MnV Walkthrough

Josue Aleman, jaleman@ucdavis.edu

2018-08-13

Contents

Step 1

Login on the Airtable and click on the **Opportunities Log** tab. Select one of the entries under **Traditional Energy Model M&V** as seen in *Figure* 1.

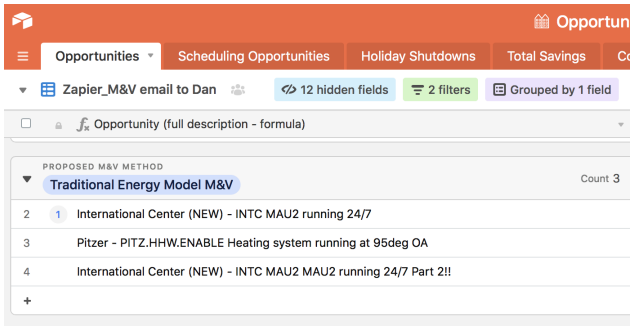


Figure 1: Traditional Energy Model MnV

Step 2

Scroll to the right of the columns until you find *Unique ID* as seen in *Figure* 2.

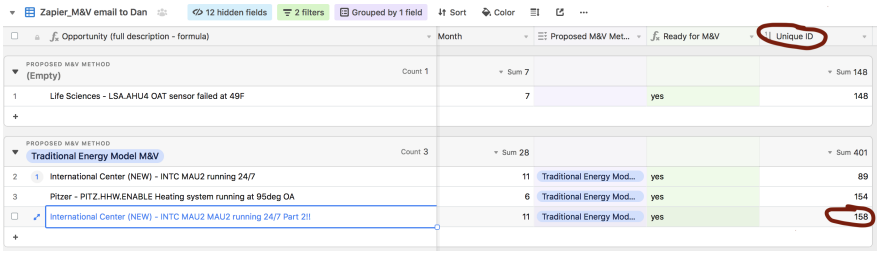


Figure 2: Unique ID

Step 3

Go to the Jupyter notebook following the path **/ UCD_ECO_coding / MnV-Tool/ ACE MnV** and make make a duplicate of **DC MnV v14-New.ipynb** notebook. Rename it as **{Building Name}_{Unique ID}_{Commodity}** as seen in *Figure* 3.

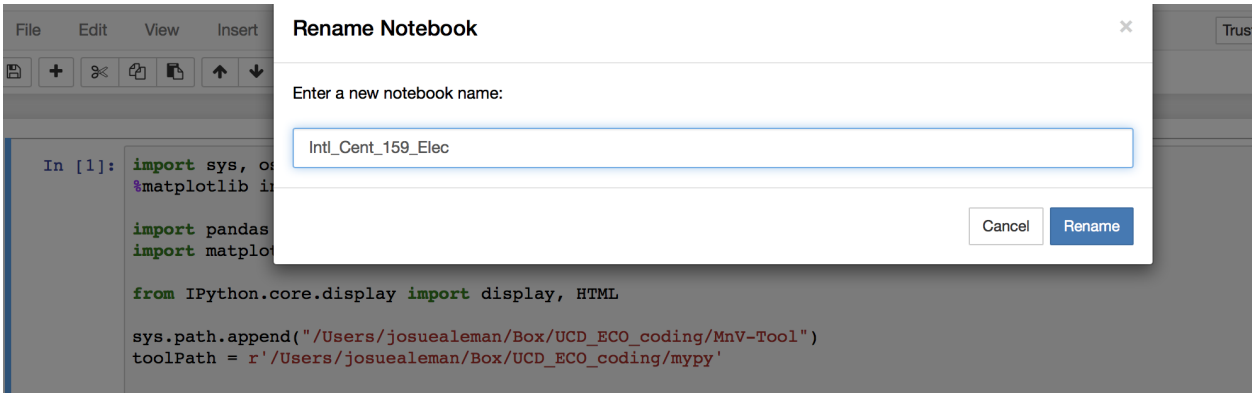


Figure 3: Notebook for Opportunity

Step 4

Ensure that the path to mypy folder and the MnV tool are correct (As seen in *Figure* 4) before running the cell. Once boht paths have been added hit **Cmd** + [**Return**] for Mac or **Ctrl** + [**Enter**] for Windows to run the cell.

Figure 4: Notebook for Opportunity

Step 5

Go to the **Pi Data Loading** cell and ensure you search for the correct building and keep *demand_kbtu* to search for the right tag. Wild cards (*) are valid to help search for the tags. Once you have that field correct run the cell (see step 4 on how to run cell). The tags available will be printed out under the cell as seen in *Figure* 5.

