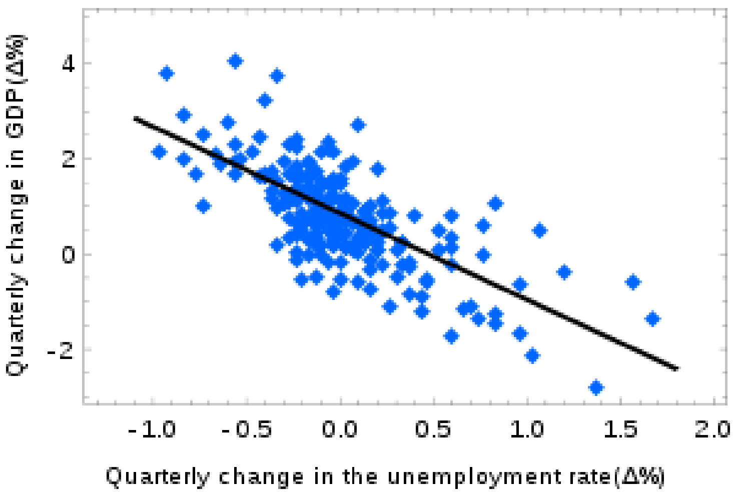
Regression: OLS
CS1D6: Introduction to data and statistics

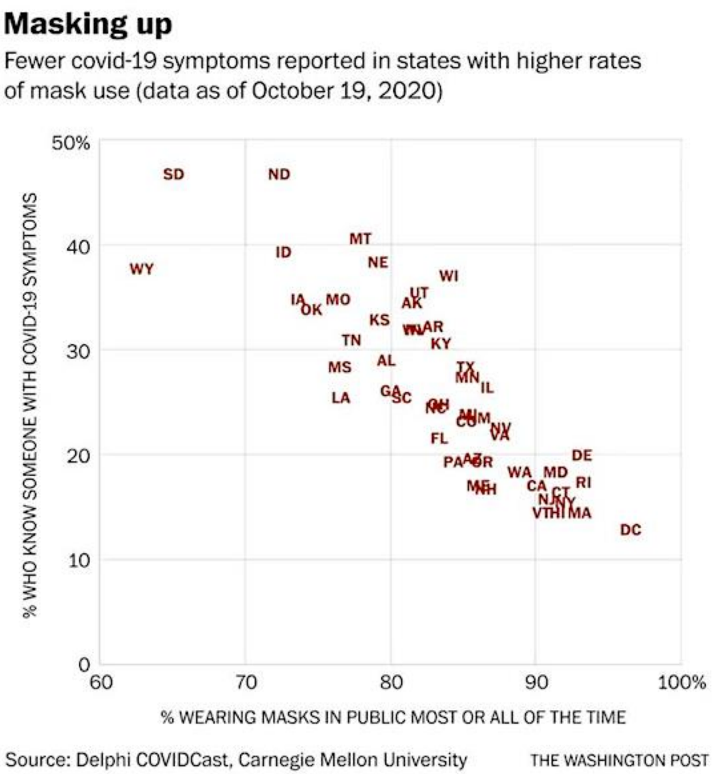
Dr. Fayyaz Minhas

Department of Computer Science
University of Warwick

The Problem

- Let's say you want to model the relationship between a scalar response and one or more explanatory variables
  - Independent variables
  - Dependent variable

