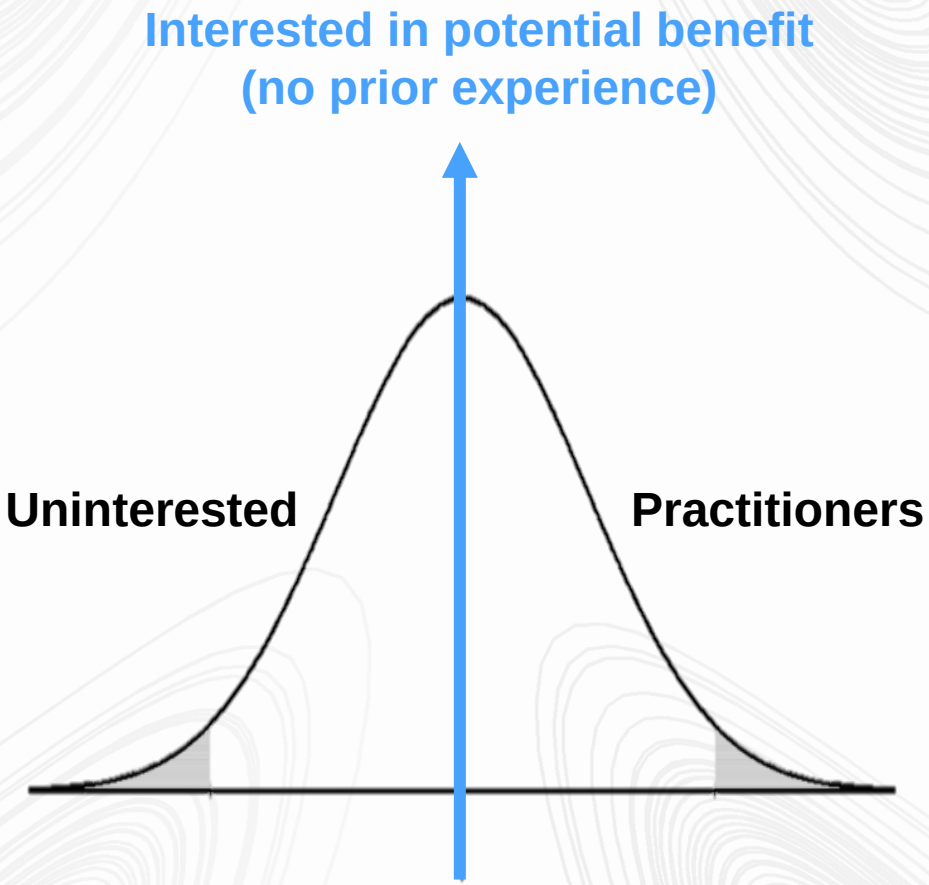
Forecasting in Energy Markets

Approaches, tools and resources for aspiring analysts

Josh Danial
February 2022
NYU SPS Center for Global Affairs

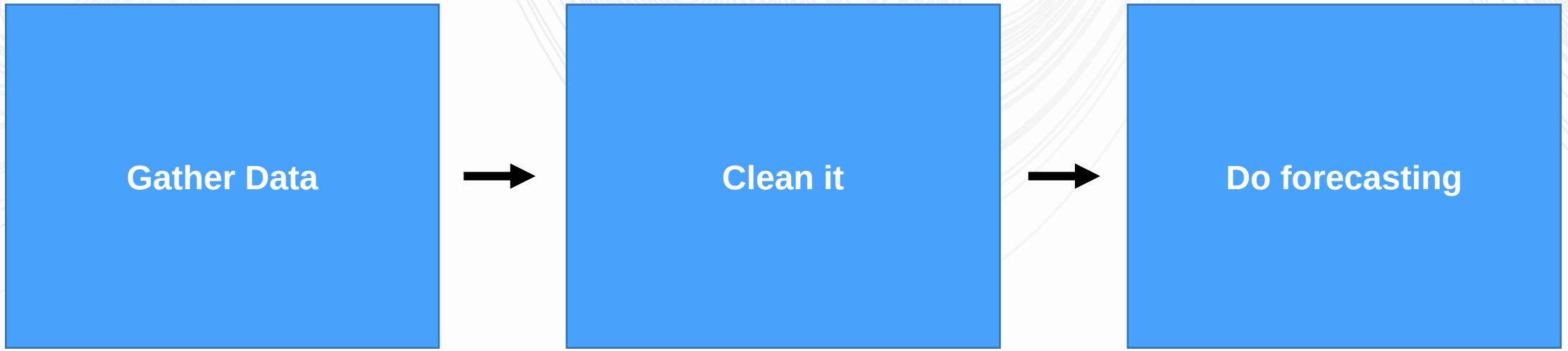
Context

The average attendee

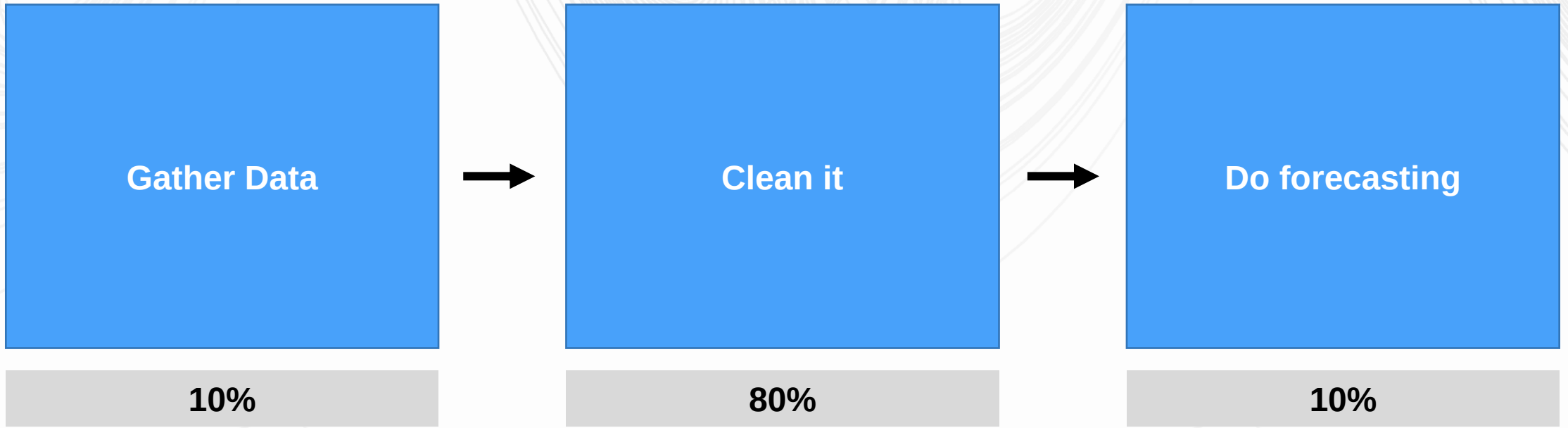


