Scikit-Learn Overview

Scikit-Learn

- We’ve seen NumPy had some built in capabilities for simple linear regression, but when it comes to more complex models, we’ll need **Scikit-Learn**!
- Before we jump straight into machine learning with **Scikit-Learn** and Python, let’s understand the philosophy behind **sklearn.**

Scikit-Learn

- Scikit-learn is a library containing many machine learning algorithms.
- It utilizes a generalized “estimator API” framework to calling the models.
- This means the way algorithms are imported, fitted, and used is uniform across all algorithms.

Scikit-Learn

- This allows users to easily swap algorithms in and out and test various approaches.
- Important Note:
    - *This uniform framework also means users can easily apply almost any algorithm effectively without truly understanding what the algorithm is doing!*

Scikit-Learn

- Scikit-learn also comes with many convenience tools, including train test split functions, cross validation tools, and a variety of reporting metric functions.
- This leaves Scikit-Learn as a “one-stop shop” for many of our machine learning needs.

Scikit-Learn

- Philosophy of Scikit-Learn
    - Scikit-Learn’s approach to model building focuses on **applying models** and **performance metrics.**
    - This is a more pragmatic industry style approach rather than an academic approach of describing the model and its parameters.

Scikit-Learn

- Philosophy of Scikit-Learn
  - Academic users used to **R** style reporting may also want to explore the **statsmodels** python library if interested in more statistical description of models such as significance levels.

Scikit-Learn

- Let’s quickly review the framework of Scikit-Learn for the **supervised** machine learning process.
- We will quickly see how the code directly relates to the process theory!

Supervised Machine Learning Process

- Recall that we will perform a Train | Test split for supervised learning.

	Area m^2	Bedrooms	Bathrooms	Price
TRAIN	200	3	2	$500,000
	190	2	1	$450,000
	230	3	3	$650,000
TEST	180	1	1	$400,000
	210	2	2	$550,000

Supervised Machine Learning Process

- Also recall there are 4 main components after a Train | Test split:

	Area m^2	Bedrooms	Bathrooms	Price	
X TRAIN	200	3	2	$500,000	Y TRAIN
	190	2	1	$450,000	
	230	3	3	$650,000	
X TEST	180	1	1	$400,000	Y TEST
	210	2	2	$550,000	

Supervised Machine Learning Process

- Scikit-Learn easily does this split (as well as more advanced cross-validation)

TRAIN

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000

TEST

Area m^2	Bedrooms	Bathrooms	Price
180	1	1	$400,000
210	2	2	$550,000

Scikit-Learn

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Supervised Machine Learning Process

- Also recall that we want to compare predictions to the y test labels.

Predictions	Area m^2	Bedrooms	Bathrooms	Price
\$410,000	180	1	1	\$400,000
\$540,000	210	2	2	\$550,000

Scikit-Learn


```
from sklearn.model_family import ModelAlgo
```

Scikit-Learn

```python
from sklearn.model_family import ModelAlgo
mymodel = ModelAlgo(param1,param2)
```

Scikit-Learn

```
from sklearn.model_family import ModelAlgo
mymodel = ModelAlgo(param1,param2)
mymodel.fit(X_train,y_train)
```

Scikit-Learn

```
from sklearn.model_family import ModelAlgo
mymodel = ModelAlgo(param1,param2)
mymodel.fit(X_train,y_train)
predictions = mymodel.predict(X_test)
```

Scikit-Learn

```
from sklearn.model_family import ModelAlgo
mymodel = ModelAlgo(param1,param2)
mymodel.fit(X_train,y_train)
predictions = mymodel.predict(X_test)

from sklearn.metrics import error_metric
```

Scikit-Learn

```
from sklearn.model_family import ModelAlgo
mymodel = ModelAlgo(param1,param2)
mymodel.fit(X_train,y_train)
predictions = mymodel.predict(X_test)

from sklearn.metrics import error_metric
performance = error_metric(y_test,predictions)
```

Scikit-Learn

- This framework will be similar for any supervised machine learning algorithm.
- Let’s begin exploring it further with Linear Regression!

Linear Regression with Scikit-Learn

Part One:
Data Setup and Model Training

Linear Regression

- Previously, we explored "*Is there a relationship between **total advertising spend** and **sales**? "*
- Now we want to expand this to "*What is the relationship between **each advertising channel (TV,Radio,Newspaper)** and **sales?**"*

01-Linear-Regression-with-Scitkit-Learn[LEC4].ipynb

Performance Evaluation

Regression Metrics

Evaluating Regression

- Now that we have a fitted model that can perform predictions based on features, how do we decide if those predictions are any good?
- Fortunately we have the known test labels to compare our results to.

Evaluating Regression

- Let’s take a moment now to discuss evaluating Regression Models
- Regression is a task when a model attempts to predict continuous values (unlike categorical values, which is classification)

Evaluating Regression

- For example, attempting to predict the price of a house given its features is a **regression task.**
- Attempting to predict the country a house is in given its features would be a classification task.

Evaluating Regression

- You may have heard of some evaluation metrics like accuracy or recall.
- These sort of metrics aren’t useful for regression problems, we need metrics designed for **continuous** values!

Evaluating Regression

- Let's discuss some of the most common evaluation metrics for regression:
    - Mean Absolute Error
    - Mean Squared Error
    - Root Mean Square Error

Evaluating Regression

- The metrics shown here apply to any regression task, not just Linear Regression!

