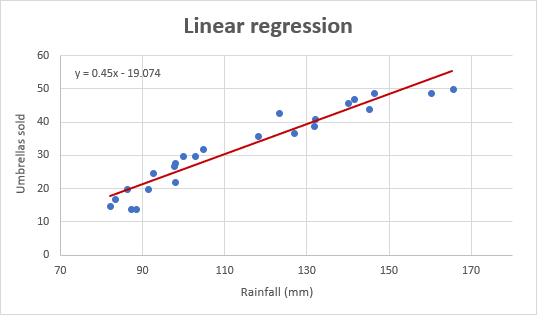
**MACHINE LEARNING**

Machine Learning isn't magic; it's just geometry!

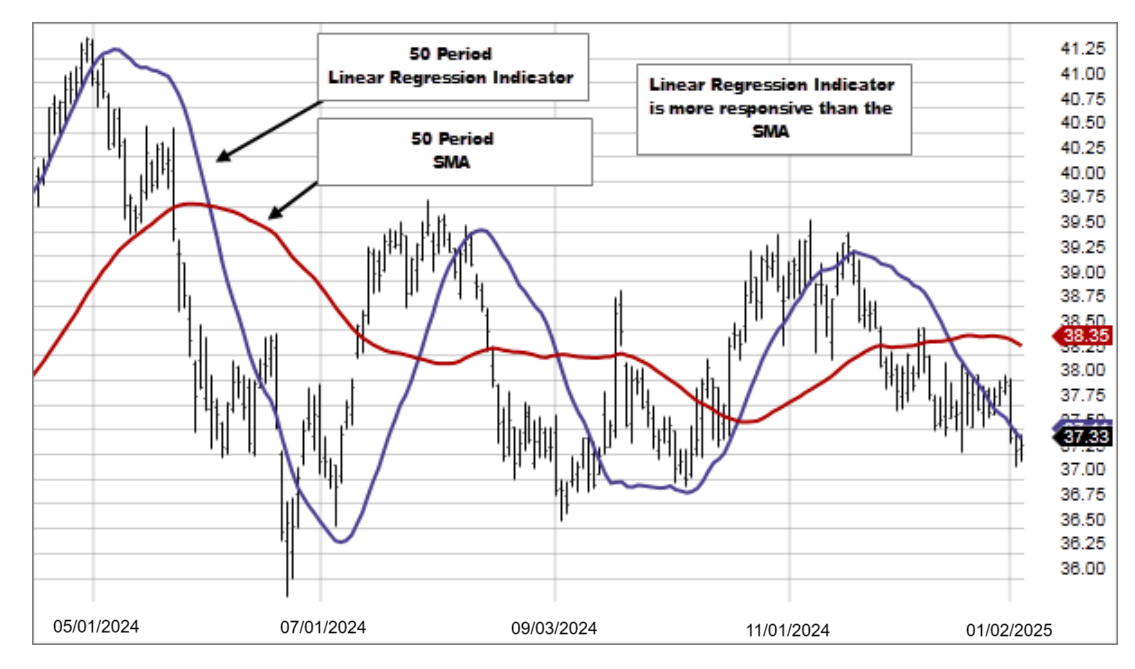
We will see how this "Works" for both classification and regression.

**Regression :**

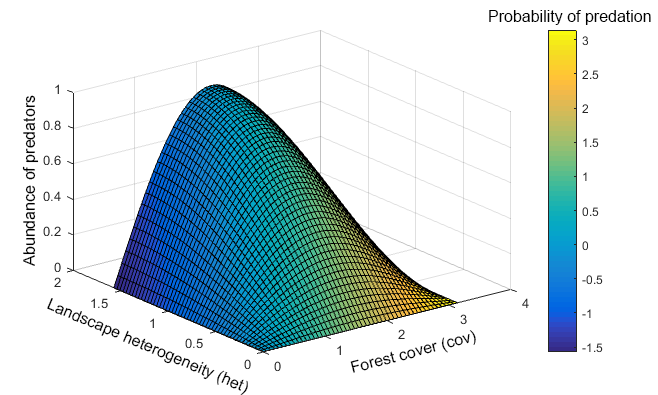
* It’s just a line of best fit.
  + Yes, the same thing you studied in grade school - except you’re not going to use a ruler graph paper and eyeball it.