The goal: Get our feet wet

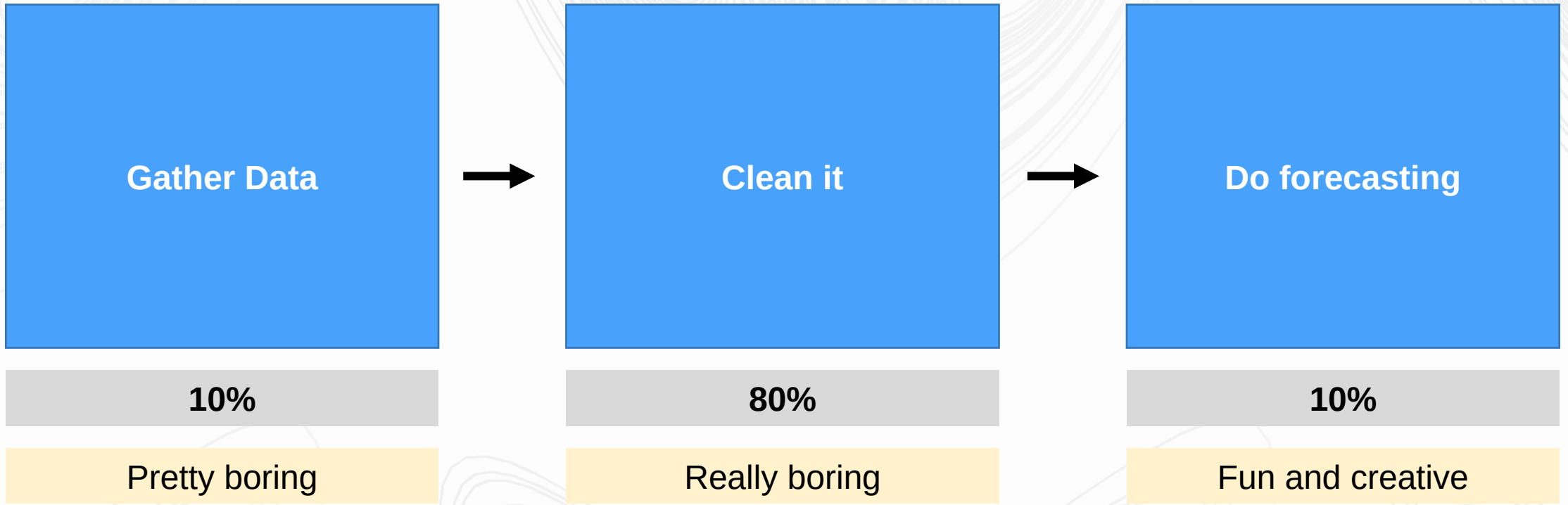
Typical analyst workflow



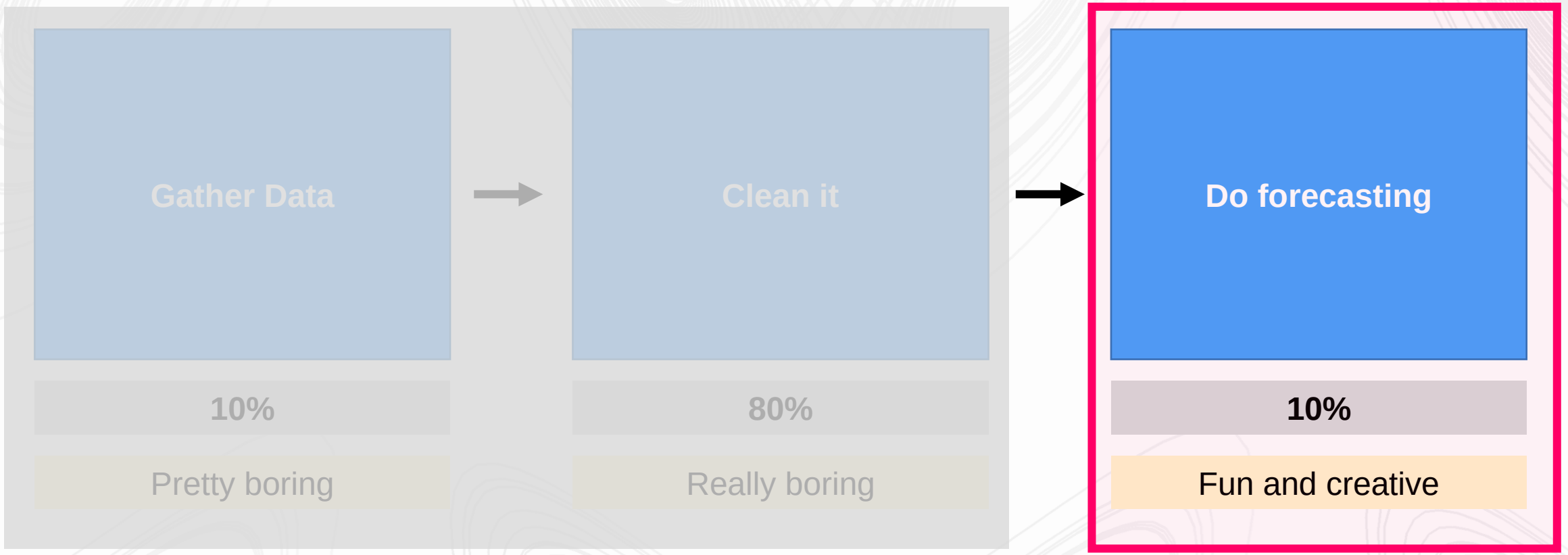
Time allocation



Excitement


Gather Data
Clean it
Do forecasting
10%
80%
10%
Pretty boring
Really boring
Fun and creative

