MACHINE LEARNING WITH PYTHON FOR SPACE WEATHER APPLICATIONS

by ITU Upper Atmosphere and Space Weather Laboratory

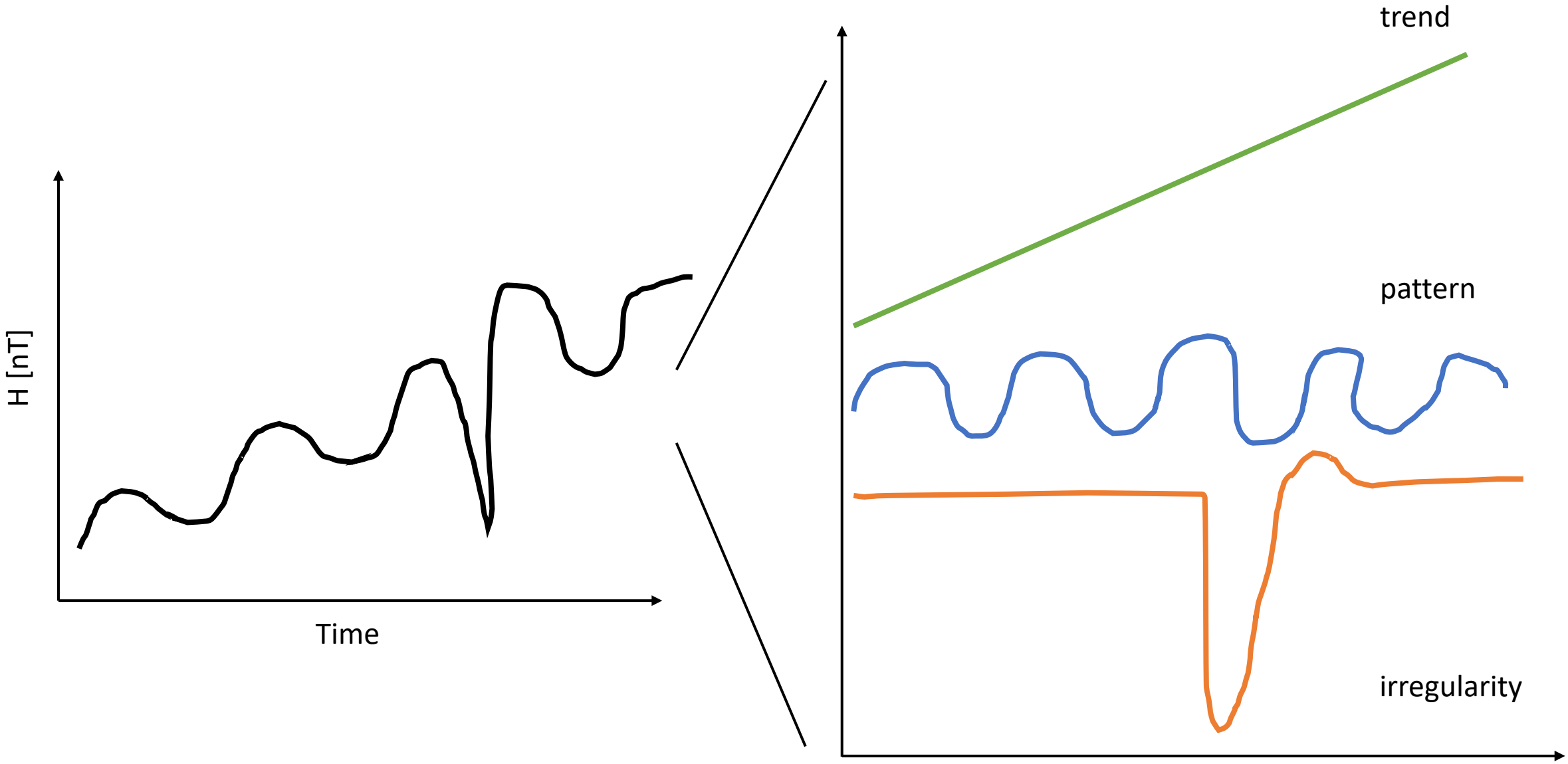
Lecture 2: Regression Techniques

Instructor: Dogacan Su Ozturk

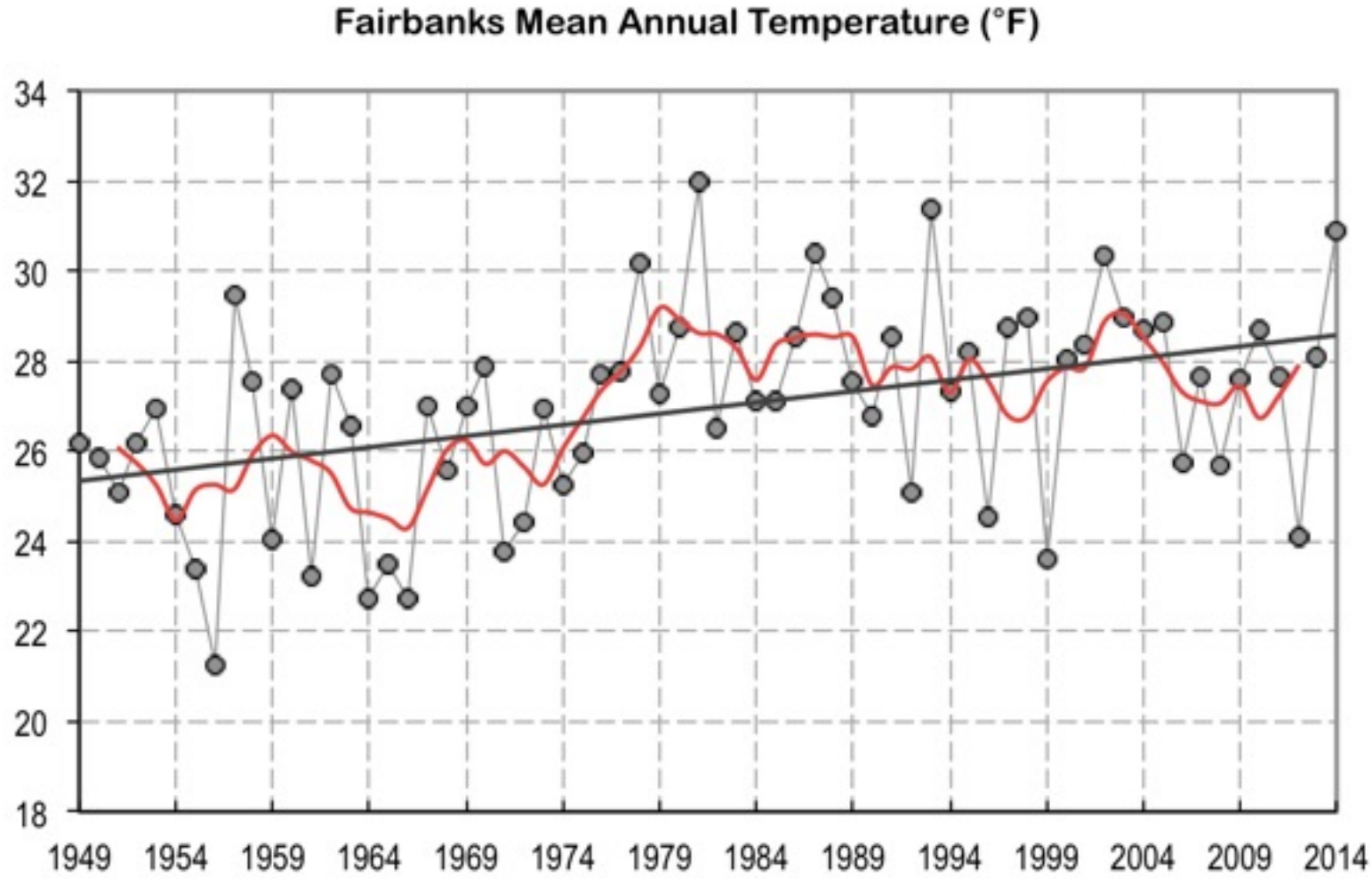
Contact: dsozturk@alaska.edu

11-12 May 2022 at Upper Atmosphere and Space Weather Laboratory

What is in a signal/data?



What is in a signal?



Source: climate.gi.alaska.edu

What is regression?

- Regression is the basic relationships pertaining to the data/signal.
- Regression aims to find the most simple relationship between two sets of data. It is the first step in Machine-Learning.

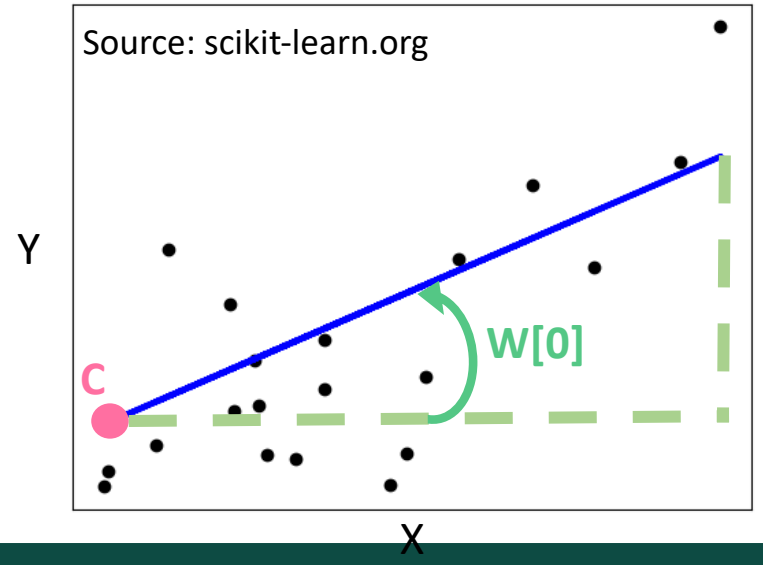
1. Linear Regression
2. Polynomial Regression
3. Logistic Regression

1. Linear Regression

Linear regression finds the parameters, weight and constant, to minimize mean squared errors between predictions and actual values.

$$\hat{y} = w[0]x[0] + w[1]x[1] + \cdots + w[n]x[n] + c$$

Here the $\hat{y}$ denotes the prediction model makes, w values are the weights, x are the input parameters (feature columns), and c is a constant.



C: Constant or intercept

W[0]: Slope or first order weight

Objective/cost function: $minimize\ \|y - Xw\|_2^2$

1. Linear Regression

Linear regression finds the parameters, weight and constant, to minimize mean squared errors between predictions and actual values.

$$\hat{y} = w[0]x[0] + w[1]x[1] + \cdots + w[n]x[n] + c$$

Here the $\hat{y}$ denotes the prediction model makes, w values are the weights, x are the input parameters (feature columns), and c is a constant.

Since middle school, you have used y=mx+n, which is the formula for a line.

```
from sklearn.linear_model import LinearRegression
```

