ChiPy Mentorship Program

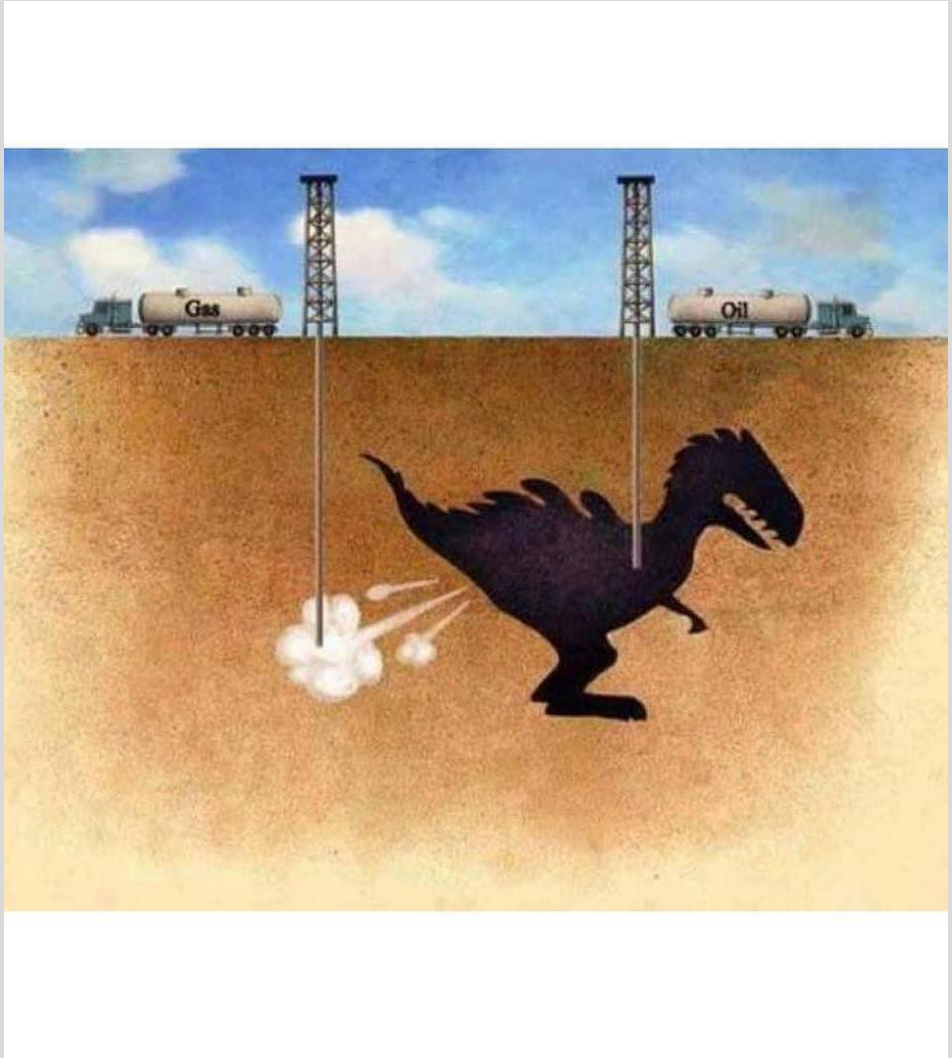
Spring 2017

https://github.com/kurtiskerstein/Chipy_Mentorship

Goals for my project

- Very large dataset
- Machine Learning
- High Level Quantitative Analysis
- Make Predictions
- Distributed Computing

Crude Oil



Disclosure

- The research and opinion contained in this presentation are solely that of the presenter and not reflective of Reliance Capital Markets llc, their officers, employees or affiliates. Futures trading is complex and involves substantial risk of loss. It is not suitable for all investors. You may lose all or more than your original investment.
- Past performance is not indicative of future results.