This talk is about the fun part

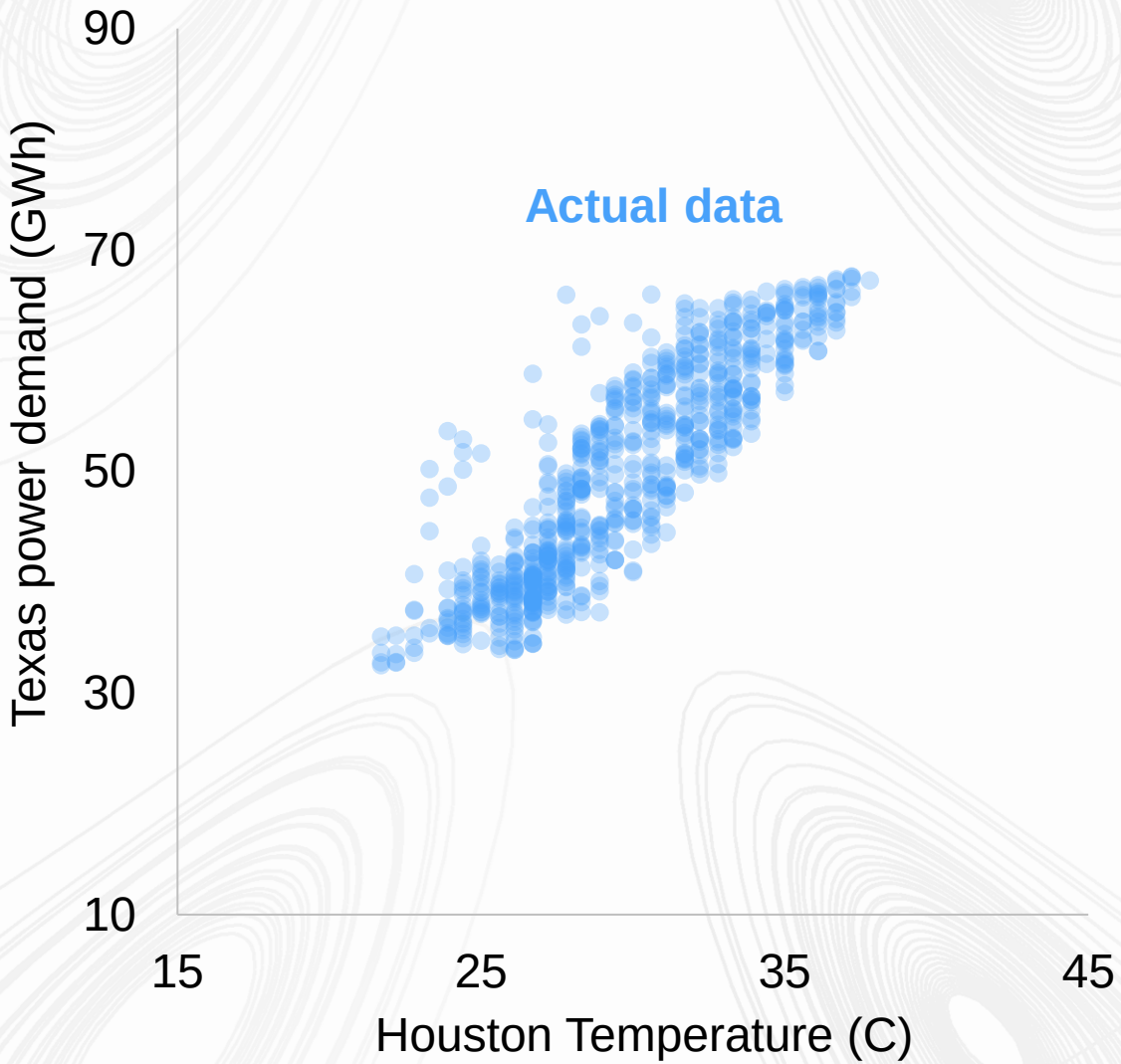


How this relates to machine learning

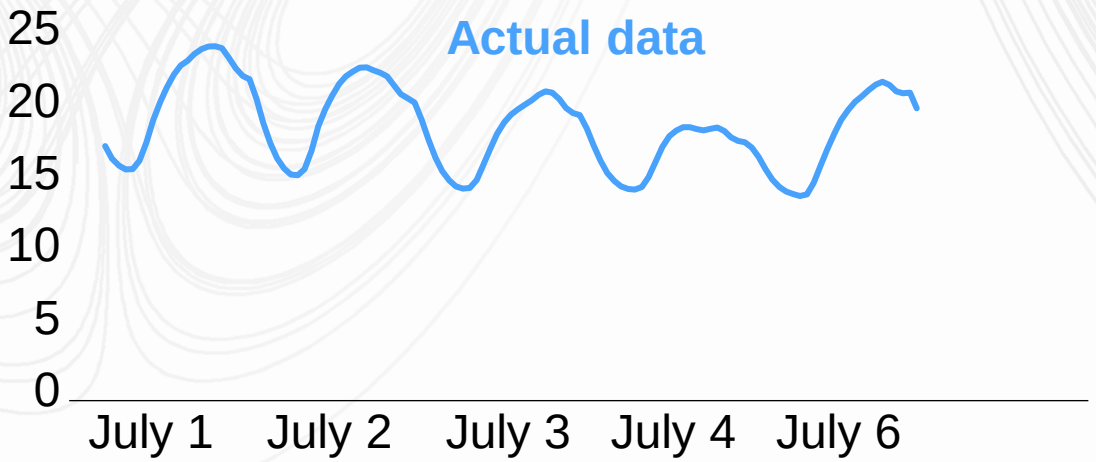
For those interested

Regressions

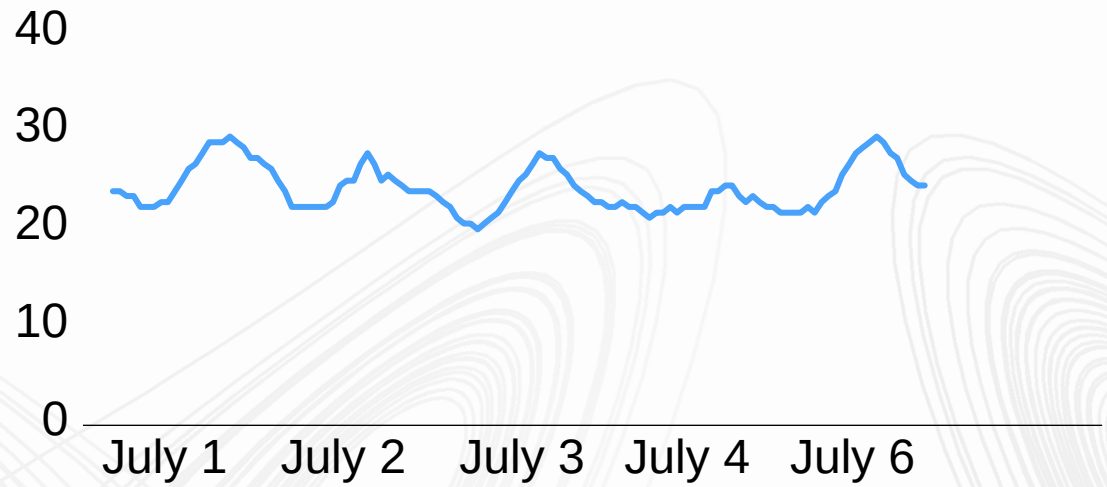
Temperature vs. demand in July, hourly



Power demand (GWh), hourly

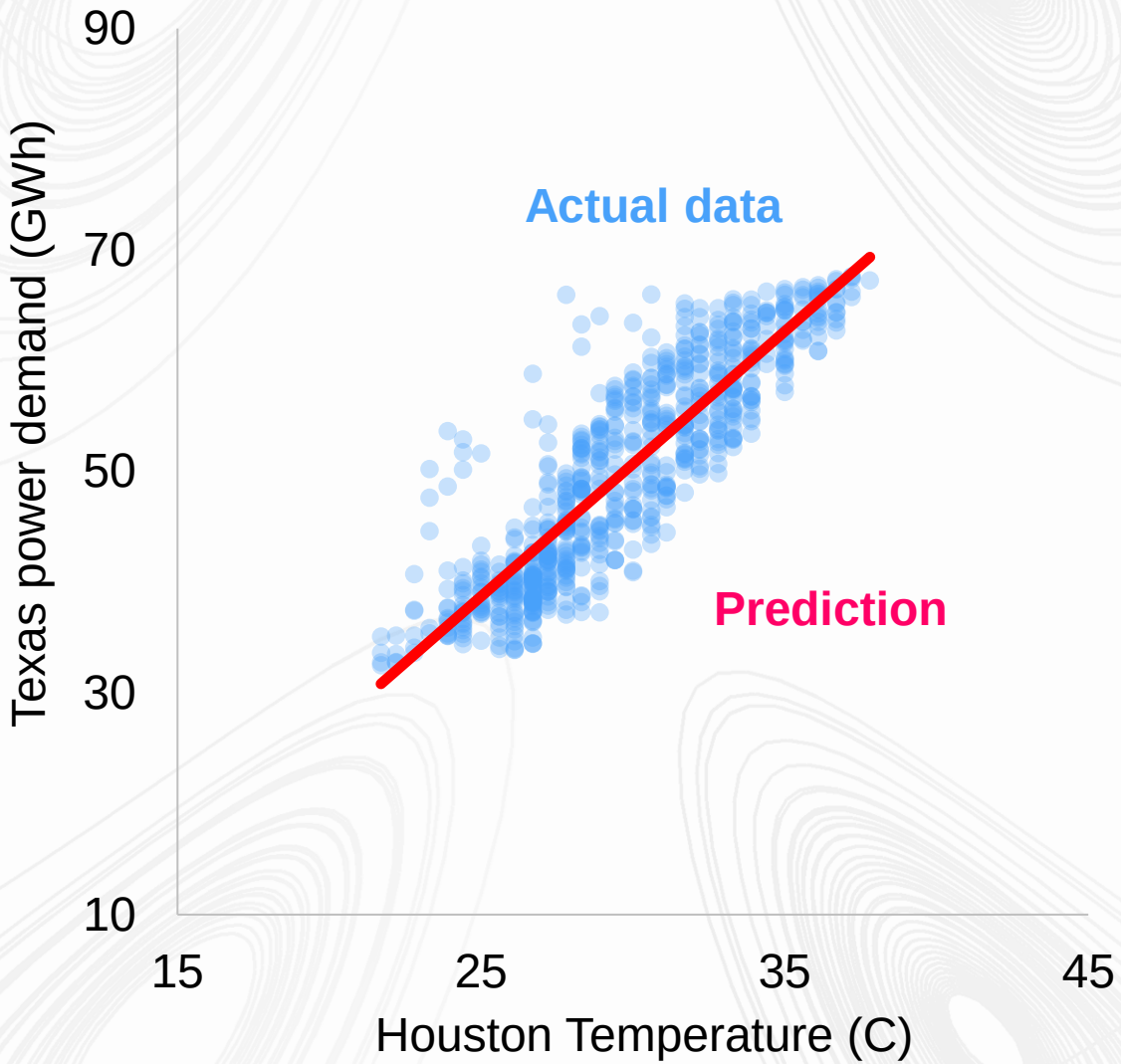


Temperature (C), hourly

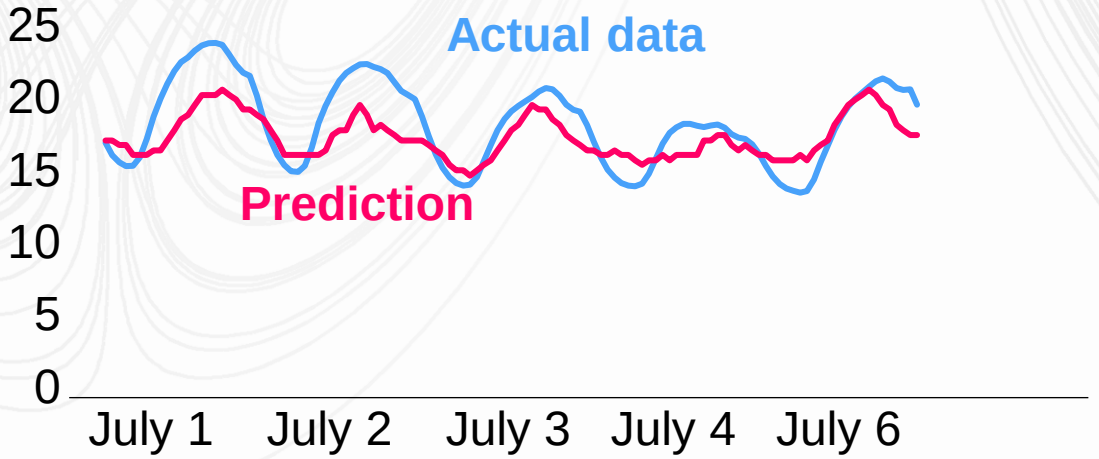


Hunting for correlation

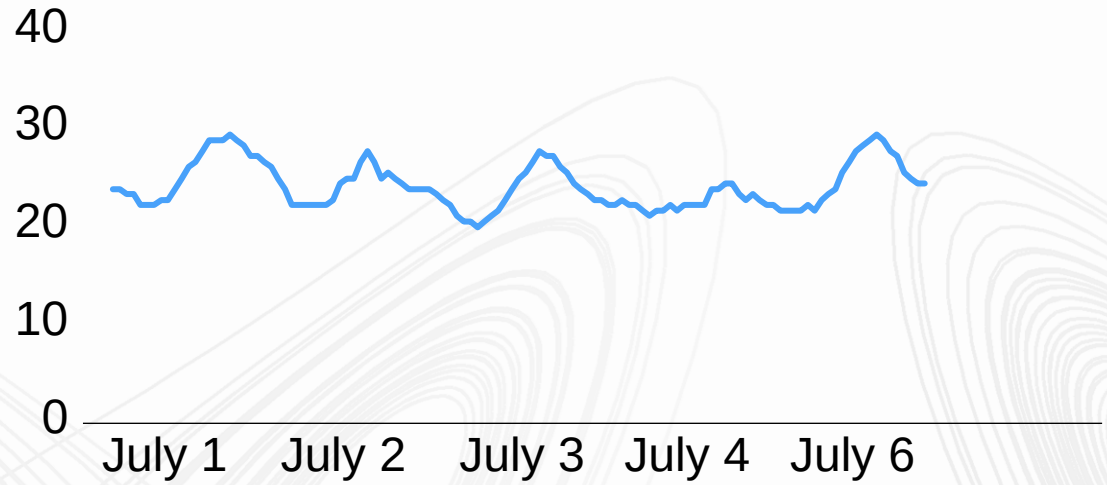
Temperature vs. demand in July, hourly



Power demand (GWh), hourly



Temperature (C), hourly



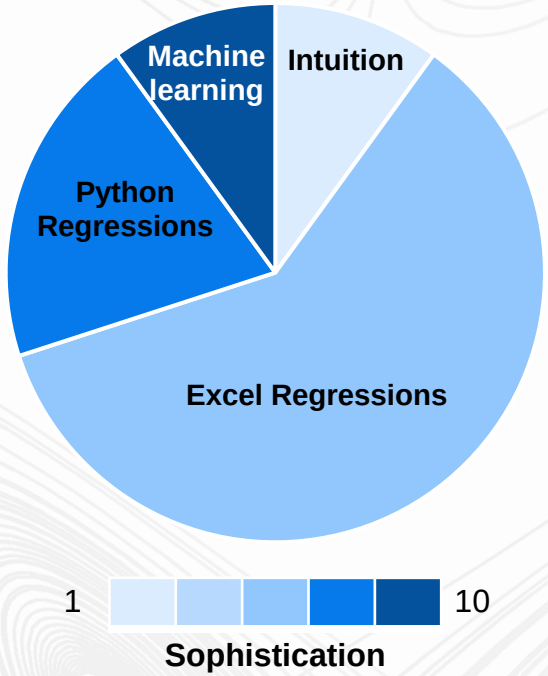
Three misconceptions:
I can't…

1. Get the data I need

- All the data you need is publicly available
- Even at energy trading shops, most of the data feeding analysis is public

2. Access the best tools

Forecasting tools used in power and gas industry (my take)



3. Apply those techniques

- Most practitioners don't know the math that goes into a linear regression, even after using it every day
- All you need is a basic understanding of "black box testing"
  - Wikipedia
  - Great lecture

1. Can’t get the data I need

Energy

- EIA open data
  - Consolidated EIA data inventory
- EIA 860 (monthly)
  - The U.S. plant stack. Information about every power plant >1 MW.
- EIA 923
  - “Stack series.” Monthly generation and fuel burn. Maps to EIA 860.
- EIA 930
  - Balancing authority level generation by tech + power demand, hourly.
- EIA 861
  - Monthly utility load and revenues. Retail power prices.
- EIA weekly/monthly petroleum supply estimates
  - Production, storage, imports, exports, refinery outputs (demand).
- EIA drilling productivity report
  - U.S. basin-level well activity by completion status, monthly.
- JODI
  - Global, monthly oil production and demand

Socio-economic

- FRED
  - One of the easiest and most comprehensive databases of US data.
- Bureau of Labor Statistics