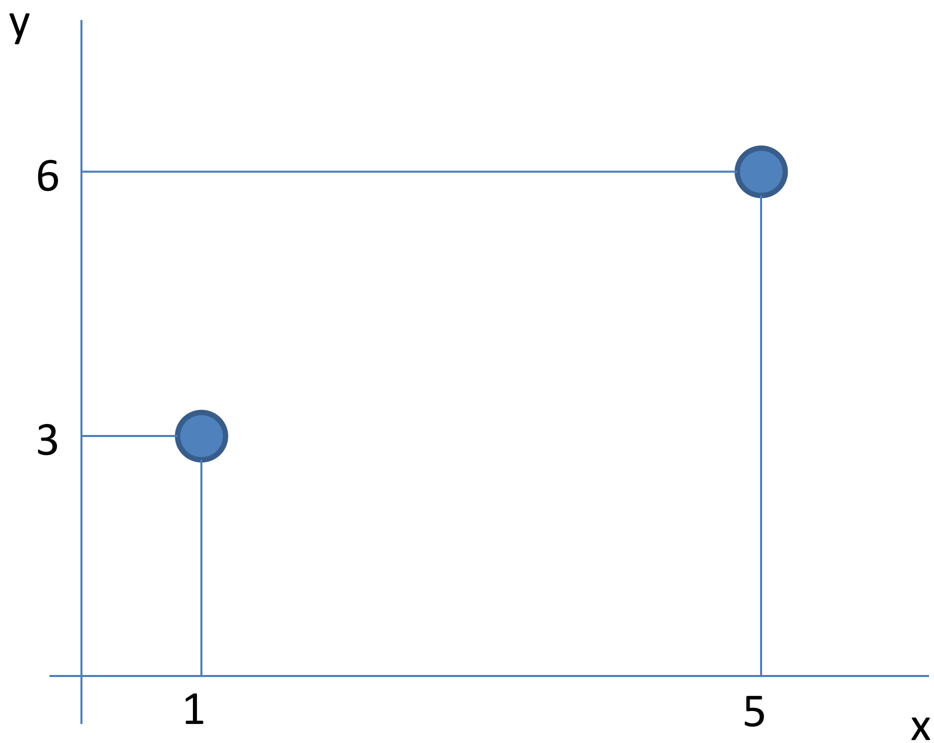




$$y = f(\boldsymbol{x}; \boldsymbol{w}) + \epsilon$$

Regression

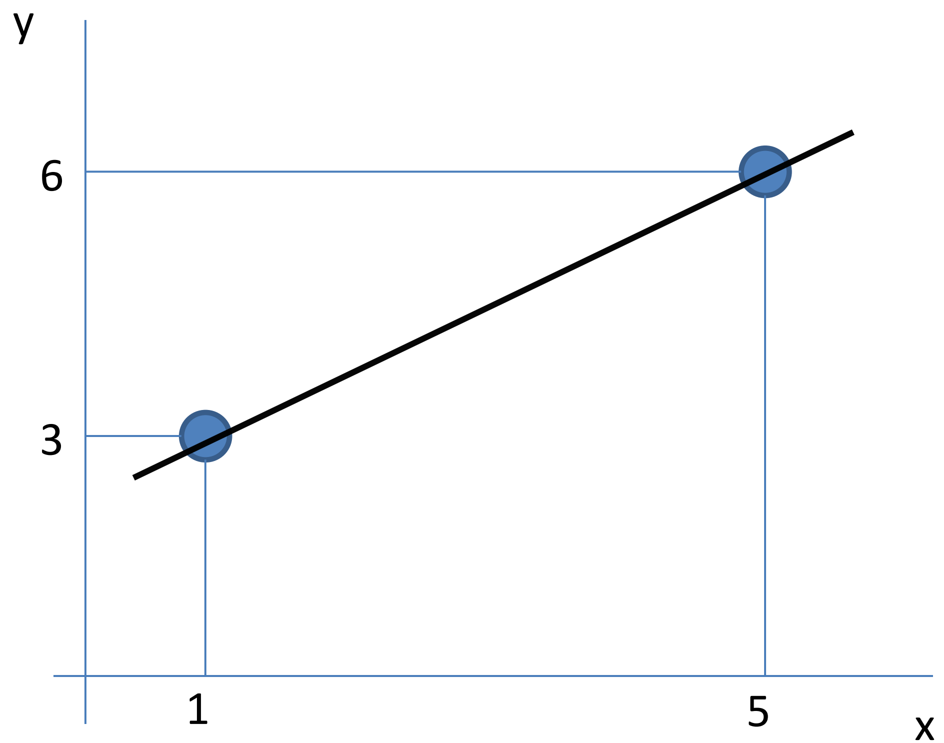
- Minimalistic Example



Can you write a formula for y in terms of x_1?

Regression

- Minimalistic Example



Can you write a formula for y in terms of x?

Solution

- The points are: (1,3) and (5,6)
- Using the two point form of a line
- $y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$
- $y = 3 + \frac{6-3}{5-1}(x - 1) = 3 + \frac{3}{4}x - \frac{3}{4} = \frac{3}{4}x + \frac{9}{4}$

https://mathworld.wolfram.com/Two-PointForm.html

Alternate solution

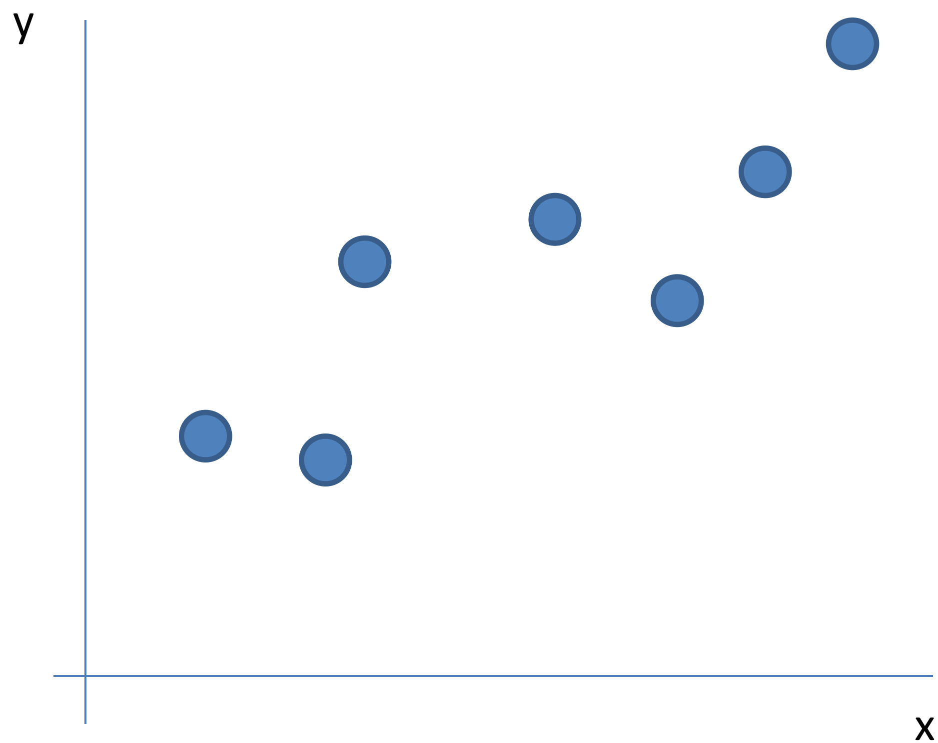
- $y_1 = mx_1 + c$
- $y_2 = mx_2 + c$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix}$$