PI Data Loading

Bldg

In [2]:
```
from PI_client import pi_client
pi = pi_client()

tags = pi.search_by_point('*international*demand_kbtu')[0]
#tags += pi.search_by_point('*shields*kbtu*')[0]

print(tags)
```

```
[u'International_Center_Electricity_Demand_kBtu', u'International_Center_Gas_Demand_kBtu']
```

Figure 5: Notebook for Opportunity

Step 6

Go back to the **Opportunities Log** and look for the column *Date issue resolved* and save that date as it will be useful in determining how much data to extract from PI as seen in *Figure* 6.

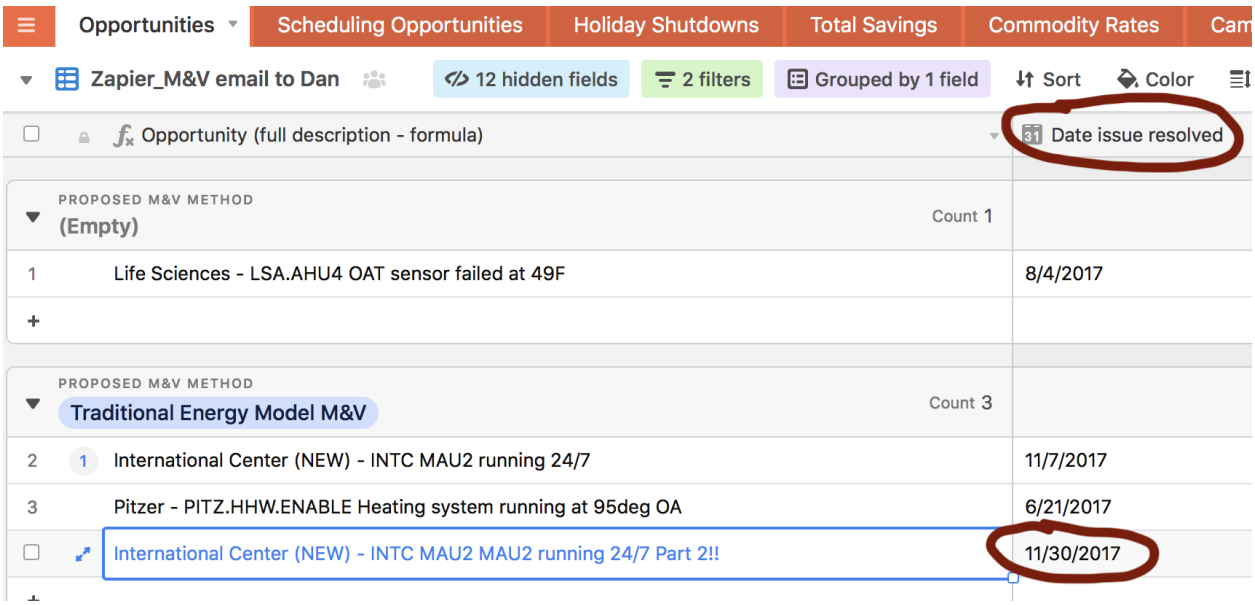


Figure 6: Notebook for Opportunity

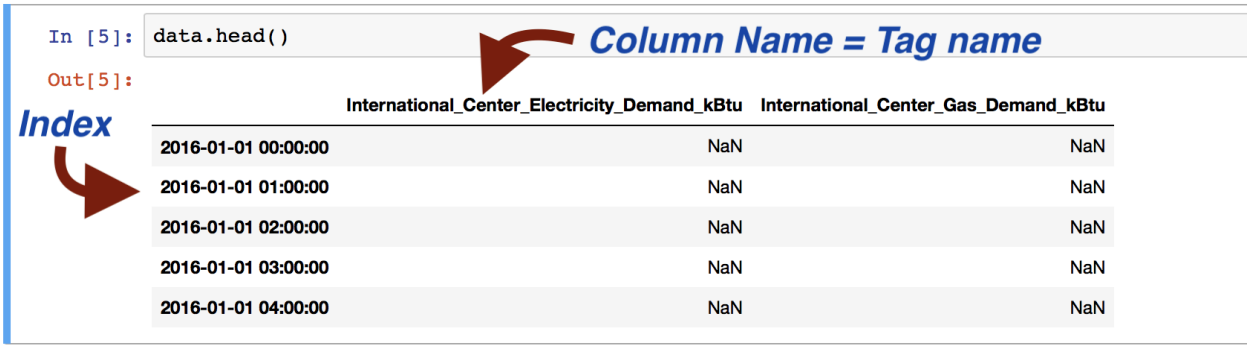
Step 7

In the notebook, go to the cell called **Pull tags** and select a start date that is at least one year before the *Date issue resolved*. To keep the end date to today's date use the "*". The parameter *interval* should be "1 hour" and the parameter *calculation* should be "calculated" for these models. Run the cell and ensure that you have the right amount of Repsonses as seen in *Figure* 7.