  - Wages, consumer price index (inflation, and much more.
- Census
  - Population/demographics– critical for a lot of hypotheses.
- World Bank
  - I don’t really use this, but a lot of people do. I think most of the data is fake
- Google's Community Mobility Reports
  - Google provides county-level, global, mobility indices. Its an incredible dataset for tracking the severity of lockdowns and recoveries.
- JHU CSSE COVID-19 Data
  - County-level confirmed cases, deaths and population.

Other

- Data.gov
  - A government project to consolidate data from a bunch of federal, state and local governments. Pretty sweet, and new datasets are added perpetually. I found a lot of the prior datasets on here.
- Yahoo stock prices
  - Stock prices
- Fundamental data
  - Consolidated, clean data quarterly data from public companies
- VMT
  - U.S. vehicle miles traveled by state and road-type, monthly.

2. Three Python packages, endless potential

Data wrangling: pandas

- Nicknamed “Excel on steroids”
- Does Excel-like operations on data
- 90% of my coding time involves this package
- Resources
  - Data Analysis with Python
  - Complete Python Pandas Data Science Tutorial
  - Web Scraping with Python Pandas
  - Web Scraping with Python

Regression: statsmodels + scikit-learn

- Statistical/machine learning regressions
- statsmodels: basic statistical models
  - OLS: ordinary regression
- scikit-learn: machine learning models
  - HuberRegressor: Linear regression model that is robust to outliers
  - LinearSVR: Very fancy regression
  - AdaBoostRegressor: Very fancy regression
  - GradientBoostingRegressor: Very fancy regression

Auto-regression: prophet

- Easy to use/automatic forecasting
- Looks at seasonality and trend in timeseries data to make a prediction
- What to use when you don’t have explanatory data
  - Quick start
  - Time series prediction using Prophet in Python
  - Forecasting gas and electricity utilization using Facebook prophet