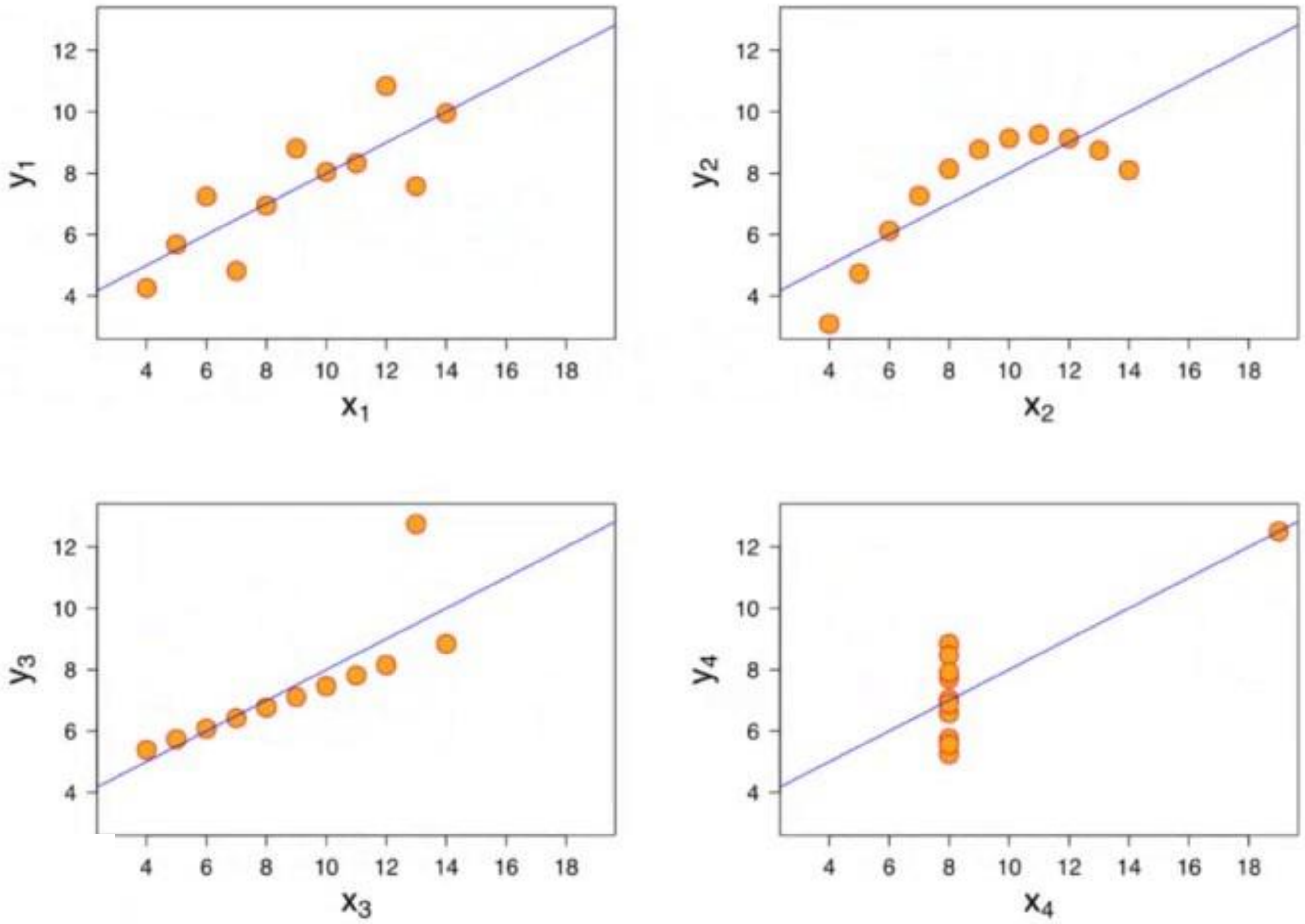
Evaluating Regression

- Mean Absolute Error (MAE)
  - This is the mean of the absolute value of errors.
  - Easy to understand

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

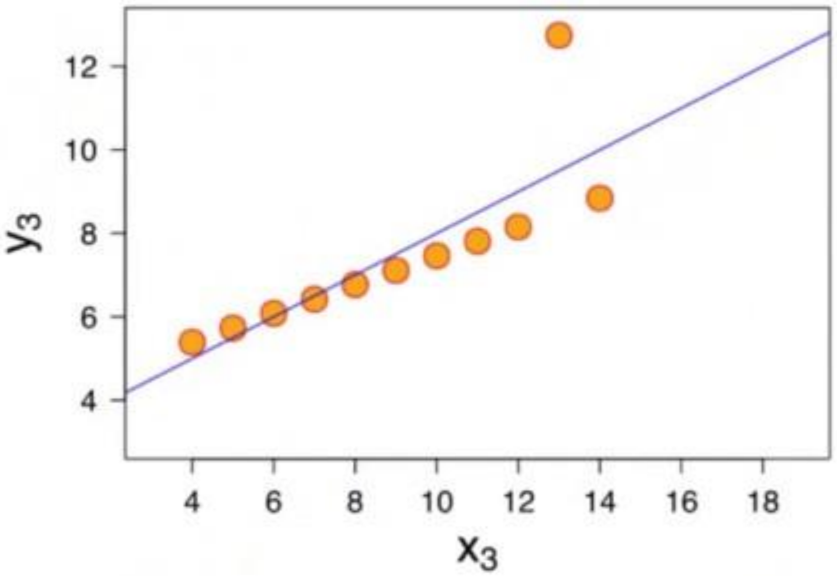
Evaluating Regression

- MAE won't punish large errors however.



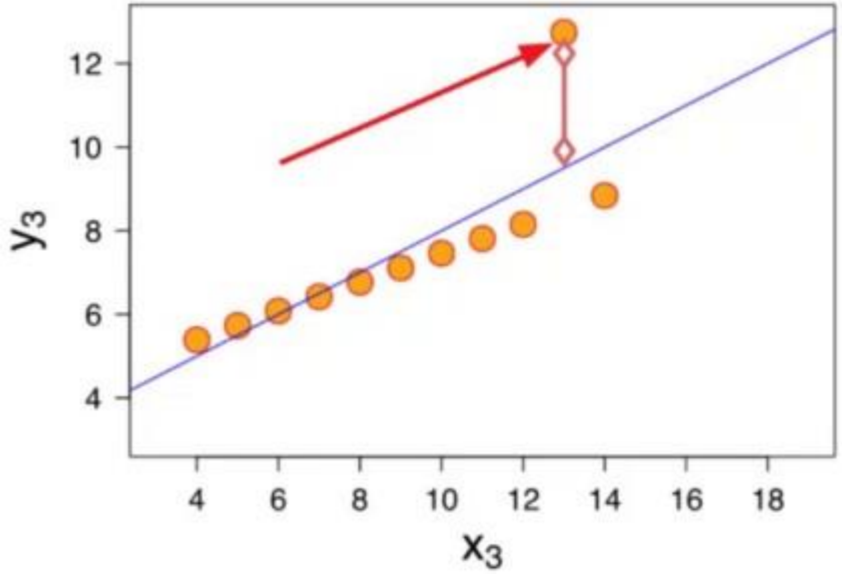
Evaluating Regression

- MAE won't punish large errors however.



Evaluating Regression

- We want our error metrics to account for these!



Evaluating Regression

- Mean Squared Error (MSE)
  - Larger errors are “punished” more than with MAE, making MSE more popular.

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

Evaluating Regression

- Mean Squared Error (MSE)
  - Issue with MSE:
    - Different units than y.
    - It reports units of y squared!

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

Evaluating Regression

- Root Mean Square Error (RMSE)
  - This is the root of the mean of the squared errors.
  - Most popular (has same units as y)

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

Machine Learning

- Most common question from students:
  - *"What is a good value for RMSE?"*
- Context is everything!
- A RMSE of $10 is fantastic for predicting the price of a house, but horrible for predicting the price of a candy bar!

Machine Learning

- Compare your error metric to the average value of the label in your data set to try to get an intuition of its overall performance.
- Domain knowledge also plays an important role here!