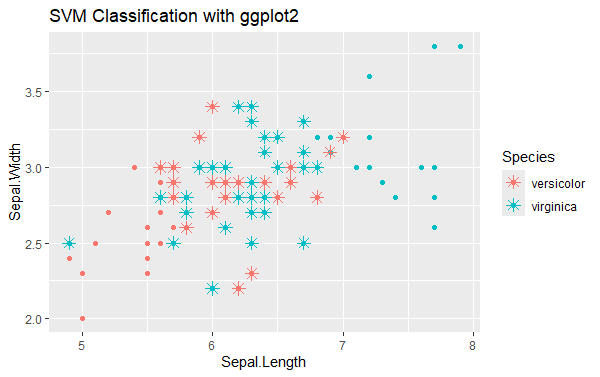
* 2 ways to make it harder
* #1 : Make it curvier than just a line.

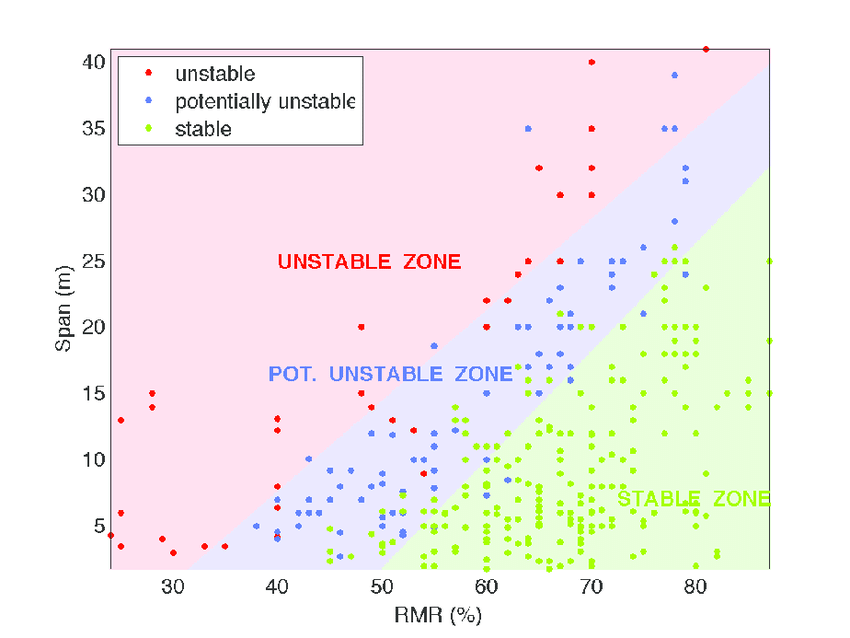


* #2 : make it multi dimensional

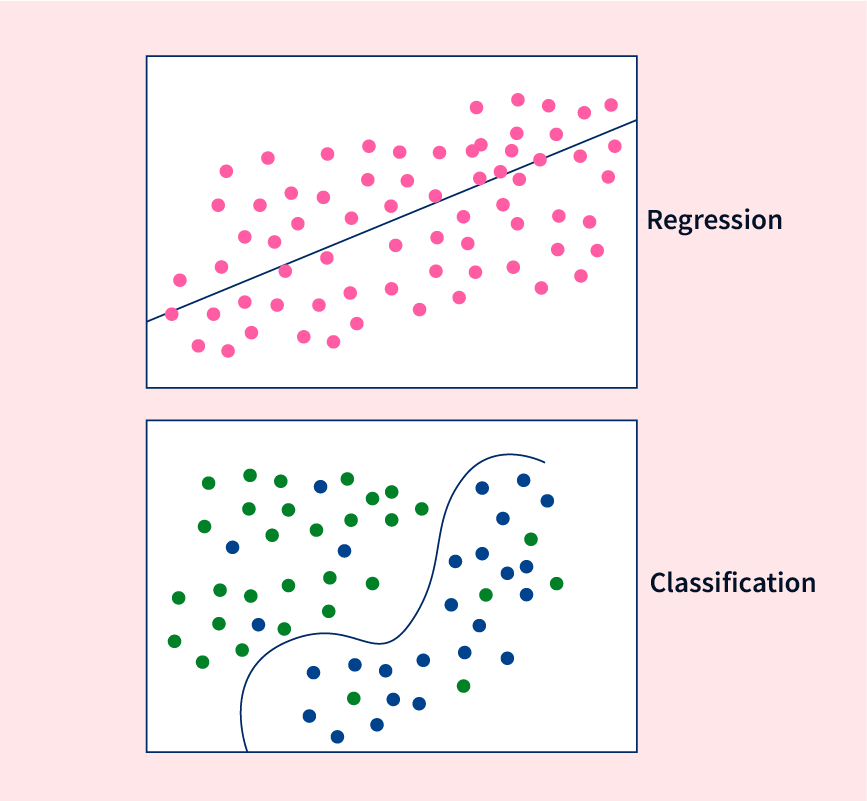


**CLASSIFICATION :**





**Machine Learning is nothing more than glorified curve-fitting.**



Recall our code examples:

X,Y = load\_data()

model = MakeModel() # instantiate the model.

model.fit(X, Y) # train the model

model.score(X, Y) # evaluate the model

model.predict(X) # Use model

When studying “Machine Learning” , what is the focus?

* It’s the algorithm, not the dataset you are using the algorithm on!
* The algorithm is the same no matter the dataset.
* The data is irrelevant.
* You don’t need more datasets.
* You need to understand the algorithm.

All machine learning interfaces are the same

* A blackbox with 2 tasks : Learn and make predictions

model = RandomForestClassifier()

model.fit(X,Y)

