# MatchIT: Python for Data Science

## Lecture: Introduction to Machine Learning in Pandas

# Lecture 3: Content

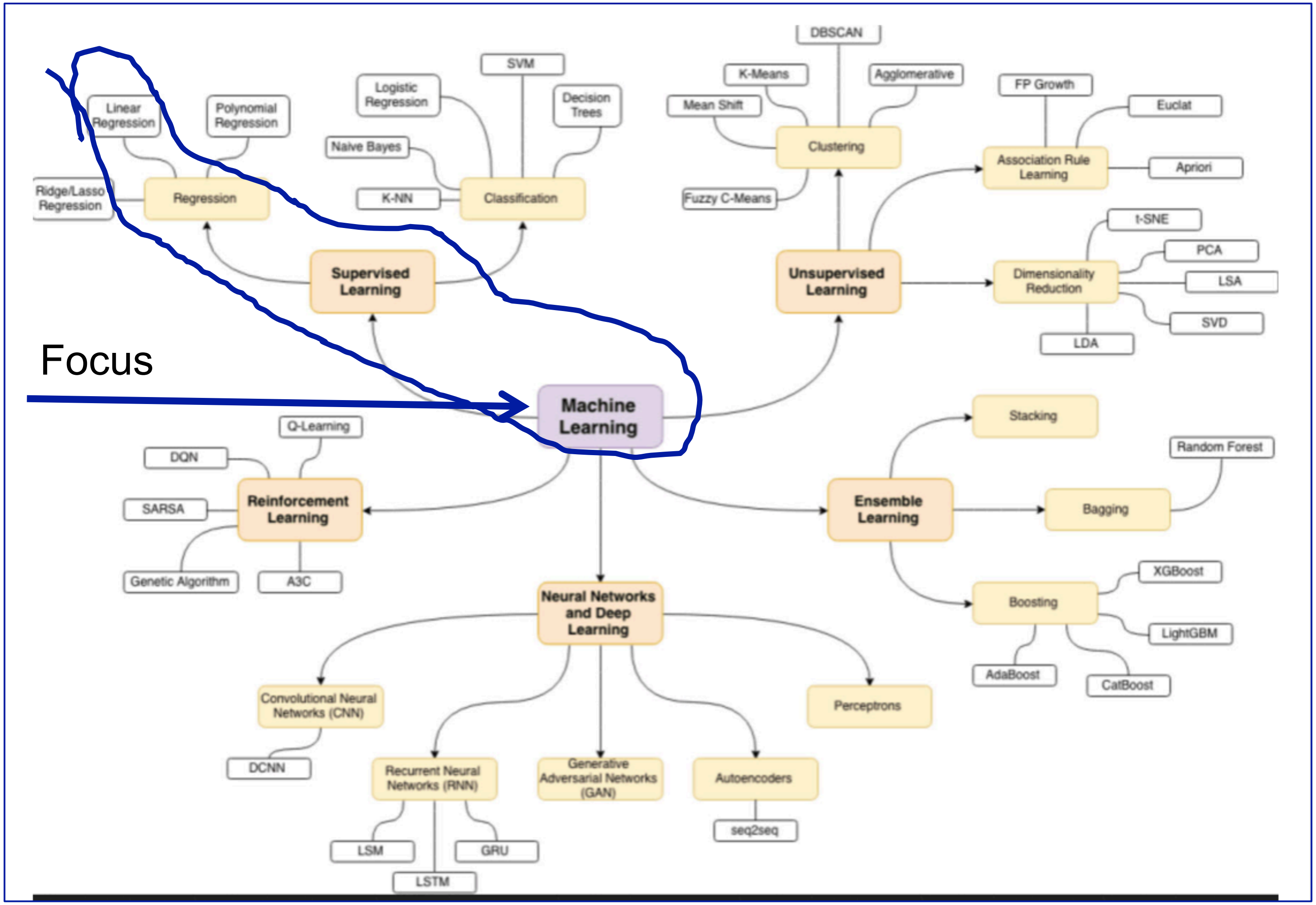
- Introduction to Machine Learning
  - Slides 3-6 materials come from the articles:
    - Grady Booch on the Future of AI : https://www.infoq.com/news/2019/02/Grady-Booch-Future-AI/?utm_source=sumome&utm_medium=linkedin&utm_campaign=sumome_share
    - The Mathematics of Machine Learning: https://www.datasciencecentral.com/profiles/blogs/the-mathematics-of-machine-learning?utm_content=bufferbed70&utm_medium=social&utm_source=facebook.com&utm_campaign=buffer
- Categories of Machine Learning
  - Supervised Learning
  - Unsupervised Learning
- Scikit-Learn Pandas Library
- Linear Regression
  - Simple
  - Multi valued

# Machine Learning in Pandas: Introduction

- Machine Learning theory is a field that intersects **statistical, probabilistic, computer science and algorithmic** aspects arising from learning iteratively from data and finding hidden insights which can be used to build intelligent applications.

- Despite the immense possibilities of Machine and Deep Learning, a thorough mathematical understanding of many of these techniques is necessary for a good grasp of the inner workings of the algorithms and obtaining good results.

# Machine Learning in Pandas: Introduction



Focus

# Machine Learning in Pandas: Introduction

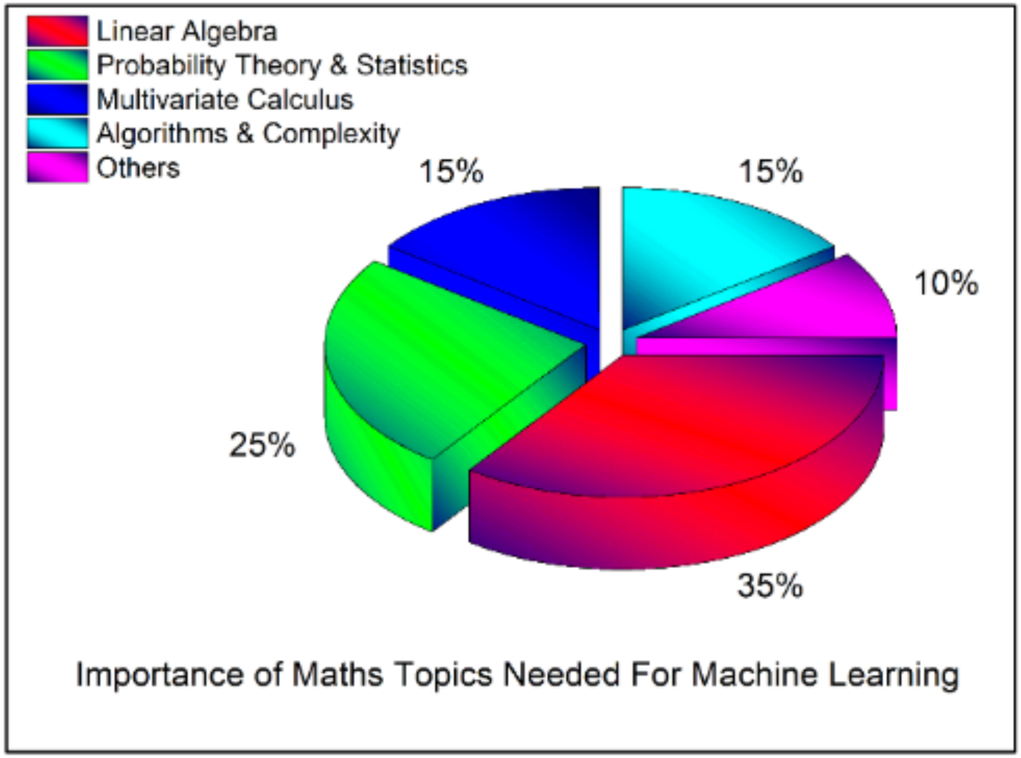
There are many reasons why the mathematics of Machine Learning is important :

1. Selecting the right algorithm which includes giving considerations to accuracy, training time, model complexity, number of parameters and number of features.
2. Choosing parameter settings and validation strategies.
3. Identifying underfitting and overfitting by understanding the Bias-Variance tradeoff.
4. Estimating the right confidence interval and uncertainty.

- Terms **highlighted in blue** along with linear regression will be focus of our lectures for this week.

# Machine Learning in Pandas: Introduction

What level of mathematics is needed:



Importance of Maths Topics Needed For Machine Learning

# Machine Learning in Pandas: Introduction

- Machine learning is not a magic pill where one can apply any approach or algorithm to analyze a data set – it is an attempt to build a mathematical model in order to understand data.  It is a meeting point of computational and algorithmic aspects of data science and various fields of mathematics theory.

- Learning part of the machine learning term is when the mathematical models are given tunable parameters that can be applied to the observed data.