Figure 7: Pulling Data From PI

Step 8

To get a glimpse of the data run the **data.head()** cell, this will display the first five rows of the data. As seen in *Figure* 8 each column is a tag and each row index is the timestamp.



	International_Center_Electricity_Demand_kBtu	International_Center_Gas_Demand_kBtu
2016-01-01 00:00:00	NaN	NaN
2016-01-01 01:00:00	NaN	NaN
2016-01-01 02:00:00	NaN	NaN
2016-01-01 03:00:00	NaN	NaN
2016-01-01 04:00:00	NaN	NaN

Figure 8: Pulling Data From PI

Step 9

Go to the **Data Section** cell and ensure the parameters are correct. Under *column* type the column number you wish to run (remember that is is 0 based meaning that the column numbers begin with 0). *IQRmult* should typically be in the range of **3 - 4** depending on how much we may need to exclude and consider outliers. *IQR*, *resampleRate*, *OATsource*, *OATname*, *sliceType*, and *midDate* should remain as the values alrready listed. Lastly, *dateRanges* should be [**'Date Start Pre', 'Date End Pre', 'Date Start Post', 'Date End Post'**] all in the format of **YYYY-MM-DD** shown in *Figure* 9. Run the cell to display the various plots shown in *Figure* 10 and *Figure* 11. For a deeper understanding on the various plots please consult the data science team.

Data Section

In [7]:
```
dataParams = {'column': 0,
              'IQRmult' : 3,
              'IQR' : 'y',
              'resampleRate' : 'D', #'D' for daily 'H' for hourly
              'OATsource' : 'file', #'self' or 'file'
              'OATname' : 'OAT', #Name of OAT column if OATsource is 'self'} #only needed with sliceType : 'ranges'
              'sliceType' : 'ranges', #half, middate, ranges
              'midDate' : '2017-01-01', #only needed with sliceType : 'middate'
              'dateRanges' : ['2016-11-01','2017-11-30','2017-11-30','2018-08-08'], #only needed with sliceType : 'rang
             }

dk = mnv.data_keeper(data, dataParams)
dk.default_clean()


# Plots
dk._outlier_plot()
dk._resampled_plot(yrange=(-100,dk.modifiedData[dk.com].max()*1.1))
dk._pre_post_plot()
```

Figure 9: Data Section Cell

```
Q(75%): 281.96 Q(25%): 83.99
IQR value is 197.97
('IQRupper', 875.88, 'IQRlower', -509.93)
Ceiling adjusted by IQR - Now 875.88
OAT being loaded from master file
```

```
/anaconda2/lib/python2.7/site-packages/seaborn/categorical.py:462: FutureWarning: remove_na is deprecated and is a pr
ivate function. Do not use.
  box_data = remove_na(group_data)
```