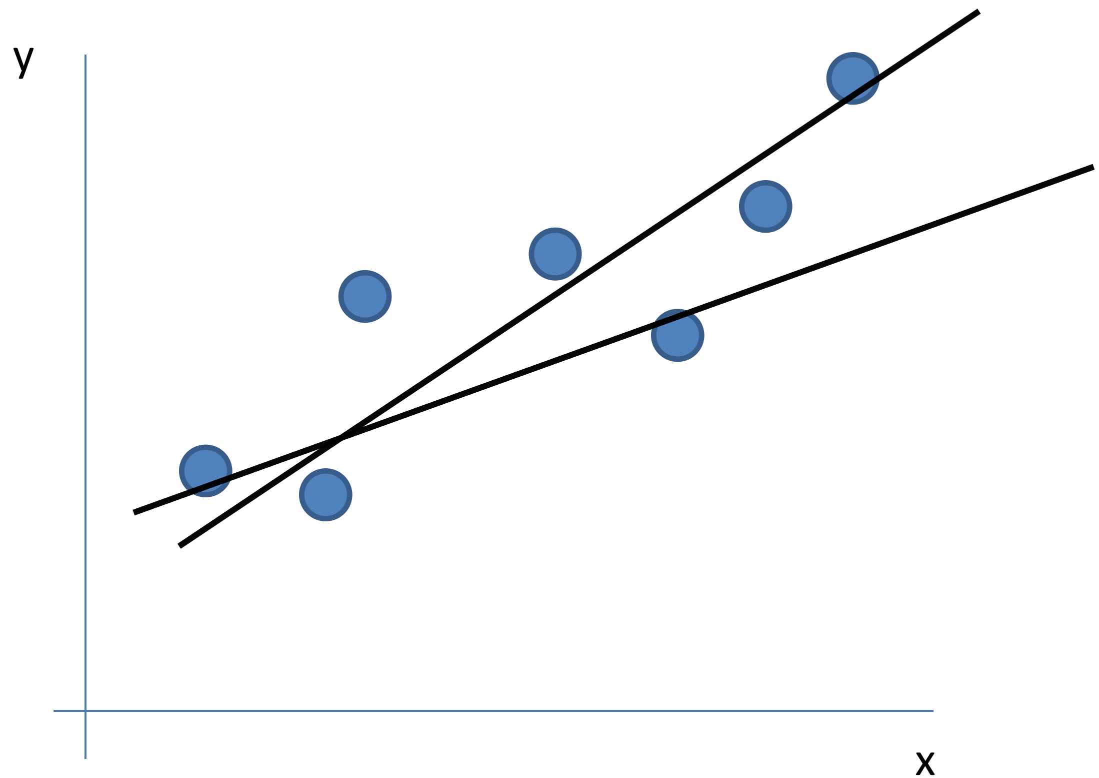
$$\begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{9}{4} \end{bmatrix}$$

```
import numpy as np
A = np.array([[1,1],[5,1]]);y = np.array([[3],[6]])
w = np.linalg.inv(A)@y
print(w)
```

But what if we have more points?



Which line is a better representation?



How do we find it?

- We can use “linear” regression
- Linear because our function f in $y = f(\boldsymbol{x}; \boldsymbol{w}) + \epsilon$ is linear in $\boldsymbol{x}$
- Linear function:

$$f(\boldsymbol{x}; \boldsymbol{w}) = w_1 x^{(1)} + w_1 x^{(2)} + b$$

Find the function means finding its parameters w_1, w_2 and b

Preliminaries

– Love dot products (and learn to spot them!)

$$ab + cd + ef = \begin{bmatrix} a & c & e \end{bmatrix} \begin{bmatrix} b \\ d \\ f \end{bmatrix} = \boldsymbol{p}^T \boldsymbol{q} = \boldsymbol{q}^T \boldsymbol{p} = \boldsymbol{q} \cdot \boldsymbol{p} \qquad \boldsymbol{q} = \begin{bmatrix} b \\ d \\ f \end{bmatrix}$$

$$a^2 + c^2 + e^2 = \boldsymbol{p}^T \boldsymbol{p} = \|\boldsymbol{p}\|^2$$

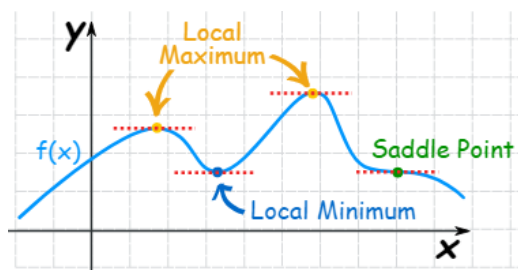
– Love matrix-vector products (and learn to spot them) $\boldsymbol{p} = \begin{bmatrix} a \\ c \\ e \end{bmatrix}$

$$ab + cd + ef = u$$
$$ag + ch + ek = v$$

$$\begin{bmatrix} b & d & f \\ g & h & k \end{bmatrix} \begin{bmatrix} a \\ c \\ e \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

– Love derivatives (and learn to solve them!)

- Allow us to find minima or maxima



REO for Ordinary Least Squares Linear Regression

- Representation

$$f(x;w) = w_1 x^{(1)} + w_2 x^{(2)} + b$$

$$\boldsymbol{w} = \begin{bmatrix} \mathrm{w}_1 \\ \mathrm{w}_2 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}$$

Alternatively, $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$

$$\boldsymbol{w} = \begin{bmatrix} \mathrm{w}_1 \\ \mathrm{w}_2 \\ b \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ 1 \end{bmatrix}$$

Or, without loss of generality, $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$

- In matrix form

$$\boldsymbol{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ b \end{bmatrix}$$