# Machine Learning in Pandas: Categories of Machine Learning

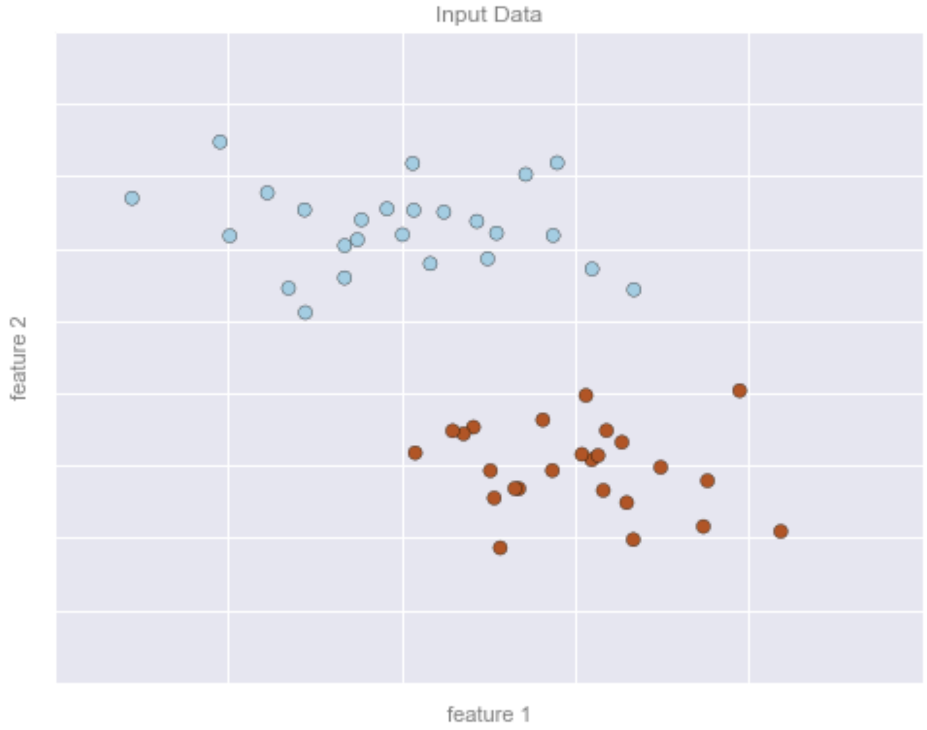
- The fundamental categorization of machine learning:

  - Supervised learning : Modeling relationship between measured features (input) and some labels (output) associated with data.
    - Tasks associated with supervised learning are:
      - Classification tasks
        - Labels are discrete categories
      - Regression tasks
        - Labels are continuous quantities

  - Unsupervised leaning: Modeling the features of dataset without reference to any label – “letting the data speak for itself”
    - Tasks associated with such models are:
      - Clustering tasks that identify distinct groups of data
      - Dimensionality Reduction tasks deal with finding of more brief-concise representation of data

# Machine Learning in Pandas: Categories of Machine Learning – Supervised Learning

- Supervised learning – Classification tasks
  - Two dimensional data : two *features* for each *point (x,y)*
  - Two class *labels* for each point – blue or red.
  - Goal: From features and labels create a model that will let us decide weather a new point should be labeled blue or red.



- *Model*: A straight line separates two classes
- *Model parameters*: A particular numbers describing the location and orientation of the line.
- Finding optimal values of parameters is *learning from the data or training the model*

# Machine Learning in Pandas: Categories of Machine Learning : Example Supervised Classification Model

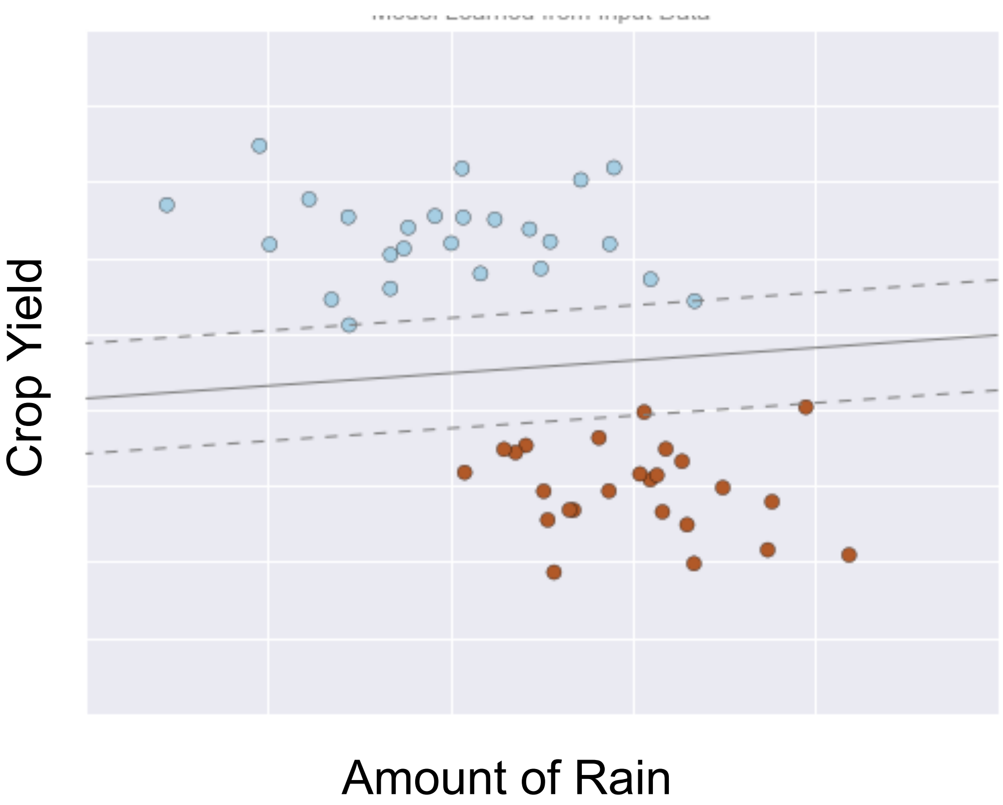
- Once the model has been trained it can be generalized to new data: It can be used to *predict* into which category a new data will fall into.
- Simple idea which can be applied to different datasets having more than two features and two or more discrete labels.

- Example 1: Answer yes/no question: “Is this email spam?”
  - Spam detection:
    - feature1, feature 2 -> normalized counts of important words of pharses (“Viagra”, “Nigerian prince”, “Lottery”, etc)
    - label -> “spam” or “not spam” (binary classification)

- Example 2: Classify handwrting: “Is this letter 0,1,2,3,...?”
  - Handwritten number recognition
    - many features
    - label -> “0”, “1”, “2” ,..., “9”

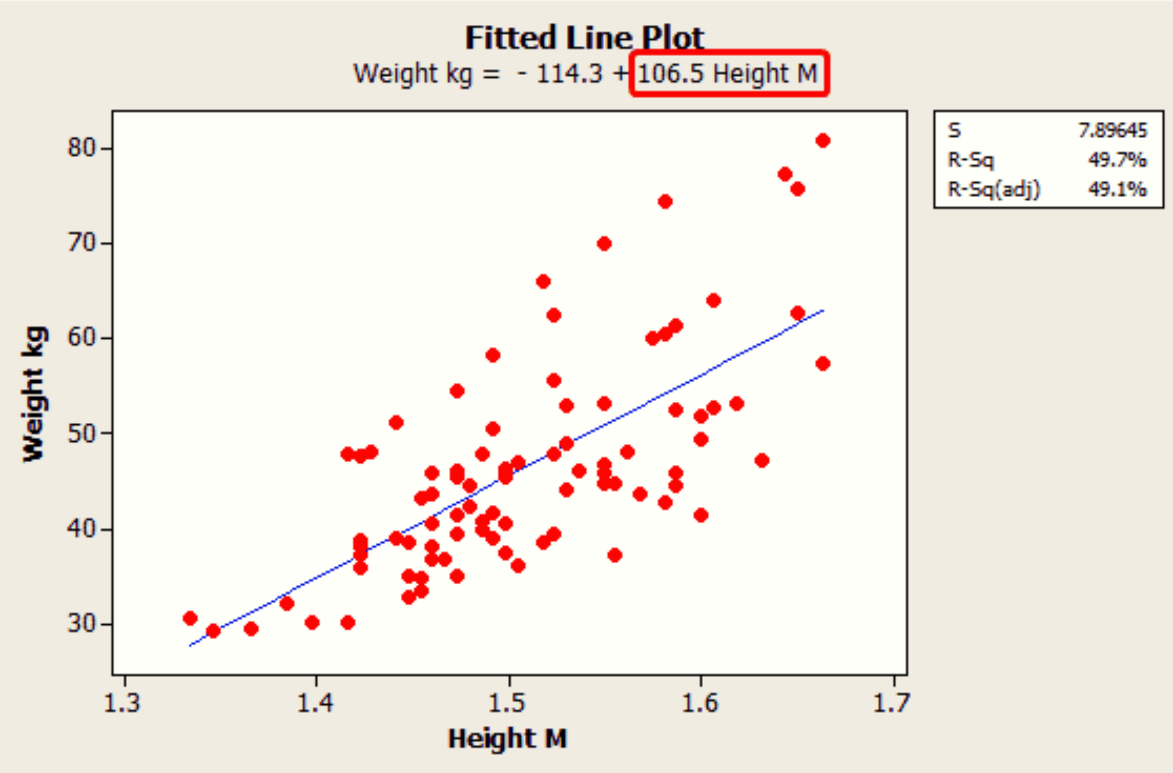
# Machine Learning in Pandas: Categories of Machine Learning : Example Supervised Classification Model