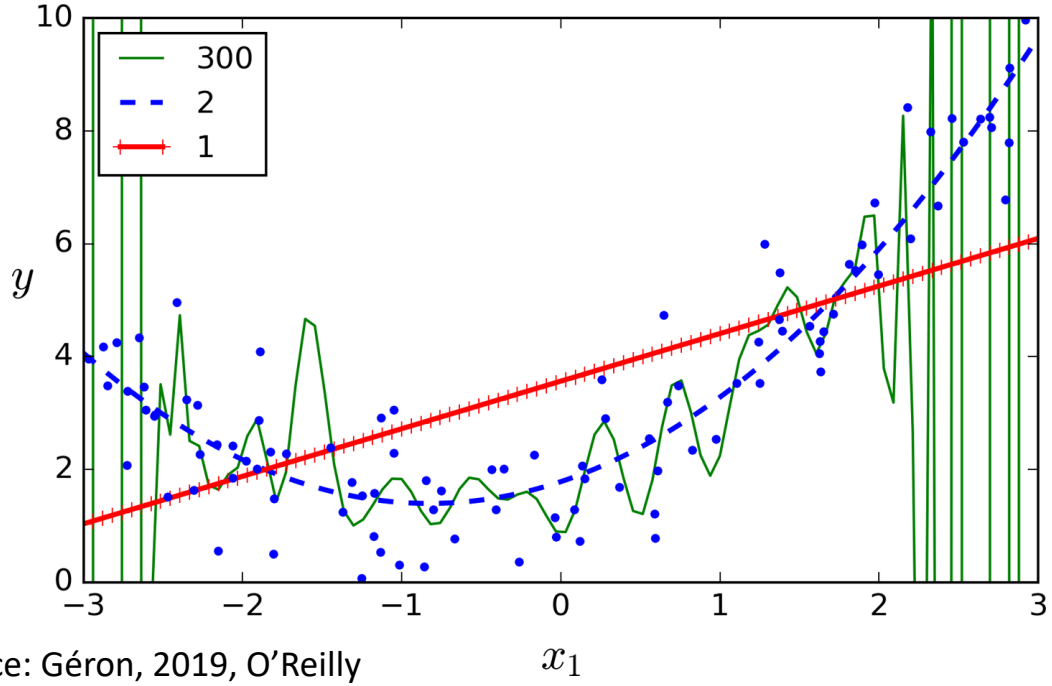
class sklearn.linear_model.LinearRegression(**, fit_intercept=True, normalize=False, copy_X=True, n_jobs=None, positive=False*)

2. Polynomial Regression

The linear representation can be improved by adding polynomial features to the fitting function.

$$\hat{y} = w_1 x + w_2 x^2 + c$$

More polynomial terms can be added to increase the order of regression.



Source: Géron, 2019, O'Reilly

But it is not always a good thing.

2. Polynomial Regression

The linear representation can be improved by adding polynomial features to the fitting function.

```
from sklearn.preprocessing import PolynomialFeatures
```

class sklearn.preprocessing.PolynomialFeatures(*degree=2*, ***, *interaction_only=False*, *include_bias=True*, *order='C'*)

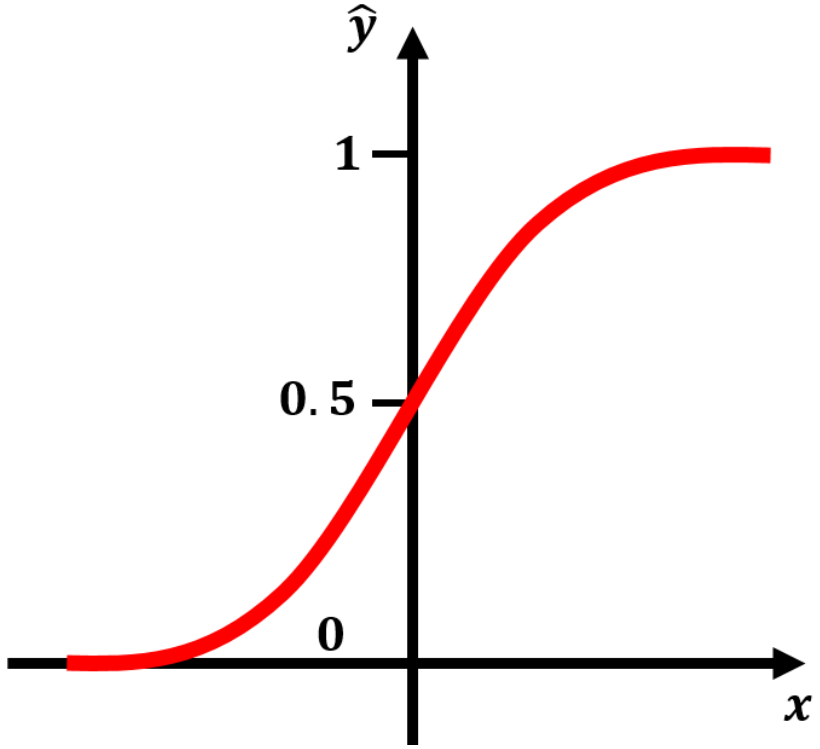
- degree is the order of polynomial

3. Logistic Regression

Logistic regression is commonly used to provide a probability (pass/fail, win/lose).

It is the binary classification analog of linear regression. Uses One vs Rest (OvR) for training.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




```
from sklearn.linear_model import LogisticRegression
class sklearn.linear_model.LogisticRegression(
penalty='l2', *, dual=False, tol=0.0001, C=1.0,
fit_intercept=True, intercept_scaling=1,
class_weight=None, random_state=None,
solver='lbfgs', max_iter=100, multi_class='auto',
verbose=0, warm_start=False, n_jobs=None,
l1_ratio=None)
```

Regularized Linear Regression Models

- What happens when we have a model trying too much?
  - “Overfitting”: When a model learns the training set too well, hence fails to perform well with test sets.

- What causes a model to overfit?
  - There could be too many dependencies (degrees of freedom).
  - The data size might not be sufficient.

We can work around overfitting in linear models by **regularization**.

- What do you think regularization will do?