3. Study how people apply these tools

Some resources with methodologies and code

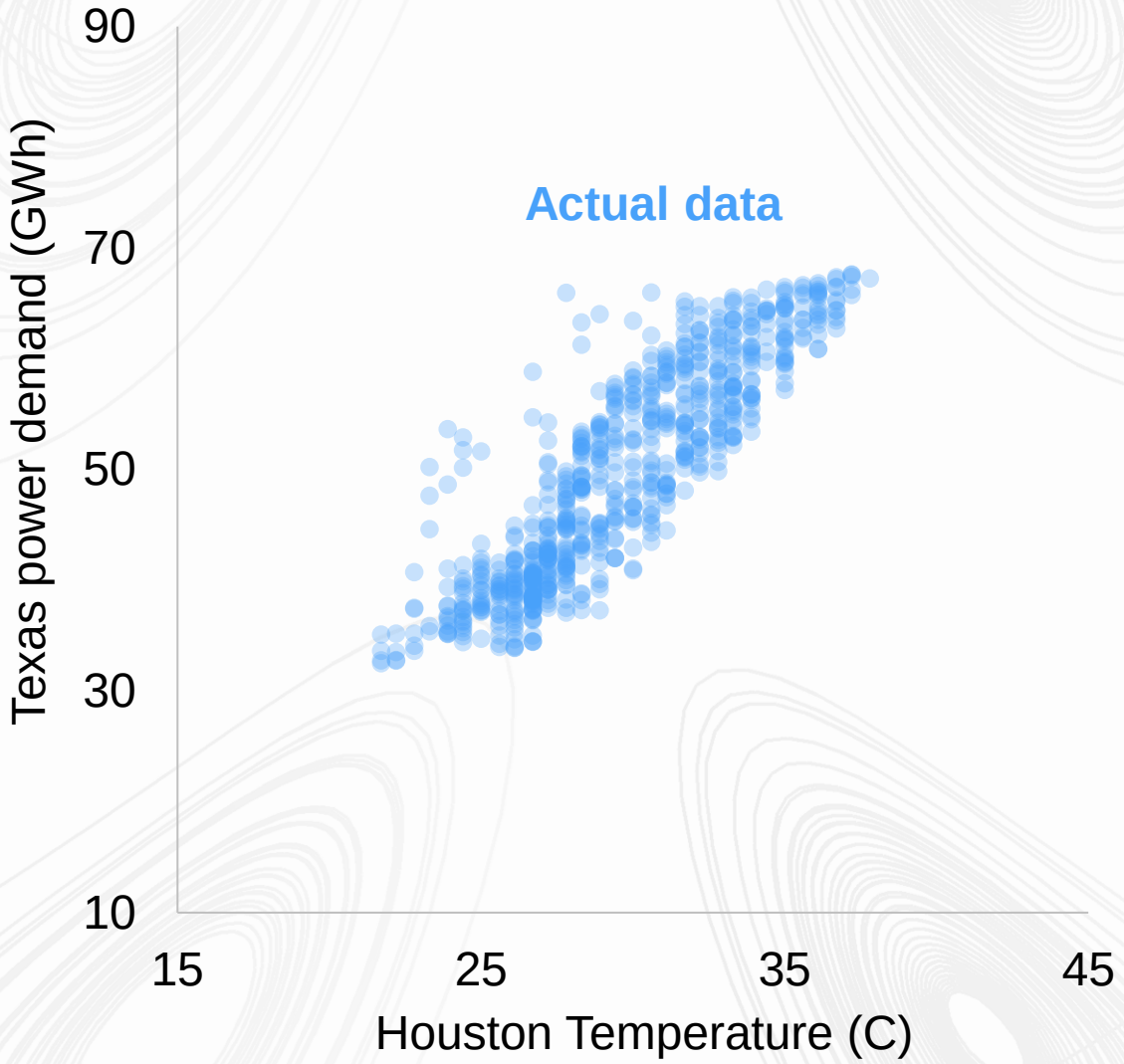
- Electricity demand
  - Electricity demand forecasting using machine learning
  - Forecasting electricity demand with Python
- Renewable generation
  - Wind Energy Forecasting with Python
  - Forecasting Renewable Energy Generation with Streamlit and sktime
- Power prices
  - Data-driven modeling for long-term electricity price forecasting
  - Forecasting day-ahead electricity prices

3. Code adventure

Let’s see how far we can get with our current knowledge base, and a couple more details

Hunting for correlation

Temperature vs. demand in July, hourly

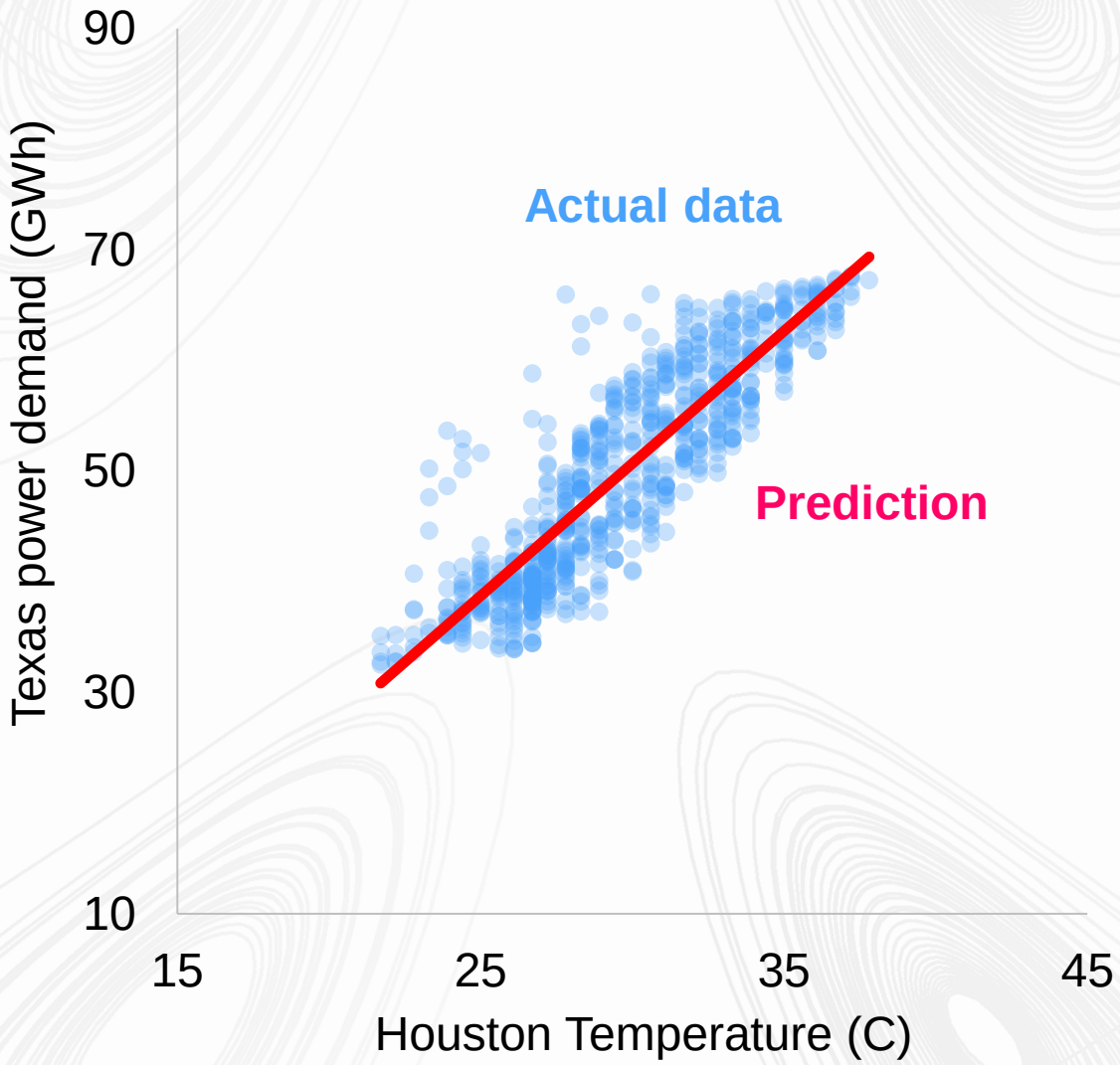


Load data/make a scatter plot

```
import pandas as pd
# read
data = pd.read_csv('data.csv')
# chart
data.plot.scatter('temp', 'load')
print(data.head(5))
'''Out:
             datetime     load  temp
0  2015-07-01 01:00:00  37456.0  22.8
1  2015-07-01 02:00:00  35119.0  21.7
2  2015-07-01 03:00:00  33638.0  21.7
3  2015-07-01 04:00:00  32798.0  21.7
4  2015-07-01 05:00:00  32805.0  22.2
'''
```