$h(x) = \theta_0 + \theta_1 X$



# Machine Learning in Pandas: Categories of Machine Learning : Example Supervised Learning: Linnear Regression

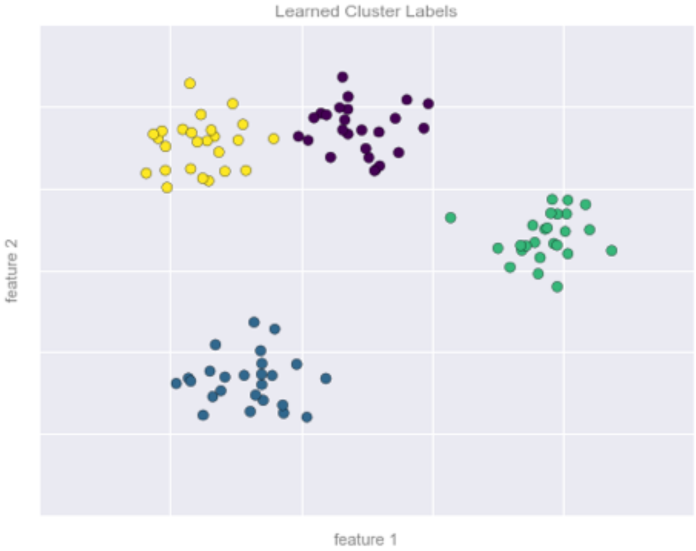
- Regression task: Labels are continuous quantities.
- $h(x) = \theta_0 + \theta_1 X$ : Predictor function



# Machine Learning in Pandas: Categories of Machine Learning : Unsupervised Learning

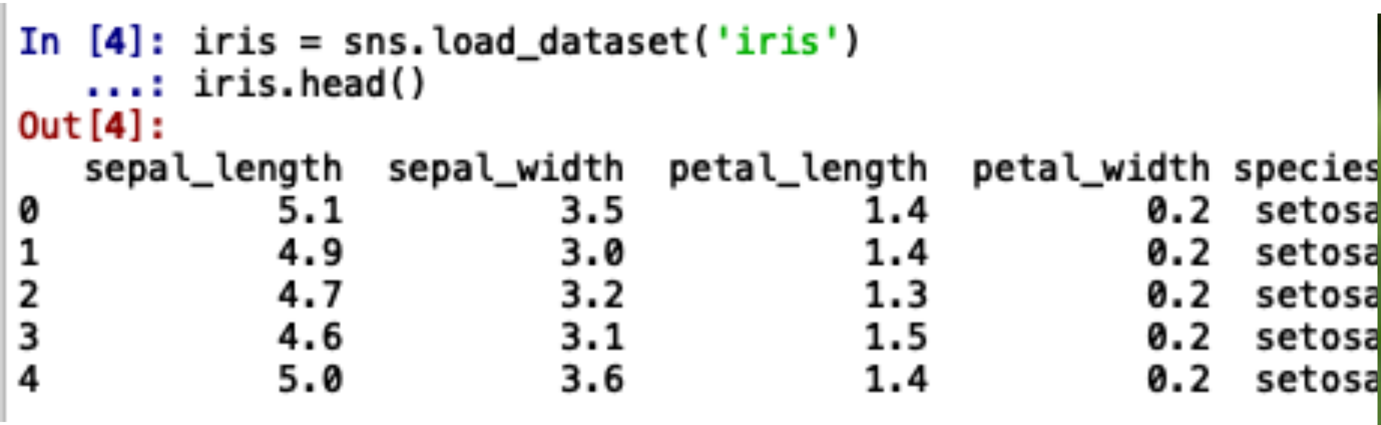
- Review: In the classification and regression examples we observed supervised learning algorithms where goal is to build model that will predict labels for new data.
- A cluster model uses the intrinsic structure of data to determine which points are related ( k-Means clustering)
  - Determine groups of contributors who work more closely

- Dimensionality reduction
  - Labels are inferred from structure.
  - Helps are reduce number of features only include important ones.