Machine Learning

- Context of importance is also necessary to consider.
  - We may create a model to predict how much medication to give, in which case small fluctuations in RMSE may actually be very significant.

Machine Learning

- Context of importance is also necessary to consider.
  - If we create a model to try to improve on existing human performance, we would need some baseline RMSE to compare to.

Evaluating Regression

- Let's quickly jump back to the notebook and calculate these metrics with SciKit-Learn!

01-Linear-Regression-with-Scitkit-Learn[LEC4].ipynb

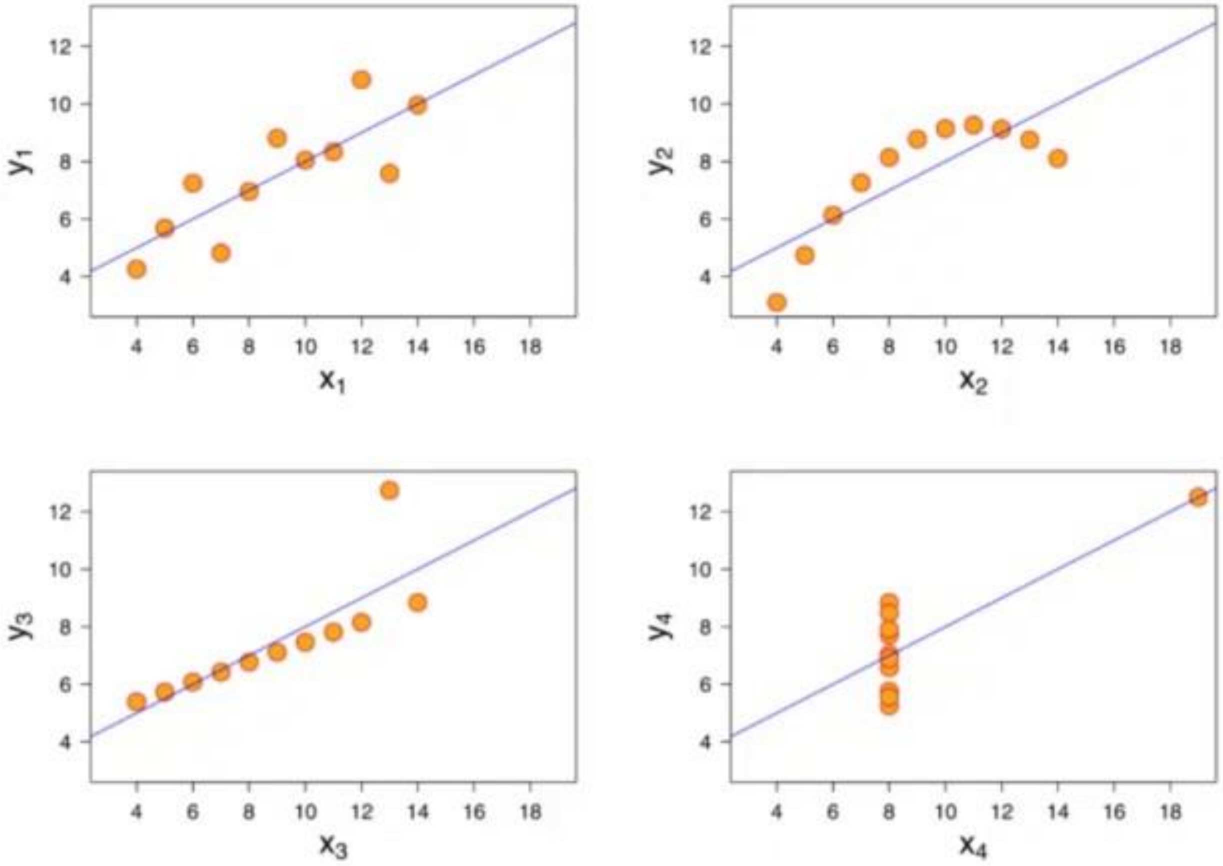
Evaluating Residuals

Linear Regression

- Often for Linear Regression it is a good idea to separately evaluate residuals ($y-\hat{y}$) and not just calculate performance metrics (e.g. RMSE).
- Let's explore why this is important...

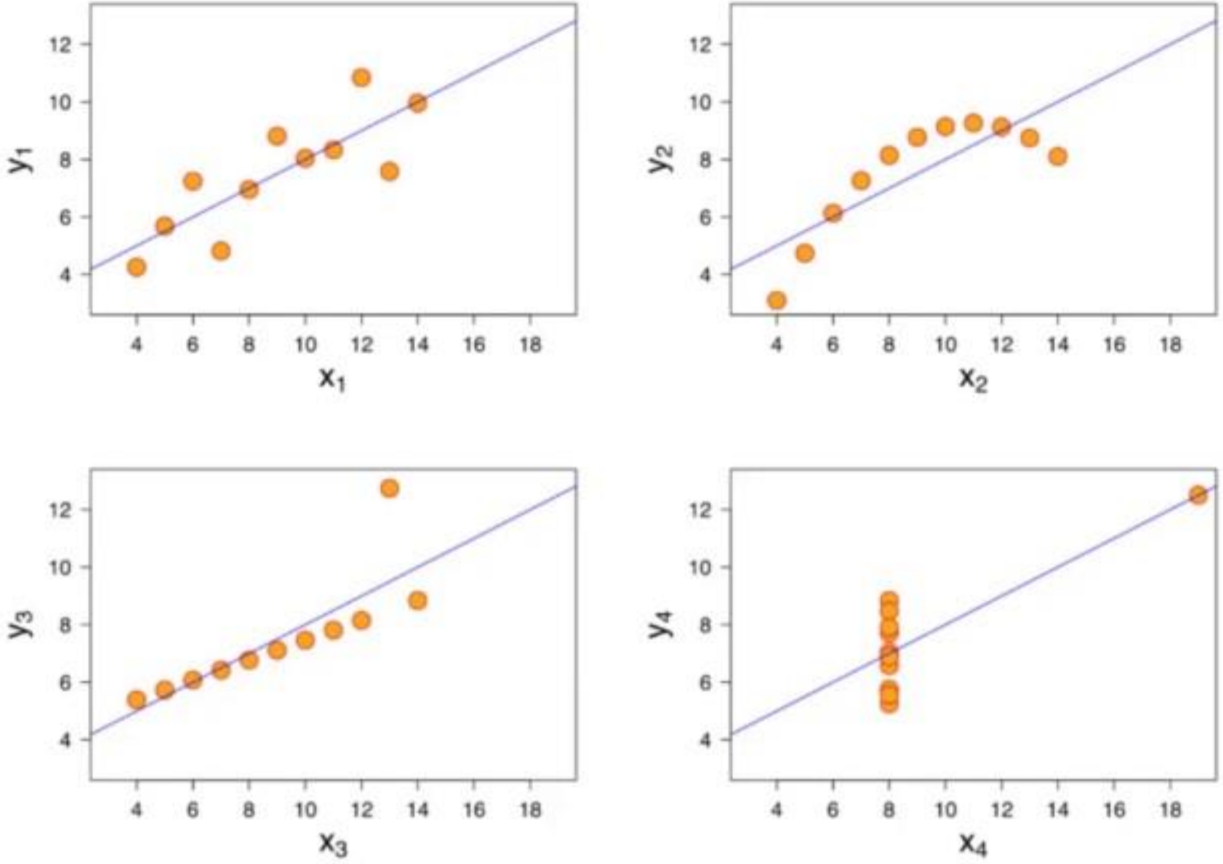
Linear Regression

- Recall Anscombe's quartet:



Linear Regression

- Clearly Linear Regression is not suitable!