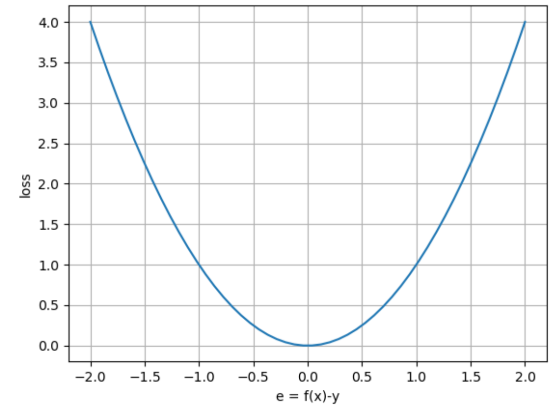
- $f(\boldsymbol{x_1}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_1} = \boldsymbol{x_1}^{\boldsymbol{T}}\boldsymbol{w}$
- $f(\boldsymbol{x_2}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_2} = \boldsymbol{x_2}^{\boldsymbol{T}}\boldsymbol{w}$
- …
- $f(\boldsymbol{x_N}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_N} = \boldsymbol{x_N}^{\boldsymbol{T}}\boldsymbol{w}$
- OR
- $\boldsymbol{F} = \boldsymbol{X}\boldsymbol{w}$

$$\boldsymbol{X}_{(N\times(d+1))} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(d)} & 1 \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(d)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \cdots & x_N^{(d)} & 1 \end{bmatrix}$$

Ordinary Least Squares Linear Regression

- Calculating Error
  - Let's define error for a prediction as (Actual Output – Target Output)^2

- $L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = \sum_{i=1}^{N}(f(\boldsymbol{x}_i;\mathbf{w}) - y_i)^2 = \sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 = \sum_{i=1}^{N}(e_i)^2$



- $e_1 = \boldsymbol{w}^T\boldsymbol{x}_1 - y_1 = \boldsymbol{x}_1{}^T\boldsymbol{w} - y_1$
- $e_2 = \boldsymbol{w}^T\boldsymbol{x}_2 - y_2 = \boldsymbol{x}_2{}^T\boldsymbol{w} - y_2$
- …
- $e_N = \boldsymbol{w}^T\boldsymbol{x}_N - y_N = \boldsymbol{x}_N{}^T\boldsymbol{w} - y_N$

- Or, in matrix form

- $\boldsymbol{e} = \boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$

$$\boldsymbol{y}_{(N\times 1)} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \boldsymbol{e}_{(N\times 1)} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}$$

- Note:

$$L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

Optimization

- Find **w** that minimize $L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$
- Or: $min_{\boldsymbol{w}} L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$
- Or: $\boldsymbol{w}^* = argmin_{\boldsymbol{w}} L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$
  - Differentiate $L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$ wrt $\boldsymbol{w}$ and substitute it to zero

$$\boldsymbol{L} = (\boldsymbol{Xw} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{Xw} - \boldsymbol{y}) = \left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}} - \boldsymbol{y}^{\boldsymbol{T}}\right)(\boldsymbol{Xw} - \boldsymbol{y})$$

$$= \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Xw} - \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} - \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{Xw} - \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}$$

$$\frac{\partial L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{2X^TXw} - \boldsymbol{2X^Ty} = \boldsymbol{0}$$

$$\boldsymbol{X^TXw} = \boldsymbol{X^Ty}$$

$$\boldsymbol{w} = \left(\boldsymbol{X^TX}\right)^{-\boldsymbol{1}}\boldsymbol{X^Ty}$$

$$\boldsymbol{w} = \boldsymbol{X^+y}$$

$$\frac{\partial \boldsymbol{w^TA}}{\partial \boldsymbol{w}} = \boldsymbol{A}$$

$$\frac{\partial \boldsymbol{Aw}}{\partial \boldsymbol{w}} = \boldsymbol{A^T}$$

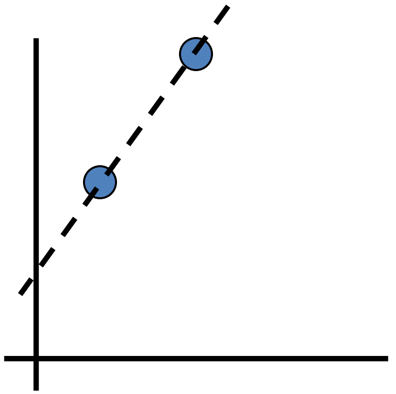
$$\frac{\partial \boldsymbol{w^TAw}}{\partial \boldsymbol{w}} = \boldsymbol{2A^Tw}$$

$$\boldsymbol{X^+} = \left(\boldsymbol{X^TX}\right)^{-\boldsymbol{1}}\boldsymbol{X^T}$$

