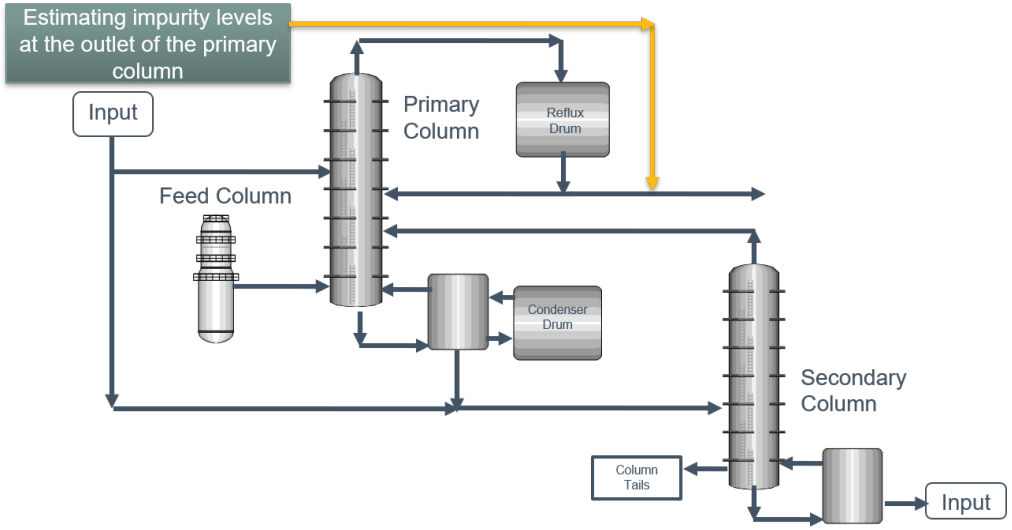
Goals for Today:

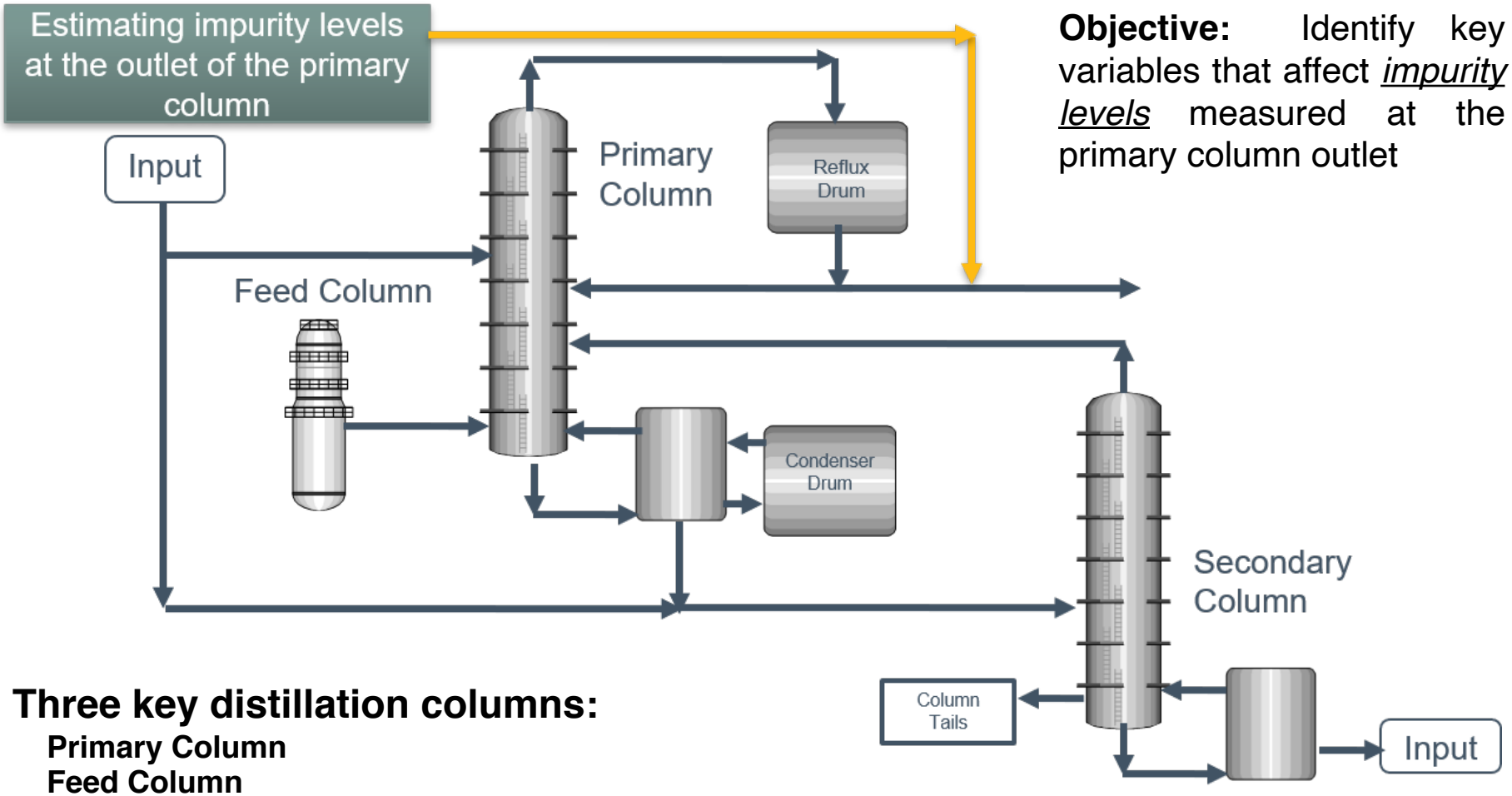
Basic Data Exploration:

- Dow Challenge Dataset
- Preliminary Investigation
- Cleaning Data
- Correlation



Dow Dataset - Overview

Dow has generously given us operating data for a plant:



Objective: Identify key variables that affect *impurity levels* measured at the primary column outlet

Three key distillation columns:

- **Primary Column**
- **Feed Column**
- **Secondary Column**

Dow Dataset - Overview

Summary of Variables

Primary Column Variables	Secondary Column Variables	Feed Column Variables
x1:Primary Column Reflux Flow	x22: Secondary Column Base Concentration	x36: Feed Column Recycle Flow
x2:Primary Column Tails Flow	x23: Flow from Input to Secondary Column	x37: Feed Column Tails Flow to Primary Column
x3:Input to Primary Column Bed 3 Flow	x24: Secondary Column Tails Flow	x38: Feed Column Calculated DP
x4:Input to Primary Column Bed 2 Flow	x25: Secondary Column Tray DP	x39: Feed Column Steam Flow
x5:Primary Column Feed Flow from Feed Column	x26: Secondary Column Head Pressure	x40: Feed Column Tails Flow
x6:Primary Column Make Flow	x27: Secondary Column Base Pressure	
x7:Primary Column Base Level	x28: Secondary Column Base Temperature	
x8:Primary Column Reflux Drum Pressure	x29: Secondary Column Tray 3 Temperature	
x9:Primary Column Condenser Reflux Drum Level	x30: Secondary Column Bed 1 Temperature	
x10:Primary Column Bed1 DP	x31: Secondary Column Bed 2 Temperature	
x11:Primary Column Bed2 DP	x32: Secondary Column Tray 2 Temperature	
x12:Primary Column Bed3 DP	x33: Secondary Column Tray 1 Temperature	
x13:Primary Column Bed4 DP	x34: Secondary Column Tails Temperature	
x14:Primary Column Base Pressure	x35: Secondary Column Tails Concentration	
x15:Primary Column Head Pressure		
x16:Primary Column Tails Temperature		
x17:Primary Column Tails Temperature 1		
x18:Primary Column Bed 4 Temperature		
x19:Primary Column Bed 3 Temperature		
x20:Primary Column Bed 2 Temperature		
x21:Primary Column Bed 1 Temperature		
Avg_Reactor_Outlet_Impurity		
Avg_Delta_Composition Primary Column		
y:Impurity		
Primary Column Reflux/Feed Ratio		
Primary Column Make/Reflux Ratio		

Relatively high dimensionality dataset, with many “x” variables that may or may not be important for understanding our “y” variable (impurity in the output stream).

Dow Dataset - Overview

Additional Details from Dow:

- The variables associated with the primary column are of higher importance than the variables associated with the other columns

- During the period this data was collected, some of the units were shut down due to maintenance. Identifying these regions and removing them will be important for building a model.

- Some of the sensors failed to report values and/or report anomalous values. Finding missing values and dealing with outliers will be important.

Dow Dataset – Reading in Data

The first thing is to load the data. Dow has made this easy by providing an organized excel spreadsheet (`ImpurityDataset_Training.xlsx`) so we don't need to write our own parser:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
dow = pd.read_excel("ImpurityDataset_Training.xlsx")
```

In Jupyter, we can view pandas dataframes by calling the dataframe directly:

```
dow
```

	Date	x1:Primary Column Reflux Flow	x2:Primary Column Tails Flow	x3:Input to Primary Column Bed 3 Flow	x4:Input to Primary Column Bed 2 Flow	x5:Primary Column Feed Flow from Feed Column	x6:Primary Column Make Flow	x7:Primary Column Base Level	x8:Primary Column Reflux Drum Pressure	x9:Primary Column Condenser Reflux Drum Level	...
0	2015-12-01 00:00:00	327.813	45.7920	2095.06	2156.01	98.5005	95.4674	54.3476	41.0121	52.2353	...
1	2015-12-01 01:00:00	322.970	46.1643	2101.00	2182.90	98.0014	94.9673	54.2247	41.0076	52.5378	...
2	2015-12-01 02:00:00	319.674	45.9927	2102.96	2151.39	98.8229	96.0785	54.6130	41.0451	52.0159	...
3	2015-12-01 03:00:00	327.223	46.0960	2101.37	2172.14	98.7733	96.1223	54.9153	41.0405	52.9477	...
4	2015-12-01 04:00:00	331.177	45.8493	2114.06	2157.77	99.3231	94.7521	54.0925	40.9934	53.0507	...
...	...	...	...	...	...	...	...	...	...	...	...

Dow Dataset – Examining the Data

We have a large block of data, but what exactly is it? We need to figure out what our headers are, how many missing values we have, and what our relevant output quantities are.

First lets just summarize the headers and datatypes:

```
# Check the columns and datatypes
for i in dow.columns:
    print("{}: {}".format(i,dow[i].dtype))
```

```
Date: datetime64[ns]
x1:Primary Column Reflux Flow: float64
x2:Primary Column Tails Flow: float64
x3:Input to Primary Column Bed 3 Flow: float64
x4:Input to Primary Column Bed 2 Flow: float64
x5:Primary Column Feed Flow from Feed Column: float64
x6:Primary Column Make Flow: float64
x7:Primary Column Base Level: float64
x8:Primary Column Reflux Drum Pressure: float64
x9:Primary Column Condenser Reflux Drum Level: float64
...
y:Impurity: float64
Primary Column Reflux/Feed Ratio: float64
Primary Column Make/Reflux Ratio: float64
```

There are lots of variables here!

Based on the description, the `xINT:` variables correspond to sensor readings and the `y:Impurity` variable is the process figure of merit that we are intent on predicting.

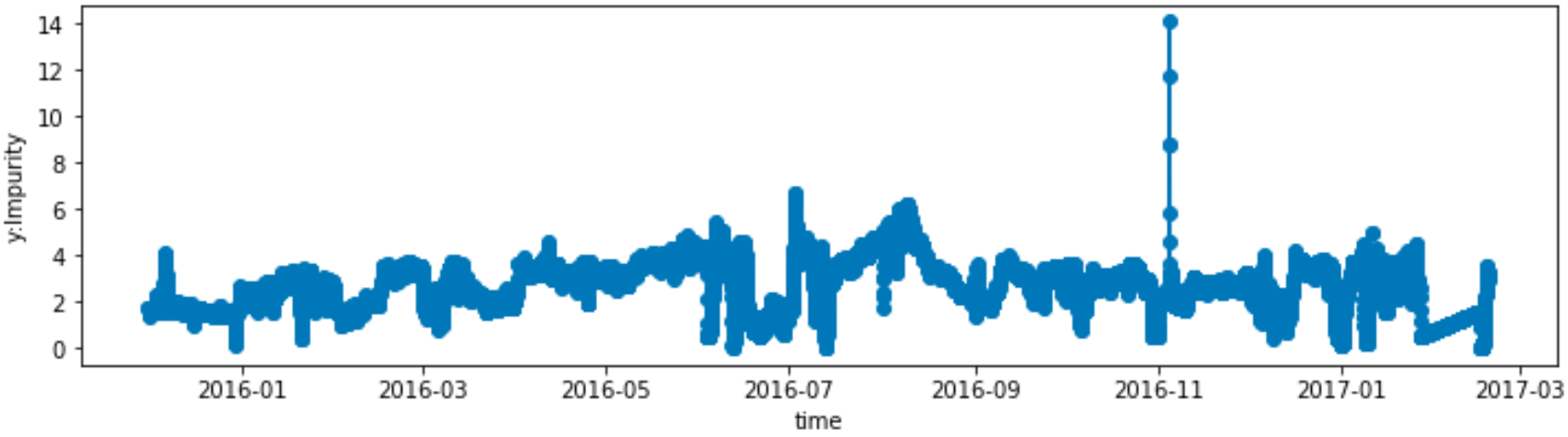
Dow Dataset – Examining the Data

One of the first things you should do with a new dataset is **look at it**.

With big datasets this is impractical to do naively, but we can still look at our most important variables.

Since `y:Impurity` vs `time` is our figure of merit, let's start here:

```
plt.figure(figsize=(12,3))
plt.plot(dow.iloc[:,0],dow.iloc[:,-3],marker='o')
plt.xlabel("time")
plt.ylabel("y:Impurity")
plt.show()
```

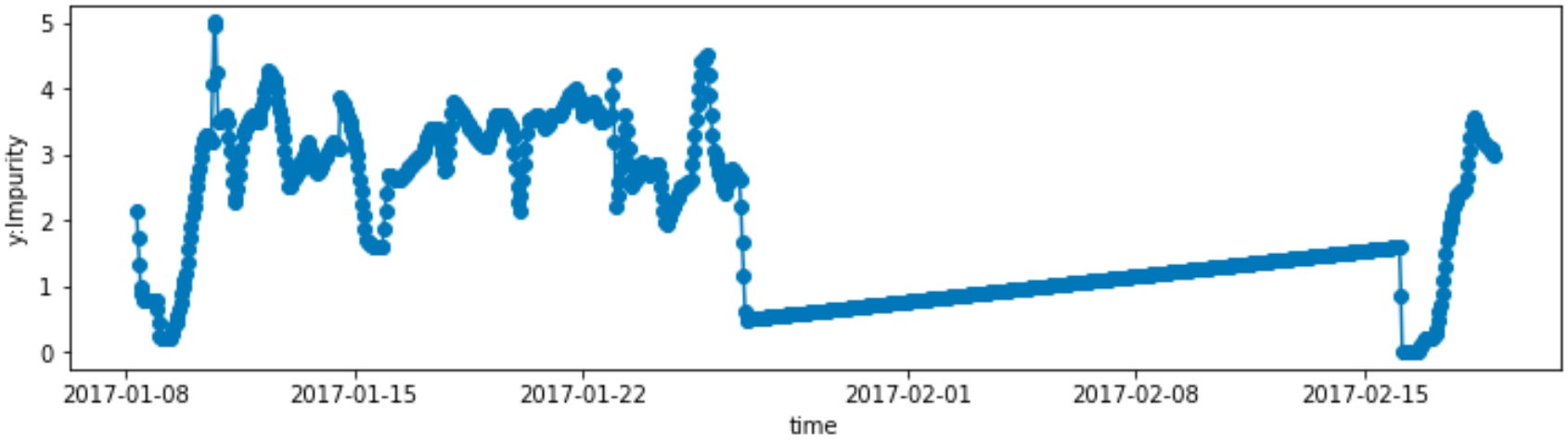


We can see potential issues with outliers. We can also see some strange behavior near the end, where the noise (which we expect) seems to go away.

Dow Dataset – Examining the Data

Let's take a closer look at the linear region near the end:

```
plt.figure(figsize=(12,3))
plt.plot(dow.iloc[-1000:,0],dow.iloc[-1000:,-3],marker='o')
plt.xlabel("time")
plt.ylabel("y:Impurity")
plt.show()
```



Something is clearly odd here. Our friends at Dow have told us that some of the units were shutdown during the period that these readings were taken, and identifying these periods is one of the challenges we'll discuss with this dataset in the next lecture.

Dow Dataset – Identifying Missing Values

A second generic consideration with any dataset is **identifying missing values.**

Let's write a pandas operation to identify how many missing values we have in every column:

```
# Loop over columns and use .isna() and .sum() methods to identify NaN and s
um their quantity, respectively
N = len(dow)
for i in dow.columns:
    N_missing = dow[i].isna().sum()
    print("{}: {} ({:4.2f}%)".format(i,N_missing,N_missing/N*100.0))
```