model.predict(X)

model = MLPClassifier()

model.fit(X,Y)

model.predict(X)

Caveats :

* You’ll learn about the exceptions to this “rule” in the future.
* We’ve already encountered score().
* For classification, it returns accuracy, but for regression, it returns the R2
* Unsupervised algorithms don’t use labels(there is no “Y”).
* More generally, models that do the same task have the same interface.

**SUMMARY**

* All data is the same. A random forest works the same way whether it’s a biology dataset or a finance dataset.
* All machine Learning interfaces are the same. Learn and Predict.

**COMPARING DIFFERENT MACHINE LEARNING MODELS.**

* Which model should I choose?
* The same lines of code are used regardless of the model I choose.
* Shouldn’t I always choose the most powerful model?
* How do I know which one is the most powerful?

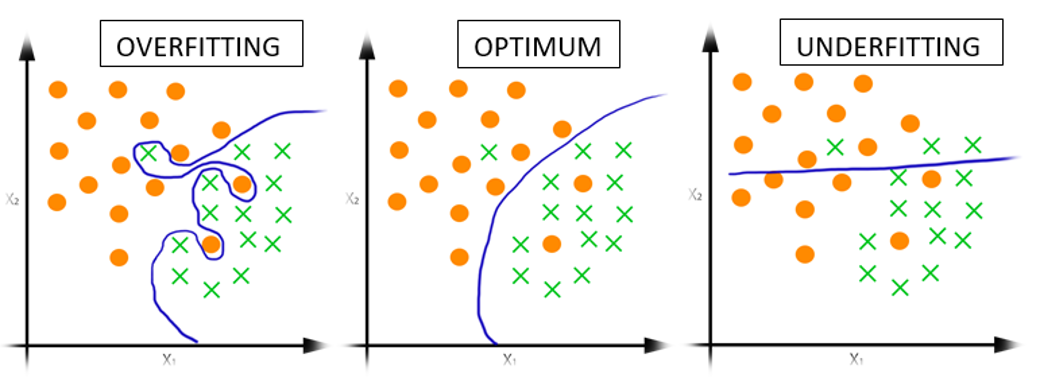


Can’t you just tell me what the BEST model is?

