residuals:	1686303	BIC:	9.789e+06
Df Model:	4		
Covariance Type:	nonrobust		

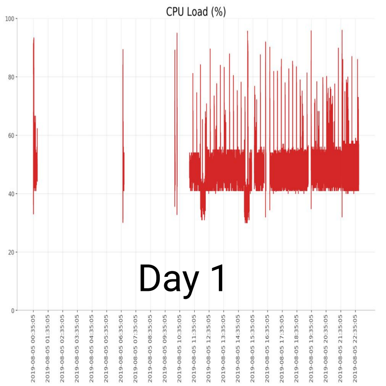
	coef	std err	t	P>|t|	[0.025	0.975]
const	-33.8231	0.132	-256.293	0.000	-34.082	-33.564
x1	1.1370	0.002	582.600	0.000	1.133	1.141
x2	-0.6087	0.001	-1001.456	0.000	-0.610	-0.607
x3	0.3967	0.001	272.600	0.000	0.394	0.400
x4	0.8608	0.001	1139.289	0.000	0.859	0.862

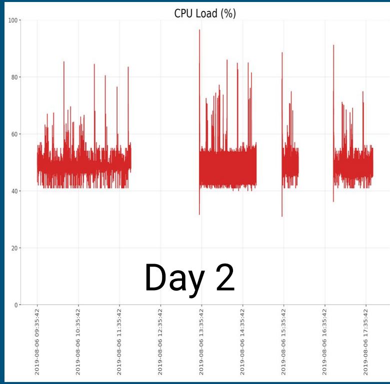
Omnibus:	347849.307	Durbin-Watson:	0.002
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1102257.968
Skew:	1.056	Prob(JB):	0.00
Kurtosis:	6.351	Cond. No.	6.23e+03

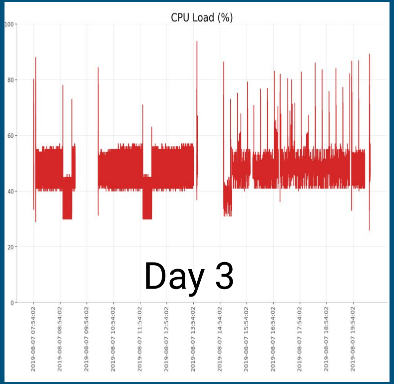
CPU and Time Series Analysis - James

- To determine if CPU output is related to sensor capture
- Perform Time Series Analysis to determine outliers

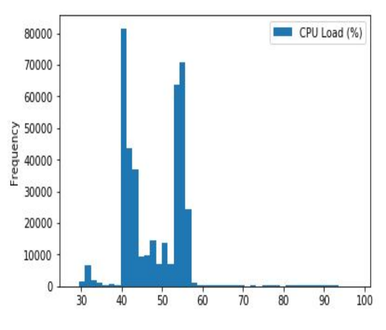
CPU Load - James

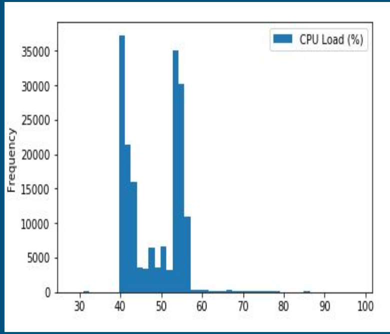


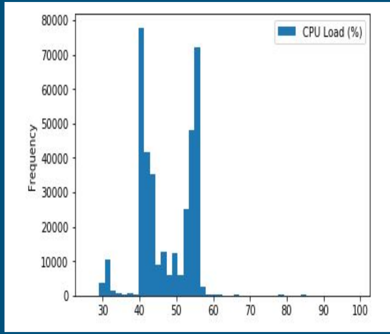


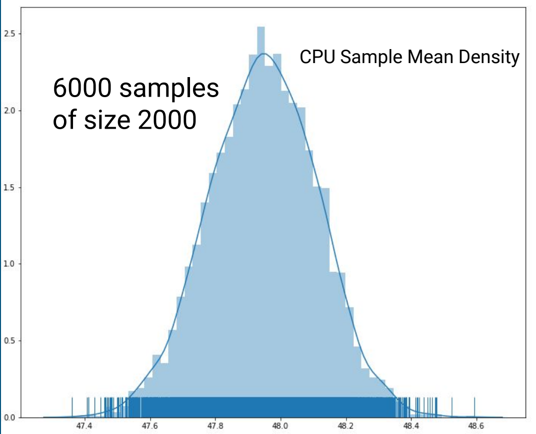


CPU Load (%)	
count	1,200,861
mean	47.94898
std	7.225541
min	26.00000
25%	42.00000
50%	47.00000
75%	54.00000
max	99.00000