1. Ridge Regression

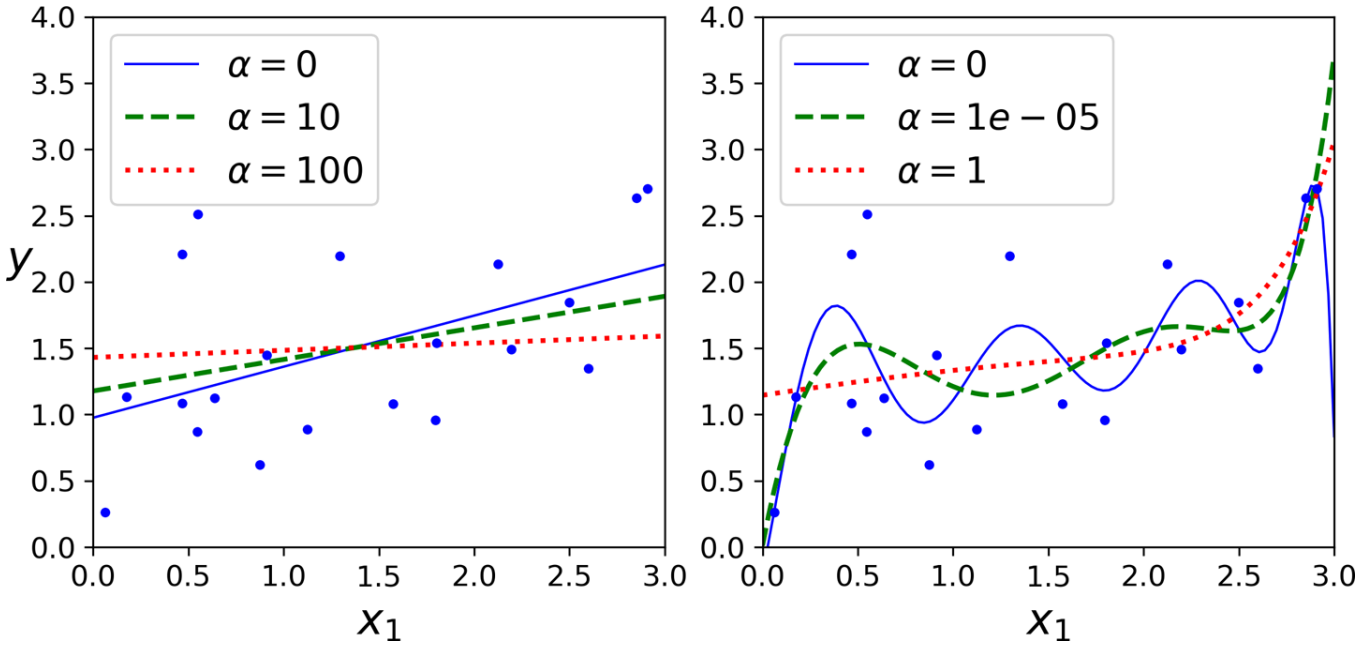
Ridge or Tikhonov regularization, uses a regularization term added to the cost function, aka the learning algorithm, to not only fit the data but to keep the weights as small as possible.

- Regularized terms are only added during training.
- Performance of the learning algorithm is evaluated with an unregularized performance measure.
- Scale the data before performing regression.
- Regularization term:

$$J(w) = MSE(w) + \alpha \frac{1}{2} \sum_{i-1}^{n} w_i^2$$

1. Ridge Regression

- Regularization term: $J(w) = MSE(w) + \alpha \frac{1}{2} \sum_{i-1}^{n} w_i^2$
- Objective/cost function: $\|y - Xw\|_2^2 + \alpha \|w\|_2^2$



from sklearn.linear_model import Ridge

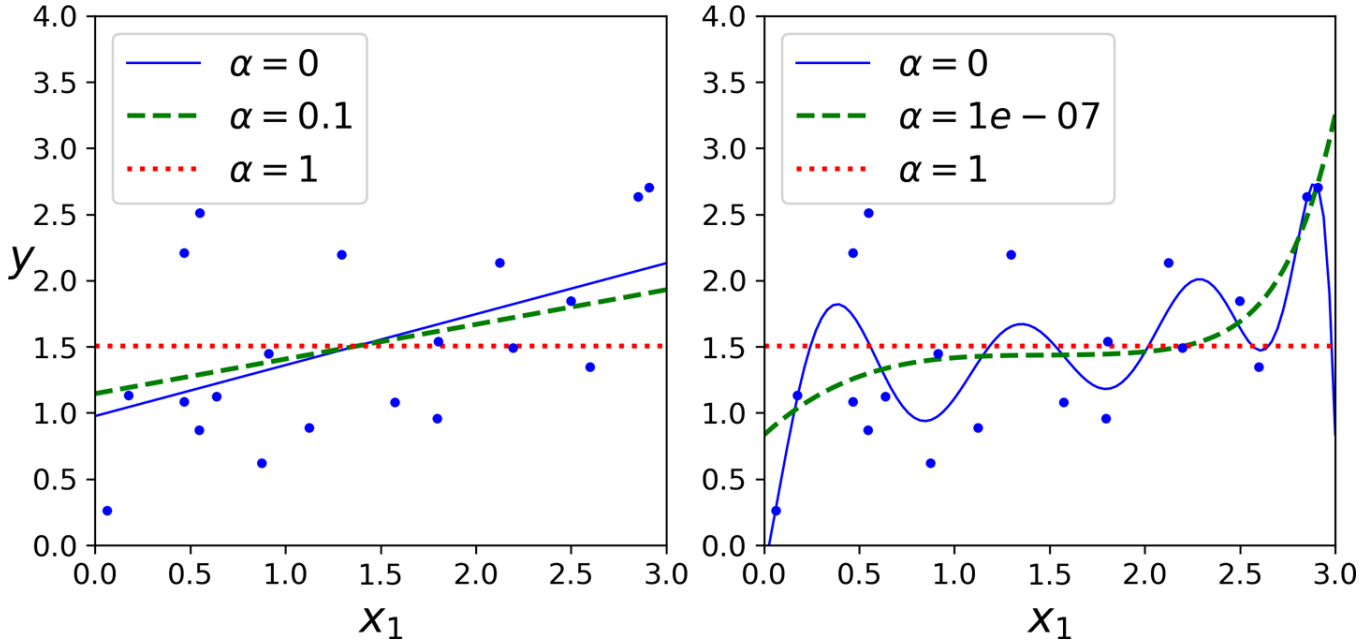
class sklearn.linear_model.Ridge(*alpha=1.0, *, fit_intercept=True, normalize=False, copy_X=True, max_iter=None, tol=0.001, solver='auto', random_state=None*)

- alpha is the regularization strength.
- tol is the stopping tolerance.
- solver is the optimization algorithm.