Topics

- Feature elimination
  - 54,000+ Features
- Prediction
  - Scikit-learn train/test split
    - Limitations
  - Seasonality
    - Optimizing
    - Forecasting

JSON Data


```
{
  "series_id":"NG.RL2R04SOK_1.A",
  "name":"Oklahoma Natural Gas Plant Liquids, Reserves Revision Decreases, Annual",
  "units":"Million Barrels",
  "f":"A",
  "unitsshort":"MMbbl",
  "description":"Oklahoma Natural Gas Plant Liquids, Reserves Revision Decreases",
  "copyright":"None","source":"EIA, U.S. Energy Information Administration",
  "iso3166":"USA-OK",
  "start":"1979",
  "end":"2008",
  "last_updated":"13-AUG-13 11.49.51 AM",
  "data":[
     ["2008","136"],
     ["2007","73"],
     ...
     ["1980","69"],
     ["1979","54"]
  ]
}
```

$$ Price Data $$

Let's make sure that the Price data is available

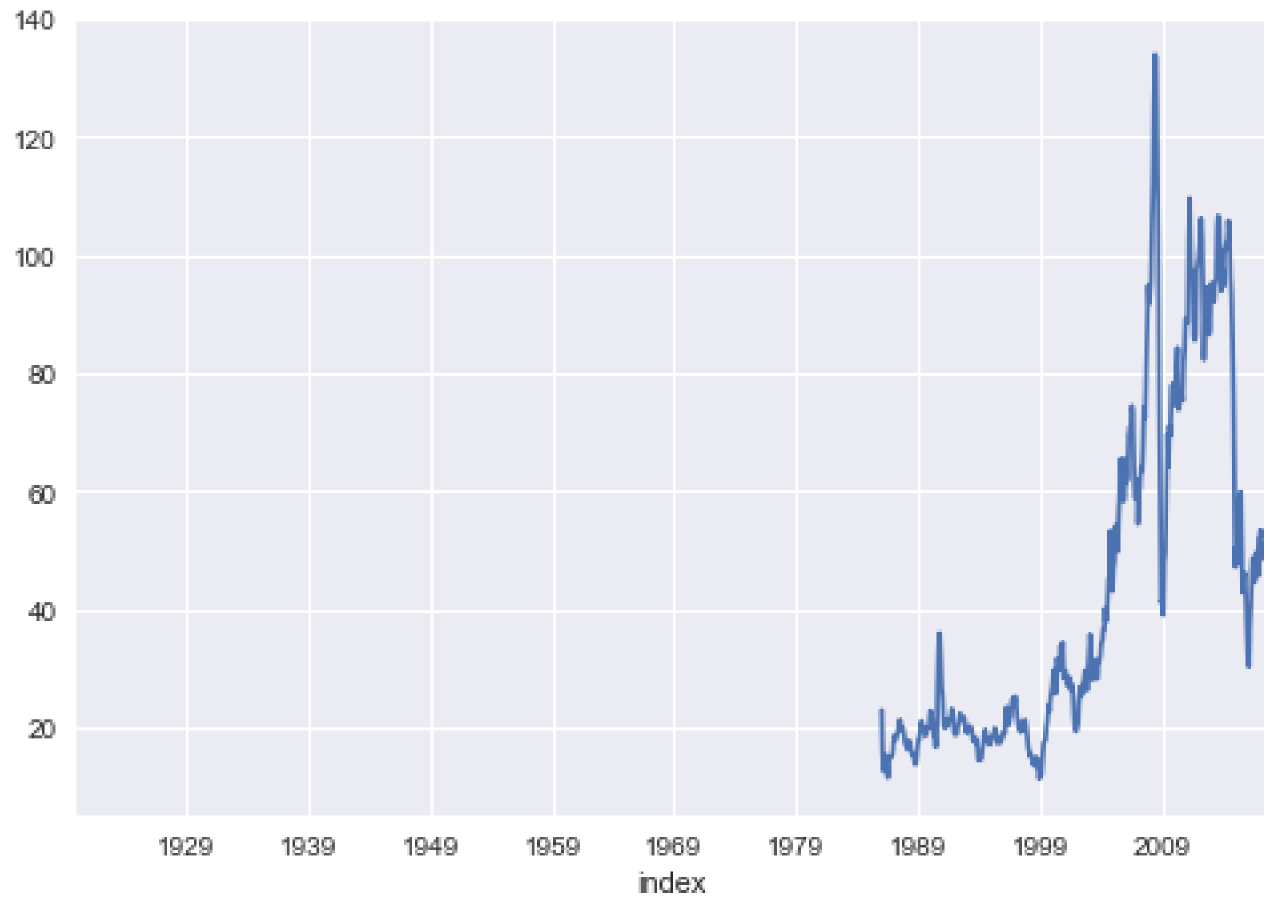
In [7]:
```
df = monthly_crude[['PET.RCLC1.M','PET.RCLC2.M','PET.RCLC3.M','PET.RCLC4.M','PET.RWTC.M']]
df.tail()
```