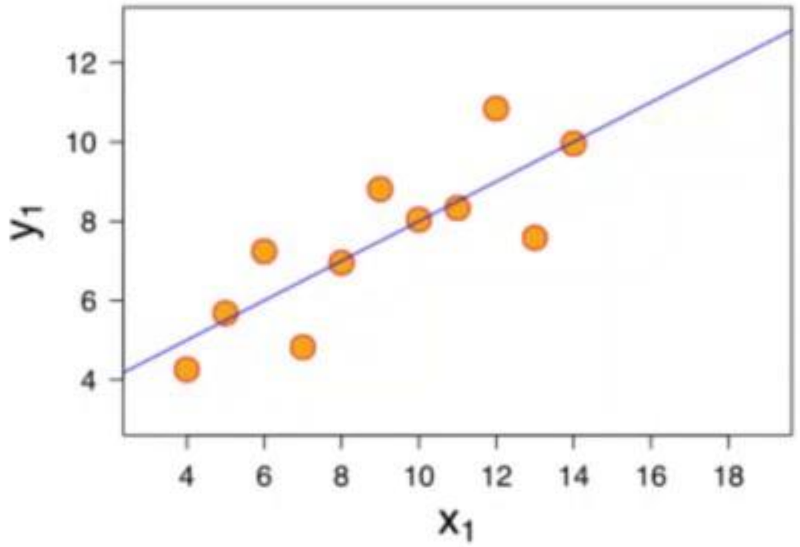


Linear Regression

- But how can we tell if we're dealing with more than one x feature?
- We can not see this discrepancy of fit visually if we have multiple features!

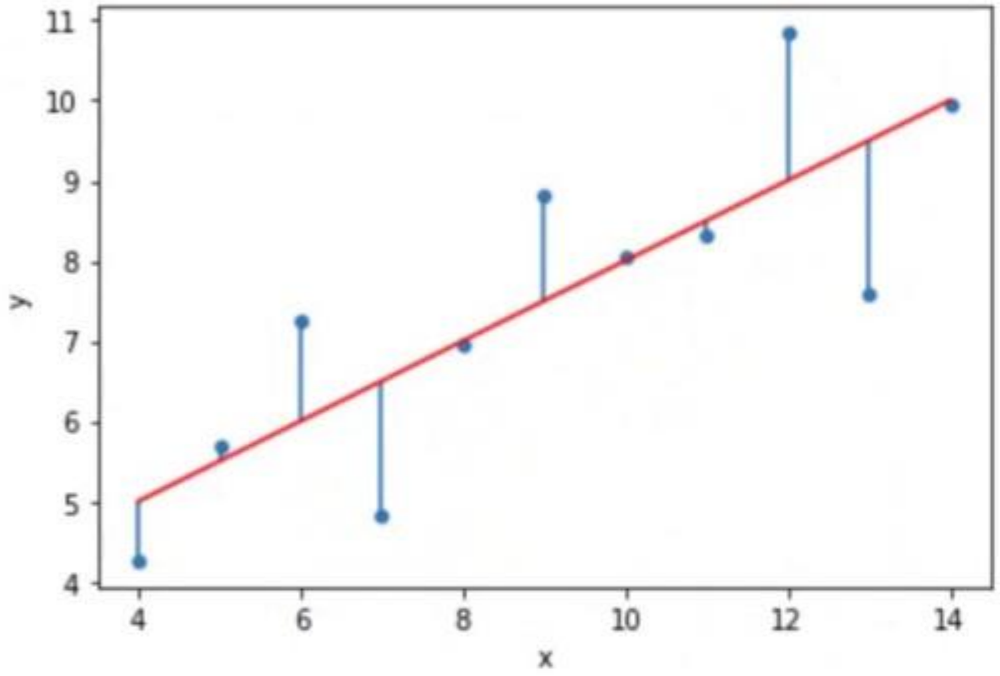
Linear Regression

- What we could do is plot residual error against true y values.
- Consider an appropriate data set:



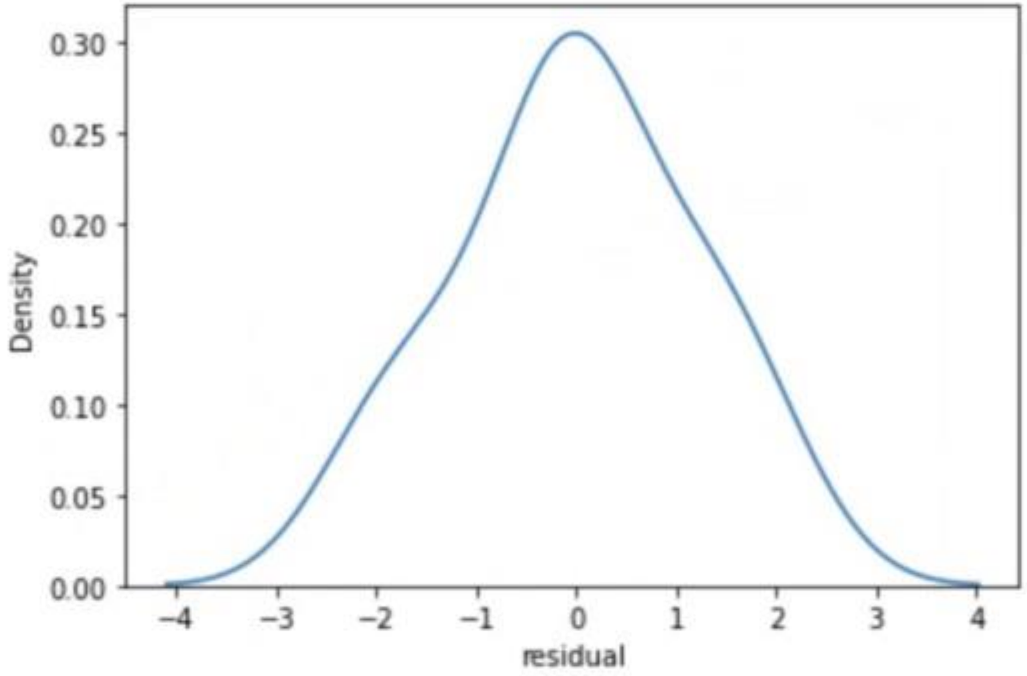
Linear Regression

- The residual errors should be random and close to a normal distribution.