2. Lasso Regression

Least Absolute Shrinkage and Selection Operator Regression (Lasso) is very similar to Ridge, but uses l1 norm of the weight vector instead of the half of the square of the l2 norm.

- Regularization term: $J(w) = MSE(w) + \alpha \sum_{i-1}^{n} w_i$
- Objective/cost function: $\frac{1}{2n} \|y - Xw\|_2^2 + \alpha \|w\|_1^2$

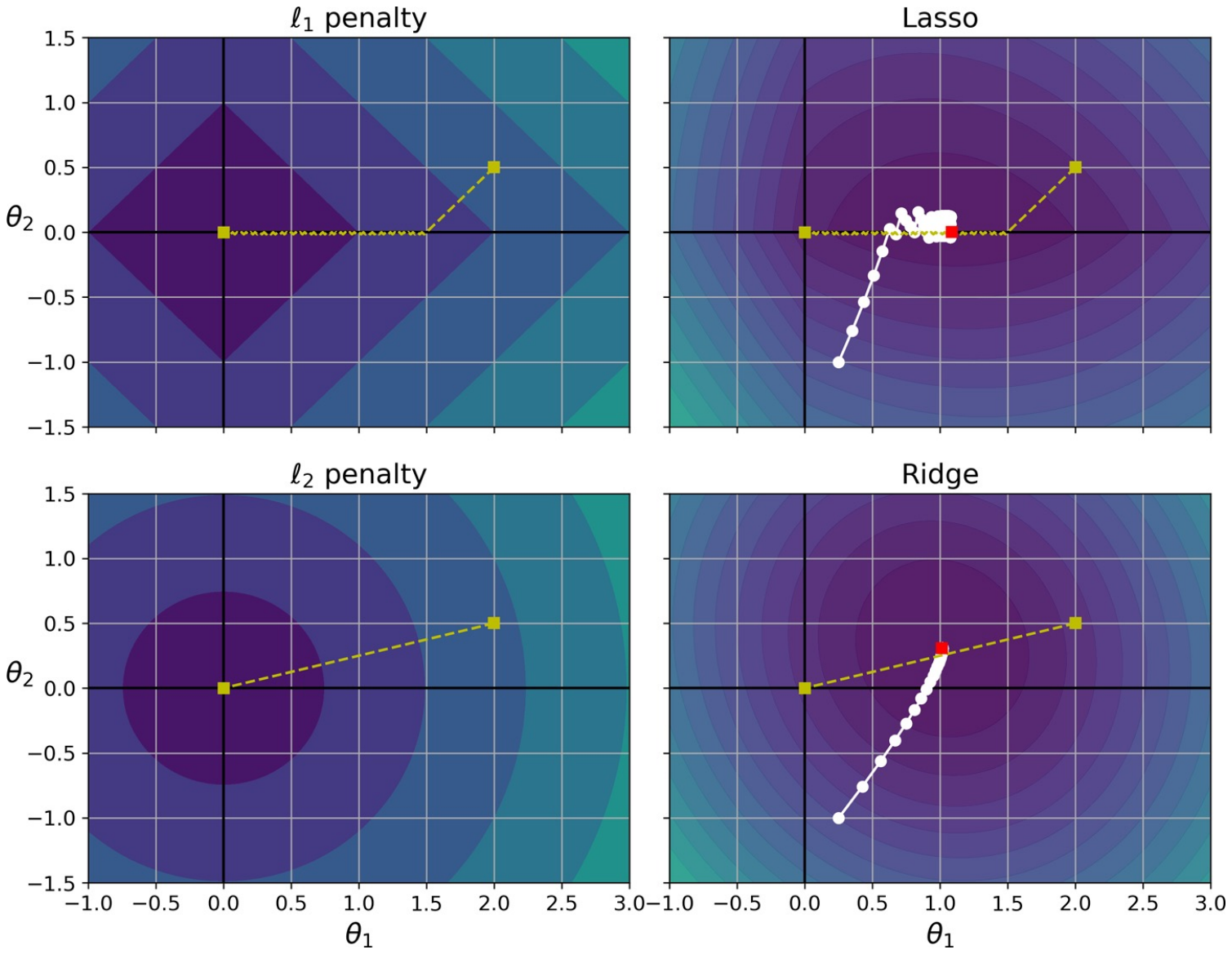


from sklearn.linear_model import Lasso

class sklearn.linear_model.Lasso(*alpha=1.0, *, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*)

- alpha is the regularization strength.
- tol is the stopping tolerance.

Lasso vs Ridge Regression



- Yellow lines are unregularized.
- Red square is the global optimum.
- Gradient descent algorithm is used for optimization.
- Can you explain the behavior of Lasso?
- Can you explain the behavior of Ridge?

3. Enet Regression: “The middle ground”

Elastic Net, or Enet, is the middle ground between Lasso and Ridge regression.