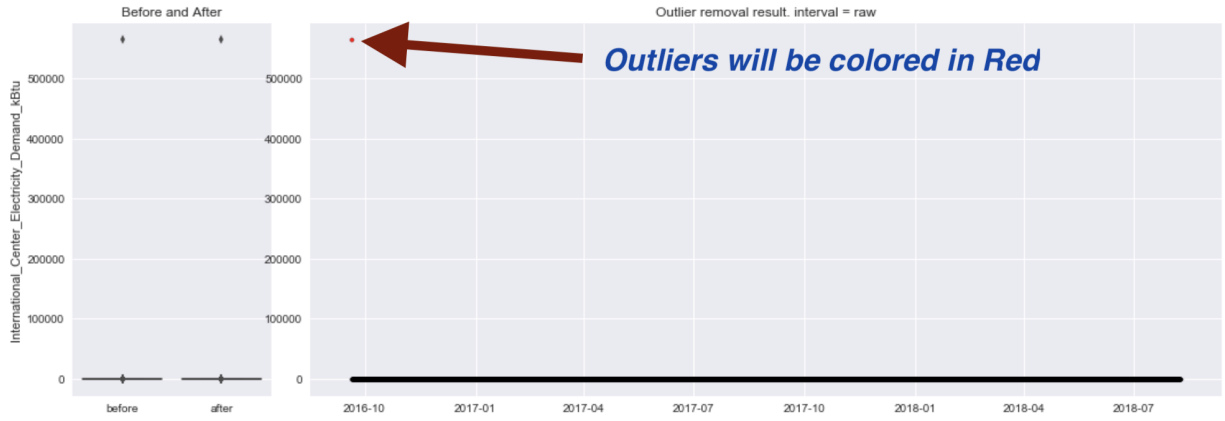


Figure 10: Results of Evaluation and Outlier Plot

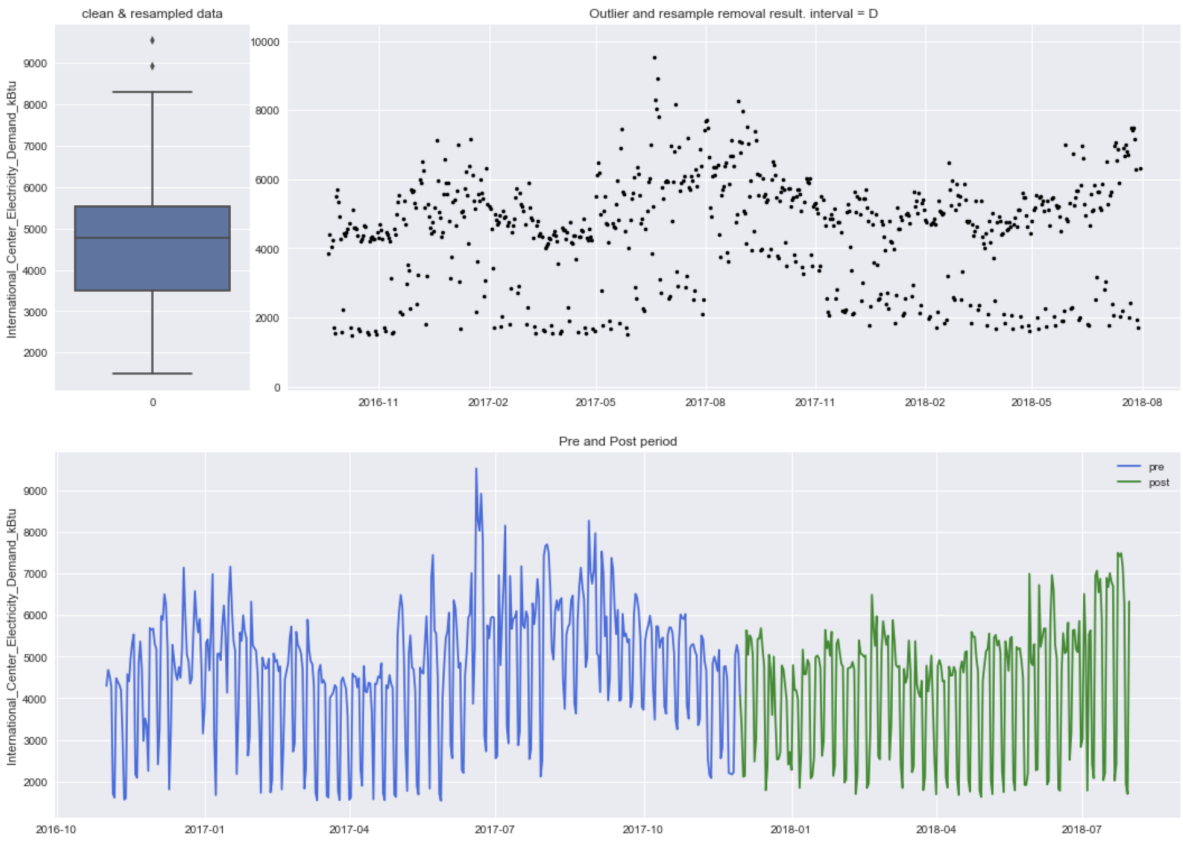


Figure 11: Outlier Resample and Pre and Post Period

Step 10

Once you have ensure that the data and plots look correct proceed to the **Many Linear Models** cell. The only parameter that will need to change is the *commodityRate*. To uncomment the right commodity rate, delete the "#" that is before the desired rate. Remember to only have one commodityRate uncommented, making sure the other rates have the "#" before it as shown in *Figure* 12. You may also select how many results you want to see by changing the *allmod.statsPool[0:5]* to a different range. To view more plots of the top results you may also change the *allmod.plot_pool(1)* to the number of plots you want to see. Once you have the right rate, run the cell which will output the summary of the models as seen in *Figure* 13. As a rule of thumb, a good model will have an **AR2** (Adjusted R^2) value that is **greater** than 0.75 and a **cvrmse** (Coefficient of Variation Root Mean Squared Error) value **less** than 0.3. If the top choice does not meet these requirements please consult the data science team for evaluation.

Many Linear Models

In [8]:

```
modelParams = {'params': ['CDH','HDH','month','hour','weekday'],
               'testTrainSplit': 'random',
               'randomState': None,
               'testSize': 0.2,
               #'commodityRate': 0.0157, #CHW (kBtu)
                'commodityRate': 0.0190504, #Elec(kBtu)
                #'commodityRate': 0.0059172, #Steam(kBtu)
               #'commodityRate':0.0070000, #Gas (kBtu)
               'paramPermuteList': ['', 'C(weekday)', 'C(month)']}

allmod = mnv.many_ols(dk.pre, dk.post, modelParams)

allmod.run_all_linear()
print(allmod.statsPool[0:5])
allmod.plot_pool(1)
modelParams['params'] = allmod.statsPool.iloc[0]['params']
```

Only ONE rate uncommented!