* Forget about this.
* There is no shortcut.
* Learn about the models, including their strengths and weaknesses.

THE APPROACH:

* Some general ideas and concepts.
* Nothing will replace learning about how these models work.

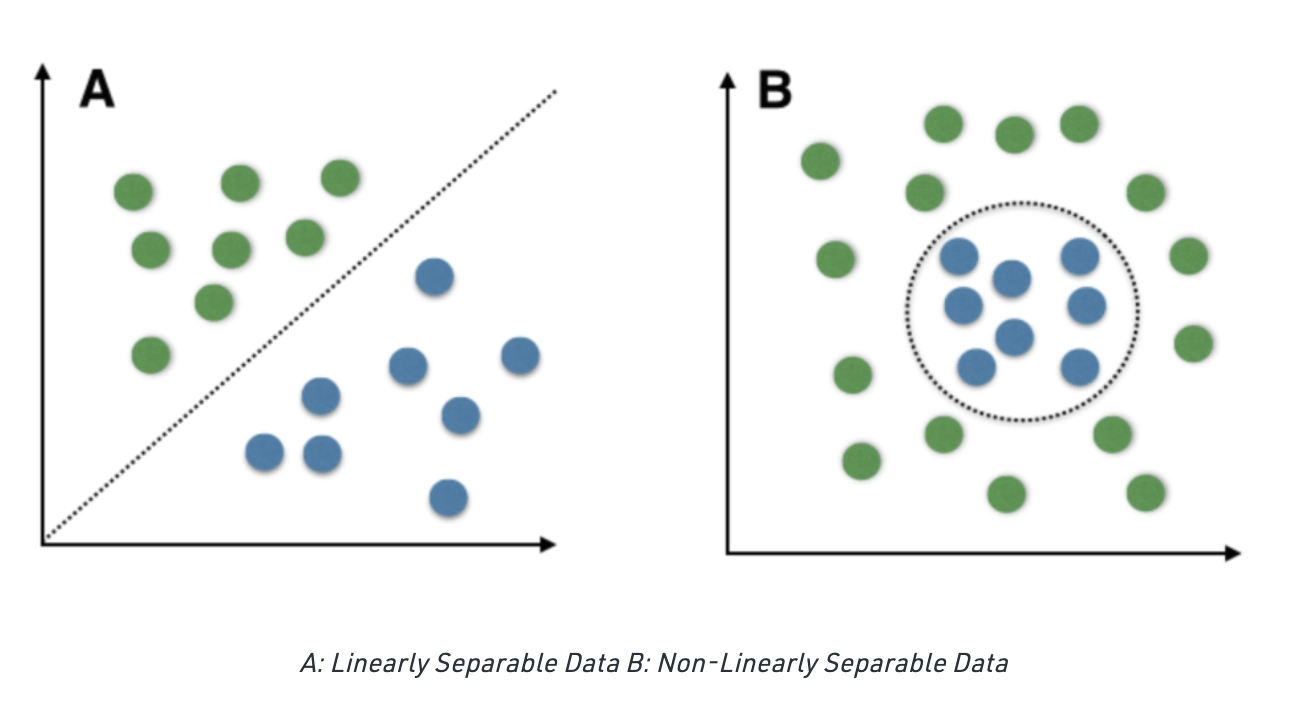


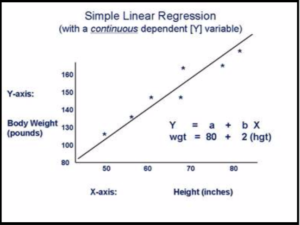
* Learn the algorithms, their pros and cons, where they fail, and succeed.