Regularization term: $J(w) = MSE(w) + r\alpha \sum_{i-1}^{n} w_i + \alpha \frac{1-r}{2} \sum_{i-1}^{n} w_i^2$

So when r=0 : Ridge Regression

when r=1 : Lasso Regression

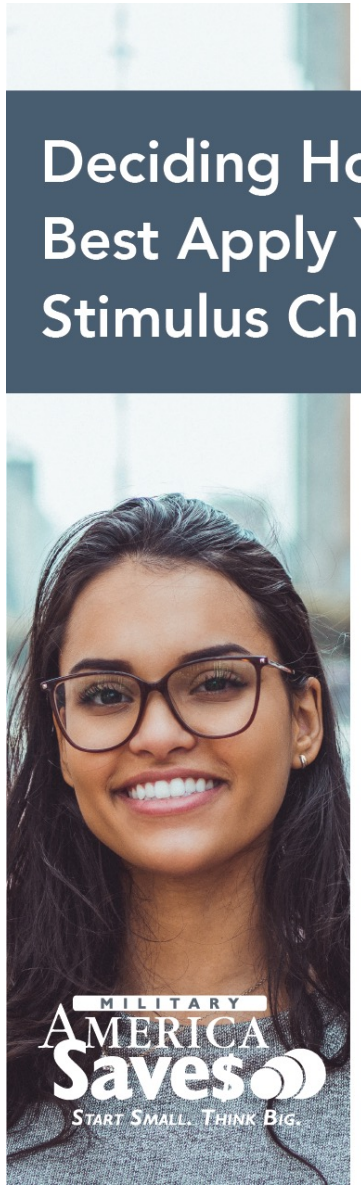
You need to decide based on how many features are useful: Lasso or Enet for few features.

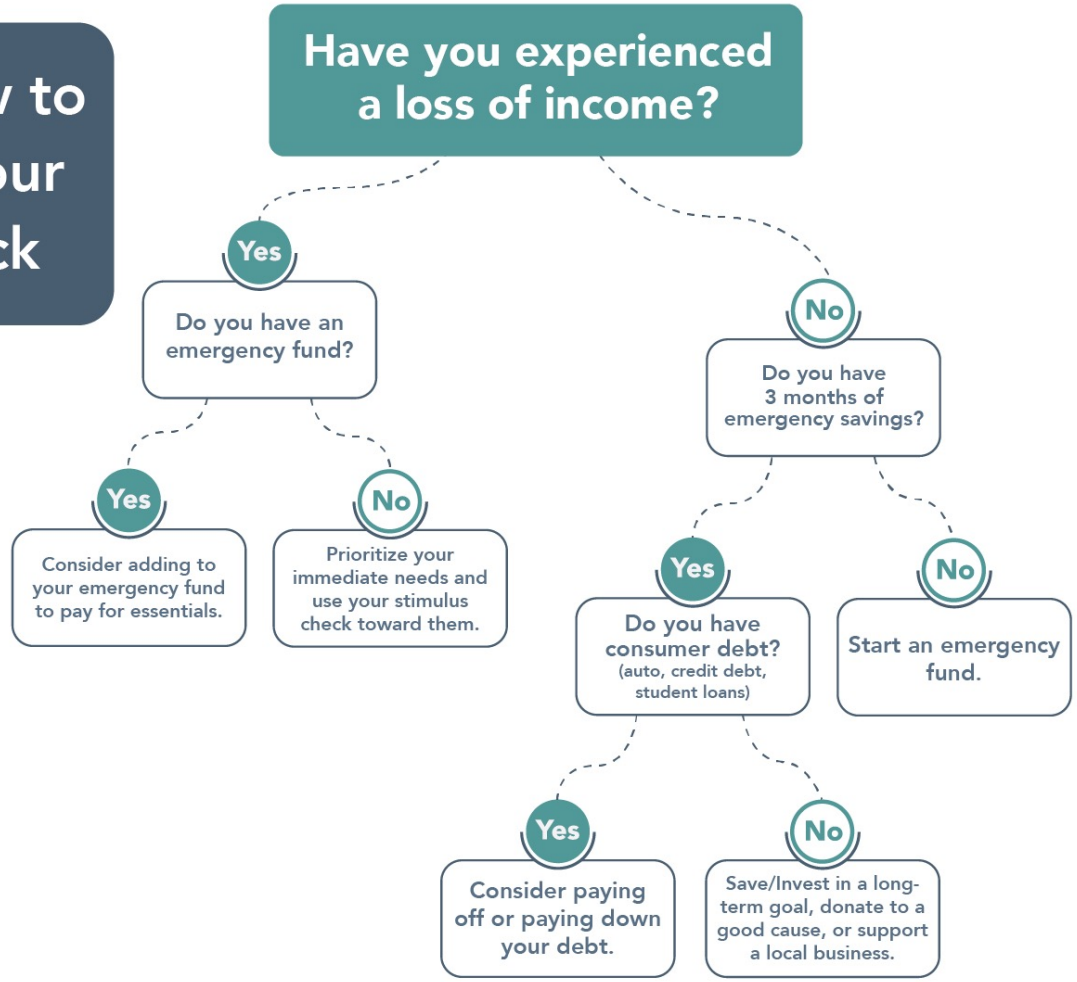
```
from sklearn.linear_model import ElasticNet
```

class sklearn.linear_model.ElasticNet(*alpha=1.0, *, l1_ratio=0.5, fit_intercept=True, normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*)

4. Decision Trees

Deciding How to Best Apply Your Stimulus Check





MilitarySaves.org

HUMAN RESOURCES

Exposure to COVID-19

EXPOSURE CRITERIA:

Contact within 6 feet for 10 minutes or more with a person with suspected or confirmed COVID-19.

EXAMPLES INCLUDE:

- Working together in a close proximity with someone with COVID-19 symptoms or has tested positive for COVID-19.
- Being in a meeting with someone with COVID-19 symptoms or has tested positive for COVID-19 with no social distancing.
- Traveling in the same vehicle with someone with COVID-19 symptoms or has tested positive for COVID-19.
- Living with someone with COVID-19 or has tested positive for COVID-19.

Have you been exposed to someone who has COVID-19 symptoms or has tested positive for COVID-19?