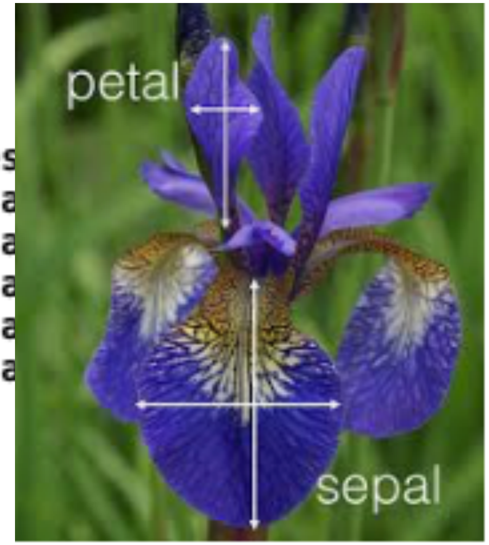


# Machine Learning in Pandas: Scikit-Learn

- Package that contains efficient implementation of large number of commonly used algorithms in machine learning with clear and well documented API.



```
In [4]: iris = sns.load_dataset('iris')
   ...: iris.head()
Out[4]:
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```



- Row: Single Observed Flower
  - Rows = Samples
  - Number of rows = n_samples

- Columns: Quantitative piece of information
  - Columns = Feature
  - Number of columns = n_features

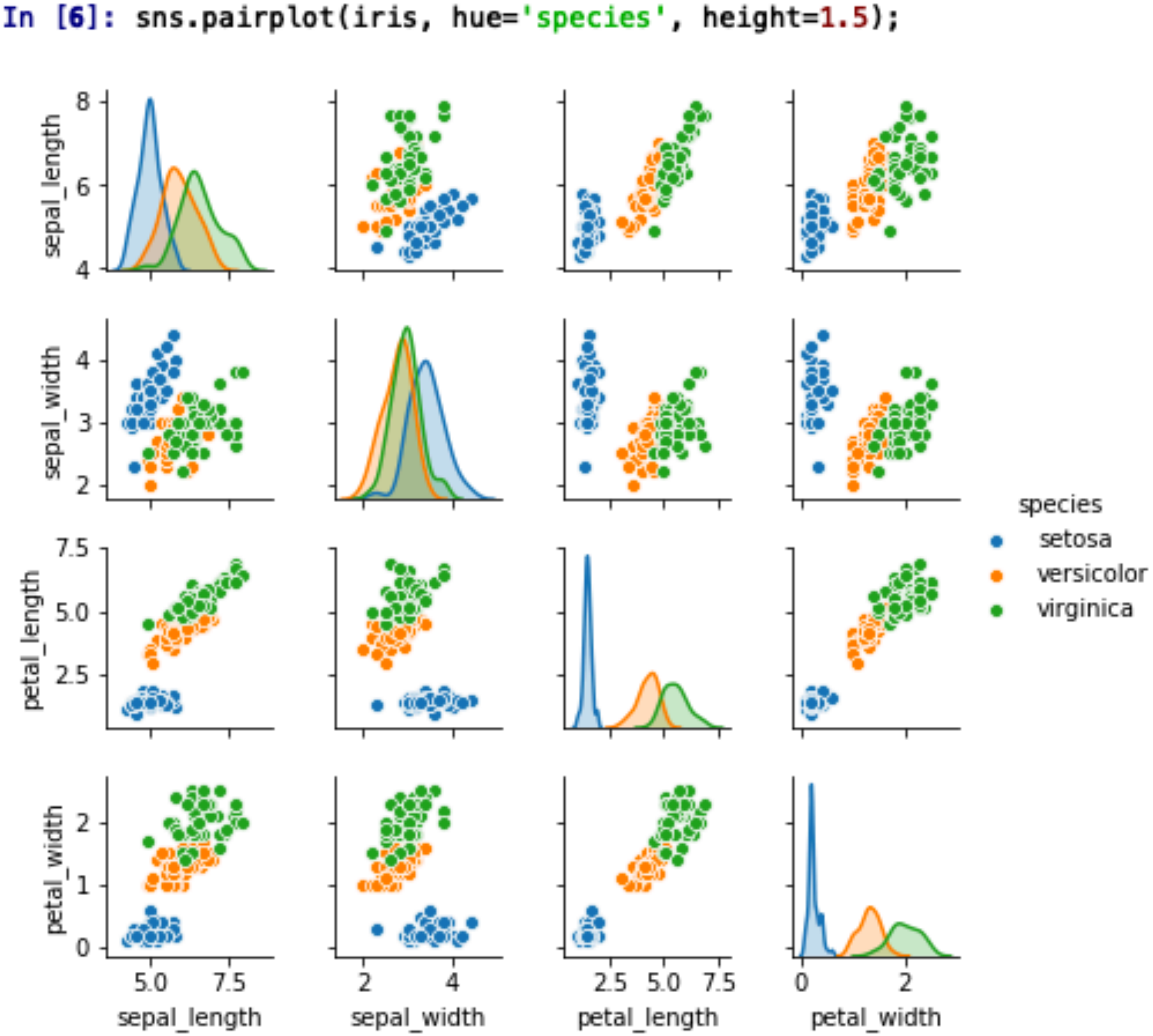
# Machine Learning in Pandas: Scikit-Learn

- Features matrix = the table, is a two dimensional with shape [n_samples, n_features] and it is most often contained by NumPy array or Pandas DataFrame.

- Label or target array conventionally called “y”
  - The quantity we want to predict from the data – the dependent variable
  - One dimensional with length of n_samples and contained in a NumPy array or Pandas Series
  - Can have continuous numerical values, or discrete classes/labels

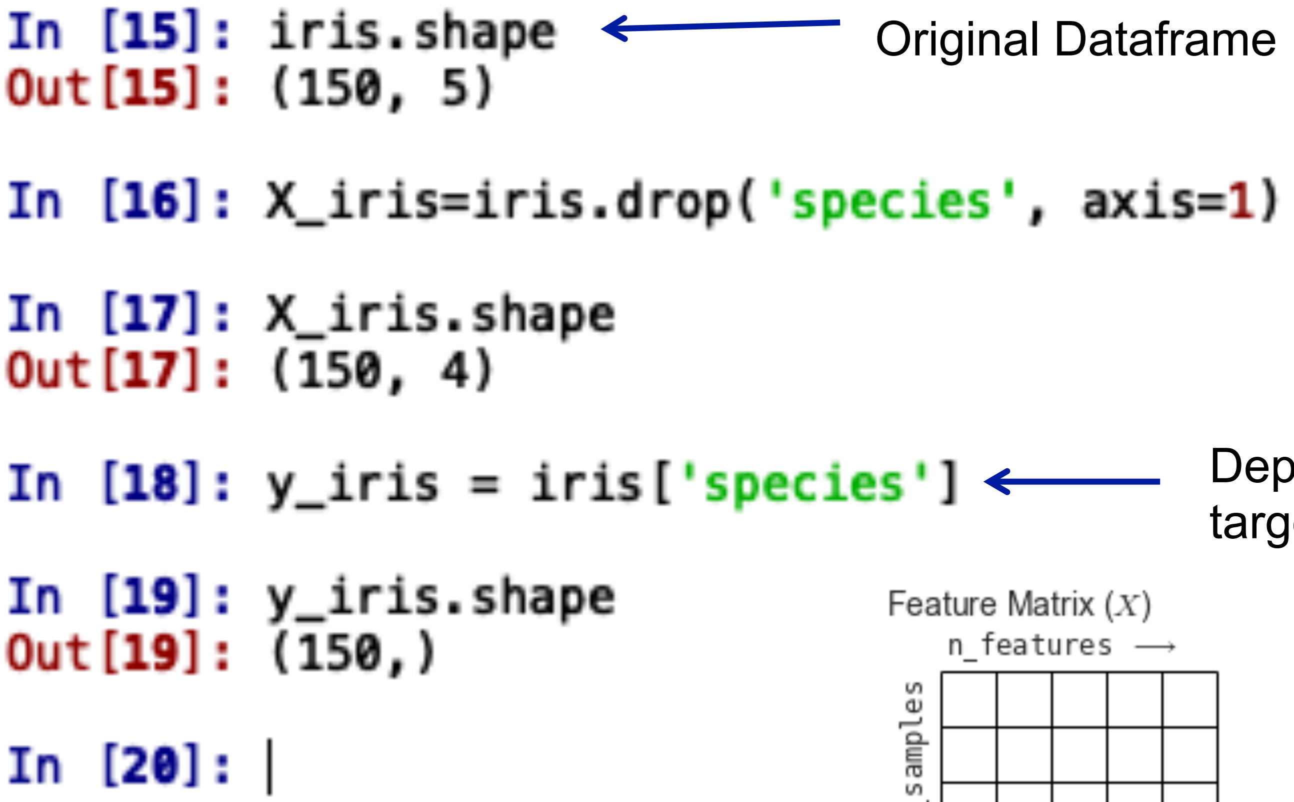
- Example:
  - Construct a model that can predict the species of flower based on the other measurements.

# Machine Learning in Pandas: Scikit-Learn



```
In [6]: sns.pairplot(iris, hue='species', height=1.5);
```

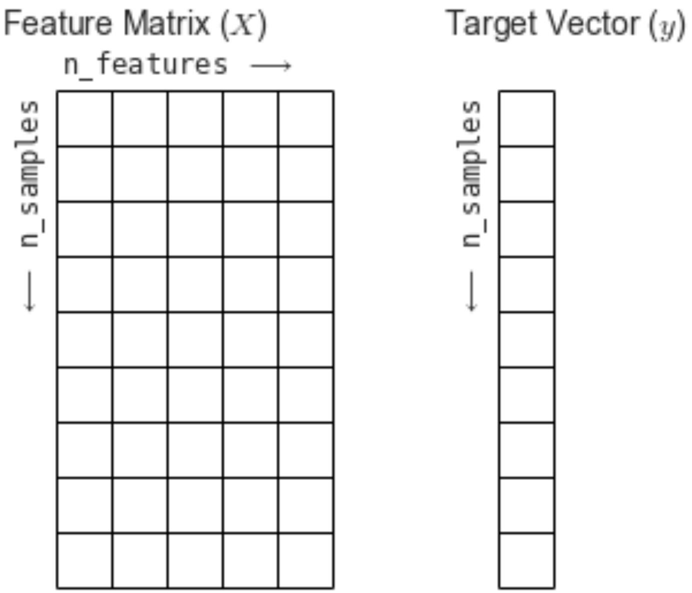
# Machine Learning in Pandas: Scikit-Learn



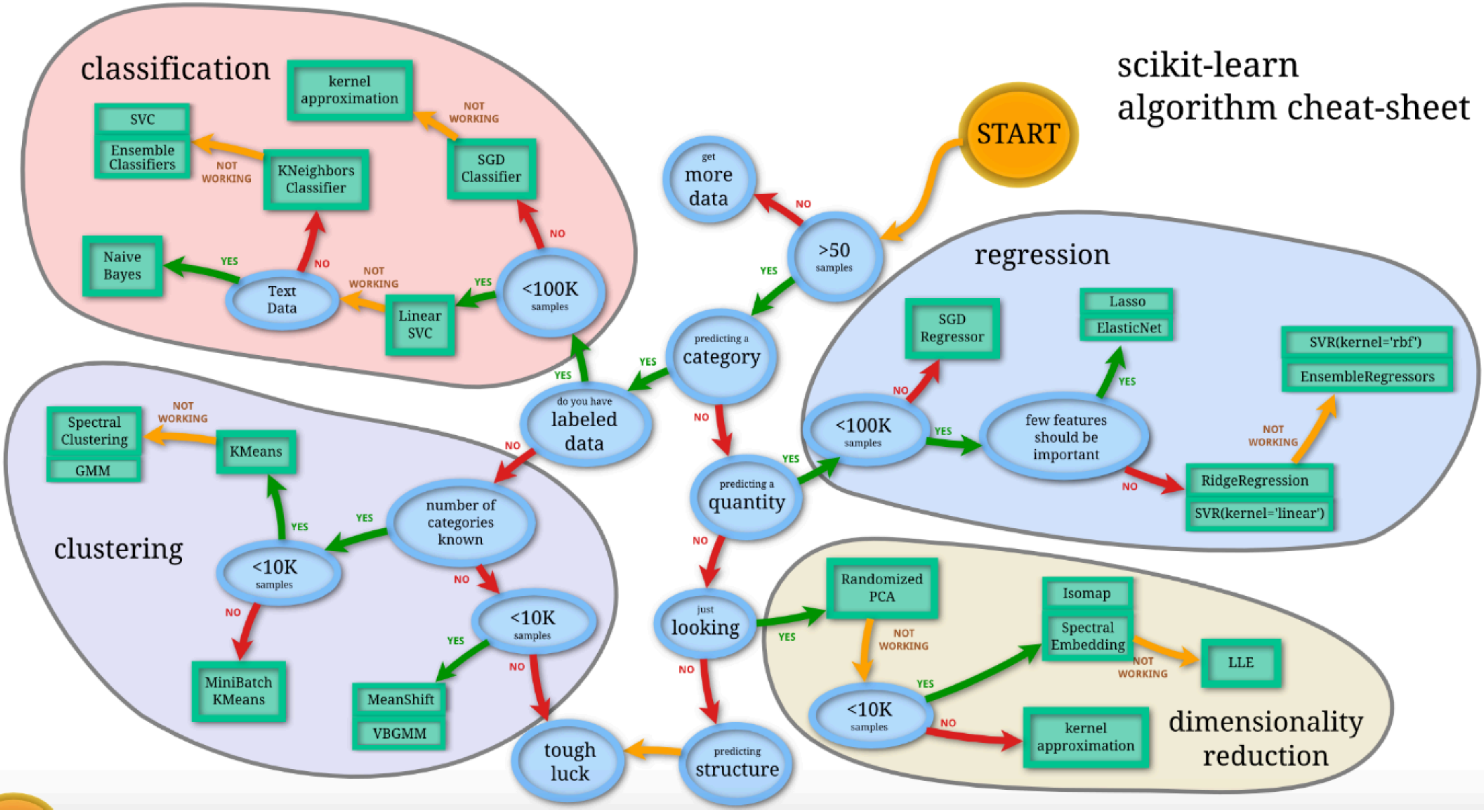
```
In [15]: iris.shape
Out[15]: (150, 5)