```
Date: 0 (0.00%)
x1:Primary Column Reflux Flow: 30 (0.28%)
x2:Primary Column Tails Flow: 22 (0.21%)
x3:Input to Primary Column Bed 3 Flow: 21 (0.20%)
x4:Input to Primary Column Bed 2 Flow: 21 (0.20%)
x5:Primary Column Feed Flow from Feed Column: 21 (0.20%)
x6:Primary Column Make Flow: 21 (0.20%)
x7:Primary Column Base Level: 21 (0.20%)
x8:Primary Column Reflux Drum Pressure: 21 (0.20%)
...
y:Impurity: 0 (0.00%)
Primary Column Reflux/Feed Ratio: 30 (0.28%)
Primary Column Make/Reflux Ratio: 30 (0.28%)
```

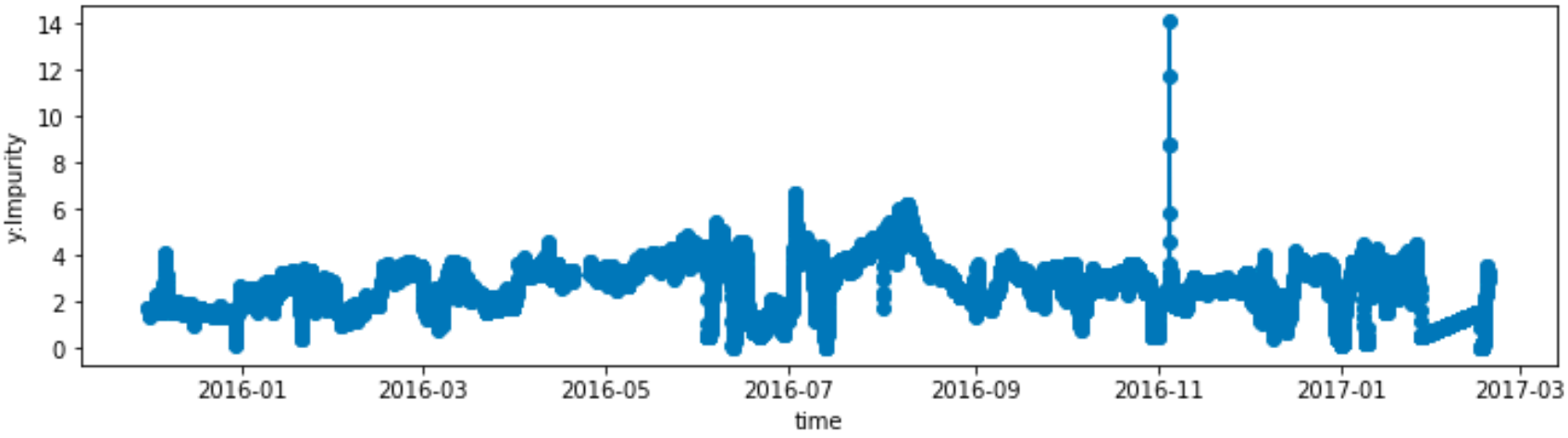
We have missing values, but in all cases it is only a fraction of a percent of the total data.

Dow Dataset – Identifying Missing Values

Since the missing values are such a small fraction, let's remove all rows with missing data, check our work, and replot the result:

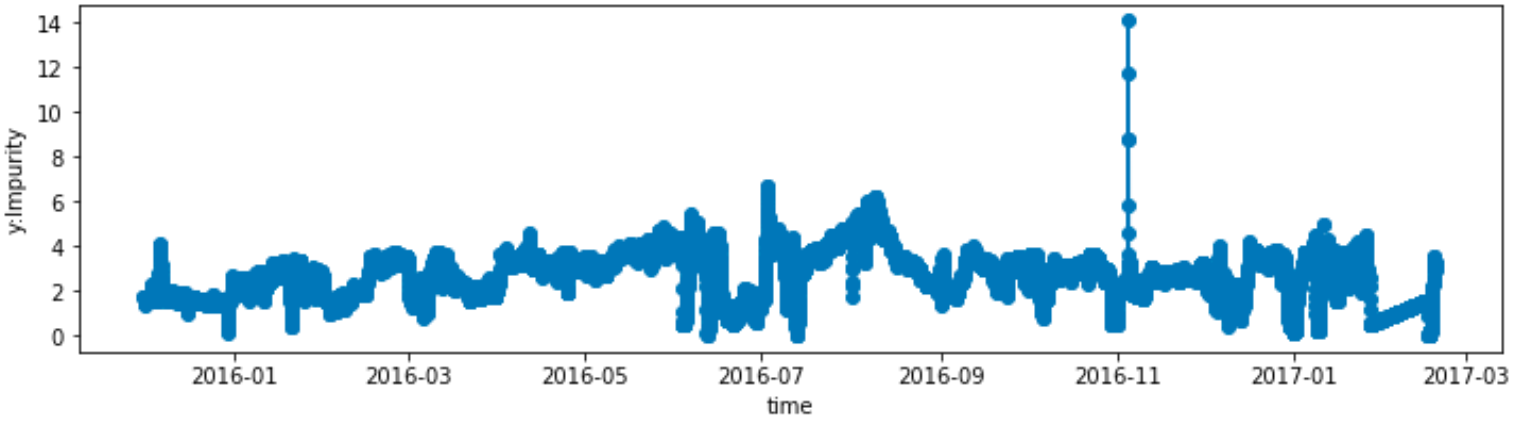
```
# Drop rows with any missing data
dow.dropna(how='any',inplace=True)

# Check that we successfully removed the missing values
print("number of missing values: {}".format(dow.isna().sum().sum()))

# Check the impurity plot
plt.figure(figsize=(12,3))
plt.plot(dow.iloc[:,0],dow.iloc[:,-3],marker='o')
plt.xlabel("time")
plt.ylabel("y:Impurity")
plt.show()
```