Out[7]:

index	PET.RCLC1.M	PET.RCLC2.M	PET.RCLC3.M	PET.RCLC4.M	PET.RWTC.M
2017-02-28	53.46	53.93	54.32	54.64	53.47
2017-03-31	49.67	50.22	50.61	50.91	49.33
2017-04-30	51.12	51.52	51.84	52.09	51.06
2017-05-31	48.54	48.86	49.12	49.34	48.48
2017-06-30	NaN	NaN	NaN	NaN	NaN

'PET.RWTC.M' Will be the value we are trying to predict

```
In [10]: monthly_crude['PET.RWTC.M'].plot()

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x11174d4e0>
```



In [5]:
```
monthly_crude = pd.read_csv('./monthly_crude_data.csv',index_col='index',parse_dates=True,infer_datetime_format=True)
monthly_crude.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1170 entries, 1920-01-31 to 2017-06-30
Columns: 54309 entries, PET.EMM_EPMM_PTE_SFL_DPG.M to PET.MBAEXUS2.M
dtypes: float64(54309)
memory usage: 484.8 MB
```

Prepare dateset for Dimensionality Reduction

In [9]:
```
# Set all data series to start January 1986
monthly_crude = monthly_crude.loc['1986-01-31':]

# Remove columns that are missing more than 10% of values
monthly_dataset = monthly_crude[monthly_crude.columns[monthly_crude.count() >= (.90 * monthly_crude['PET.RWTC.M'].count(


#fill in values instead of NaN's
monthly_dataset.fillna(method='ffill',axis=0,inplace=True)
monthly_dataset.fillna(0,inplace=True)

# Drop price Data
monthly_dataset = monthly_dataset.drop(['PET.RCLC1.M','PET.RCLC2.M','PET.RCLC3.M','PET.RCLC4.M','PET.RWTC.M'],axis=1)

monthly_dataset.tail(10)
```

Removed Thousands of features just by filtering!

In [10]:
```
monthly_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 378 entries, 1986-01-31 to 2017-06-30
Columns: 3864 entries, PET.C130013451.M to PET.MPFEXP32.M
dtypes: float64(3864)
memory usage: 11.1 MB
```

Feature Elimination

- scikit-learn
  - Linear Regression
  - RFE
- Train/test split
- Score
- Test set

Recursive Feature Elimination (RFE)

In [15]:

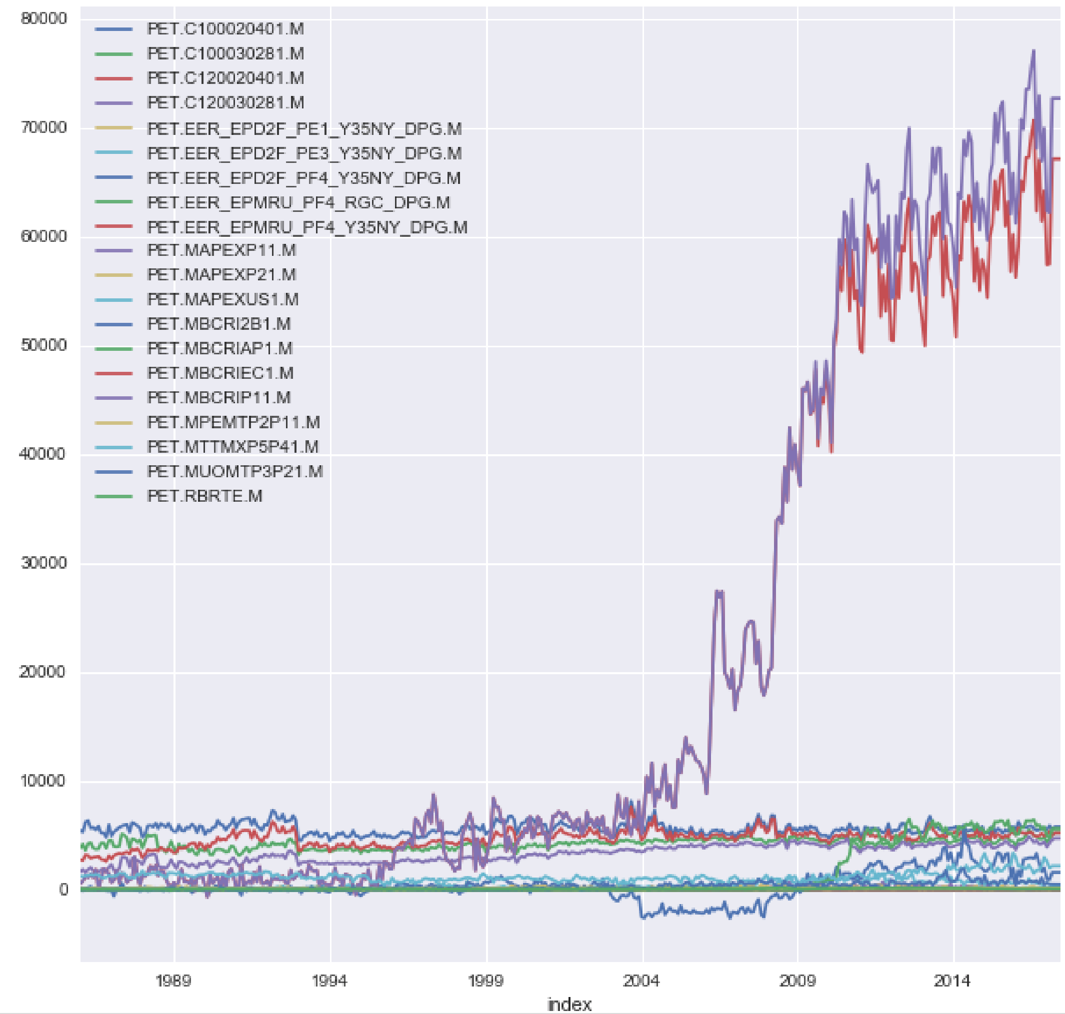
```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
```

In [36]:

```
#use linear regression as the model
lr_rfe = LinearRegression()
rank all features, i.e continue the elimination until the last one
rfe = RFE(lr_rfe,step=20, n_features_to_select=20)
rfe.fit(monthly_dataset,y)

with open('RFE_linearregression.pickle','wb') as f:
    pickle.dump(rfe, f)

# pickle_in = open('RFE_linearregression.pickle','rb')
# rfe = pickle.load(pickle_in)
```

Remaining Features

```
In [19]: results_rfe.plot(figsize=(10,10))

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x10cfb3400>
```