In [16]: X_iris=iris.drop('species', axis=1)

In [17]: X_iris.shape
Out[17]: (150, 4)

In [18]: y_iris = iris['species']

In [19]: y_iris.shape
Out[19]: (150,)

In [20]: |
```

Original Dataframe

Dependent variable y, target array



# Machine Learning in Pandas: Scikit-Learn API: Estimator Class implements many algorithms



From: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

# Machine Learning in Pandas: Basics of the API

1. Choose a class of model by importing the appropriate estimator class from Scikit-Learn

2. Choose model hyperparameters by instantiating this class with desired values. Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes.

3. Arrange data into a features matrix and target vector.

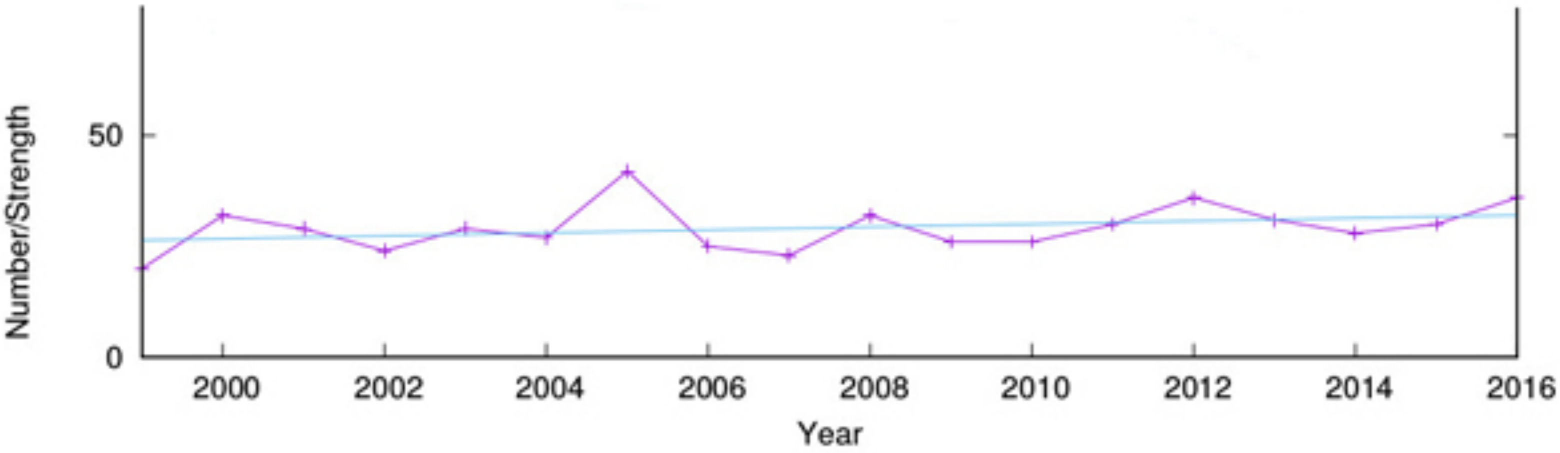
4. Fit the model to your data by calling the fit(method) of the model instance.

5. Apply the model to new data

   - For supervised learning, we can predict labels for unknown data using the predict() method
   - For unsupervised learning, we often transform to infer properties of the data using the transform() or predict() method.

# Linear Regression

- A linear regression tries to estimate a linear relationship that best fits a given set of data.
  - Example : How the number of tropical storms has changed over the years? Plot number of storms against the time and using linear regression find the straight line that fits the best the plotted data.



Yearly tropical storms. The blue line indicates the result of a linear regression on the number of storms over time.

From: [https://plus.maths.org/content/maths-minute-linear-regression](https://plus.maths.org/content/maths-minute-linear-regression)

# Linear Regression

- Dependent variable y is linearly dependent on one or more explanatory variables

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n.$$

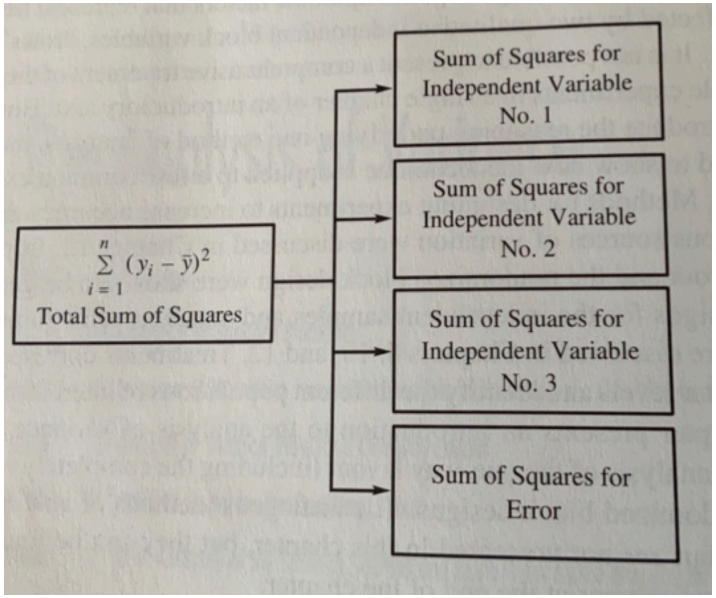
- In case of tropical storm, dependent variable is one: number of hurricanes and the single explanatory variable is year.

$$hurricanes = -637.9 + 0.3323 \times year.$$

- The values of parameters are chosen to minimize the error over the years – in case of linear regression a squared error.

# Linear Regression

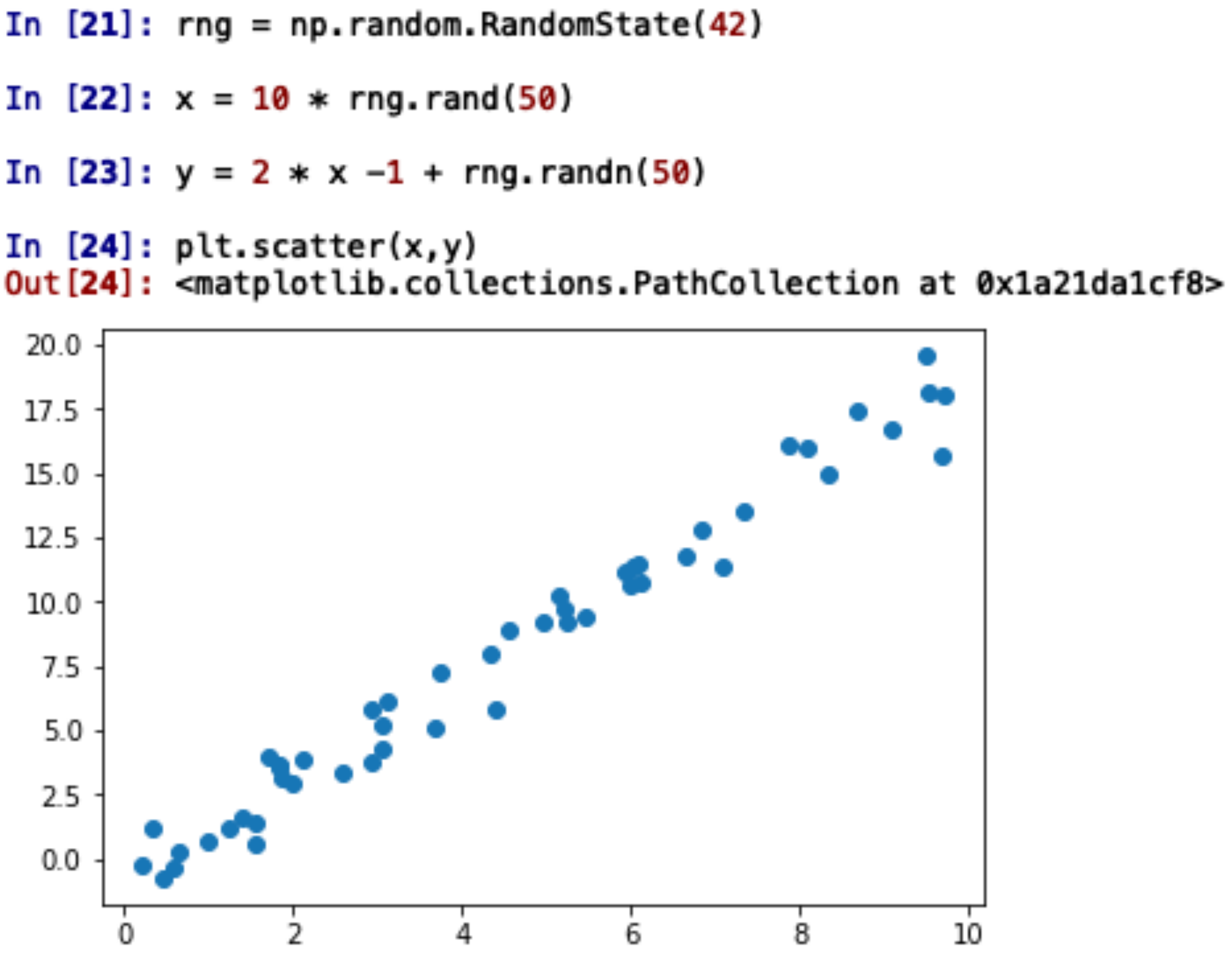
- What is a mean squared error (MSE)?
  - MSE measures the average of the squares of the errors, the average squared difference between the estimated values and actual values.


$\sum_{i=1}^{n} (y_i - \bar{y})^2$
Total Sum of Squares
Sum of Squares for Independent Variable No. 1
Sum of Squares for Independent Variable No. 2
Sum of Squares for Independent Variable No. 3
Sum of Squares for Error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

$Y_i$ = actual value
$Y_i$ hat = estimated value

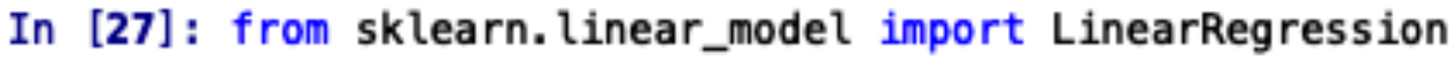
# Machine Learning in Pandas: Simple Linear Regression