```
number of missing values: 0
```

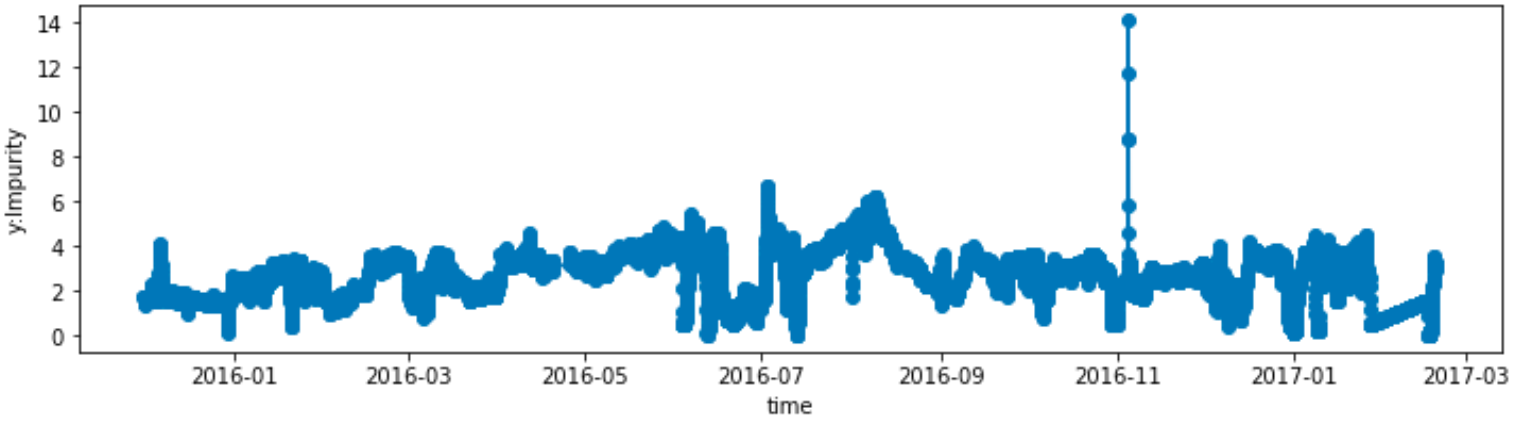


Dow Dataset – Identifying Missing Values

Before removing missing values:



After removing missing values:



Comparing the resulting impurity plot with the original, we can see that it is largely unchanged.

Even with the removed values we still have plenty of data for training an impurity model (this might not always be the case, and sometimes missing values are themselves a useful predictor for modeling).

Dow Dataset – Correlations

We have a lot of sensor readings related to the unit operations (40 of them). Absent other knowledge, we don't know if (1) these are all important, (2) if none of them are important, or (3) if we actually need even more variables to construct a useful model.

When you have a lot of variables, a simple and informative thing to calculate is the linear correlation coefficient (r_{xy}):

$$r_{xy} = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_i (x_i - \mu_x)^2}\sqrt{\sum_i (y_i - \mu_y)^2}}$$

Where x_i and y_i are the values of properties X and Y at a given time *i*, and μ_x and μ_y are the average values of X and Y over the dataset.

This quantity is bounded between -1 and 1, and tells us if the two variables X and Y tend to move in the opposite direction or same direction, respectively.

The denominator normalization accounts for the fact that x and y may have different units and variability.

Dow Dataset – Correlations

We care about correlation, because it is simplest opportunity for finding important variables for predicting our property of interest.

In this course, we will typically focus on problems where linear regression doesn't work, but if you happen to have good linear predictor variables, that is wonderful and you should identify this right away.

Pandas has a built-in method for calculating the correlation between columns:

```
corr = dow.corr()
corr
```

	x1:Primary Column Reflux Flow	x2:Primary Column Tails Flow	x3:Input to Primary Column Bed 3 Flow	x4:Input to Primary Column Bed 2 Flow	x5:Primary Column Feed Flow from Feed Column	x6:Primary Column Make Flow	x7:Primary Column Base Level	x8:Primary Column Reflux Drum Pressure	x9:Primary Column Condenser Reflux Drum Level	x10:Primary Column Bed1 DP
x1:Primary Column Reflux Flow	1.000000	0.840909	0.638107	0.790147	0.915221	0.869800	0.733663	0.565074	0.788199	0.797622
x2:Primary Column Tails Flow	0.840909	1.000000	0.755112	0.879539	0.739763	0.698189	0.845652	0.622606	0.884212	0.454977
x3:Input to Primary Column Bed 3 Flow	0.638107	0.755112	1.000000	0.703178	0.621706	0.611063	0.657509	0.319365	0.687136	0.441262
x4:Input to Primary Column Bed 2 Flow	0.790147	0.879539	0.703178	1.000000	0.732774	0.702620	0.726438	0.560686	0.780996	0.525478
x5:Primary Column Feed Flow from Feed Column	0.915221	0.739763	0.621706	0.732774	1.000000	0.949683	0.662324	0.461143	0.722785	0.804096
x6:Primary Column Make Flow	0.869800	0.698189	0.611063	0.702620	0.949683	1.000000	0.616827	0.408036	0.665838	0.758701

Dow Dataset – Correlations

This is a lot of data to digest at one time. It is much better to visualize, using a so-called correlation matrix (a N_var x N_var plot with z-colored by correlation.)

You can do this in Pandas using the built-in print functionality and setting the background color:

```
corr.style.background_gradient(cmap='PRGn').set_precision(2)
```

	x1:Primary Column Reflux Flow	x2:Primary Column Tails Flow	x3:Input to Primary Column Bed 3 Flow	x4:Input to Primary Column Bed 2 Flow	x5:Primary Column Feed Flow from Feed Column	x6:Primary Column Make Flow	x7:Primary Column Base Level	x8:Primary Column Reflux Drum Pressure	x9:Primary Column Condenser Reflux Drum Level	x10:Primary Column Bed1 DP	x11:Primary Column Bed2 DP	x12:Primary Column Bed3 DP	x13:Primary Column Bed4 DP	x14:Primary Column Base Pressure	x15:Primary Column Head Pressure
x1:Primary Column Reflux Flow	1.00	0.84	0.64	0.79	0.92	0.87	0.73	0.57	0.79	0.80	0.85	0.83	0.77	0.67	0.60
x2:Primary Column Tails Flow	0.84	1.00	0.76	0.88	0.74	0.70	0.85	0.62	0.88	0.45	0.48	0.46	0.49	0.67	0.64
x3:Input to Primary Column Bed 3 Flow	0.64	0.76	1.00	0.70	0.62	0.61	0.66	0.32	0.69	0.44	0.38	0.31	0.41	0.37	0.34
x4:Input to Primary Column Bed 2 Flow	0.79	0.88	0.70	1.00	0.73	0.70	0.73	0.56	0.78	0.53	0.50	0.46	0.55	0.61	0.58
x5:Primary Column Feed Flow from Feed Column	0.92	0.74	0.62	0.73	1.00	0.95	0.66	0.46	0.72	0.80	0.83	0.79	0.80	0.57	0.50
x6:Primary Column Make Flow	0.87	0.70	0.61	0.70	0.95	1.00	0.62	0.41	0.67	0.76	0.80	0.78	0.80	0.52	0.44
x7:Primary Column Base Level	0.73	0.85	0.66	0.73	0.66	0.62	1.00	0.65	0.89	0.44	0.46	0.42	0.44	0.69	0.67
x8:Primary Column Reflux Drum Pressure	0.57	0.62	0.32	0.56	0.46	0.41	0.65	1.00	0.64	0.34	0.35	0.34	0.30	0.99	1.00
x9:Primary Column Condenser Reflux Drum Level	0.79	0.88	0.69	0.78	0.72	0.67	0.89	0.64	1.00	0.47	0.49	0.46	0.48	0.68	0.66
x10:Primary Column Bed1 DP	0.80	0.45	0.44	0.53	0.80	0.76	0.44	0.34	0.47	1.00	0.92	0.83	0.79	0.47	0.37
x11:Primary Column Bed2 DP	0.85	0.48	0.38	0.50	0.83	0.80	0.46	0.35	0.49	0.92	1.00	0.97	0.83	0.49	0.39
x12:Primary Column Bed3 DP	0.83	0.46	0.31	0.46	0.79	0.78	0.42	0.34	0.46	0.83	0.97	1.00	0.78	0.48	0.38
x13:Primary Column Bed4 DP	0.77	0.49	0.41	0.55	0.80	0.80	0.44	0.30	0.48	0.79	0.83	0.78	1.00	0.42	0.33
x14:Primary Column Base Pressure	0.67	0.67	0.37	0.61	0.57	0.52	0.69	0.99	0.68	0.47	0.49	0.48	0.42	1.00	0.99
x15:Primary Column Head Pressure	0.60	0.64	0.34	0.58	0.50	0.44	0.67	1.00	0.66	0.37	0.39	0.38	0.33	0.99	1.00
x16:Primary Column Tails Temperature	0.19	0.18	0.19	0.24	0.22	0.21	0.17	0.09	0.16	0.18	0.10	0.06	0.18	0.11	0.10

Dow Dataset – Correlations

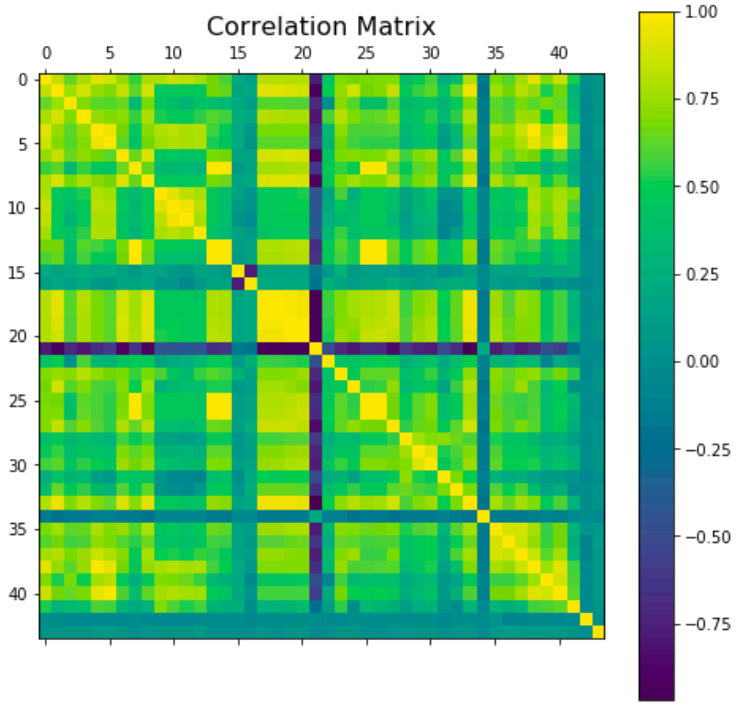
More succinctly, you can use matplotlib to make an actual plot of the matrix using the `plt.matshow()` function:

```
plt.figure(figsize=(8,8))
plt.matshow(corr,fignum=0) # makes a new fig by default, 0 adds to current
plt.colorbar() # add colorbar
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```

Two things are universal about these plots:

First, the matrix is symmetric (i.e., $A=A^{\mathsf{T}}$), because $r_{xy}=r_{yx}$. So only half of the matrix holds unique information.

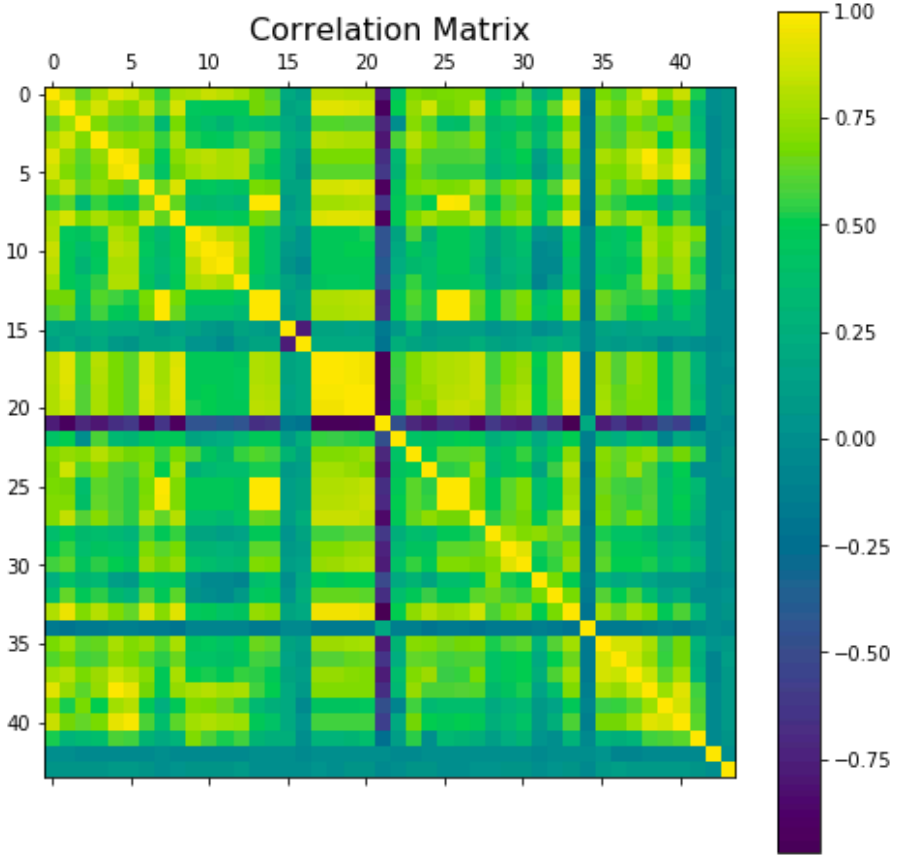
Second, there is a bright strip of yellow along the diagonal, corresponding to a correlation of 1. This is because $r_{xx}=1$ by definition.