**LINEAR MODELS :**

Examples: Linear Regression, Logistic Regression.

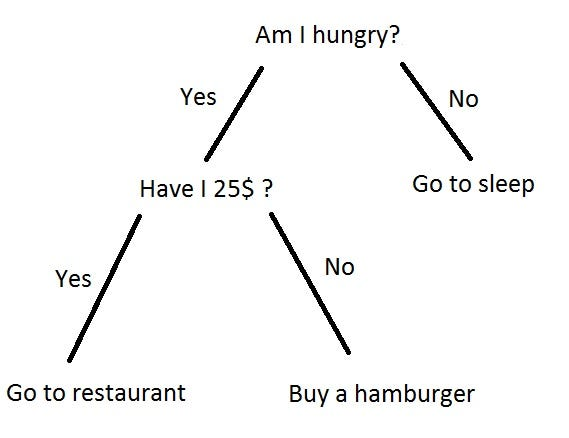
Note: You can look these up in the scikit-learn docs.



* Very easy to interpret.  
  

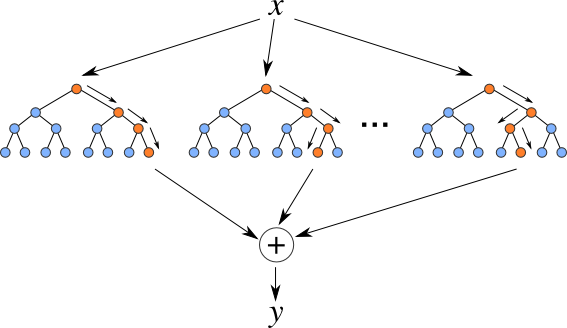
**Basic NonLinear Models:**

* Don’t be fooled! They are not necessarily “better” than a linear model.
* Examples: Naive Bayes, Decision Tree, K-Nearest Neighbor.



Ensemble Models

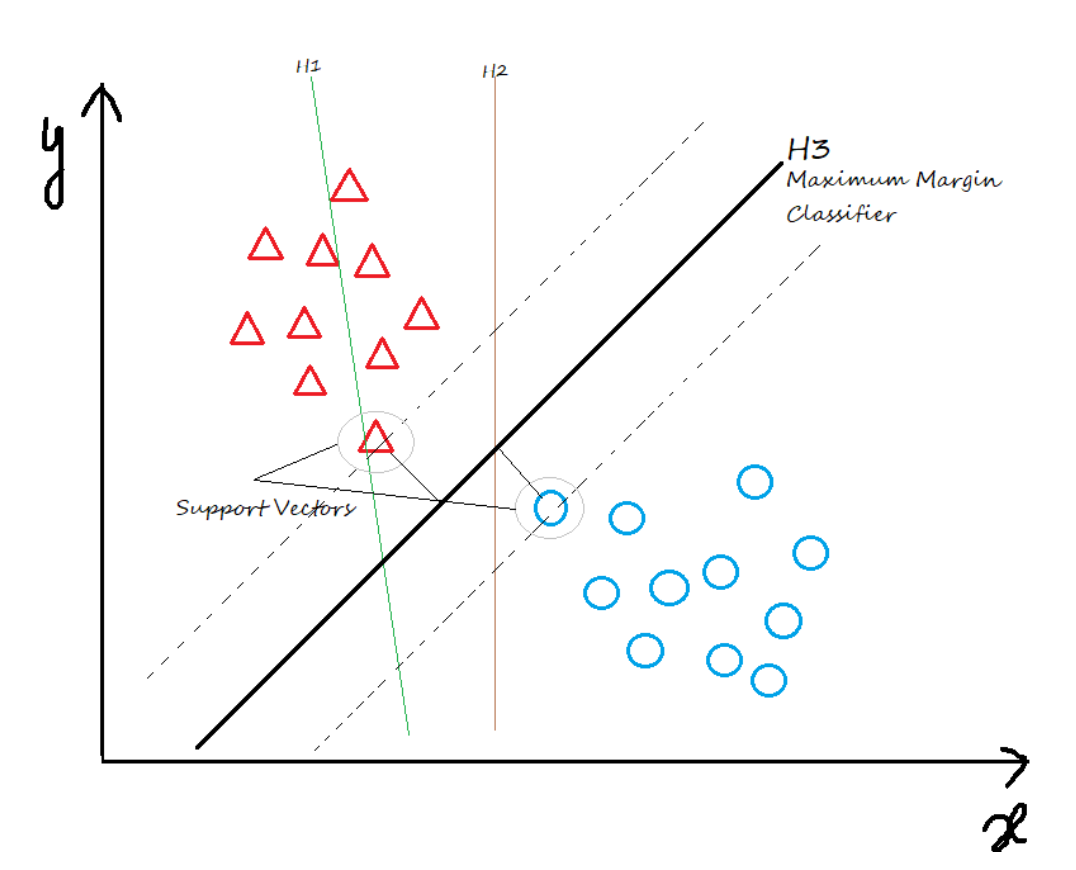
* Random Forest, AdaBoost, Extra Trees, Gradient Boosted Trees.