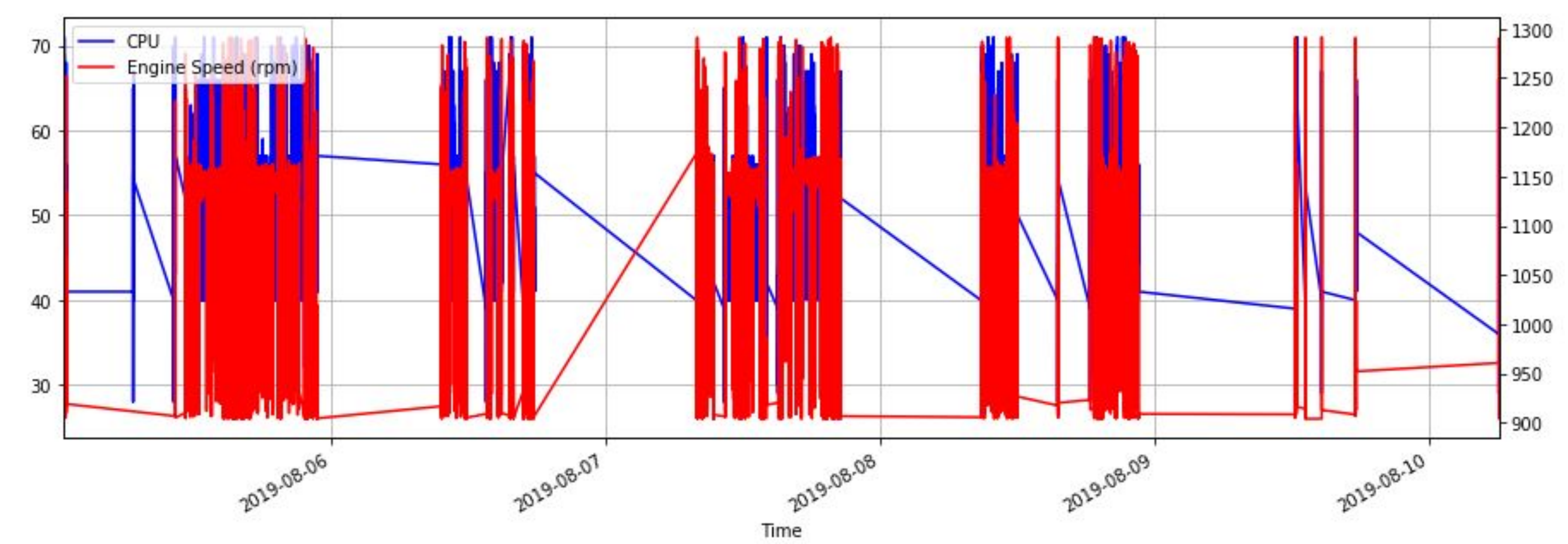






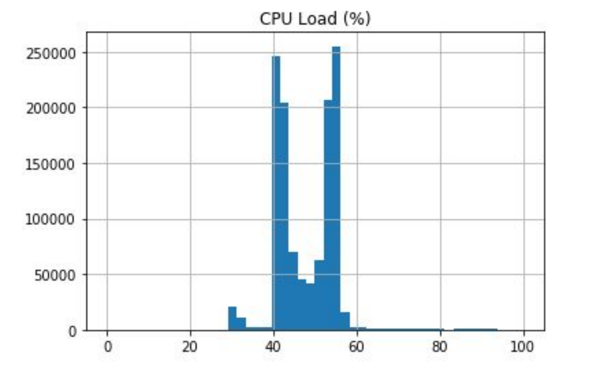


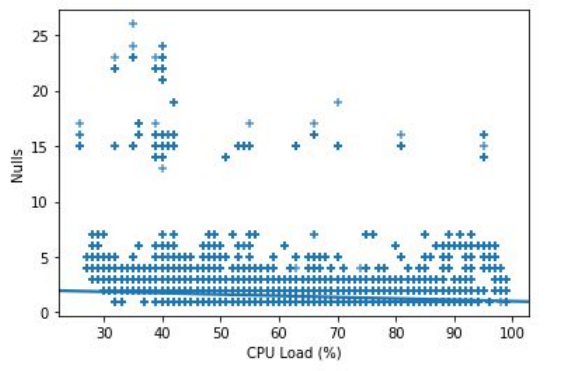
CPU Load: Why the Gaps? - James



CPU Load - Linear Regression - James

Is the CPU Load related to the number of active sensor readings?





Correlation:
-0.11680395911398092

R2 values using k-folds cross-validation (k=3):
0.01438574935982073,
0.013693048287461984,
0.01279950799256221

No significant correlation!

Time Series Analysis - James

Performed Time Series Analysis on CPU Load, Vehicle Weight, Driver Requested Torque, Outside Air Temperature, and Temperature of Air Entering Vehicle.

Problem:

Data is spotty with frequent gaps. Furthermore, data is sampled every 100 milliseconds, causing noise.

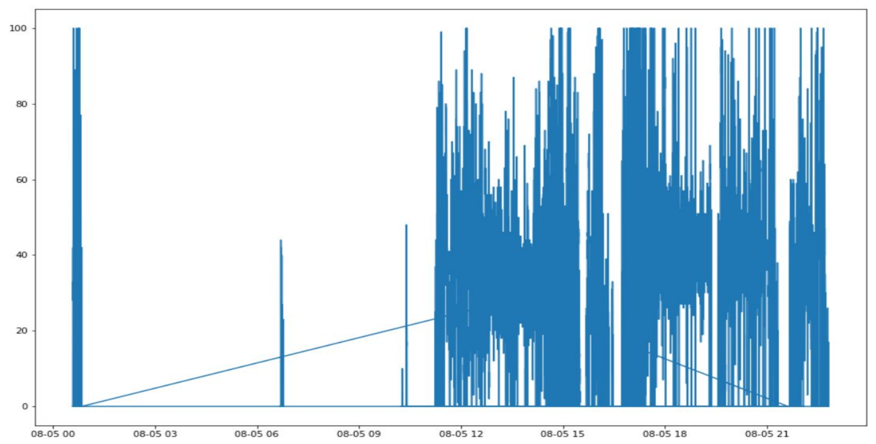
Attempted Solution

Resample data to every minute using mean.

Still frequent dropouts.

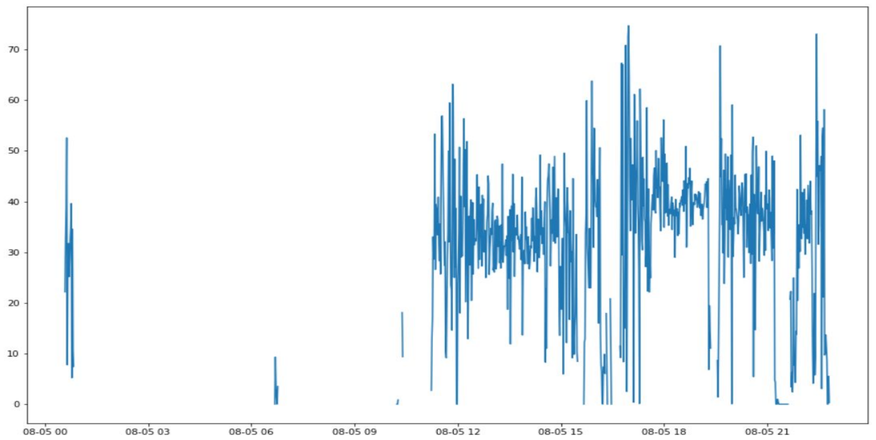
```
plt.plot(day1['Driver Requested Torque (%)'])
```

```
[<matplotlib.lines.Line2D at 0x2796037b308>]
```