Pseudo-inverse

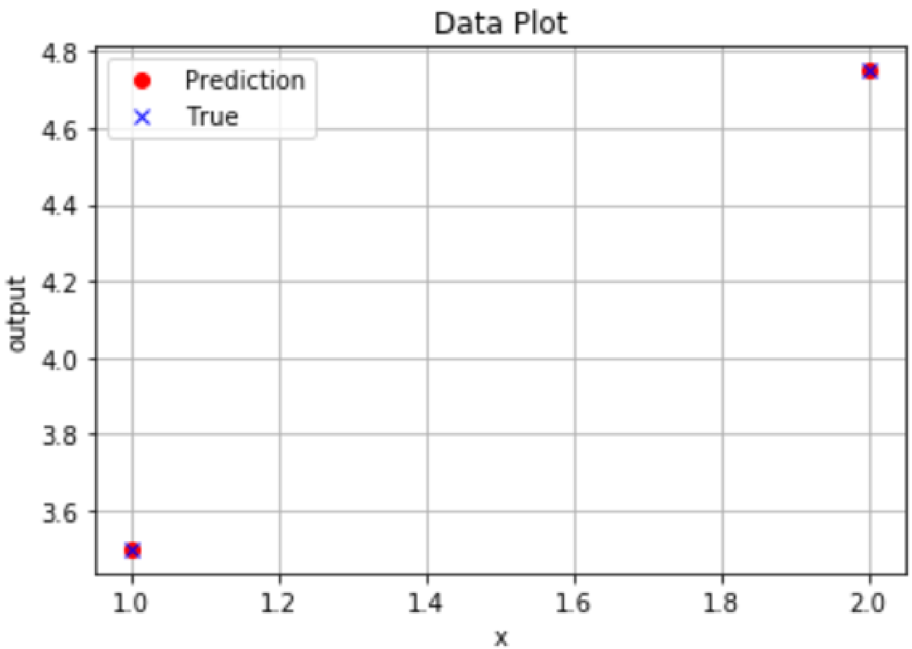
Linear Regression: Simple Example

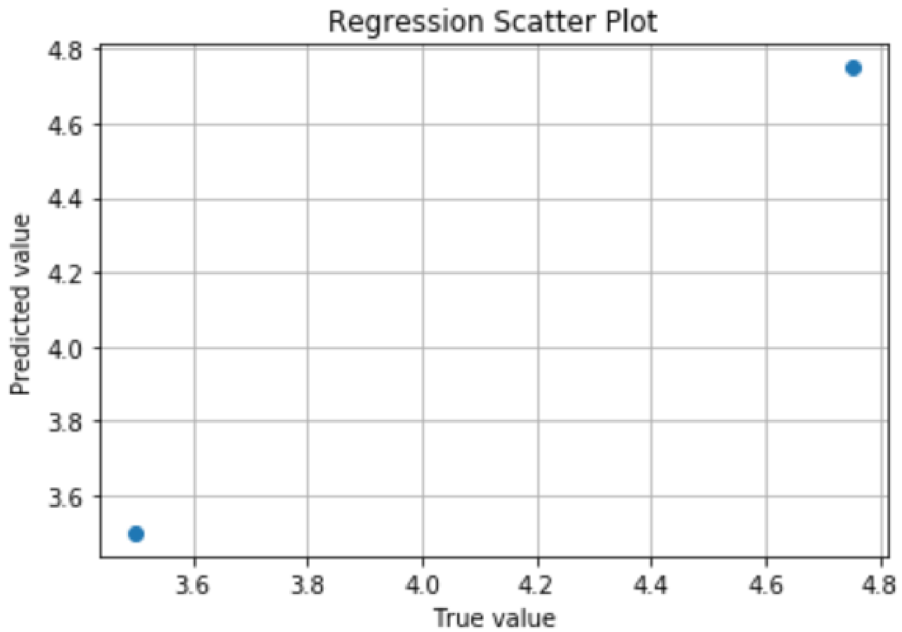
- Example
  - $\boldsymbol{X} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} 3.5 \\ 4.75 \end{bmatrix}$
  - Thus: $\boldsymbol{w} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} = \begin{bmatrix} 1.25 \\ 2.25 \end{bmatrix}$
  - Now
    - $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\mathbf{1})} = \begin{bmatrix} 1.25 \\ 2.25 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 3.5$
    - $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\mathbf{2})} = \begin{bmatrix} 1.25 \\ 2.25 \end{bmatrix}^T \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 4.75$



x	y
1	3.5
2	4.75

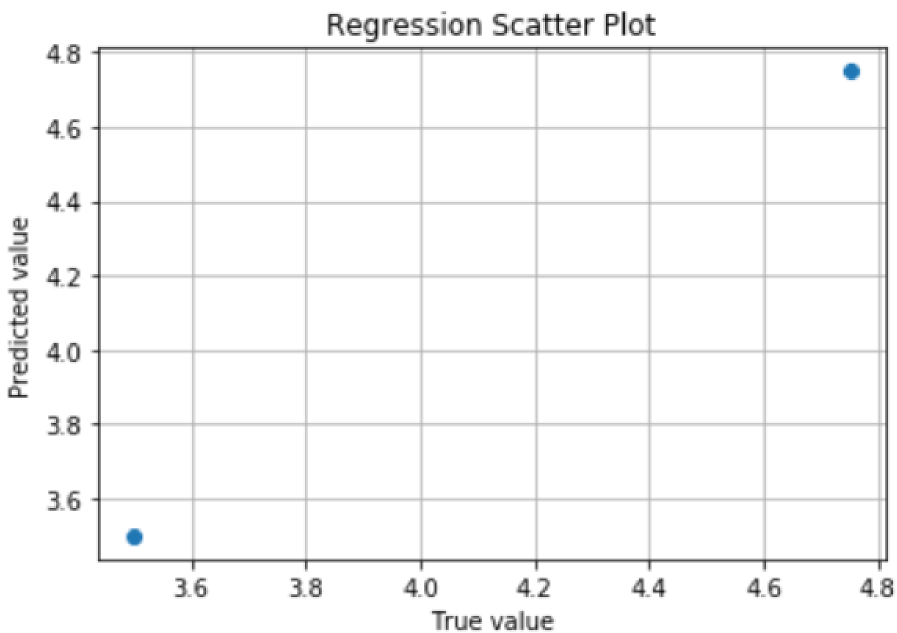
Coding






```
import numpy as np
import matplotlib.pyplot as plt
X0 = np.array([[1],[2]])
y =np.array([3.5,4.75])
X = np.hstack((X0,np.ones((X.shape[0],1)))) #append 1 to each example
w = np.linalg.pinv(X)@y
f = X@w
e = f-y
L = e@e
plt.figure();plt.plot(X0,f,'ro');plt.plot(X0,y,'bx');plt.grid();plt.xlabel('x');plt.ylabel('output');plt.legend(['Prediction','True']);
plt.title('Data Plot')
plt.figure();plt.plot(y,f,'o');plt.grid();plt.xlabel('True value');plt.ylabel('Predicted value');plt.title('Regression Scatter Plot')
```

Using Sk-learn




```
from sklearn.linear_model import LinearRegression
regr = LinearRegression(fit_intercept = False).fit(X, y)
f = regr.predict(X)
print('Weights:',regr.coef_)

plt.figure();plt.plot(y,f,'o');plt.grid();plt.xlabel('True value');plt.ylabel('Predicted value');plt.title('Regression Scatter Plot')

# No need to append 1 to feature vector using below
from sklearn.linear_model import LinearRegression
regr = LinearRegression(fit_intercept = True).fit(X0, y)
f = regr.predict(X0)
plt.figure();plt.plot(y,f,'o');plt.grid();plt.xlabel('True value');plt.ylabel('Predicted value');plt.title('Regression Scatter Plot')
```

How to measure how good the fit is?