Simple, powerful linear regression

Temperature vs. demand in July, hourly

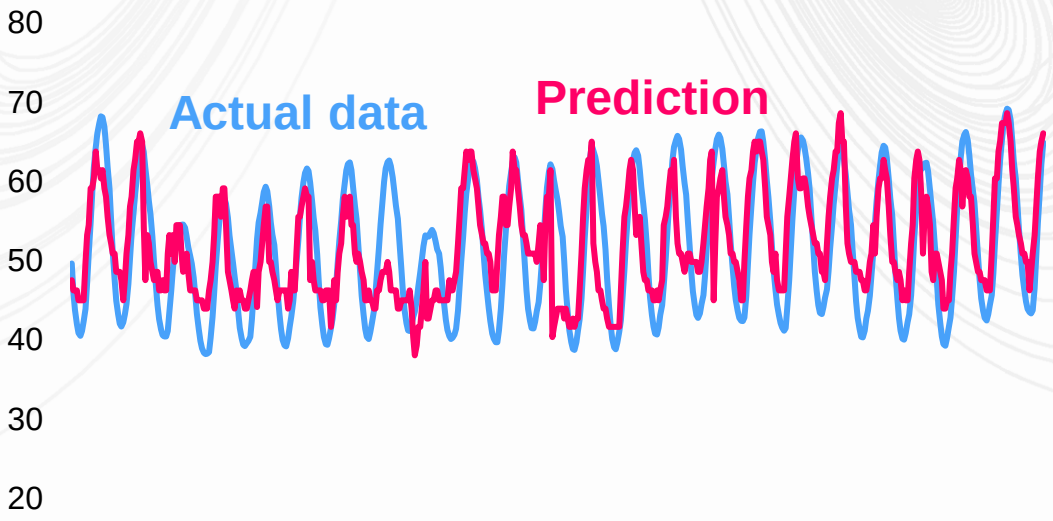


Running a regression

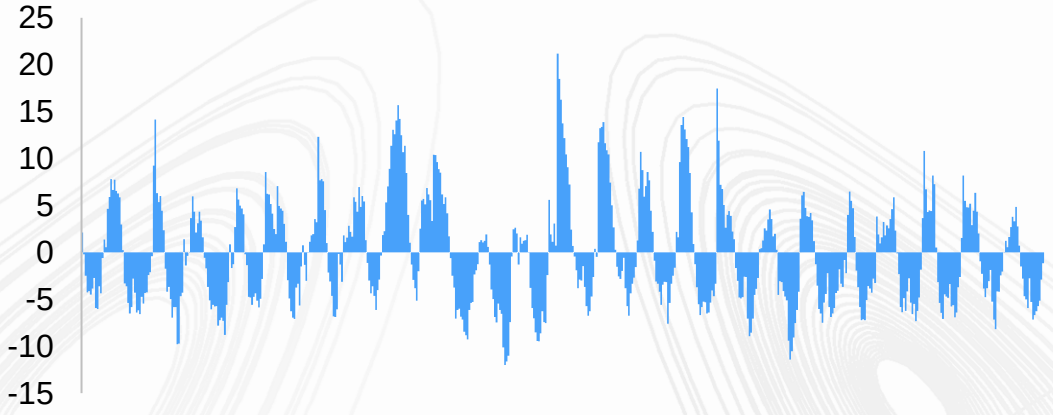
```
ort

import statsmodels.api as sm

# add constant
data['constant'] = 1

# make variables

x = data[['temp', 'constant']]

y = data['load']

# statsmodels takes in Y then X
model = sm.OLS(y, x)

out = model.fit()

# apply regression function

data['pred'] = out.predict(x)

# chart

ax = data.plot.scatter('temp', 'load')

data.plot(ax=ax, x='temp', y=pred', color='r')
```

What to look for

Actual power demand vs. expected, GWh



Errors (actual - predicted), GWh

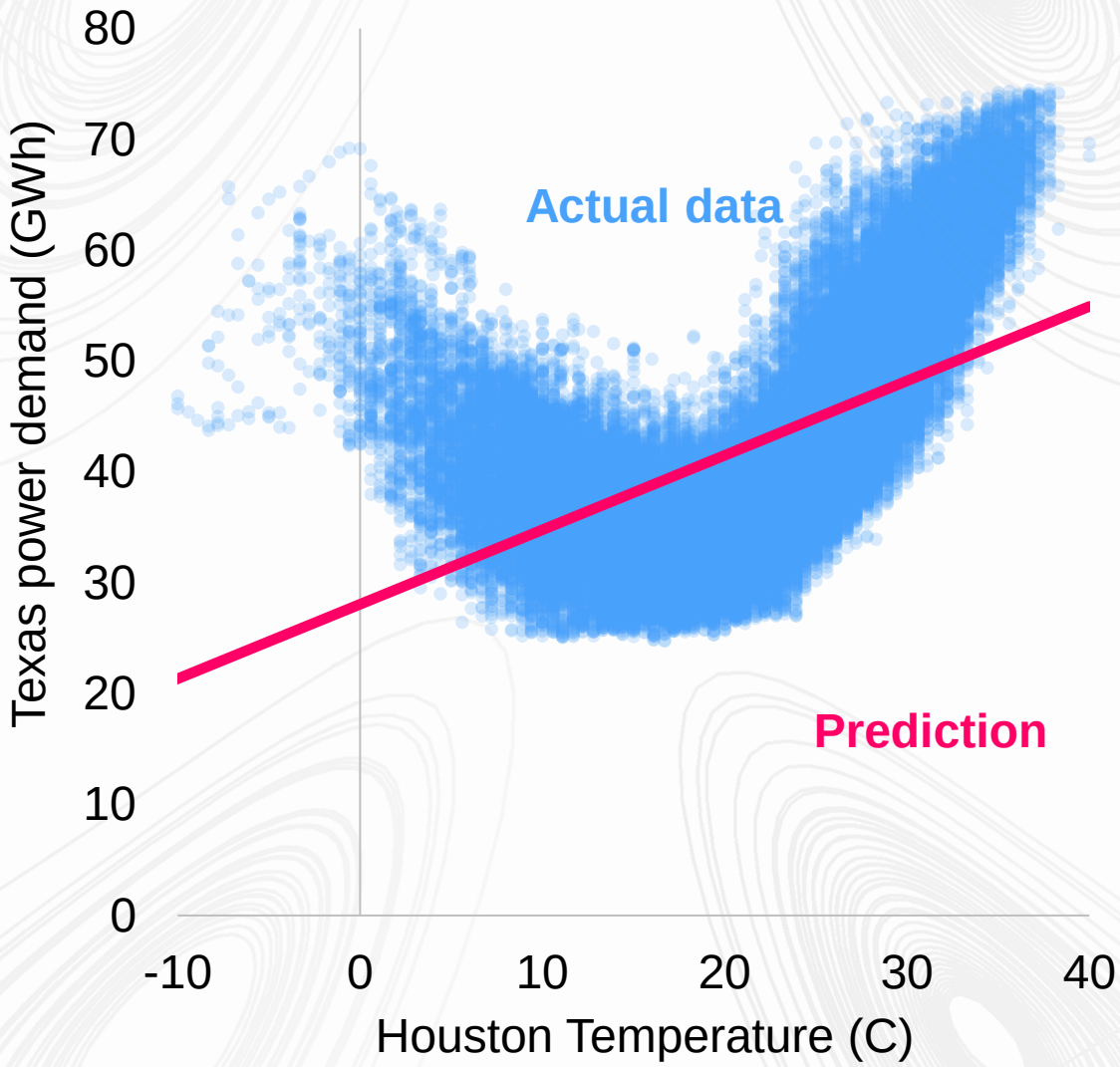


Getting regression statistics

```
print(result.summary())

'''Out:
                            OLS Regression Results
==============================================================================
Dep. Variable:                   load   R-squared:                       0.297
Model:                            OLS   Adj. R-squared:                  0.297
Method:                 Least Squares   F-statistic:                 2.430e+04
Date:                Thu, 03 Feb 2022   Prob (F-statistic):               0.00
Time:                        05:10:55   Log-Likelihood:            -5.9964e+05
No. Observations:               57565   AIC:                         1.199e+06
Df Residuals:                   57563   BIC:                         1.199e+06
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
temp         671.7196      4.309    155.871      0.000     663.273     680.166
constant    2.809e+04     98.890    284.066      0.000    2.79e+04    2.83e+04
'''
```

When it breaks down

Temperature vs. demand, hourly (2015-2021)