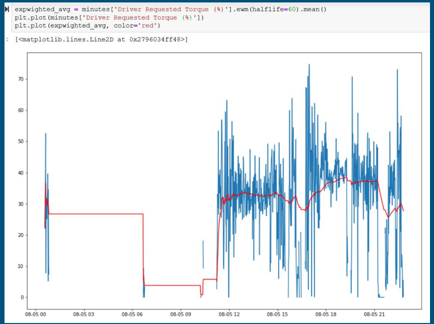

```
minutes = day1.resample('1T').mean()
plt.plot(minutes['Driver Requested Torque (%)'])
```

```
[<matplotlib.lines.Line2D at 0x27910eec788>]
```

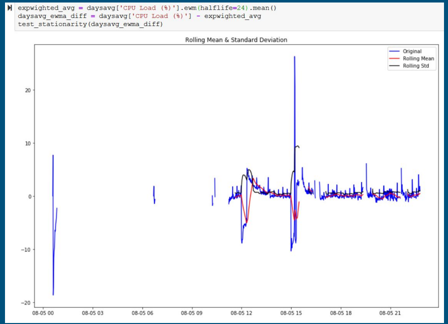


Exponentially Weighted Averages - James

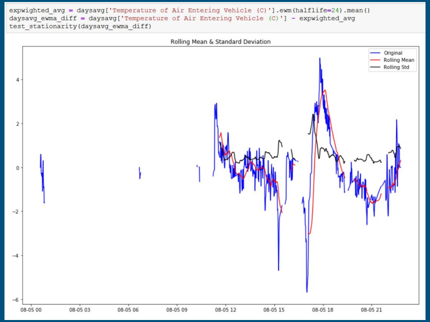
Driver Requested Torque



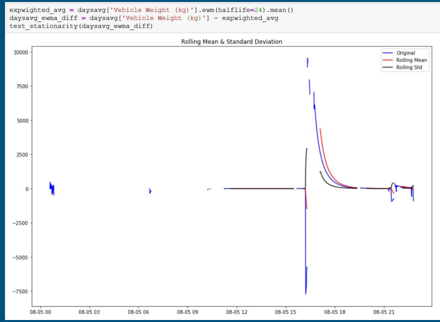
CPU Load



Outside Air Temperature



Temp. of Air Entering Vehicle



Vehicle Weight

Goals and Questions - Christopher

Confirm consistency between speed components.

- For both trucks:
  - GPS Speed.
  - Wheel-Based Vehicle Speed.
- Are both speed components reading at a similar level?
  - If not, what can be done about it?

Side goal:

- Make the column headers easier to read.
  - Makes it easier to “eyeball” columns.
  - Is this effective?

Renaming Columns - Christopher

- Implemented function to rename columns in a DataFrame.
    - Takes a Python dictionary in to convert names.
    - Modular.
    - No errors if a column is missing or if an extra column exists.
- Reads hand-typed dictionaries into usable Python dictionary.
    - Emphasis on accuracy of translations.
    - Includes units of measurement, if applicable.

Before renaming:

Time	1730_CH9_ AutomaticStartStop	1730_CH10_Truck_Batteries	4649_Ch1_Alternator_250A	4649_Ch2_BattOut_100A	4649_Ch3_Trailer_50A	4649_C

After renaming:

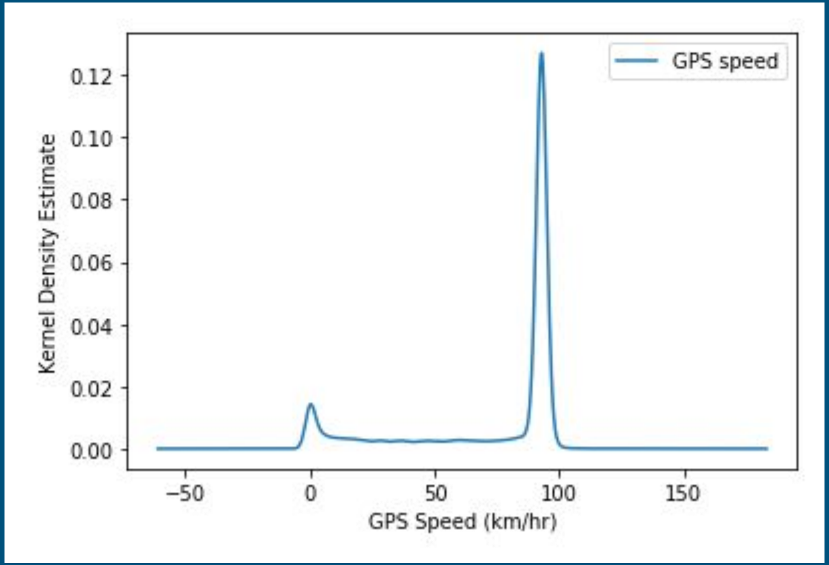
Time (DateTime)	1730 Automatic Start & Stop (V)	1730 Batteries (V)	4649 Alternator (A)	4649 Battery Out (A)	4649 Trailer (A)	4649 Inverter (A)	4649 Fridge (A)	4649 Battery Bank (A)	4649 Battery Separator (A)	...	Transmission Lube Temperature (C)	Transmission Lube Level (%)	Transn Pr