Figure 12: Many Linear Models

	AIC	AR2	R2	cvrmse	postDiff	params	summary
16	4920.5	0.871074	0.876804	0.119642	56867.9	CDH2 + HDH + C(weekday) + C(month)	OLS R...
30	4924.61	0.867643	0.873526	0.120814	57687.3	CDH + HDH2 + C(weekday) + C(month)	OLS R...
22	4933.45	0.860311	0.86652	0.123377	59449.1	CDH + HDH + C(weekday) + C(month)	OLS R...
5	4958.28	0.847552	0.854328	0.129279	50453.6	CDH2 + HDH2 + C(weekday) + C(month)	OLS R...
4	5025.79	0.812112	0.819866	0.143472	61144.7	CDH2 + C(weekday) + C(month)	OLS R...

Top Result

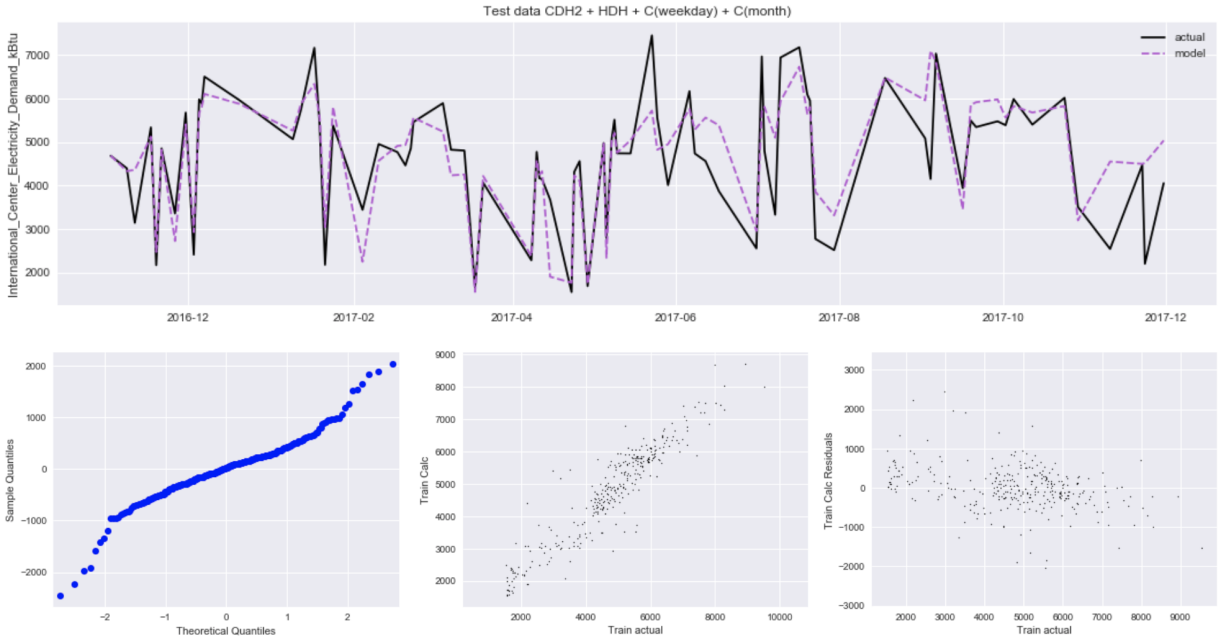


Figure 13: Many Linear Models Pt 2

Step 11

Run the **Single Linear Model** to run a *KFold* as a mean of ensuring that the predictions we are making in *Step 10* are accurate. Once again consult the data science team for further explanation on the plots and the data that is printed.

Step 12

Go to the **Savings** cell and run it. It will display the amount of savings that were made due to the fix in the opportunity as seen in *Figure* 14. The first plot displays the actual usage during the post period (black solid line) and the predicted usage if the fix had not been made (dotted purple line). The second plot shows the positive savings (black dots) and the potential loss (red dots). Lastly, the third plot shows the cummulative savings up to today's date.