- Correlation Coefficient
- Mean Squared Error
- Mean Absolute Error
- Root Mean Squared Error
- Coefficient of Determination (R2)

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} (y_i - \hat{y}_i)^2$$

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} |y_i - \hat{y}_i|$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

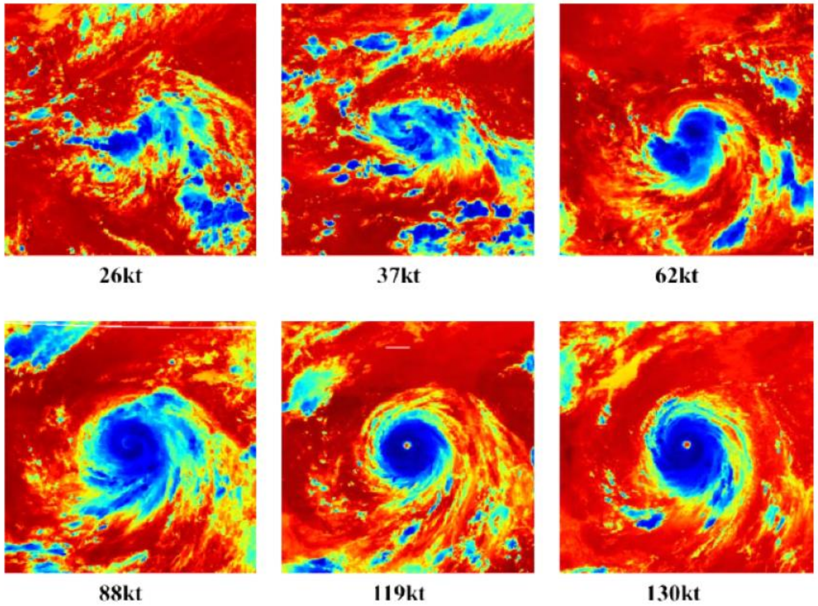
https://setosa.io/ev/ordinary-least-squares-regression/

Practical Application

Hurricane Intensity Estimation

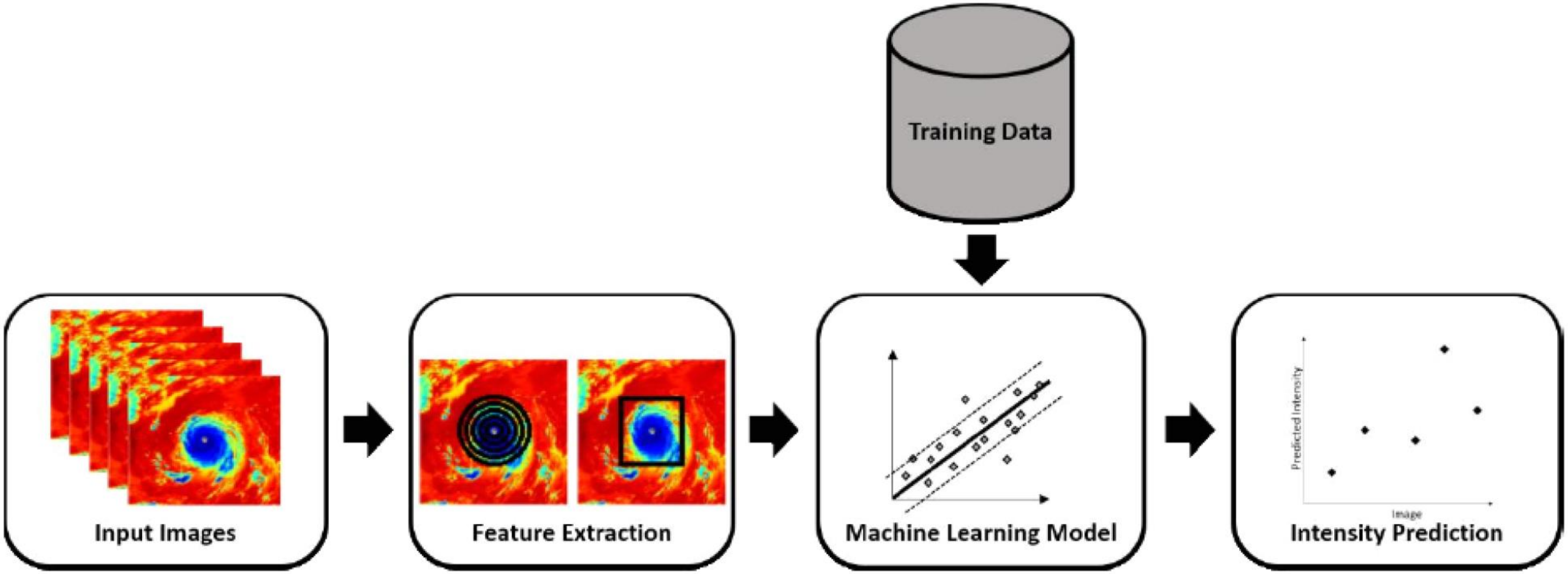
Input: Infrared Satellite Images of Hurricanes