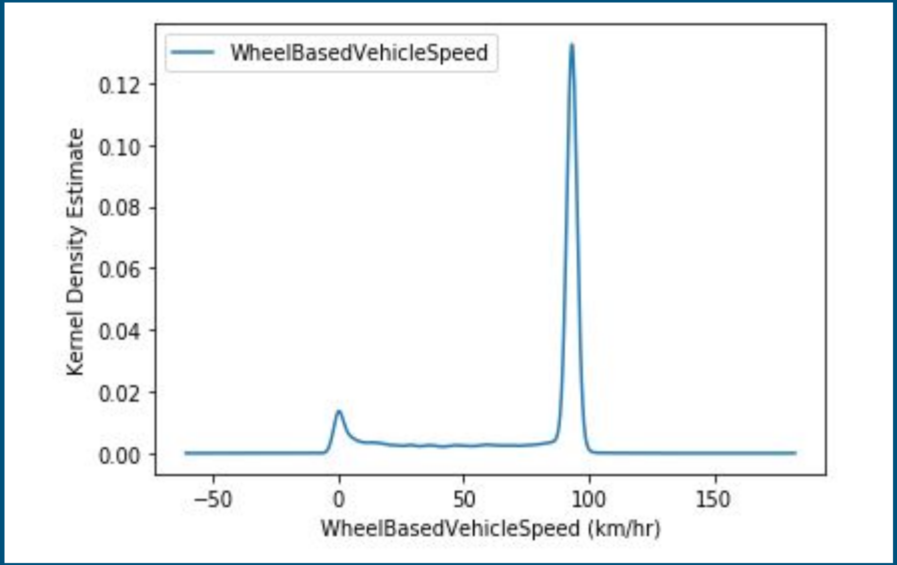
GPS Speed vs. Wheel-Based Speed - Christopher

- Goals
  - Effectiveness of speed-measuring components.
  - Consistency between these components.
  - Basic understanding & exploration of data.
  - Basic data statistics.
  - Long-haul or short-haul?

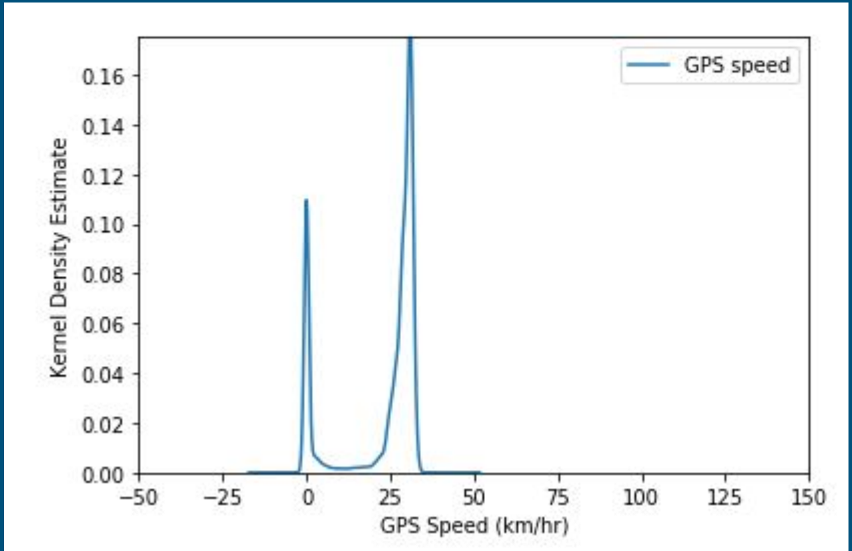
	Truck 1		Truck 2	
	GPS Speed	Wheel-Based Speed	GPS Speed	Wheel-Based Speed
Mean	74.55 km/hr (46.32 mph)	74.83 km/hr (46.50 mph)	22.69 km/hr (14.10 mph)	82.13 km/hr (51.04 mph)
Standard Deviation	31.94 km/hr (19.85 mph)	31.96 km/hr (19.86 mph)	12.18 km/hr (7.57 mph)	44.08 km/hr 27.39 mph

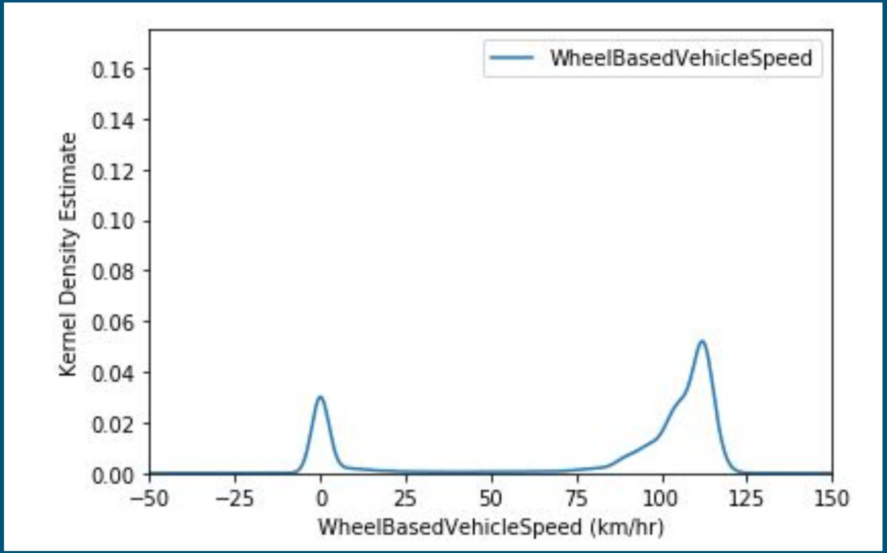
Truck 1: KDE Distributions - Christopher