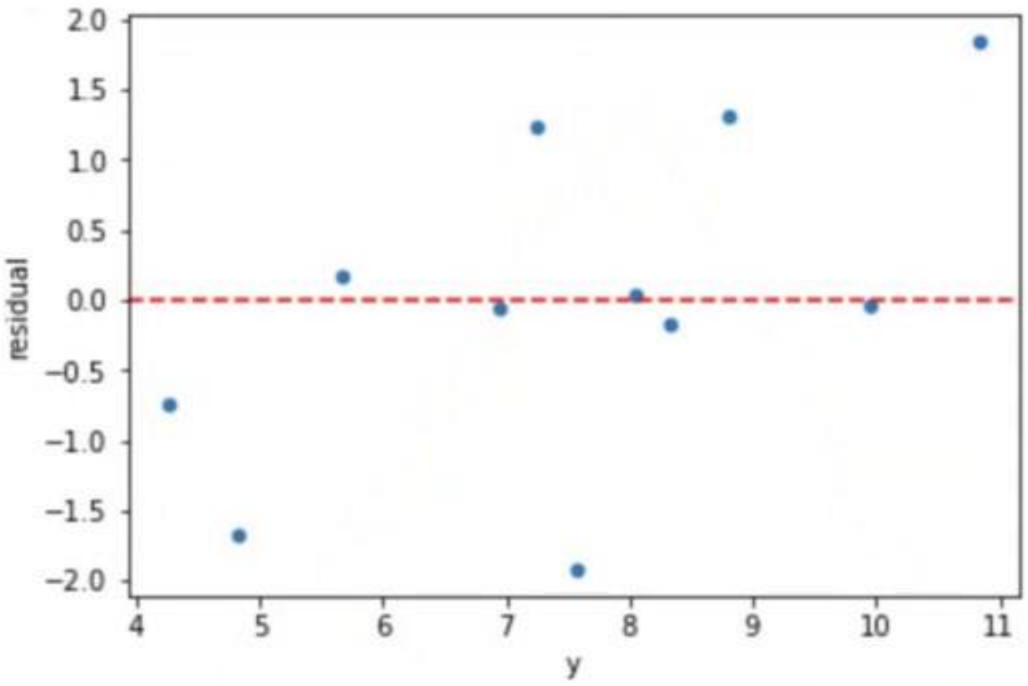
Linear Regression

- The residual errors should be random and close to a normal distribution.



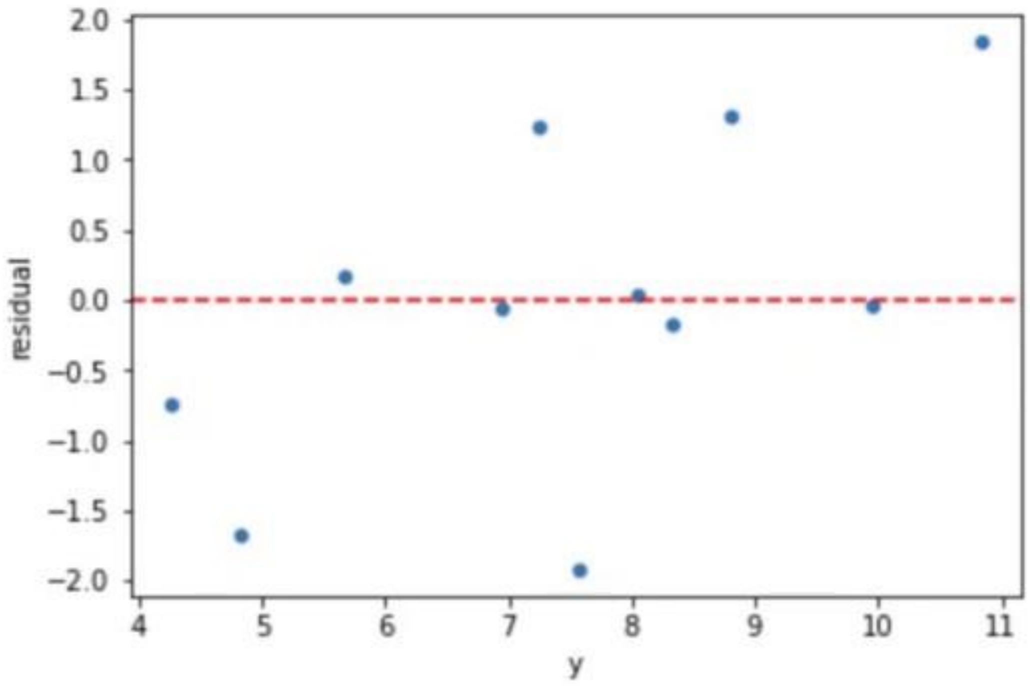
Linear Regression

- Residual plot shows residual error vs. true y value.