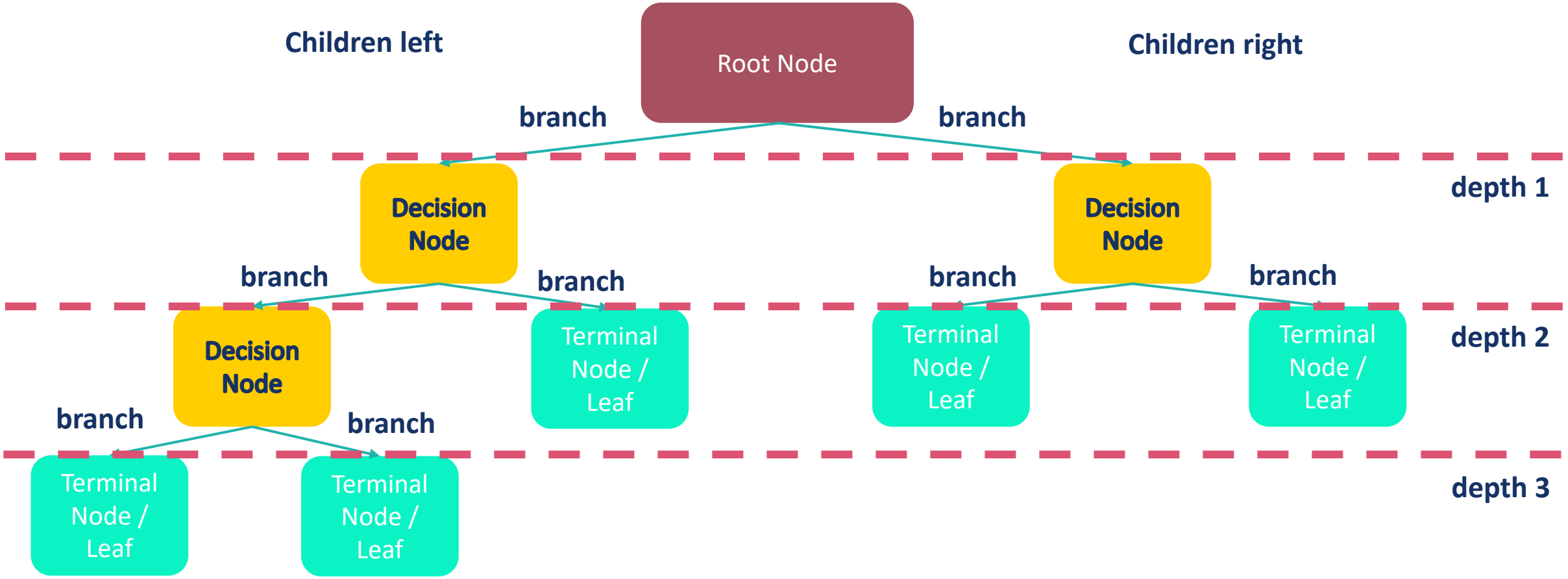
- NO → **No medical attention or quarantine needed.** Monitor for symptoms. Continue to wear a mask, socially distance, and wash your hands.
- YES → **Are you experiencing COVID-19 symptoms?**
  - NO → **Self-isolate/quarantine for 14 days** before returning to work and get tested. Continue to wear a mask, socially distance and wash your hands.
  - YES → **Have you been tested for COVID-19?**
    - NO → **Self-isolate/quarantine and get tested.** If COVID-19 test is positive, you may return to work 10 days from onset of symptoms ONLY if you are fever-free (without use of fever-reducing medication) and have improvement of symptoms for at least 24 hours
    - YES → **Did you test positive for COVID-19?**
      - NO → **Self-isolate/quarantine for 14 days after your last contact** with the person with COVID-19 before returning to work. you may return to work 10 days from onset of symptoms ONLY if you are fever-free (without use of fever-reducing medication) and have improvement of symptoms for at least 24 hours
      - YES → **Contact your supervisor and fill out the Self-isolation form at https://covid19notice.questionpro.com/.** If COVID-19 test is positive, you may return to work 10 days from onset of symptoms ONLY if you are fever-free (without use of fever-reducing medication) and have improvement of symptoms for at least 24 hours

systemhr@tennessee.edu • 865-974-8847

Tennessee Department of Health Resources for Colleges and Universities. July 31 2020
Tennessee Department of Health. What to Do If Your Test Results are Negative. April 2020.

4. Decision Trees: Principles

- Decision trees are widely used for both Regression and Classification.
- They learn through if/else questions to arrive at a decision.