* Average the predictions from multiple trees.
* XGBoost has been used to win a significant number of Kaggle Contests.
* Ensemble methods are really powerful.
* Learn decision trees first before diving into ensemble.

**Support Vector Machine(SVM)**

* It was the “go-to” method for a long time.
* Today, that is deep learning, but SVM used to beat neural networks.
* Powerful and non-linear classifier, but they do not scale.
* Most datasets these days are too large which immediately disqualifies this model.



**DEEP LEARNING:**

* State of the art in Computer Vision(CV) and NLP(Natural Language Processing)

Images / Video / Text / Speech

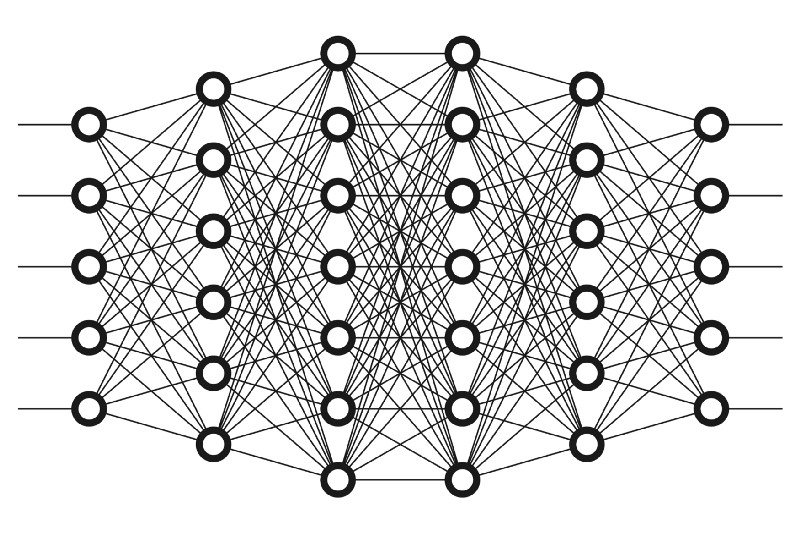
* Not “plug-and-play” (unlike Random Forest)

E.g.: If you try it on any random data, it may fail spectacularly.

* You wouldn’t normally use SKLearn.

Instead: Theano, Tensorflow, Keras, etc.

There were discussions against including MLP in SKLearn because of its non-plug-and-play nature.



**SUMMARY:**

* Do not take this table as gospel.
* ML is a field of experimentation, not philosophy.