```
In [21]: rng = np.random.RandomState(42)

In [22]: x = 10 * rng.rand(50)

In [23]: y = 2 * x -1 + rng.randn(50)

In [24]: plt.scatter(x,y)
Out[24]: <matplotlib.collections.PathCollection at 0x1a21da1cf8>
```

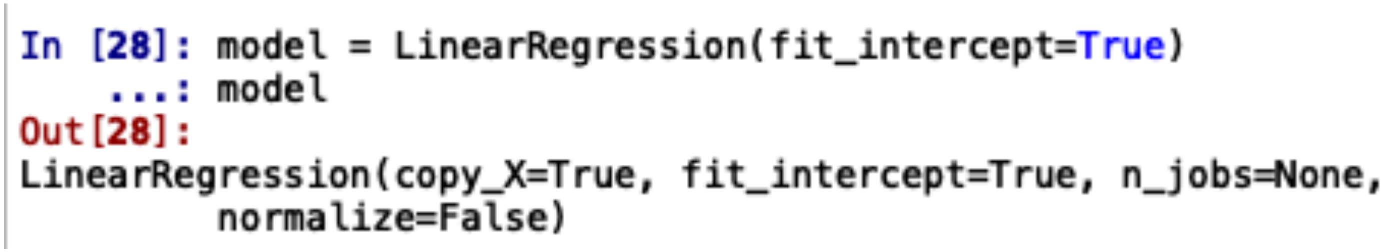
# Machine Learning in Pandas: Simple Linear Regression

1. Choose a class of model:



```
In [27]: from sklearn.linear_model import LinearRegression
```

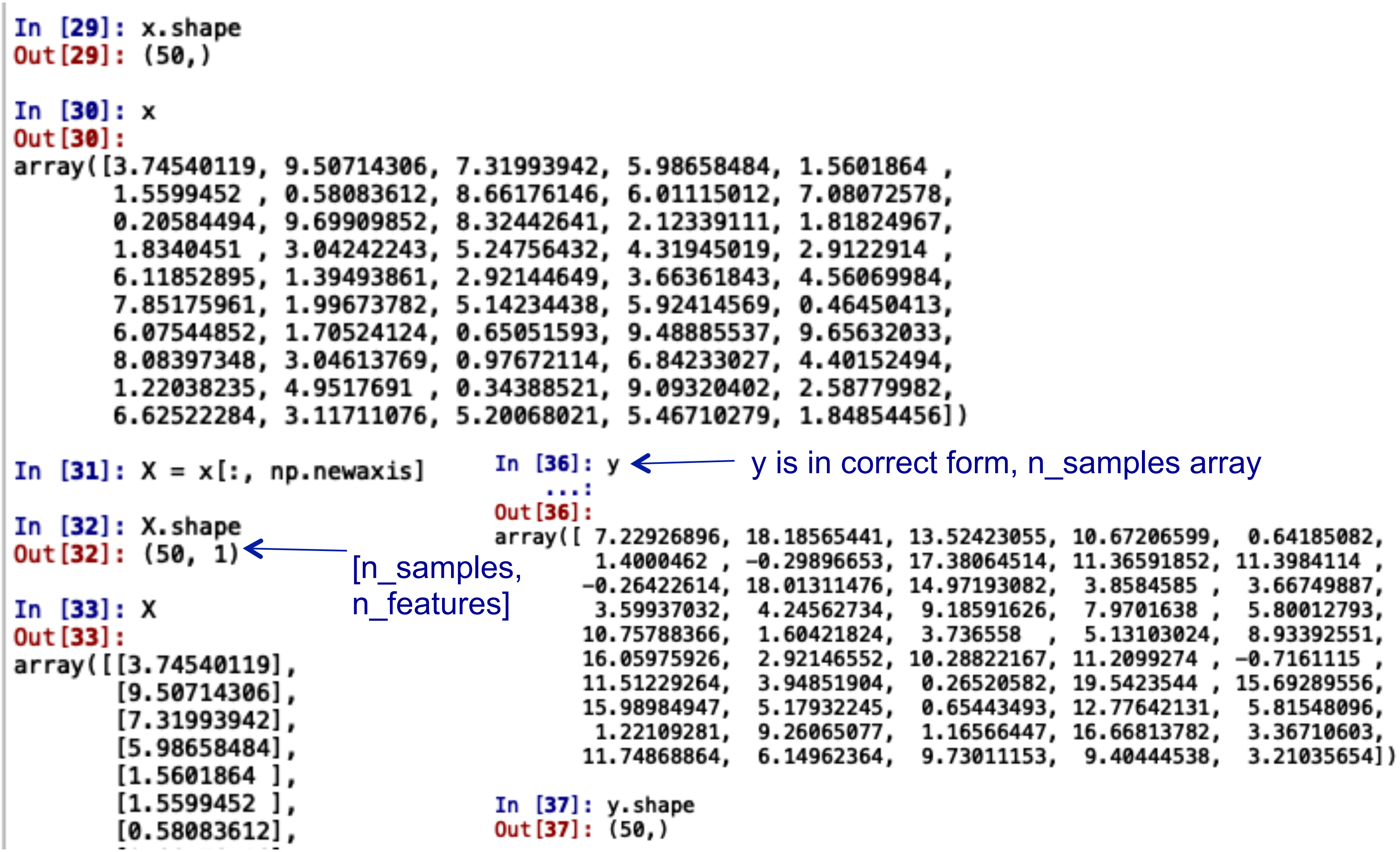
2. Choose model hyperparameters