Truck 2: KDE Distributions - Christopher





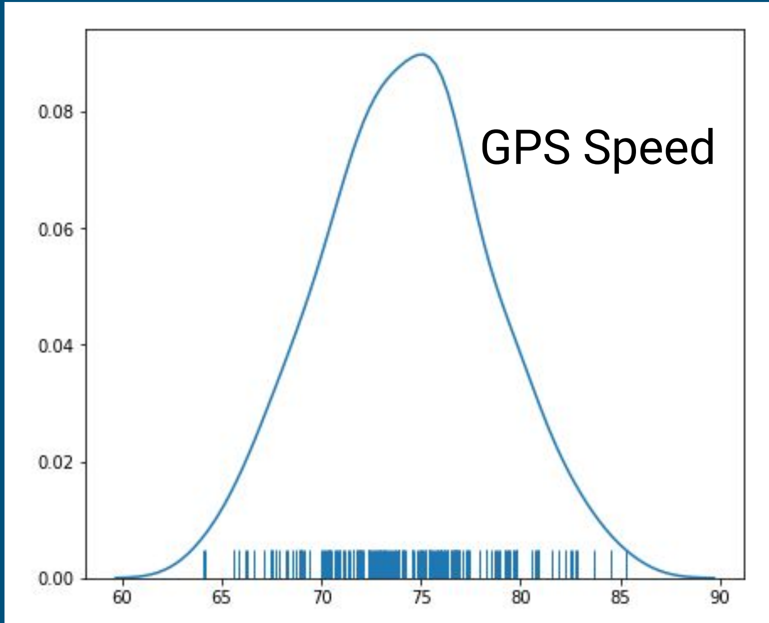
Hypothesis Testing for Speeds - Christopher

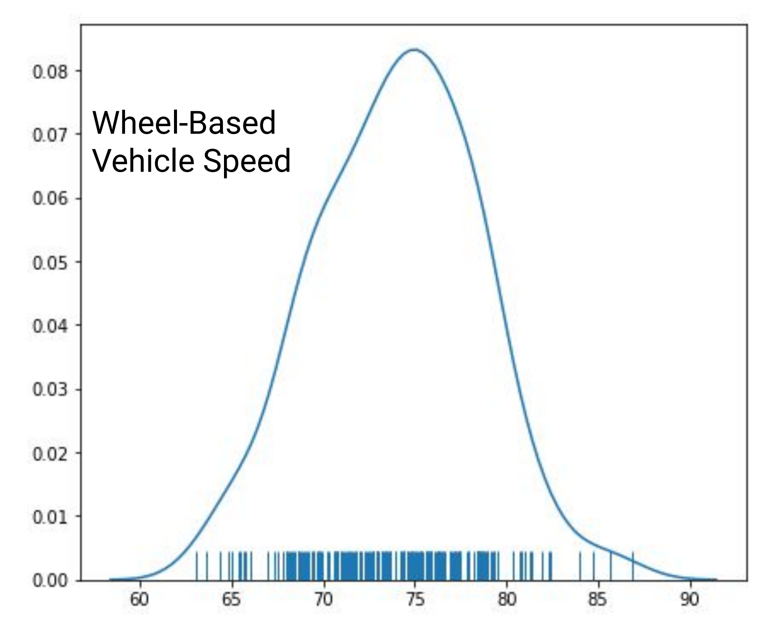
For these two speed components:

- The hypothesis test was:
  - H0: There are no differences in speeds between the two components.
  - Ha: There is a significant difference in speed between the two components.
    - Indicates two-tailed test (Scipy defaults to this).
- Bimodal Distribution:
  - Two-sample t-test will be used.
  - Central Limit Theorem required.

Using the Central Limit Theorem for Truck 1
Christopher

Truck 1: 200 Samples of Size 50 for Both Components





Two-Sample T-Test Results on Truck 1
Christopher

Two-Sample T-Test:
```
Ttest_indResult(statistic=0.5068109121220089, pvalue=0.6125689000122868)
```

- Confidence level of 95%.
  - Alpha level of 0.05.
- Using Scipy's built-in two-sample t-test:
  - P-value was ~0.613
  - Higher than alpha level of 0.05.

Thus, we fail to reject the null hypothesis and can assume that there is no significant difference between the measurements of both speed components for Truck 1.

Truck 2's P-value was lower than 0.05 (~1.12e-233) when the same test was ran. Thus, for Truck 2, we reject the null hypothesis and conclude that there is significant difference.

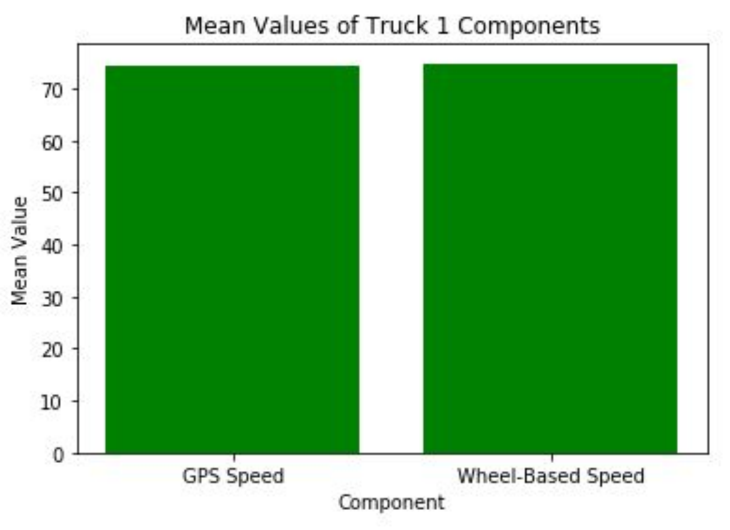
“Correcting” Truck 2’s GPS Speed Values
Christopher