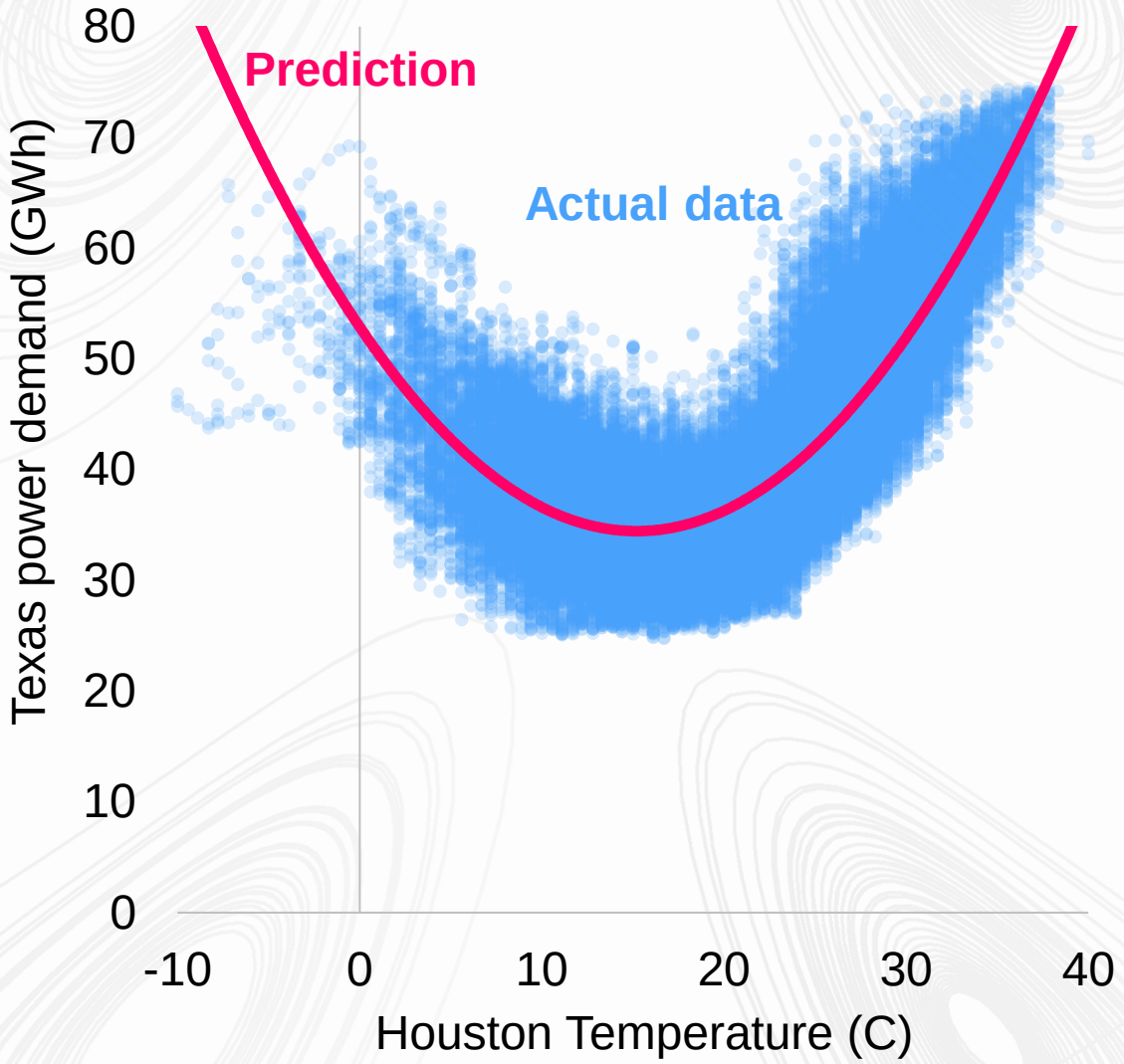


Same code from earlier

```
x = data[['temp', 'constant']]
y = data['load']
# same code from before
model = sm.OLS(y, x)
out = model.fit()
data['pred'] = out.predict(x)
# plot.. same code
ax = data.plot.scatter('temp', 'load')
data.plot(ax=ax, x='temp', y='pred', color='r')
```

Classical statistics still has an answer

Temperature vs. demand, hourly (2015-2021)

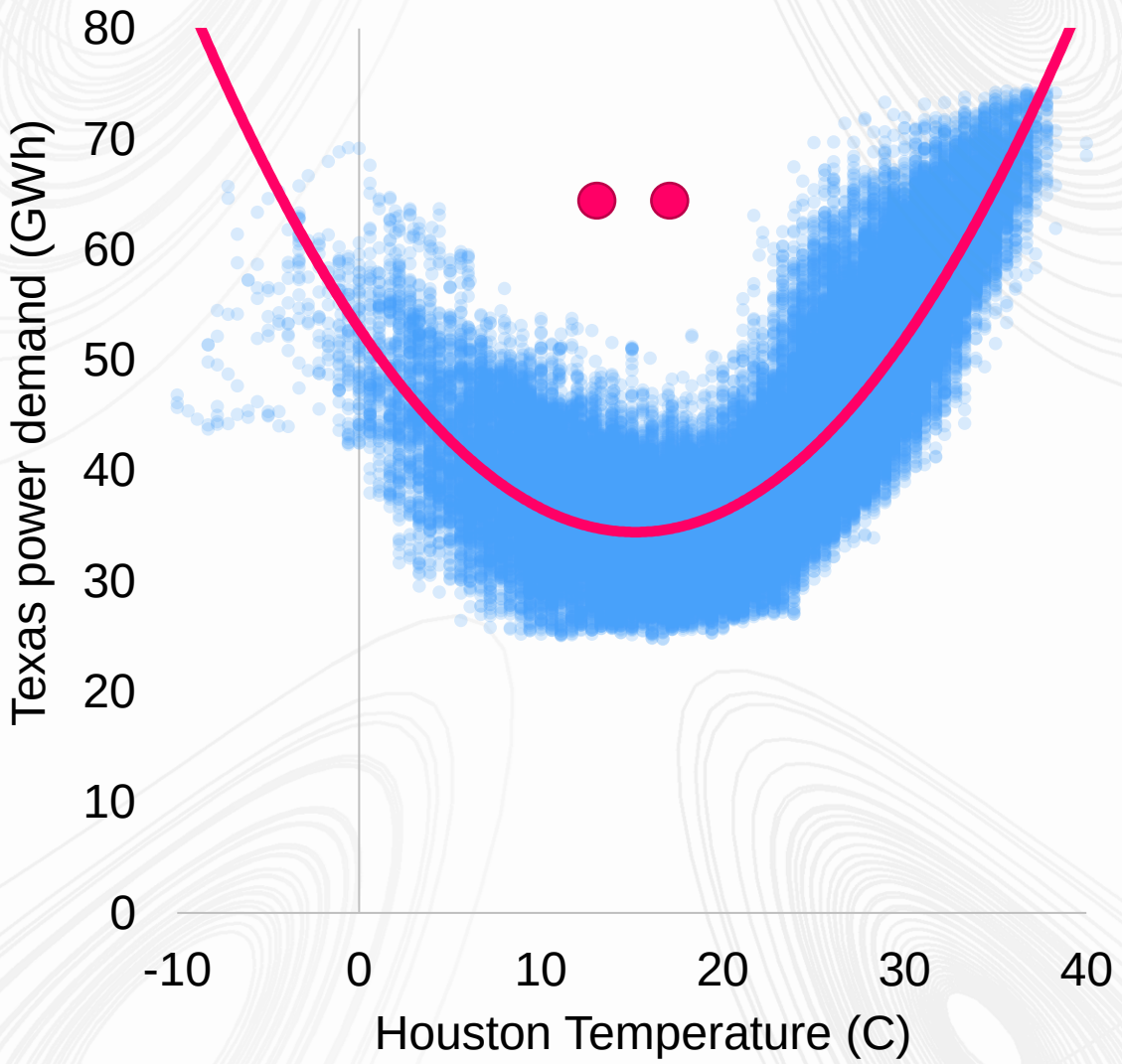


Polynomial regression in Python

```
# add polynomial
data['temp^2'] = data['temp'] ** 2
# new temp^2 variable
x = data[['temp^2', 'temp', 'constant']]
y = data['load']
# same code from before
model = sm.OLS(y, x)
out = model.fit()
data['pred'] = out.predict(x)
# plot.. same code
ax = data.plot.scatter('temp', 'load')
data.plot(ax=ax, x='temp', y='pred', color='r')
```

Classical statistics still has an answer

Temperature vs. demand, hourly (2015-2021)