```
In [28]: model = LinearRegression(fit_intercept=True)
    ...: model
Out[28]:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

# Machine Learning in Pandas: Simple Linear Regression... continued

3. Arrange data into a features matrix and target vector



```
In [29]: x.shape
Out[29]: (50,)

In [30]: x
Out[30]:
array([3.74540119, 9.50714306, 7.31993942, 5.98658484, 1.5601864 ,
       1.5599452 , 0.58083612, 8.66176146, 6.01115012, 7.08072578,
       0.20584494, 9.69909852, 8.32442641, 2.12339111, 1.81824967,
       1.8340451 , 3.04242243, 5.24756432, 4.31945019, 2.9122914 ,
       6.11852895, 1.39493861, 2.92144649, 3.66361843, 4.56069984,
       7.85175961, 1.99673782, 5.14234438, 5.92414569, 0.46450413,
       6.07544852, 1.70524124, 0.65051593, 9.48885537, 9.65632033,
       8.08397348, 3.04613769, 0.97672114, 6.84233027, 4.40152494,
       1.22038235, 4.9517691 , 0.34388521, 9.09320402, 2.58779982,
       6.62522284, 3.11711076, 5.20068021, 5.46710279, 1.84854456])

In [31]: X = x[:, np.newaxis]

In [32]: X.shape
Out[32]: (50, 1)

In [33]: X
Out[33]:
array([[3.74540119],
       [9.50714306],
       [7.31993942],
       [5.98658484],
       [1.5601864 ],
       [1.5599452 ],
       [0.58083612],
```

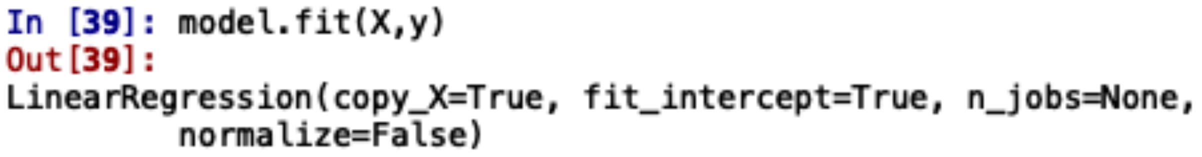
[n_samples, n_features]

```
In [36]: y
    ...:
Out[36]:
array([ 7.22926896, 18.18565441, 13.52423055, 10.67206599,  0.64185082,
        1.4000462 , -0.29896653, 17.38064514, 11.36591852, 11.3984114 ,
       -0.26422614, 18.01311476, 14.97193082,  3.8584585 ,  3.66749887,
        3.59937032,  4.24562734,  9.18591626,  7.9701638 ,  5.80012793,
       10.75788366,  1.60421824,  3.736558  ,  5.13103024,  8.93392551,
       16.05975926,  2.92146552, 10.28822167, 11.2099274 , -0.7161115 ,
       11.51229264,  3.94851904,  0.26520582, 19.5423544 , 15.69289556,
       15.98984947,  5.17932245,  0.65443493, 12.77642131,  5.81548096,
        1.22109281,  9.26065077,  1.16566447, 16.66813782,  3.36710603,
       11.74868864,  6.14962364,  9.73011153,  9.40444538,  3.21035654])

In [37]: y.shape
Out[37]: (50,)
```

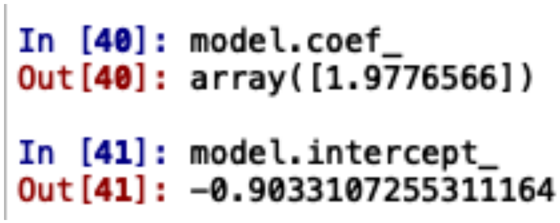
y is in correct form, n_samples array

# Machine Learning in Pandas: Simple Linear Regression... continued

4. Fit the model to your data.



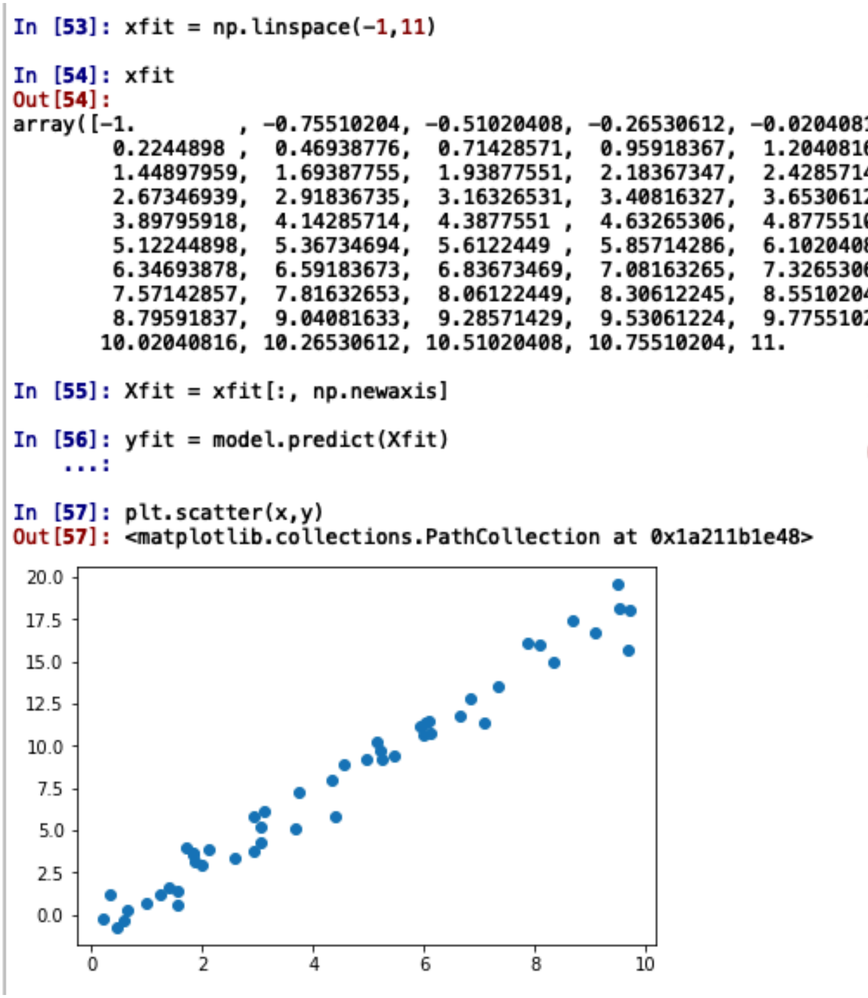
```
In [39]: model.fit(X,y)
Out[39]:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```



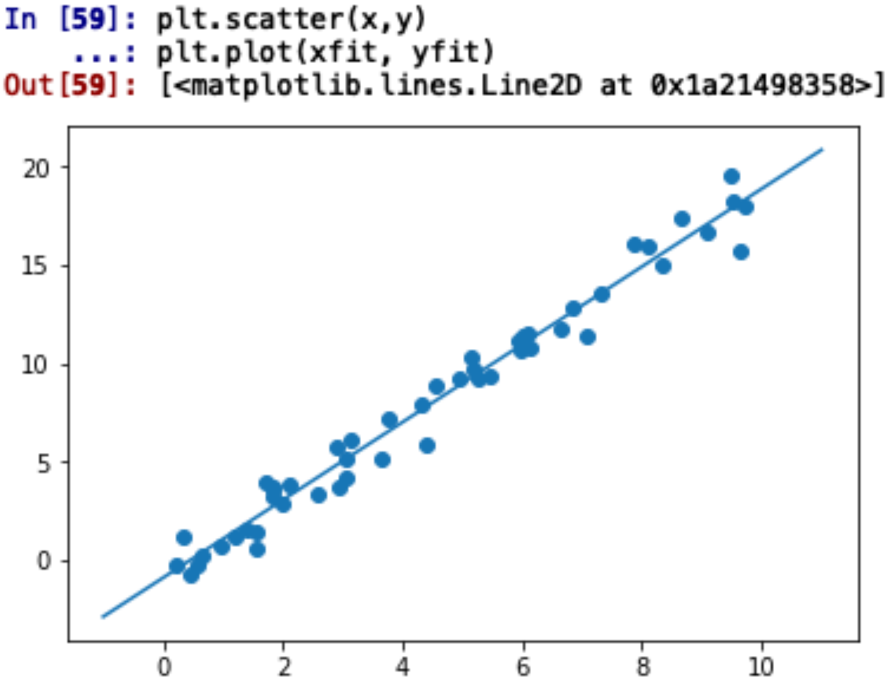
```
In [40]: model.coef_
Out[40]: array([1.9776566])