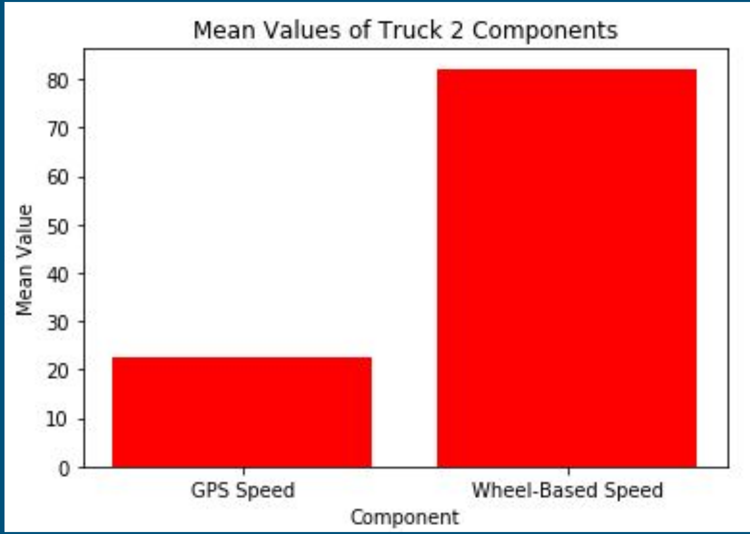
- Truck 2 has faulty GPS Speed data.
  - The component was clearly reading much lower than its Wheel-Based Speed counterpart.
  - Its faulty values are proven by the consistency in Truck 1’s data with the same components.
  - Misconfigured?
  - Different units?
  - Broken component?
- Goal:
  - Use the data and trends of Truck 1 to help predict, or correct, GPS Speed values for Truck 2.
  - Note: this was an analytical task, NOT a machine learning task.

The Data at Hand - Christopher

Recall the data for both trucks.

- Like Truck 1’s data, Truck 2’s values should have been relatively identical.



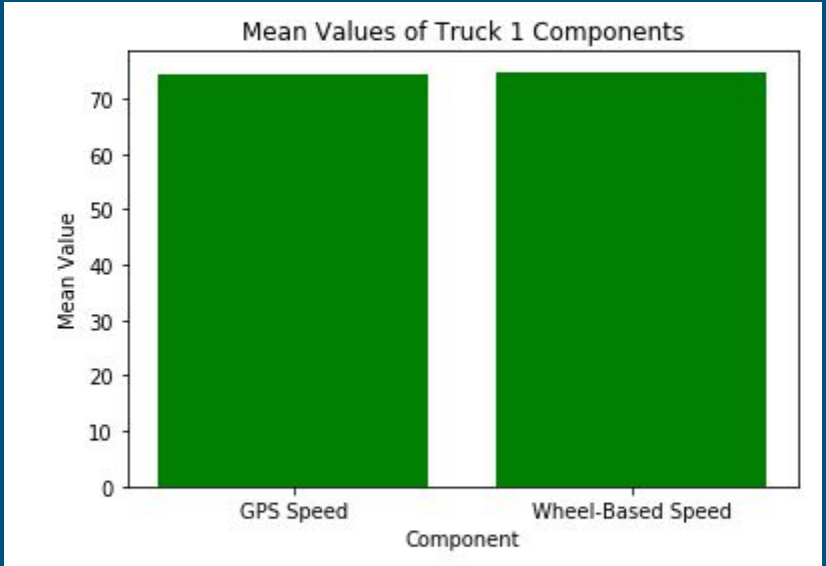


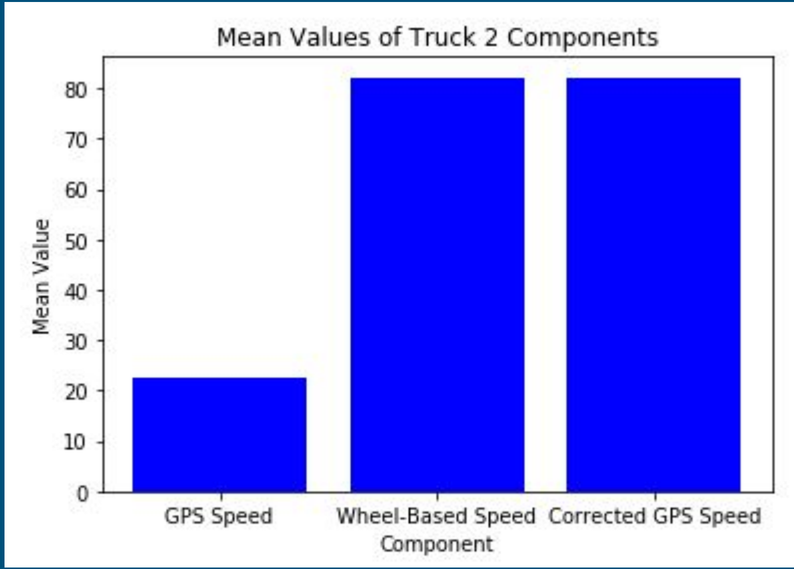
The Analytical Solution - Christopher

Since Truck 1 had “better” data than Truck 2:

- Utilize Truck 1’s data to “predict” or “correct” Truck 2’s GPS Speed.
  - Given any Wheel-Based Speed value for Truck 2.
- Implementation:
  - Calculate mean difference of GPS Speed and Wheel-Based Speed for Truck 1.
    - -0.28539807628549124 (GPS - Wheel).
    - Indicates that GPS is reading slightly lower than Wheel-Based.
  - Iterate through Truck 2’s Wheel-Based Speed data.
    - Add the difference of means to each one and store the result in a new column.

Resulting Data - Christopher






```
GPS Speed Mean: 74.5450583567317
Wheel-Based Speed Mean: 74.83045643301719
```

```
GPS Speed Mean: 22.685516192007974
Wheel-Based Speed Mean: 82.1331223040454
Corrected GPS Speed Mean: 81.84772422776032
```

APU Truck 1 data

With all things considered more data is generally better. We had a lot of data but when we resampled it we were only left with less than a weeks worth.

This was good. Our economical student machines aren't built for a huge amount of data and it made us think a little more about our problems.

It was helpful our mentor, Daniel Wingo, knew this as he is a Data Engineer at Volvo.

APU - data used to evaluate

Over 1 million rows and 51 columns of data for Truck 1, I had to scale back the un needed and possibly irrelevant columns to clean the data.

I then resampled to days and minutes.