Savings

```
In [10]: print(round(mod.postCumsum, 0))
         print("Savings = $" +str(round(mod.postCumsum * mod.params.commodityRate,1)))
         mod.savings_plot(yaxis='dollars')
```

```
59660.0
Savings = $1136.5
```

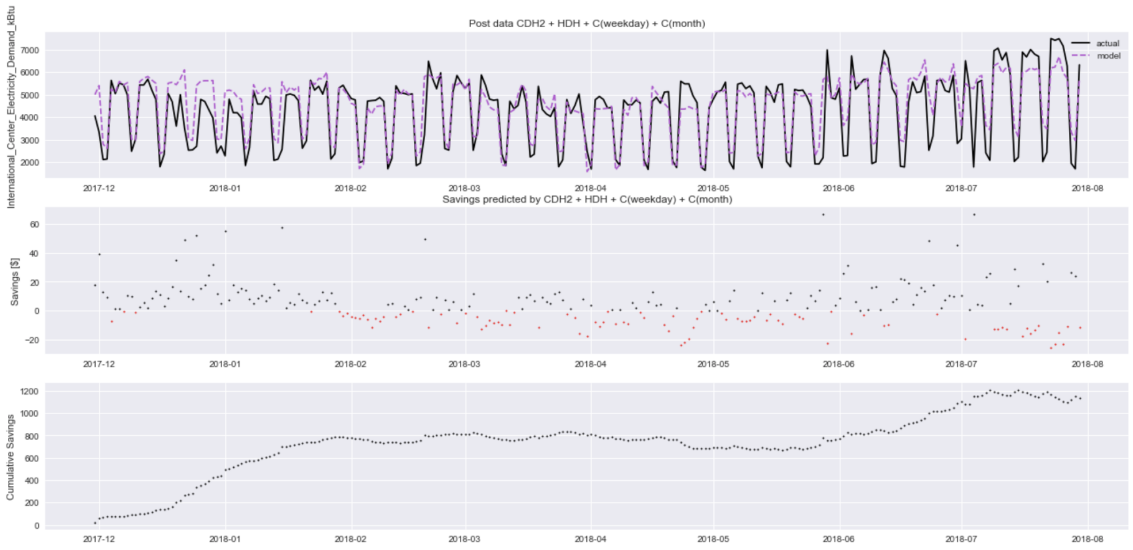


Figure 14: Savings Plots

Step 13

Run the cell which contains the "Estimated Annual Savings" to view how would be saved durign the fiscal year due to the fix as seen in *Figure* 15.

```
In [11]: print("Estimated Annual Savings: %0.f" % round(mod.postCumsum/len(dk.post)*365,0))
```

```
Estimated Annual Savings: 89612
```

Figure 15: Estimated Annual Savings

Step 14

Go back to the **Opportunites Log** and copy the field entry name as displayed in *Figure* 16. Paste the copied entry into the *Total Savings* tab under **Opportunity & Commodity** column. Fill out the **Commodity & Unit**, **Commodity & Savings** and **Opportunity Item** columns as seen in *Figure* 17. Continue filling out the **Adjusted R-Squared**, **CVRMSE** columns (From the top AR2 result in the Many Linear Models shown in *Figure* 13). The **Start Date** column will contain the date of the Start Post Period and the **Date of Modeled Savings** column will contain the End Post Period date. The **Actual FY 17-18 Savings** column should have the number from the **Savings** cell in *Figure* 14. Lastly, the **Projected FY 17-18 Savings..** column should have the number from the **Estimated Annual Savings** cell shown in *Figure* 15.