Dow Dataset – Correlations

We also see a lot of bright yellow patches and dark purple patches along the off diagonal. These correspond to situations where the variables are strongly correlated, or anti-correlated.

For the purposes of building a model, this means that these variables hold redundant information. If you know what one is, then you can predict what the other is (depending on the level of correlation), this might form a basis for removing variables from a model.



Dow Dataset – Correlations

Since we are interested in impurity, let's take a look at the five most linearly correlated features with `y:Impurity`:

```
# Check the five most linearly correlated variables with impurity
print("y-correlations:\n")
lab = corr['y:Impurity'].abs().nlargest(6).index
print(corr['y:Impurity'][lab[1:]])
```

```
y-correlations:

x10:Primary Column Bed1 DP                          0.700183
Avg_Delta_Composition Primary Column                0.618379
Avg_Reactor_Outlet_Impurity                         0.596340
x5:Primary Column Feed Flow from Feed Column        0.592390
x11:Primary Column Bed2 DP                          0.589071
Name: y:Impurity, dtype: float64
```

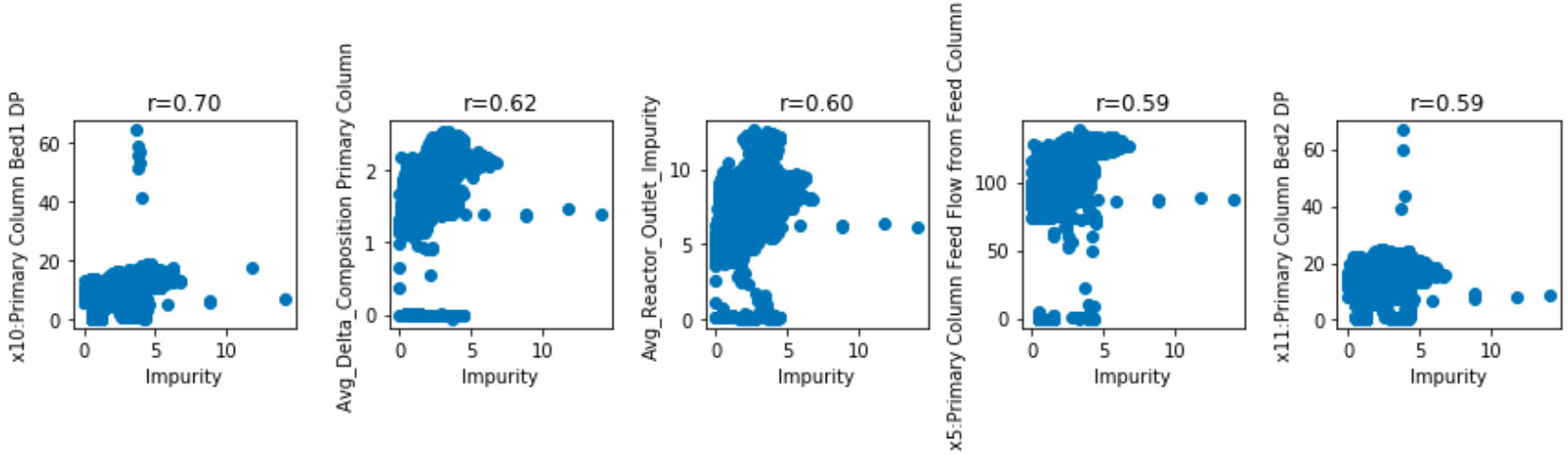
Here I have used the `abs` method so that we find the largest magnitude correlations, not just positive correlations.

I've also returned the top 6 correlated variables, and thrown out the first, because `y:impurity` is trivially the most correlated variable with itself.

Dow Dataset – Correlations

A second plot that is useful for visualizing correlations is a parity plot, where we plot two variables against one another and correlations occur along the diagonal:

```
plt.figure(figsize=(12,3))
for count_i,i in enumerate(lab[1:]):
    ax = plt.subplot(1,5,count_i+1)
    plt.scatter(dow["y:Impurity"],dow[i])
    plt.title("r={:4.2f}".format(corr.loc['y:Impurity'][i]))
    plt.xlabel("Impurity")
    plt.ylabel(i)
plt.tight_layout()
plt.show()
```



None of these are particularly correlated with impurity (in keeping with this being a difficult modeling problem).

Dow Dataset – Correlations

We also saw in the correlation matrix that several of the variables were correlated with one another. Let's find the five most correlated pairs of variables:

```
# Find linearly correlated features
corr_array = corr.abs().to_numpy()[:-5,:-5]
row,col = np.where((corr_array > 0.95) & (corr_array < 1))
labels = [ (corr.index[row[_]],corr.columns[col[_]]) for _ in range(len(row))\
            if row[_] > col[_] ] # find labels/cols
```

Here we've only retained the `xint:` type of variables and found the those with correlation magnitudes greater than 0.95.

With a `for` loop we can automate the plotting of the parity plots:

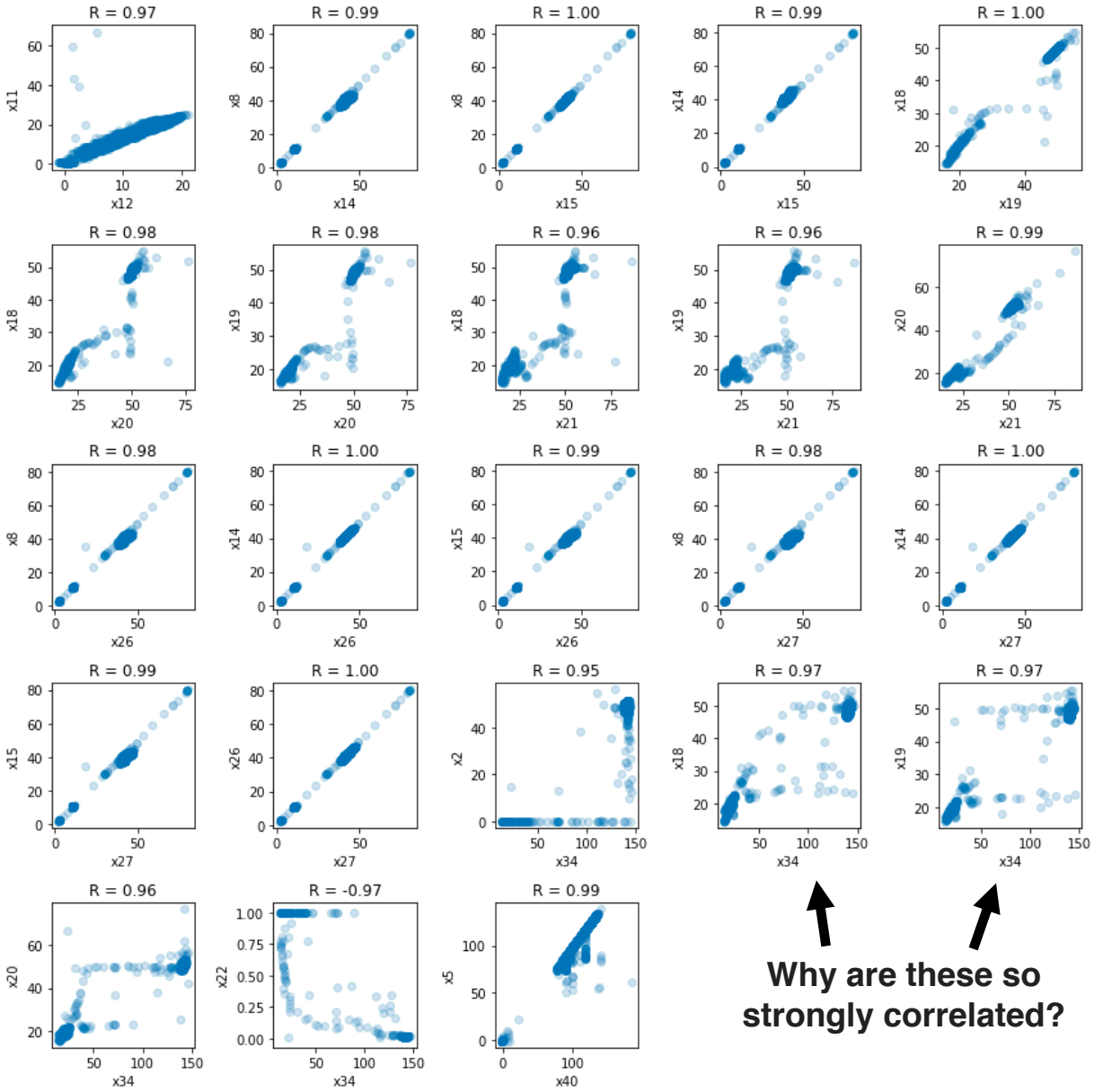
```
### Correlation Grid of features
plt.figure(figsize=(12,12))
inc = -1
for i in range(len(labels)):
  ax = plt.subplot(5,5,i+1)
  ax.scatter(dow.loc[:,labels[i][0]],dow.loc[:,labels[i][1]],alpha=0.2)
  ax.set_title("R = {:<3.2f}".format(corr.loc[labels[i][0],labels[i][1]]))
  ax.set_xlabel(labels[i][0].split(":")[0])
  ax.set_ylabel(labels[i][1].split(":")[0])
plt.tight_layout()
plt.show()
```

Dow Dataset – Correlations

We've just uncovered a lot of interesting relationships:

- First, there are several variables the are strongly correlated and likely redundant (e.g., `x8`, `x14`, `x15`, `x26`, and `x27` which all correspond to pressure in different unit ops).

- Second, we also see strong linear correlations among variables whose parity plots contain obvious outliers. For example, in the `x11` vs `x12` plot there is a strong correlation, but there are a small number of outliers that throw off the axes.



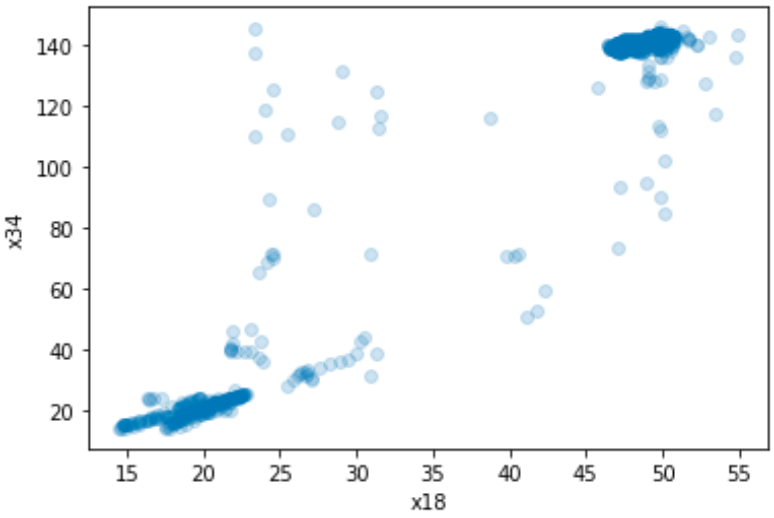
Dow Dataset – Correlations

We saw several cases where there is a strong linear correlation, but localized data. For example, `x34` vs `x18` has a strong correlation (r=0.97) but this linear correlation is driven by essentially two discrete values:

```
# Save the original header labels and rename working labels
h_0 = dow.columns
dow.columns = [ _.split(':')[0] for _ in dow.columns ]

# Make the Plot
plt.scatter(dow["x18"],dow["x34"],alpha=0.2)
plt.xlabel("x18")
plt.ylabel("x34")
plt.show()
```

These variables correspond to the temperature in the beds of the primary and secondary column. There is a large density of points at ~(20,20) and ~(50,140) respectively, with hardly any values in between.

We know that some of the units were off during the time we have data for. We might guess that the "off-temperature" of both is ambient ~20C (and strongly correlated), and the "on-temperature" is 50/140C, respectively.



The strong correlation comes from the fact that they were off or on at the same times, *not that they varied linearly during operation.*