Some info about these features

PET.MAPEXP11.M	East Coast (PADD 1) Exports of Asphalt and Roa...	Thousand Barrels	East Coast (PADD 1) Exports of Asphalt and Roa...	Mbbl	M	1970-01-03 07:01:41	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MAPEXP21.M	Midwest (PADD 2) Exports of Asphalt and Road O...	Thousand Barrels	Midwest (PADD 2) Exports of Asphalt and Road Oil	Mbbl	M	1970-01-03 07:01:41	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MAPEXUS1.M	U.S. Exports of Asphalt and Road Oil, Monthly	Thousand Barrels	U.S. Exports of Asphalt and Road Oil	Mbbl	M	1970-01-03 07:01:41	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MBCRI2B1.M	Refining District Minnesota-Wisconsin-North Da...	Thousand Barrels	Refining District Minnesota-Wisconsin-North Da...	Mbbl	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MBCRIAP1.M	Refining District Appalachian No. 1 Refinery a...	Thousand Barrels	Refining District Appalachian No. 1 Refinery a...	Mbbl	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MBCRIEC1.M	Refining District East Coast Refinery and Blen...	Thousand Barrels	Refining District East Coast Refinery and Blen...	Mbbl	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MBCRIP11.M	East Coast (PADD 1) Refinery and Blender Net I...	Thousand Barrels	East Coast (PADD 1) Refinery and Blender Net I...	Mbbl	M	1970-01-03 07:01:41	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MPEMTP2P11.M	Midwest (PADD 2) Receipts by Tanker and Barge ...	Thousand Barrels	Midwest (PADD 2) Receipts by Tanker and Barge ...	MBBL	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MTTMXP5P41.M	West Coast (PADD 5) Receipts by Pipeline, Tank...	Thousand Barrels	West Coast (PADD 5) Receipts by Pipeline, Tank...	MBBL	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.MUOMTP3P21.M	Gulf Coast (PADD 3) Receipts by Tanker and Bar...	Thousand Barrels	Gulf Coast (PADD 3) Receipts by Tanker and Bar...	MBBL	M	1970-01-03 07:10:01	1970-01-03 08:01:42	2017-05-02T14:54:09-04:00
PET.RBRTE.M	Europe Brent Spot Price FOB, Monthly	Dollars per Barrel	Europe Brent Spot Price FOB	$/bbl	M	1970-01-03 07:11:45	1970-01-03 08:01:44	2017-05-10T13:24:45-04:00

Train/Test

```
from sklearn.model_selection import train_test_split
from sklearn import cross_validation
X_train, X_test, y_train, y_test = cross_validation.train_test_split(results_rfe, y, test_size=0.2)
lr_rfe.fit(X_train,y_train)

lr_rfe.score(X_test,y_test)
```

```
/Users/Prometheus/Code/Chipy_Mentorship/venv3.6/lib/python3.6/site-packages/scipy/linalg/basic.py:1018: RuntimeWarnin
g: internal gelsd driver lwork query error, required iwork dimension not returned. This is likely the result of LAPAC
K bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gelss' driver.
  warnings.warn(mesg, RuntimeWarning)
```

```
0.99361657190795627
```

Full Dataset

	ind1	ind2	ind3	ind4	ind5	dependent
2017-07-04	0	1	2	3	4	0
2017-07-05	5	6	7	8	9	1
2017-07-06	10	11	12	13	14	2
2017-07-07	15	16	17	18	19	3
2017-07-08	20	21	22	23	24	4
2017-07-09	25	26	27	28	29	5
2017-07-10	30	31	32	33	34	6
2017-07-11	35	36	37	38	39	7
2017-07-12	40	41	42	43	44	8
2017-07-13	45	46	47	48	49	9

Training set

	ind1	ind2	ind3	ind4	ind5	dependent
2017-07-04	0	1	2	3	4	0
2017-07-05	5	6	7	8	9	1
2017-07-06	10	11	12	13	14	2
2017-07-07	15	16	17	18	19	3
2017-07-08	20	21	22	23	24	4
2017-07-09	25	26	27	28	29	5
2017-07-10	30	31	32	33	34	6

Test set

	ind1	ind2	ind3	ind4	ind5	dependent
2017-07-11	35	36	37	38	39	7
2017-07-12	40	41	42	43	44	8
2017-07-13	45	46	47	48	49	9

Test Set Dilemma

- What about tomorrow or end of month?
  - Out of sample values (AKA Steps)

	ind1	ind2	ind3	ind4	ind5
2017-07-14	50	51	52	53	54
2017-07-15	55	56	57	58	59

- Forecast my inputs
  - How do we do this?