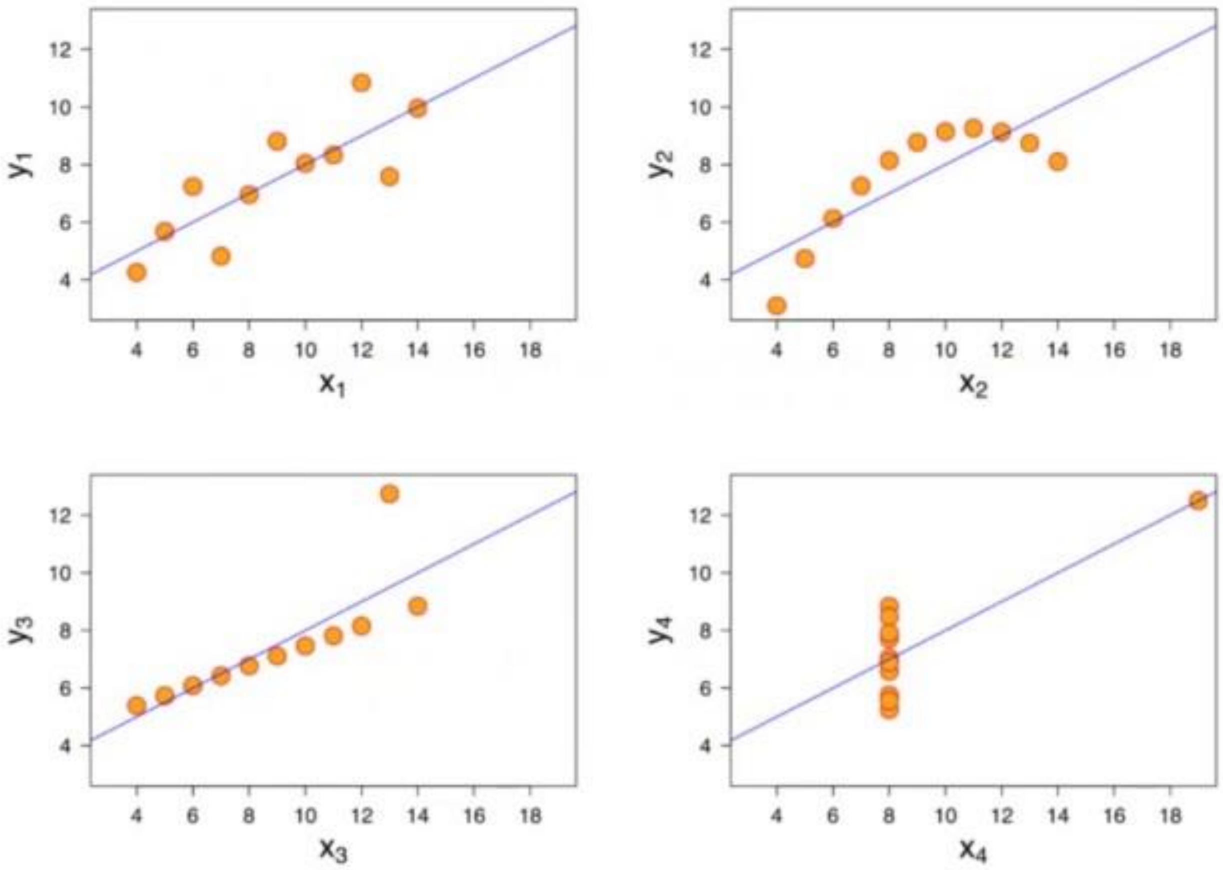
Linear Regression

- There should be no clear line or curve.



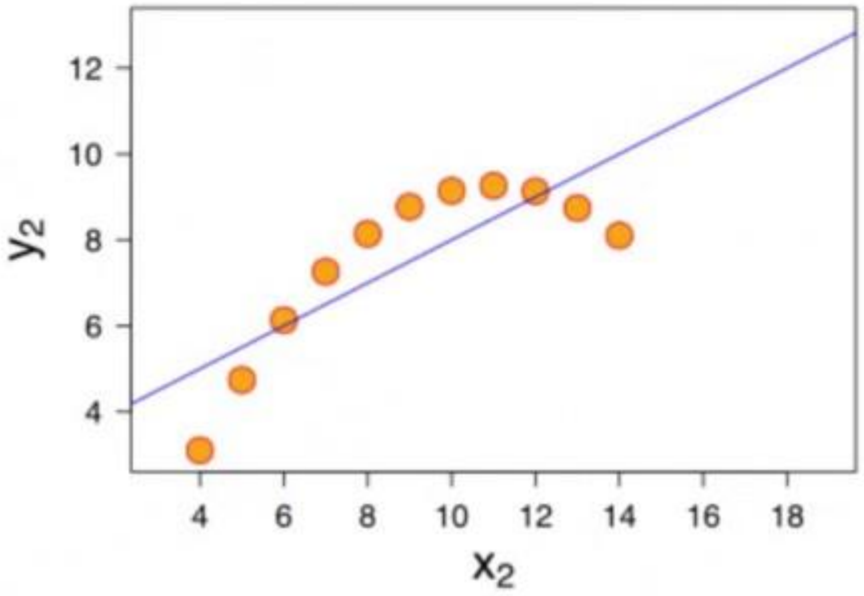
Linear Regression

- What about non valid datasets?



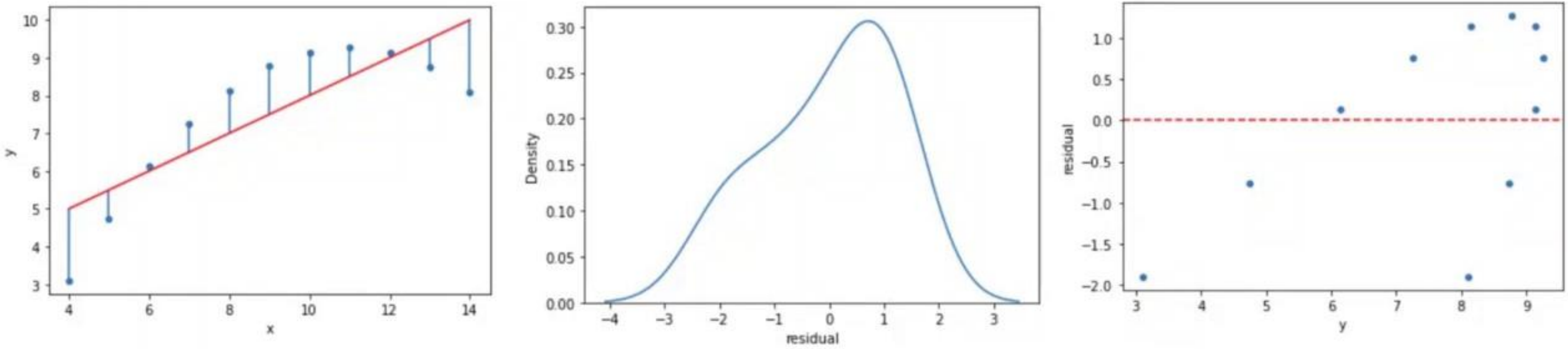
Linear Regression

- What about non valid datasets?