Output: Maximum Sustained Windspeed in knots



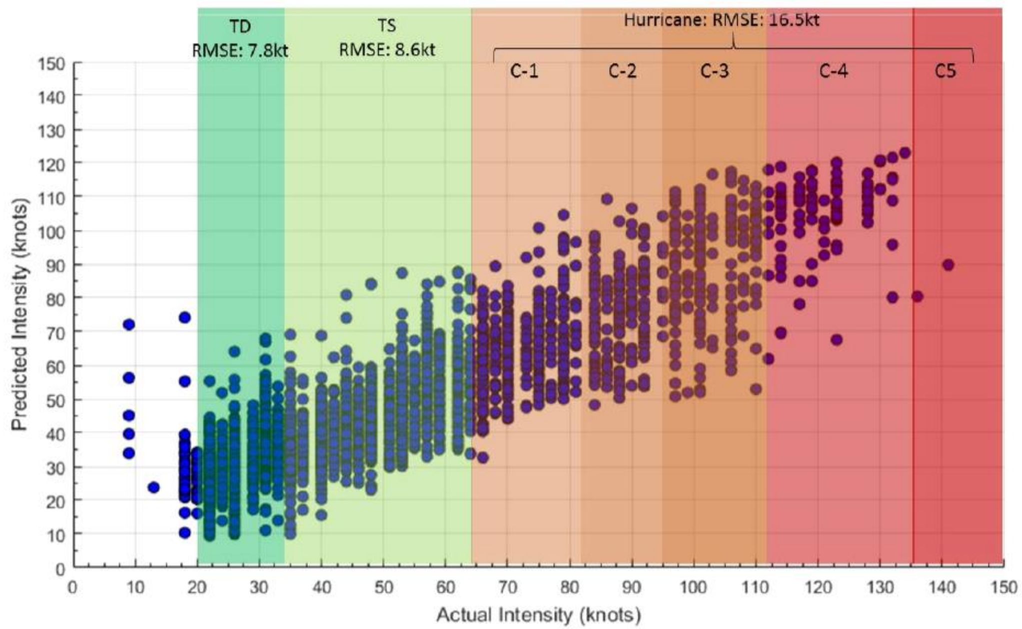
https://en.wikipedia.org/wiki/Hurricane_hunters