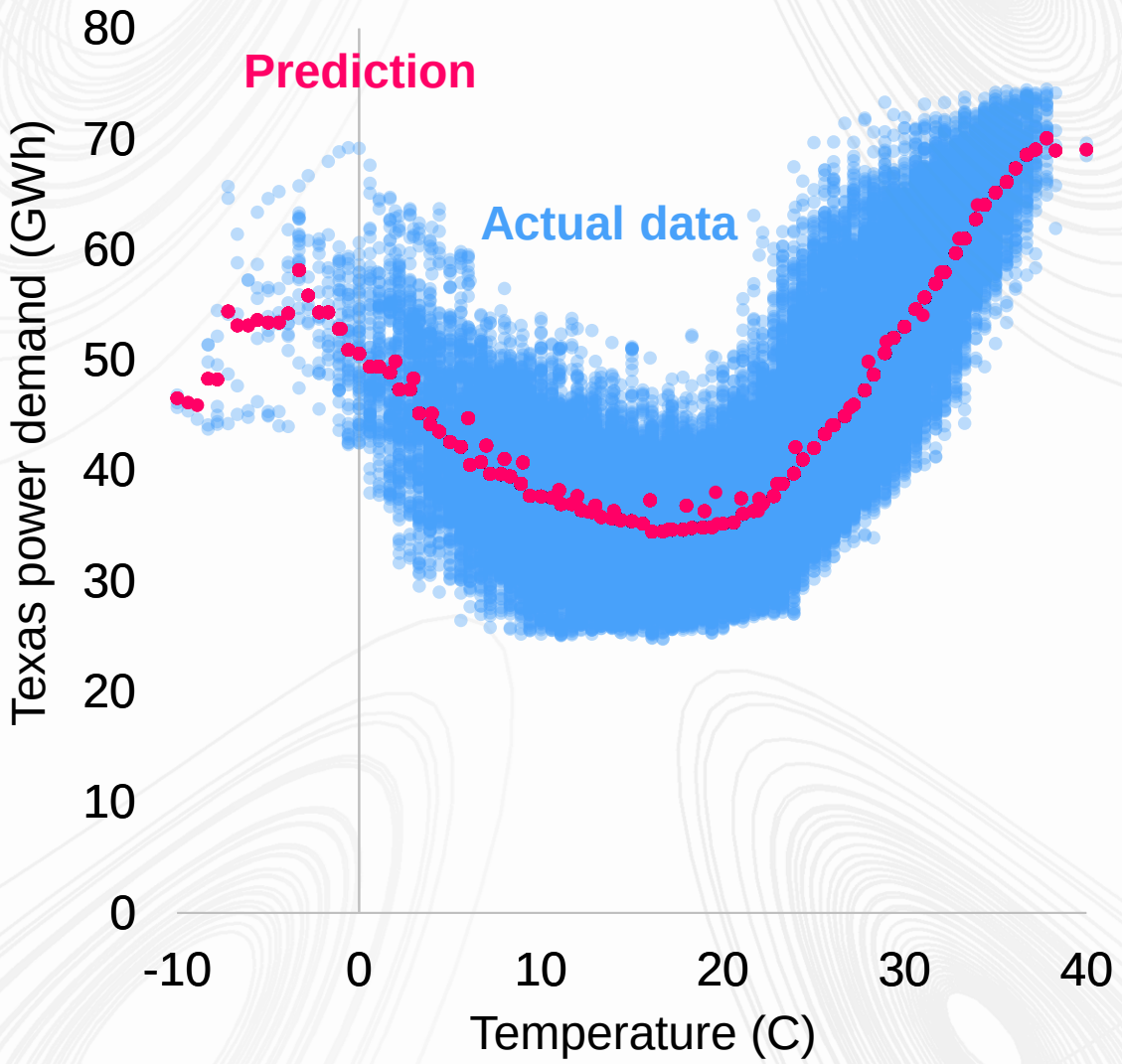


Polynomial regression in Python

```
# add polynomial
data['temp^2'] = data['temp'] ** 2
# new temp^2 variable
x = data[['temp^2', 'temp', 'constant']]
y = data['load']
# same code from before
model = sm.OLS(y, x)
out = model.fit()
data['pred'] = out.predict(x)
# plot.. same code
ax = data.plot.scatter('temp', 'load')
data.plot(ax=ax, x='temp', y='pred', color='r')
```

Let’s turn it up anyways

Temperature vs. demand, hourly (2015-2021)

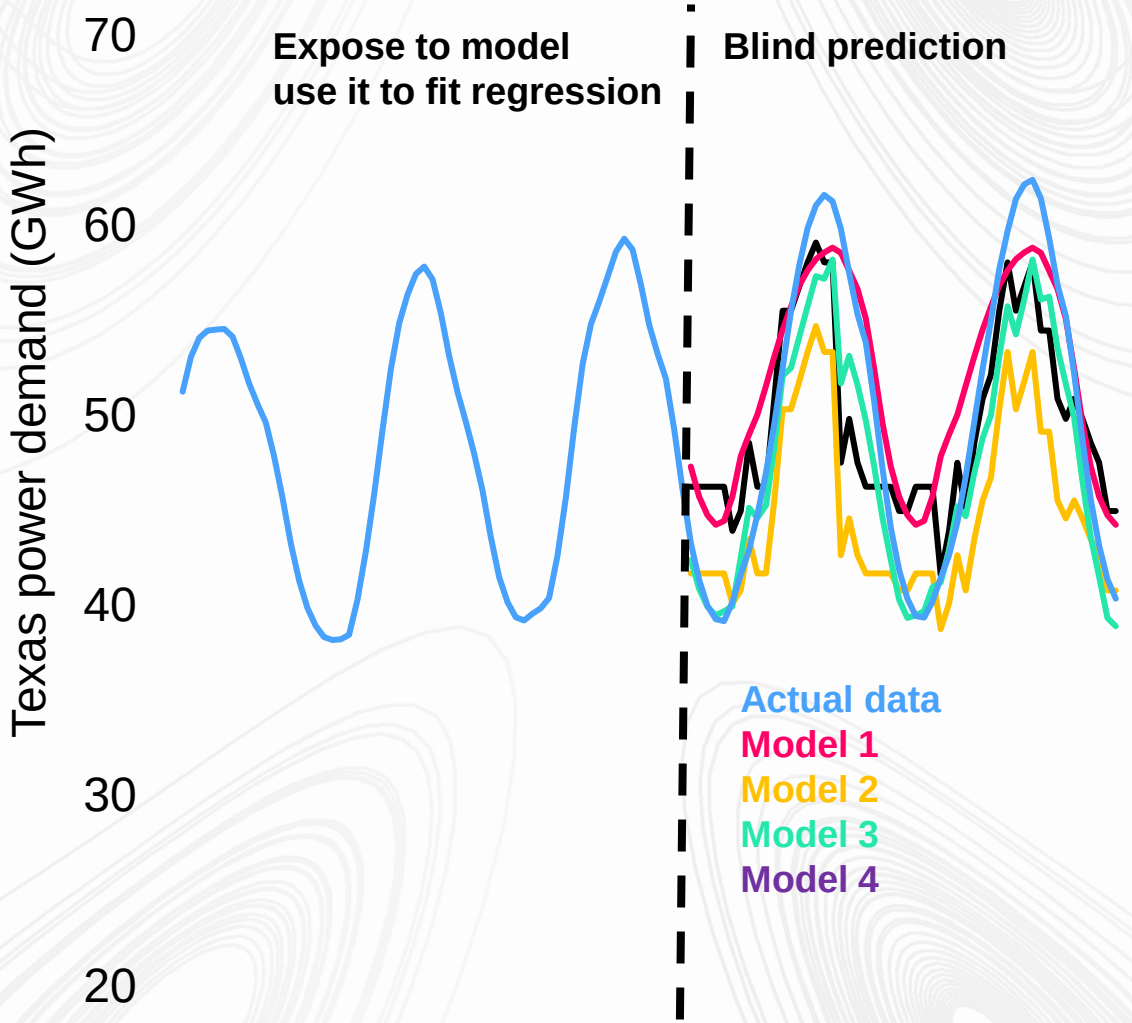


Using a machine learning regressor

```
from sklearn.ensemble import GradientBoostingRegressor
# fancy machine learning
model = GradientBoostingRegressor()
# no constant, no polynomial
x = data[['temp']]
y = data['load']
model = model.fit(x, y)
data['pred'] = model.predict(x)
# chart... same code
ax = data.plot.scatter('temp', 'load')
data.plot(ax=ax, x='temp', y='pred', color='r')
```

Backtest

Illustrative backtest



Train/test split

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
xgb = GradientBoostingRegressor()
ada = AdaBoostRegressor()
data['datetime'] = pd.to_datetime(data['datetime'])
# split data
train = data[data['datetime'] < '1/1/2019']
test = data[data['datetime'] > '1/1/2019']
x_train, y_train = train[['temp']], train[['load']]
x_test, y_test = test[['temp']], test[['load']]
# fit/predict
xgb = xgb.fit(x_train, y_train)
ada = ada.fit(x_train, y_train)
y_test['xgb'] = xgb.predict(x_test)
y_test['ada'] = ada.predict(x_test)
print(y_test.head(3))
'''Out:
      load           xgb           ada
0  36953.0  39659.922366  41642.391821
1  37114.0  39252.865195  41642.391821
2  37154.0  39659.922366  41642.391821
```

Thank you!