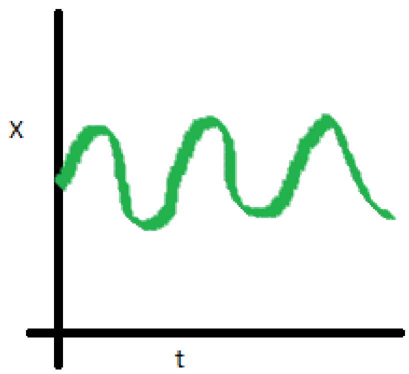
	prediction
2017-07-14	10
2017-07-15	11

With Trends!!!

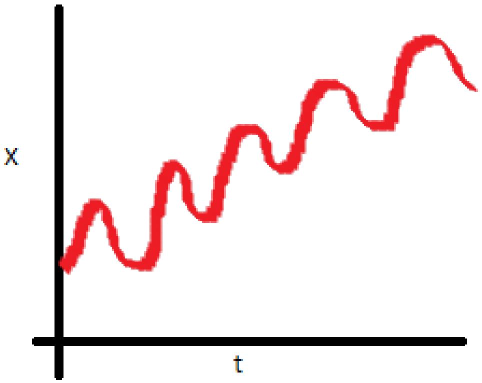
Seasonality

- Cyclical nature of the commodities markets

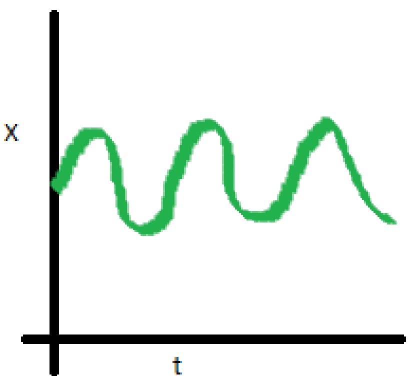
- Past data can give us information about the future