PHURIE ML Pipeline

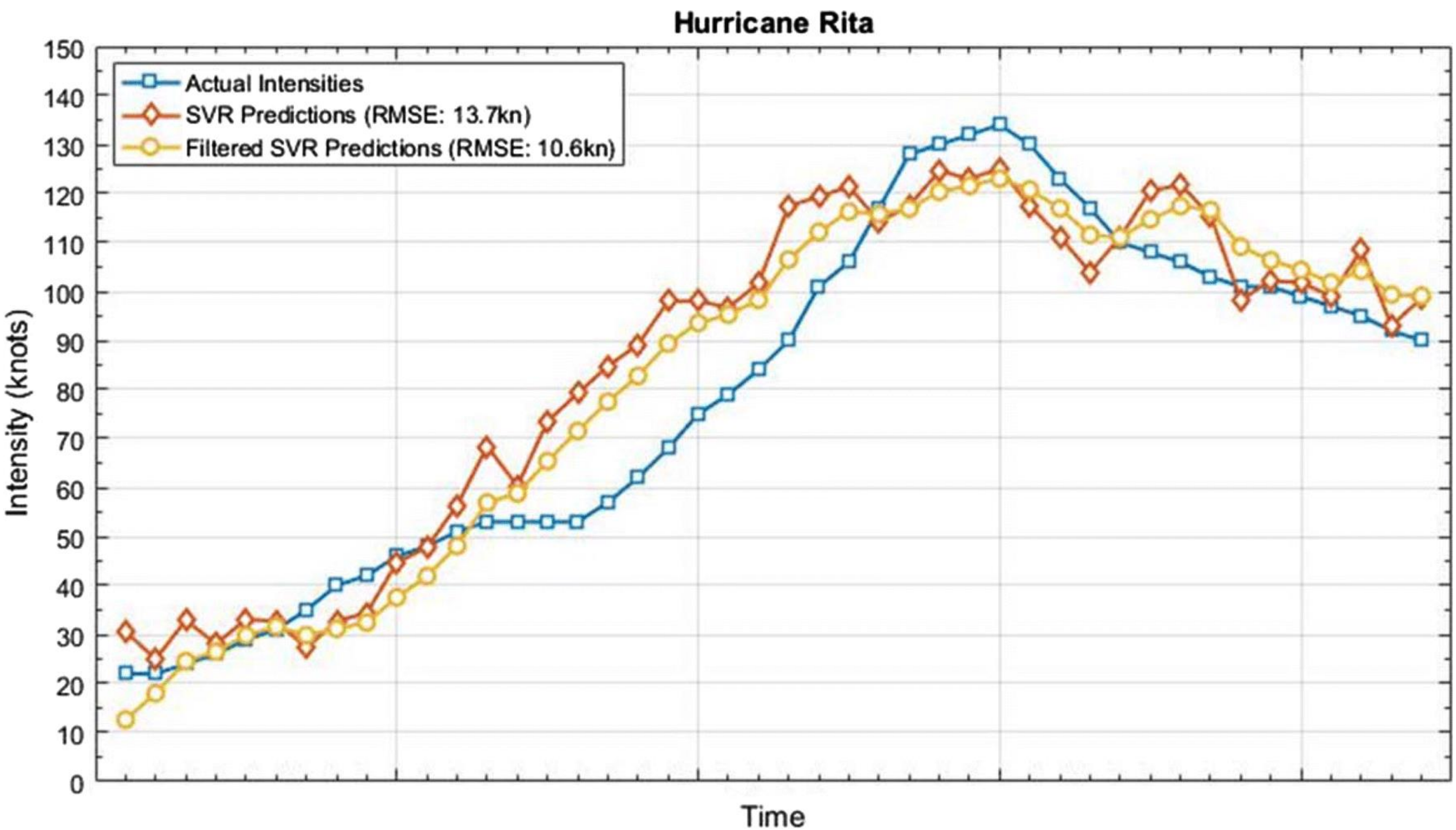


PHURIE: Hurricane Intensity Estimation from Infrared Satellite Imagery using Machine Learning, Amina Asif, Muhammad Dawood, Bismillah Jan, Javaid Khurshid, Mark DeMaria, and Fayyaz ul Amir Afsar Minhas, in Neural Computing and Applications, DOI: http://dx.doi.org/10.1007/s00521-018-3874-6, 2018. (Paper)

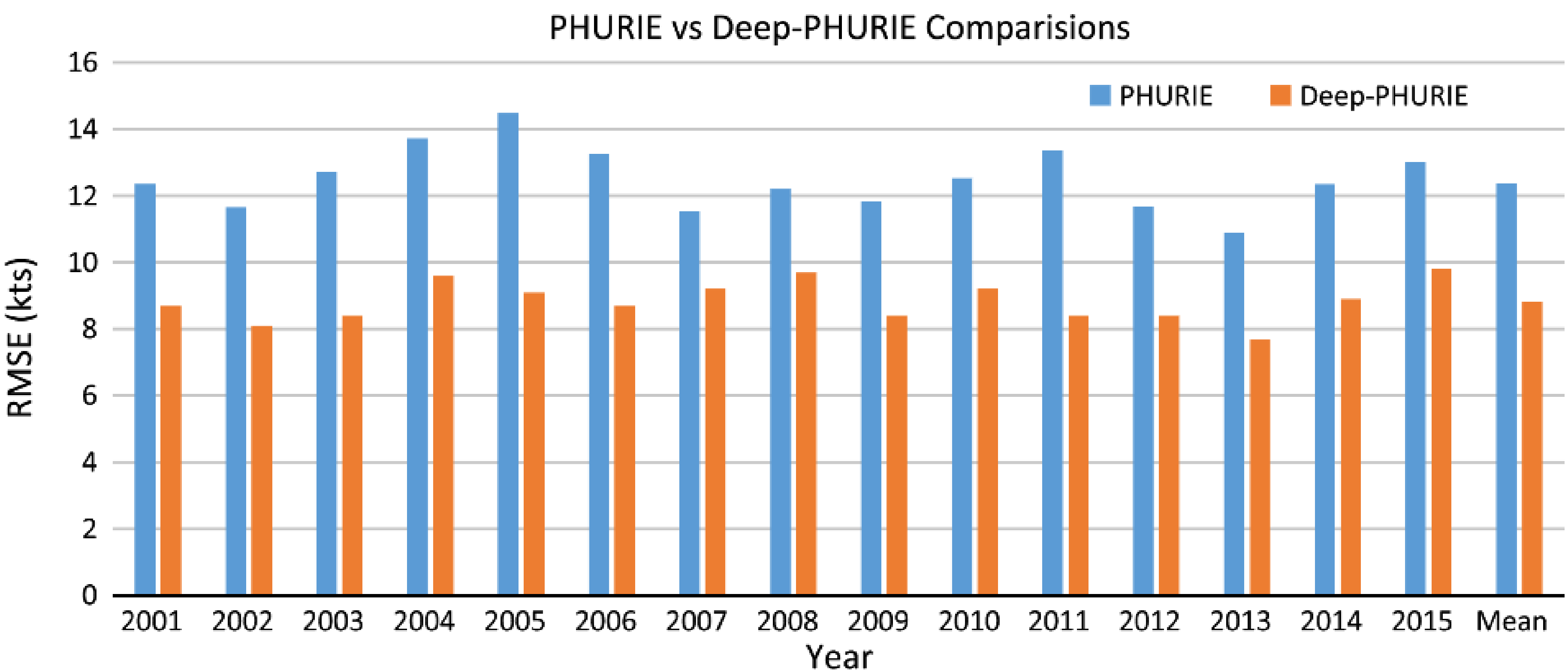
Practical Application



Method	Mean RMSE (kt)	Mean RMSE after smoothing
PHURIE: SVR	**11.2**	**9.5**
PHURIE: OLS	12.8	10.5
PHURIE: BPNN	12.0	10.1
PHURIE: XGBoost	11.3	9.8
Baseline predictor (mean)	24.3	–


Hurricane Rita
Actual Intensities
SVR Predictions (RMSE: 13.7kn)
Filtered SVR Predictions (RMSE: 10.6kn)
Intensity (knots)
Time
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0

Extension: Deep PHURIE



Deep-PHURIE: Deep Learning based Hurricane Intensity Estimation from Infrared Satellite Imagery, M. Dawood, A. Asif and Fayyaz Minhas, in Neural Computing and Applications. pp. DOI: 10.1007/s00521-019-04410-7, July 2019.

Nonlinear Regression and Generalized Linear Models

- OLS restricted to linear
- What if we want to fit a non-linear function form?
  - For example, how do we fit a polynomial to a single variable?

$$f(\boldsymbol{x};\boldsymbol{w}) = w_1 x + w_2 x^2 + b$$

  - Simply add another feature which is the square of the original one
  - This is called polynomial regression

https://scikit-learn.org/stable/modules/linear_model.html

End of Lecture

We want to make a machine that will be proud of us.

- Danny Hillis