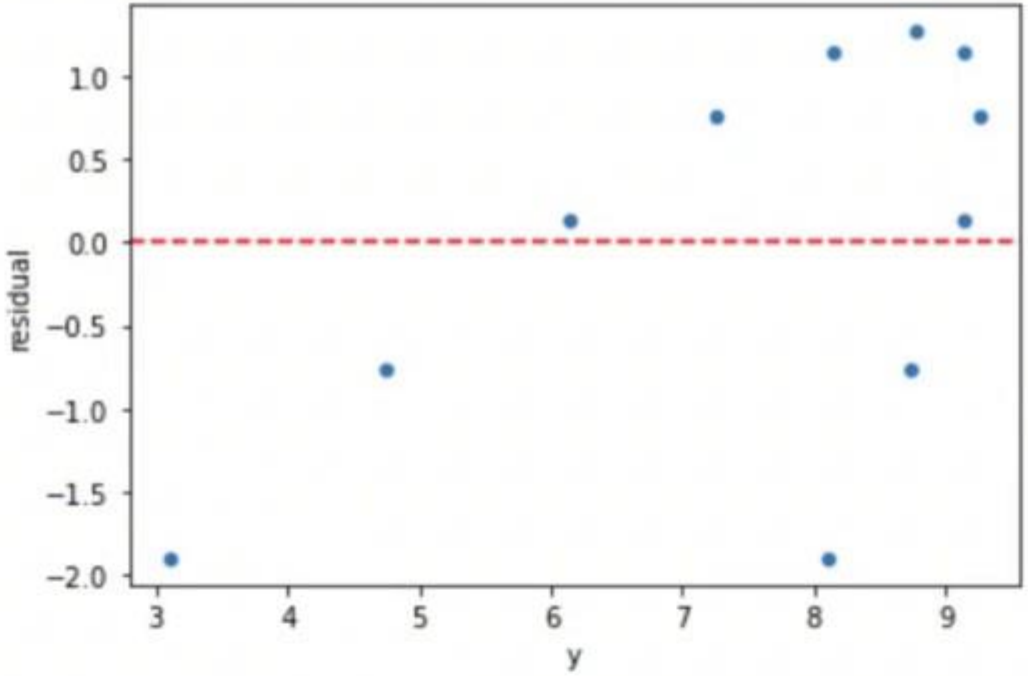
Linear Regression

- Residual plot showing a clear pattern, indicating Linear Regression no valid!



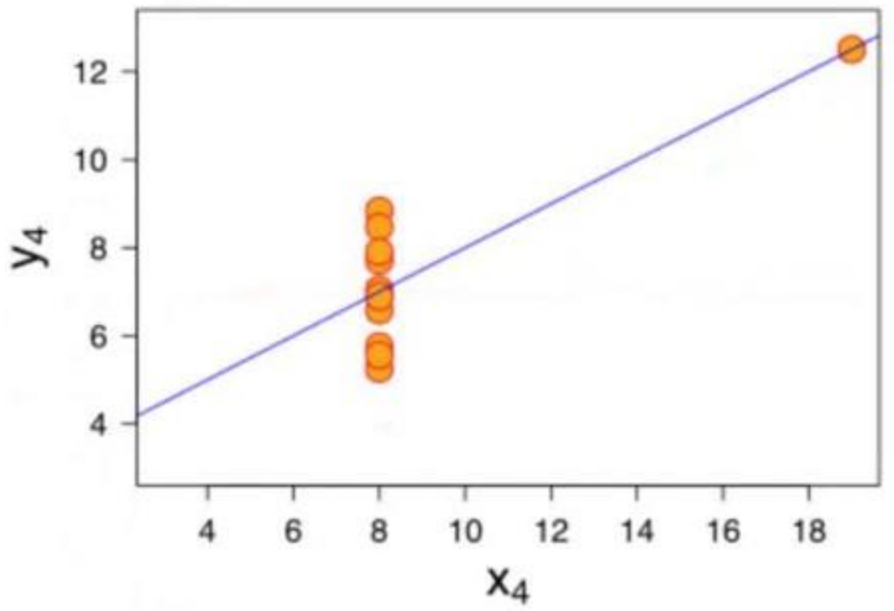
Linear Regression

- Residual plot showing a clear pattern, indicating Linear Regression no valid!



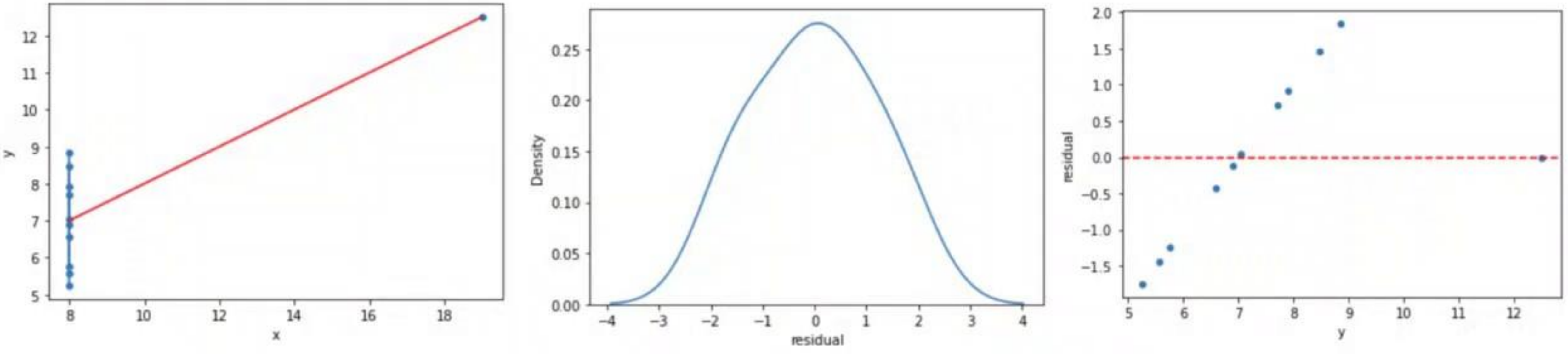
Linear Regression

- What about non valid datasets?



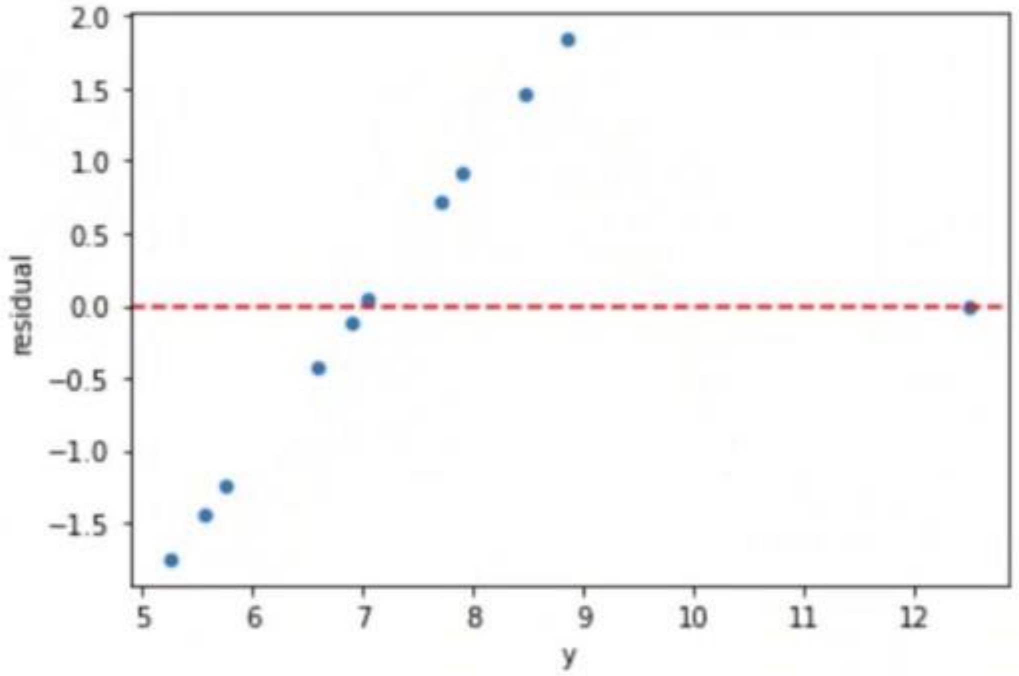
Linear Regression

- Residual plot showing a clear pattern, indicating Linear Regression no valid!