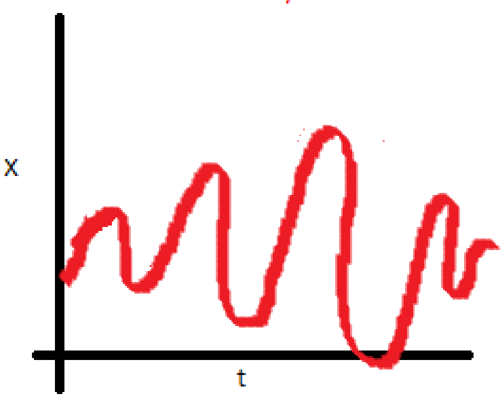
Stationary series



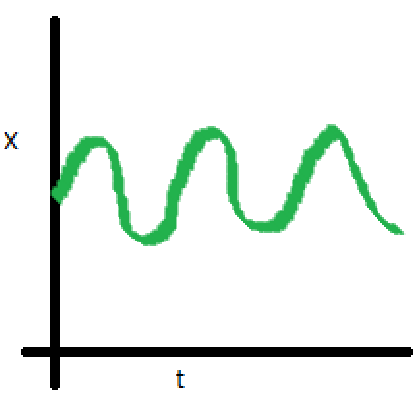
Non-Stationary series



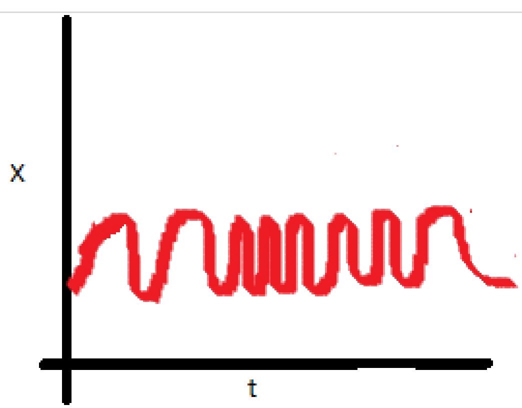
Stationary series



Non-Stationary series



Stationary series



Non-Stationary series

SARIMAX

```
def curve_fit(series):
    # Define the p, d and q parameters to take any value between 0 and 2
    p = d = q = range(0, 2)

    # Generate all different combinations of p, d and q triplets
    pdq = list(itertools.product(p, d, q))

    # Generate all different combinations of seasonal p, q and q triplets
    seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

#     warnings.filterwarnings("ignore") # specify to ignore warning messages
    param_list=[]
    for param in pdq:
        for param_seasonal in seasonal_pdq:
            try:
                mod = sm.tsa.statespace.SARIMAX(series,
                                                order=param,
                                                seasonal_order=param_seasonal,
                                                enforce_stationarity=False,
                                                enforce_invertibility=False)
                results = mod.fit()
                param_list.append([results.aic,(param,param_seasonal)])
    #             print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
            except:
                continue
    return param_list
```

Forecast Series

```
def predict_SARIMAX(series,num_months=False):
    #return optimized paramaters
    value = curve_fit(series)
    value.sort()
    value

    #pass optimized paramater to ARIMA
    mod = sm.tsa.statespace.SARIMAX(series,
                                    order=value[0][1][0],
                                    seasonal_order=value[0][1][1],
                                    enforce_stationarity=False,
                                    enforce_invertibility=False)

    results = mod.fit()
    #print(series)
    #print(results.summary().tables[1])

    #predict number of months out of sample data
    pred = results.predict(start=results_rfe.iloc[-1].name,
                                    end=results_rfe.iloc[-1].name+relativedelta(months=num_months))
    return pred
```

Forecast each column in dataset

In [40]:

```
#new dataframe for forcasted values
forecast =pd.DataFrame()

#for each column in the dataframe pass through column to ARIMA
all_cols = results_rfe.columns
for col in all_cols:
    # call function for ARIMA here and assign this to a new series
#     forecast[col] = forecast_ARIMA(results_rfe[col],5)
    forecast[col] = predict_SARIMAX(results_rfe[col],5)

forecast.to_csv('forecast.csv')
```

Created Values are now inputs

In [406]:
```
forecast
```

Out[406]:

	PET.C100020401.M	PET.C100030281.M	PET.C120020401.M	PET.C120030281.M	PET.EER_EPD2F_PE1_Y35NY_DPG.M	PET.EER_EPD2F_PE3_Y35NY_DPG.M	PI
2017-06-30	5890.865124	5301.409809	5321.678974	4774.269053	1.490431	1.497725	
2017-07-31	5553.488049	5158.627631	5012.344295	4650.170890	1.514000	1.527000	
2017-08-31	5664.631864	5115.721876	5124.599226	4649.745606	1.514000	1.527000	
2017-09-30	5610.167845	5019.991918	5140.943956	4590.451244	1.514000	1.527000	
2017-10-31	5603.478772	4986.363739	5116.909493	4552.594836	1.514000	1.527000	
2017-11-30	5525.561020	4934.699350	5045.264594	4504.756093	1.514000	1.527000	

Linear Regression Prediction

lr_rfe fit model to predict price from forcasted features

```
In [45]: forecast['Prediction'] = lr_rfe.predict(forecast)
         forecast['Prediction']

Out[45]: 2017-06-30    58.484699
         2017-07-31    51.501940
         2017-08-31    51.693834
         2017-09-30    48.764816
         2017-10-31    45.354732
         2017-11-30    44.851821
         Freq: M, Name: Prediction, dtype: float64
```

For the future

- Find other alternatives for relevant features
- Optimizing the forecast for each feature
- Distributed Computing
- Scale the project

Thank you

Patrick Boland (Mentor)

U.S. Energy Information Administration (EIA)

Questions?