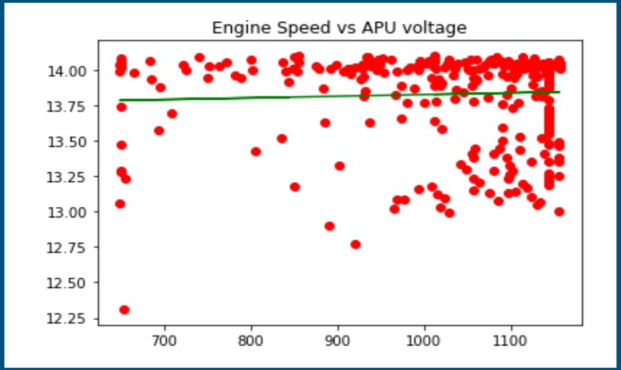
Features used: APU battery bank, Alternator amps, Ambient air temp, Refrigerator, and Inverter.

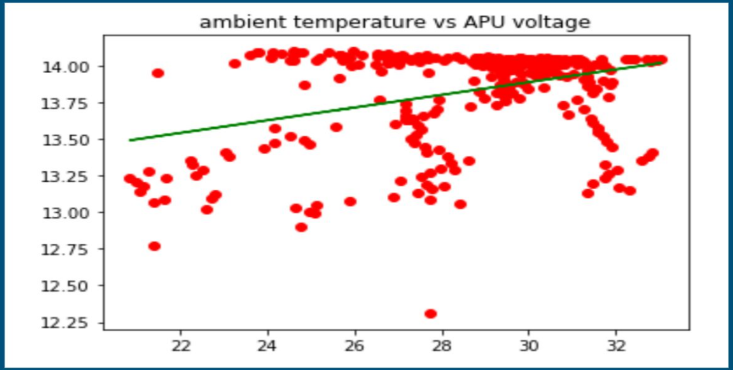
*initially I had no clue what features were associated with the APU unit.

What Volvo wanted to know

This was tricky. They wanted to know if the apu was “worth it” compared to the other options available. Obviously, we didn’t have data on the other options so my first assumptions were to investigate what would use the apu battery bank or what other factors, such as temperature could be accounted for with analytic techniques.

Some things did not have a very good correlation...

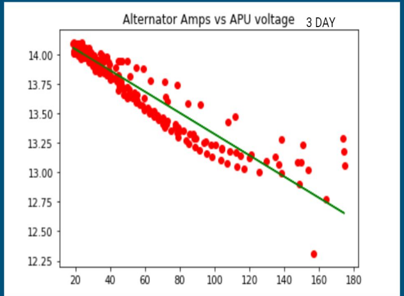


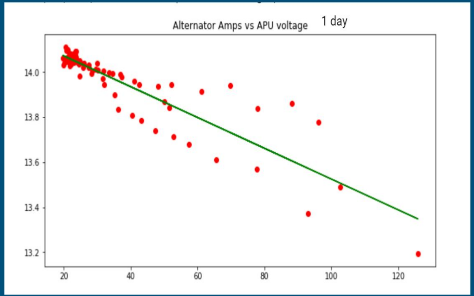


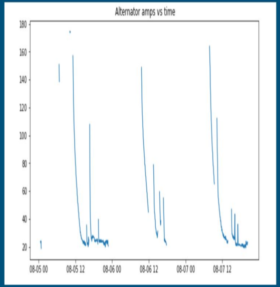

```
******Ambient temp and APU voltage**
correlation
 [[1.          0.36553273]
 [0.36553273 1.        ]]

covariance
 [[0.1030681  0.31816982]
 [0.31816982 7.35091062]]

******Engine speed and APU voltage**
correlation
 [[1.          0.05023186]
 [0.05023186 1.        ]]

covariance
 [[1.03068101e-01 2.20561877e+00]
 [2.20561877e+00 1.87058778e+04]]
```

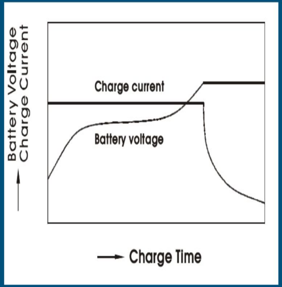
Some good correlations

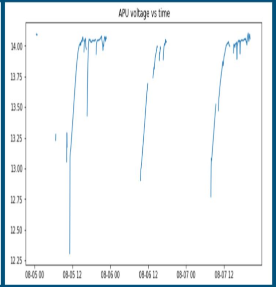
This should make us happy. Using the skills taught in computer science and applying them to see a physical attribute of the real world.











Scatter plots with regression and time series seem to match up well.
This was cool!

But.. issues :/

The only thing that that runs off APU battery bank was a compressor.. Which we didn't have data for. BUMMMER

This was good though, because in the real world as data scientist or computer scientist we won't always have the full picture and better investigation techniques or other data may need to be requested.

So more "thought experiments" were needed to find something cool.

MACHINE LEARNING!!

With the help of professor Mohanty, I came up with the idea of using logistic regression(binary classification) to predict what times of day the APU battery bank 'probably' be charged. 14 volts = charged... else not charged. Volvo E.engineers were happy with this value.

80/20, test_train_split results!

```
#get some score values for 'accu
logReg.score(X_train,y_train)
```

]: 0.948339483394834

```
logReg.score(X_test,y_test)#this
```

]: 0.8676470588235294

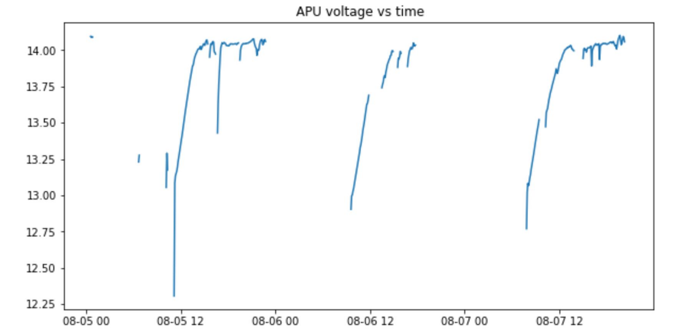
```
[[25  4]
 [ 5 34]]
```

	precision	recall	f1-score	support
0	0.86	0.83	0.85	30
1	0.87	0.89	0.88	38
micro avg	0.87	0.87	0.87	68
macro avg	0.87	0.86	0.87	68
weighted avg	0.87	0.87	0.87	68