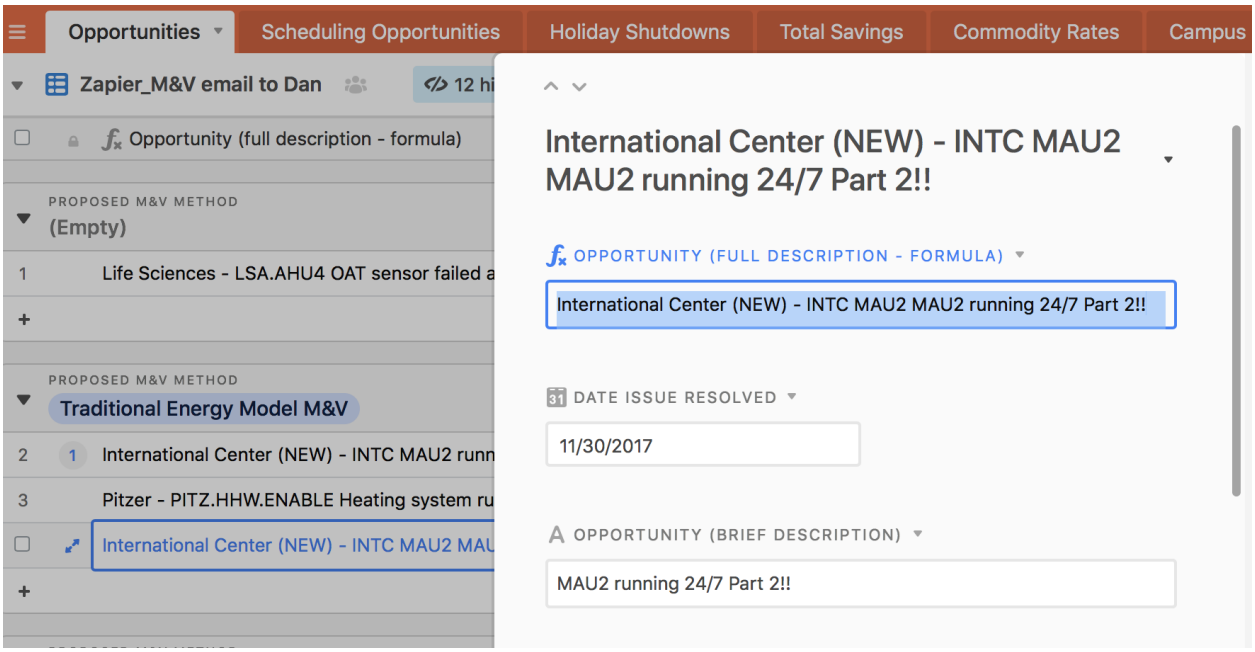


Figure 16: Opportunities Log

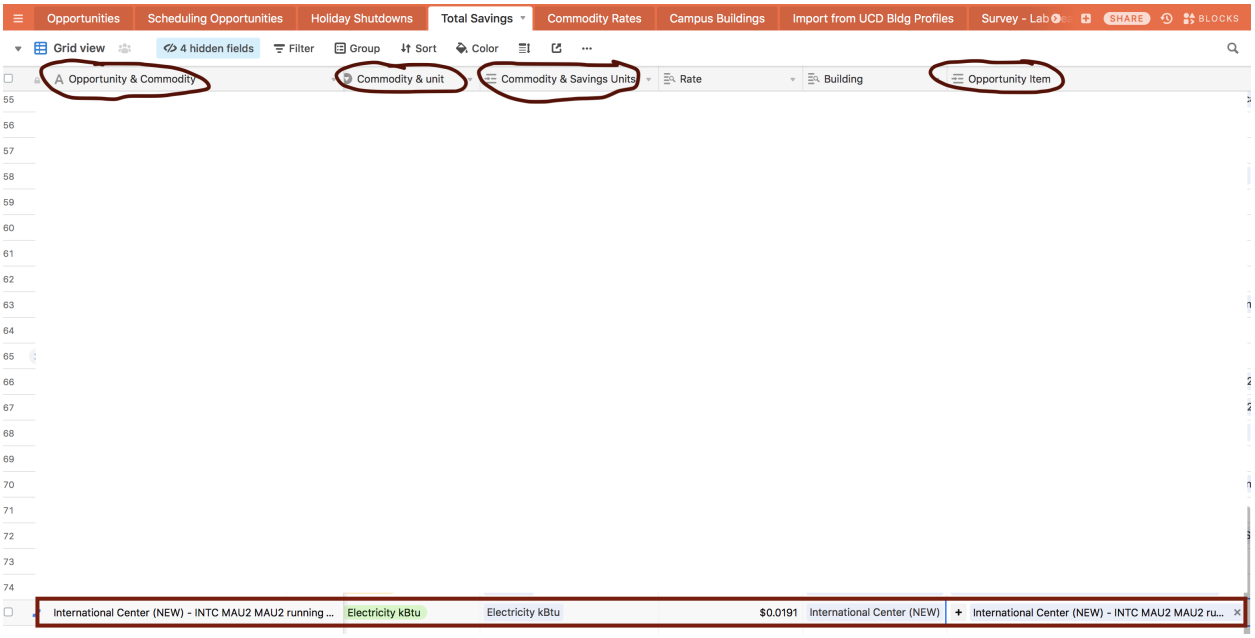


Figure 17: Total Savings Tab