In [41]: model.intercept_
Out[41]: -0.903310725531164
```

# Machine Learning in Pandas: Simple Linear Regression... continued

5. Predict labels for unknown data

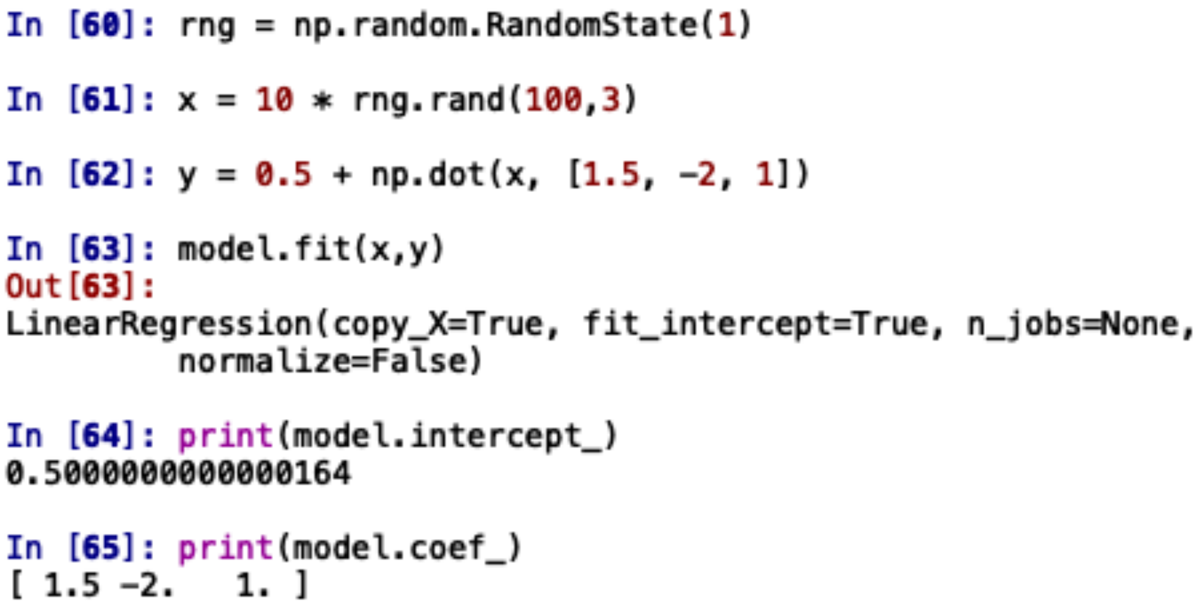


```
In [53]: xfit = np.linspace(-1,11)

In [54]: xfit
Out[54]:
array([-1.        , -0.75510204, -0.51020408, -0.26530612, -0.02040816,
        0.2244898 ,  0.46938776,  0.71428571,  0.95918367,  1.20408163,
        1.44897959,  1.69387755,  1.93877551,  2.18367347,  2.42857143,
        2.67346939,  2.91836735,  3.16326531,  3.40816327,  3.65306122,
        3.89795918,  4.14285714,  4.3877551 ,  4.63265306,  4.87755102,
        5.12244898,  5.36734694,  5.6122449 ,  5.85714286,  6.10204082,
        6.34693878,  6.59183673,  6.83673469,  7.08163265,  7.32653061,
        7.57142857,  7.81632653,  8.06122449,  8.30612245,  8.55102041,
        8.79591837,  9.04081633,  9.28571429,  9.53061224,  9.7755102 ,
       10.02040816, 10.26530612, 10.51020408, 10.75510204, 11.        ])

In [55]: Xfit = xfit[:, np.newaxis]

In [56]: yfit = model.predict(Xfit)
    ...:

In [57]: plt.scatter(x,y)
Out[57]: <matplotlib.collections.PathCollection at 0x1a211b1e48>
```



```
In [59]: plt.scatter(x,y)
    ...: plt.plot(xfit, yfit)
Out[59]: [<matplotlib.lines.Line2D at 0x1a21498358>]
```

# Machine Learning in Pandas: Linear Regression... continued



```
In [60]: rng = np.random.RandomState(1)

In [61]: x = 10 * rng.rand(100,3)

In [62]: y = 0.5 + np.dot(x, [1.5, -2, 1])

In [63]: model.fit(x,y)
Out[63]:
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)

In [64]: print(model.intercept_)
0.5000000000000164

In [65]: print(model.coef_)
[ 1.5 -2.   1. ]
```

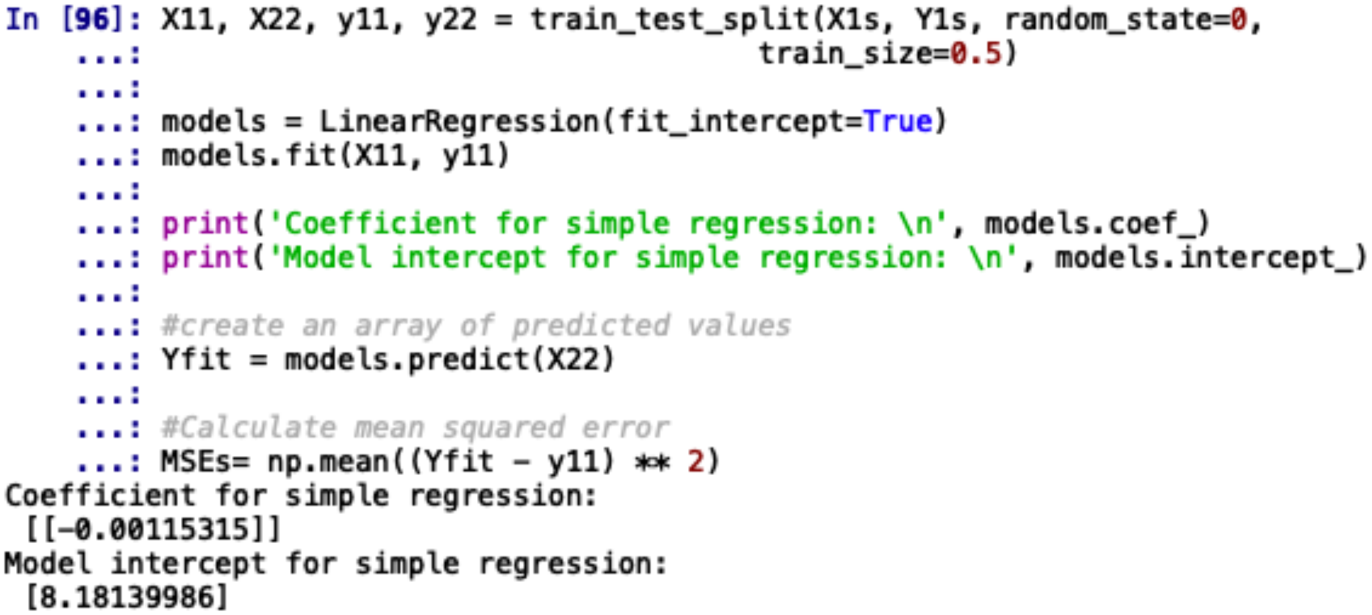
# Machine Learning in Pandas: Model validation the right way

In the previous examples we have used all data available to:

1) To create a model
2) To validate a model

For correct results we must split data set into two parts:

1) Training (to build the model)
2) Validation (to validate the model)



```
In [96]: X11, X22, y11, y22 = train_test_split(X1s, Y1s, random_state=0,
    ...:                                        train_size=0.5)
    ...:
    ...: models = LinearRegression(fit_intercept=True)
    ...: models.fit(X11, y11)
    ...:
    ...: print('Coefficient for simple regression: \n', models.coef_)
    ...: print('Model intercept for simple regression: \n', models.intercept_)
    ...:
    ...: #create an array of predicted values
    ...: Yfit = models.predict(X22)
    ...:
    ...: #Calculate mean squared error
    ...: MSEs= np.mean((Yfit - y11) ** 2)
Coefficient for simple regression:
 [[-0.00115315]]
Model intercept for simple regression:
 [8.18139986]
```

# Todo:

Read:

Chapter 5, Machine Learning:

1) What is Machine Learning
2) Introduction to Scikit-Learn
3) Hyperparameters and Model Validation

Quiz:

Thursday, June 20, 10:00 -12:00

Lab:

Due, June 20.