Prediction results

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 0])
```

But is it believable?? Maybe. More data is probably needed.



Time	4649_Ch8_	4649_Ch1_Alternator_250A		
8/7/2019 11:15	13.7934	41.34075		array([[9.99982618e-01, 1.73824566e-05],
8/7/2019 19:30	14.08159	21.25699		[8.69124474e-03, 9.91308755e-01],
8/7/2019 16:10	13.99505	27.67186		[5.67938696e-01, 4.32061304e-01],
8/5/2019 11:50	13.28952	88.28844		[1.00000000e+00, 2.24537236e-21],
8/7/2019 18:40	14.04888	21.67571		[1.22176306e-02, 9.87782369e-01],
8/7/2019 20:10	14.08686	21.38521		[9.55600452e-03, 9.90443995e-01],
8/5/2019 16:45	13.6376	71.85345		[1.00000000e+00, 9.13247721e-16],
8/5/2019 22:35	14.06674	24.41085		[9.18741704e-02, 9.08125830e-01],
8/6/2019 9:35	12.90422	148.66		[1.00000000e+00, 9.80310780e-42],
8/5/2019 11:40	13.23158	100.4002		[1.00000000e+00, 1.87058693e-25],
8/7/2019 13:25	14.03414	24.96136		[1.36447599e-01, 8.63552401e-01],
8/5/2019 16:40	13.43022	107.6826		[1.00000000e+00, 7.64291541e-28],
8/6/2019 10:00	13.07478	106.9722		[1.00000000e+00, 1.05713032e-27],
8/6/2019 17:15	14.01223	23.58174		[5.23006501e-02, 9.47699350e-01],
8/6/2019 14:25	13.93646	29.94169		[8.87343640e-01, 1.12656360e-01],
8/7/2019 10:40	13.65863	55.28275		[1.00000000e+00, 3.35511392e-10],
8/5/2019 22:20	14.07108	21.82358		[1.34879714e-02, 9.86512029e-01],
8/5/2019 17:45	14.03608	27.73642		[5.73807170e-01, 4.26192830e-01],
8/6/2019 14:00	13.84889	39.91026		[9.99945646e-01, 5.43542615e-05],
8/7/2019 15:40	14.01707	27.38171		[5.08846606e-01, 4.91153394e-01],
8/7/2019 11:35	13.87242	35.76111		[9.98641269e-01, 1.35873109e-03],
8/7/2019 13:10	14.02427	25.28241		[1.69261536e-01, 8.30738464e-01],
8/5/2019 17:50	14.03196	26.74513		[3.85618378e-01, 6.14381622e-01],
8/5/2019 13:05	13.76529	44.09613		[9.99997968e-01, 2.03235196e-06],
8/5/2019 13:40	13.92353	30.31995		[9.14061711e-01, 8.59382885e-02],
8/5/2019 17:25	14.04727	27.22046		[4.72964583e-01, 5.27035417e-01],
8/5/2019 0:45	14.09451	24.04218		[6.95704936e-02, 9.30429506e-01],
8/7/2019 17:30	14.04748	20.92801		[6.90001308e-03, 9.93099987e-01],
8/7/2019 15:05	14.00622	34.63013		[9.96465576e-01, 3.53442357e-03],
8/7/2019 15:15	14.00354	28.96895		[7.80756646e-01, 2.19243354e-01],
8/7/2019 7:55	13.02187	153.8548		[1.00000000e+00, 1.90908913e-43],
8/6/2019 17:35	14.03098	23.04861		[3.48672123e-02, 9.65132788e-01],
8/7/2019 17:35	14.04485	21.78971		[1.33607444e-02, 9.86639256e-01],
8/6/2019 15:30	13.88244	59.31233		[1.00000000e+00, 1.71870705e-11],
8/5/2019 21:10	14.07798	23.99426		[6.78693974e-02, 9.32130603e-01],
8/7/2019 15:25	13.98732	26.63547		[3.72316476e-01, 6.27683524e-01],
8/5/2019 16:25	13.97393	24.64454		[1.13867840e-01, 8.86132160e-01],
8/7/2019 19:25	14.06612	19.87486		[3.03460431e-03, 9.96965396e-01],
8/5/2019 16:20	13.98586	21.51933		[1.12777054e-02, 9.88722295e-01],
8/7/2019 15:55	14.02369	25.08682		[1.49111214e-01, 8.50888786e-01],
8/5/2019 19:00	14.04617	24.31895		[8.71479881e-02, 9.12852012e-01],
8/5/2019 19:40	14.02389	23.80275		[6.10158271e-02, 9.38984173e-01],
8/5/2019 14:40	14.01355	22.51167		[2.35569586e-02, 9.76443041e-01],
8/5/2019 14:25	14.01467	24.21654		[8.25507516e-02, 9.17449248e-01],
8/5/2019 12:25	13.51681	63.31943		[1.00000000e+00, 6.17394284e-13],
8/7/2019 11:50	13.86059	31.53672		[9.65906829e-01, 3.40931709e-02],
8/7/2019 11:30	13.84261	38.42224		[9.99829123e-01, 1.70876717e-04],
8/7/2019 15:10	14.01411	28.85791		[7.64521341e-01, 2.35478659e-01],
8/5/2019 18:15	14.04241	24.08706		[7.40739809e-02, 9.25926019e-01],
8/5/2019 14:15	14.00966	24.39401		[9.38145695e-02, 9.06185431e-01]

ML Testdata

Volvo Trucks Data Science Conclusions

The project was not easy. Coming up with questions was hard as we are not electrical or mechanical engineers, but it gave us a good intro into "how to ask questions" on engineering type data.

Good data or bad data or wacky data; some insights can be made if there is enough data, and we had more than enough data to evaluate many different sensors over a couple of days time. Any data than what was given may have been limited due to our hardware. Compiling many rows sometimes took a while.

Did we answer anything for Volvo? We think so! in the case Daniel Wingo is asked about a particular sensor and if that data generated is worth investigating because of some questionable correlation he may have a roadmap now to save time on where to start.