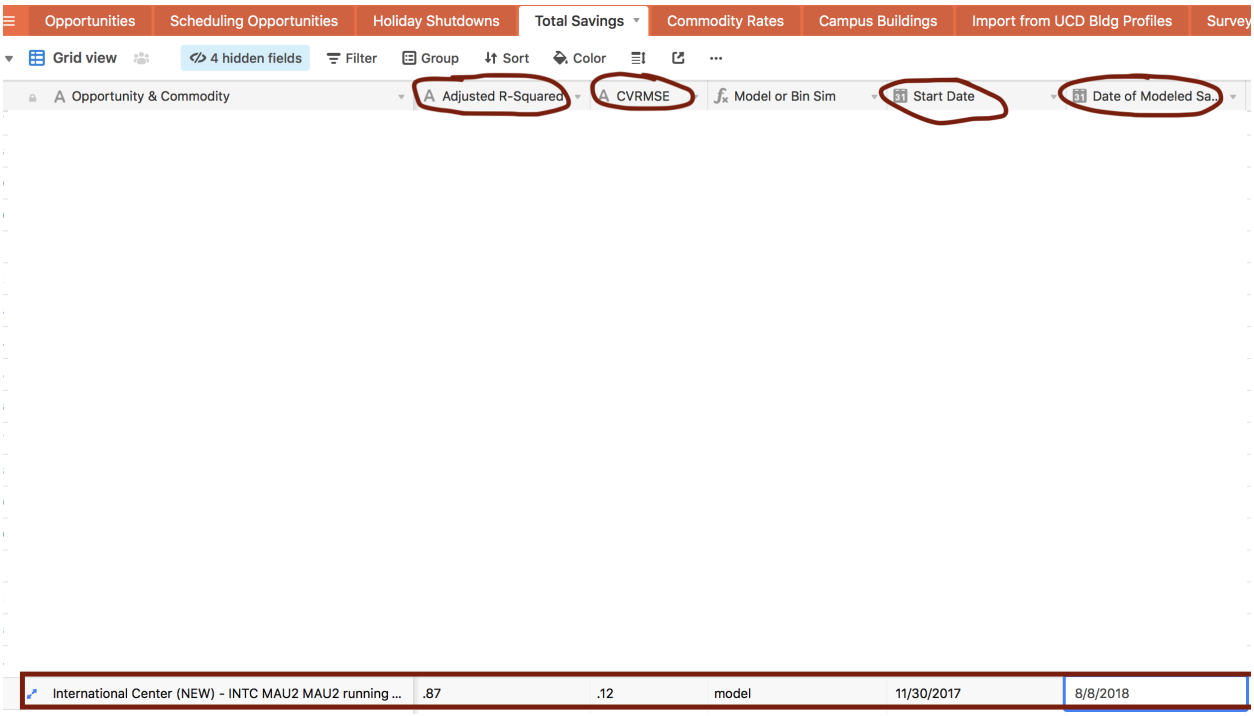


Figure 18: AR2, CVRMSE, Start/End Post Period

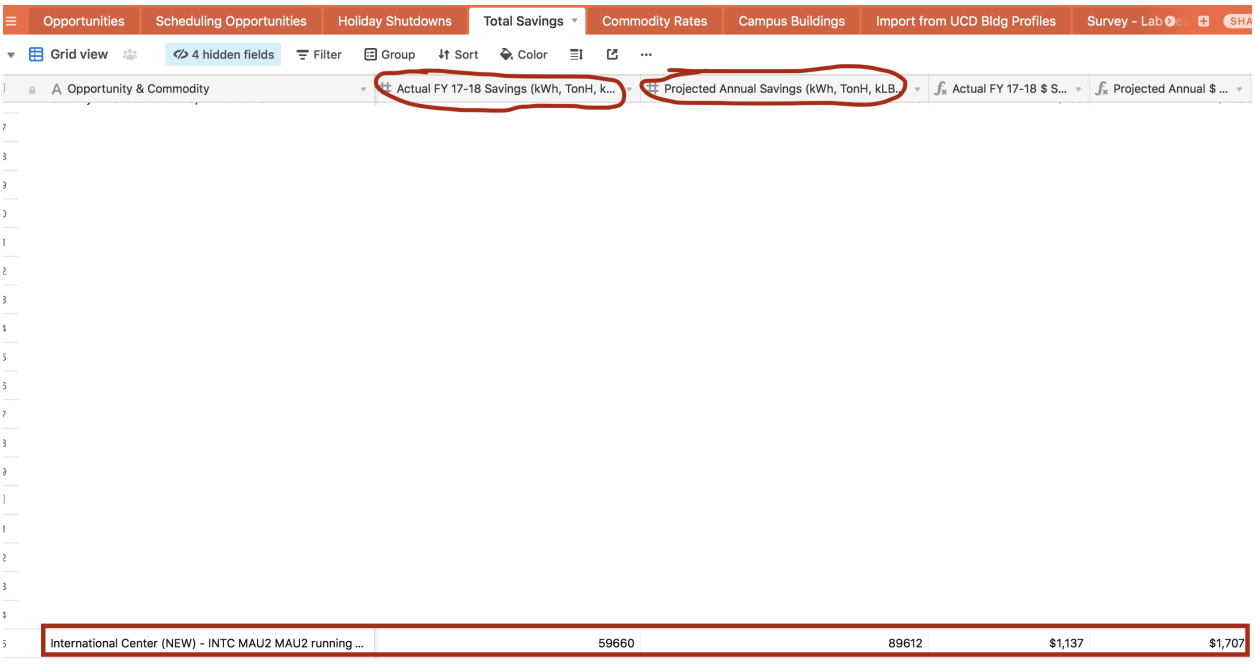


Figure 19: Actual/Projected Savings