Linear Regression

- Residual plot showing a clear pattern, indicating Linear Regression no valid!



Linear Regression

- Let's explore creating these plots with Python and our model results!

01-Linear-Regression-with-Scitkit-Learn[LEC4].ipynb

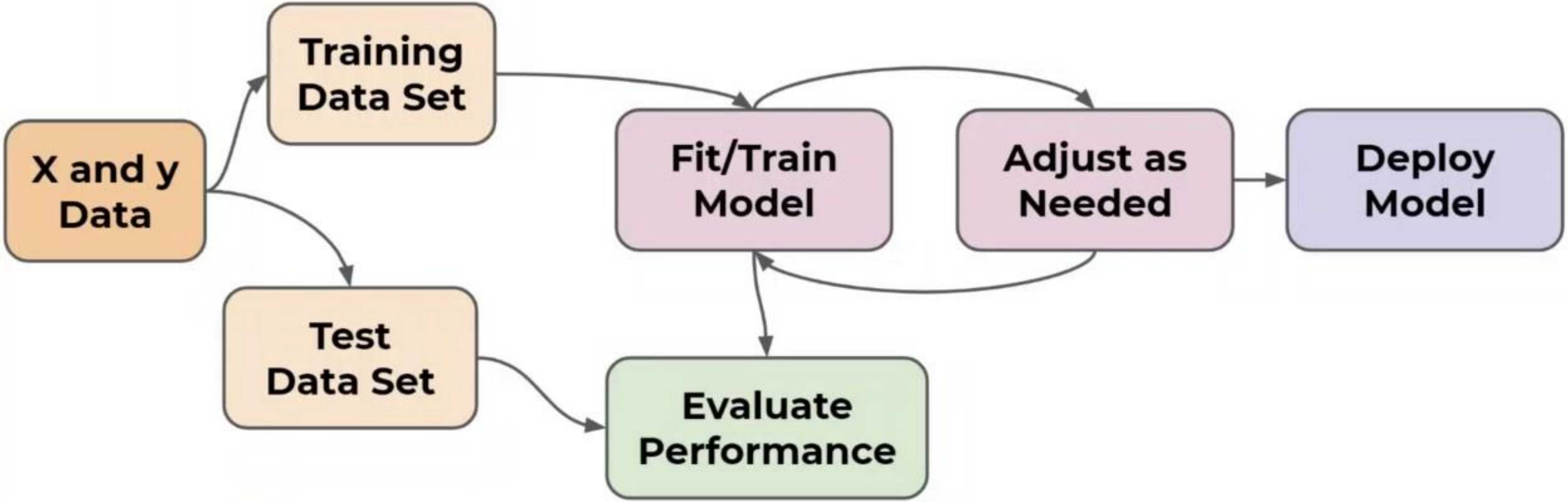
Model Deployment

Linear Regression

- We’re almost done with our first machine learning run through!
- Let’s quickly review what we’ve done so far in the ML process.

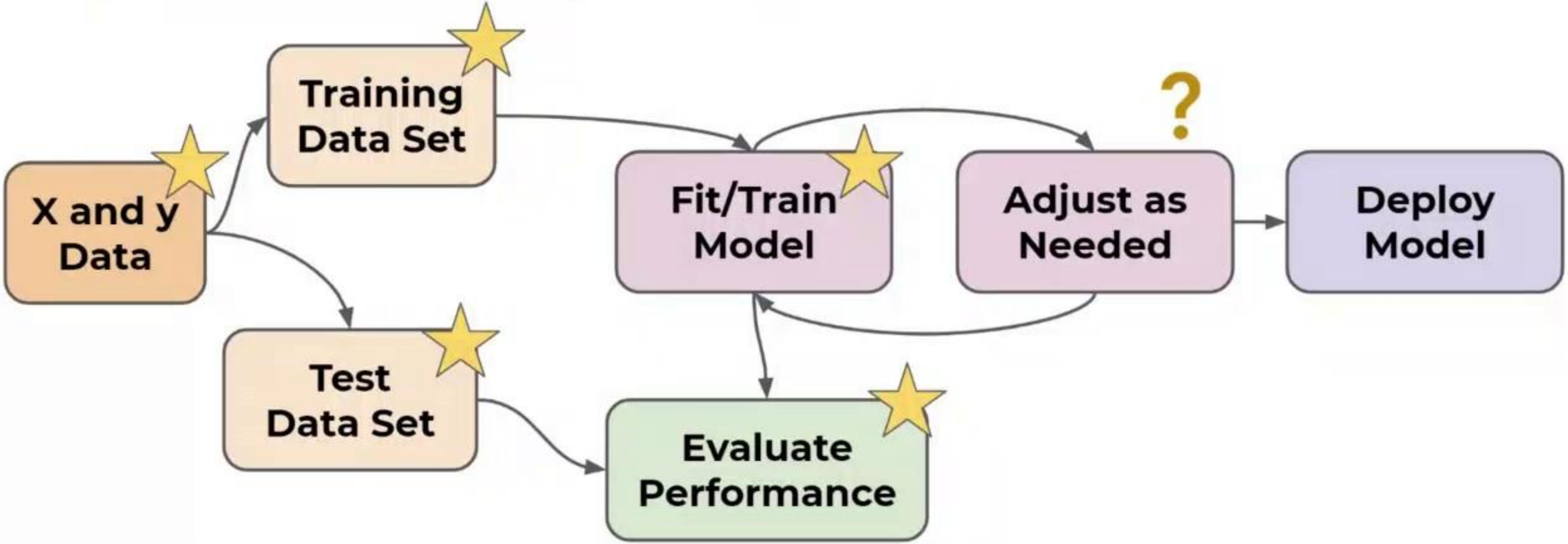
Supervised Machine Learning Process

- Recall the Supervised ML Process



Supervised Machine Learning Process

- Recall the Supervised ML Process



Linear Regression

- Later on we will explore polynomial regression and regularization as model adjustments.
- For now, let's focus on a simple "deployment" of our model by saving and loading it, then applying to new data.

01-Linear-Regression-with-Scitkit-Learn[LEC4].ipynb