4. Decision Trees: Usage

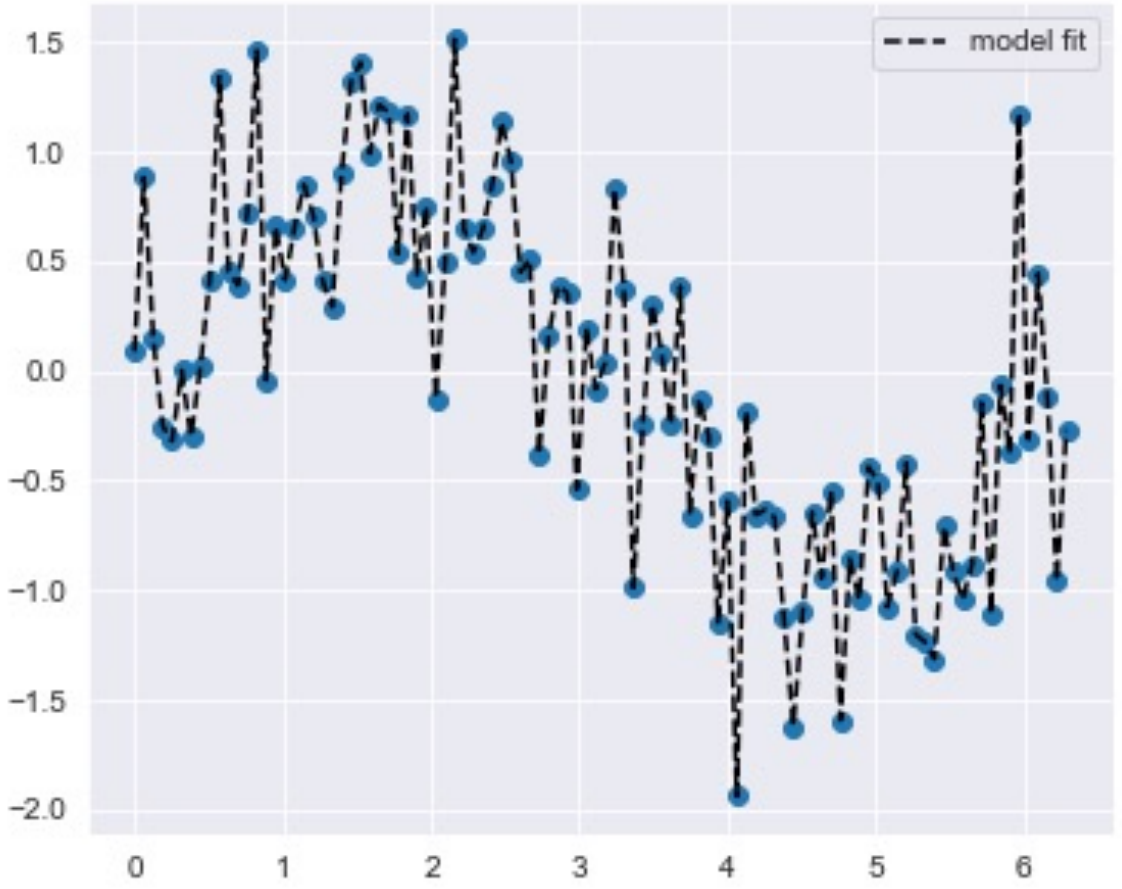
- Decision trees are widely used for both Regression and Classification.
- They learn through if/else questions to arrive at a decision.
- Each node has a certain amount of sample in it.
- Decision trees are:
  - Very likely to overfit.
  - Very likely to be limited by the range of the data.
- Decision trees can predict both Regression and Classification.
- In this class we will focus on the Regression.
- from sklearn.tree import DecisionTreeRegressor

  class sklearn.tree.DecisionTreeRegressor(*, *criterion='mse'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *ccp_alpha=0.0*)

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

4. Decision Trees: Constraints

If left unconstrained decision tree tends to make a decision for every node.



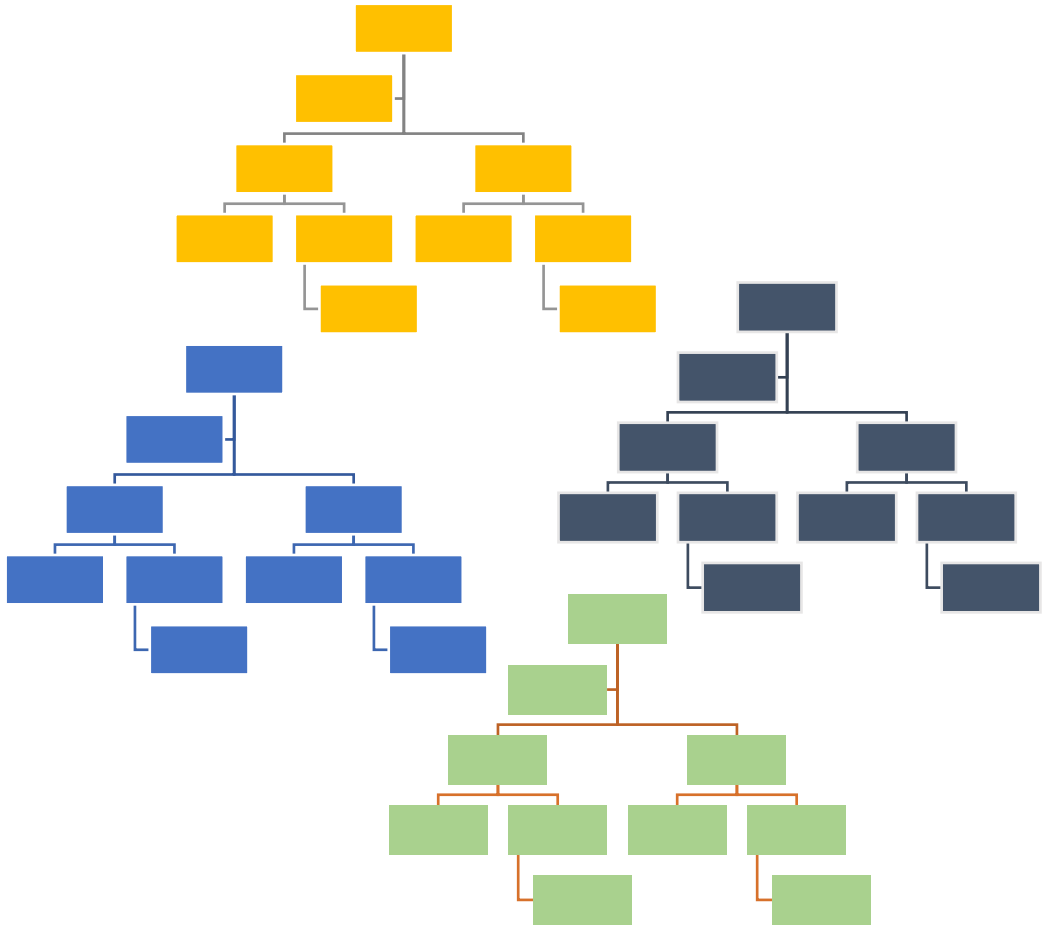
What can we tune in a decision tree?

- max_depth
- min_samples_leaf
- min_samples_split
- min_weight_fraction_leaf

5. Ensemble Decision Trees: Random Forest

- Another way to overcome overfitting is growing a forest of random decision trees with constraints.
- Random forest trains multiple decision trees in parallel and ensembles their predictions into a single decision tree.
- Each decision tree is trained with a random subset of observations.
- Random forests can be further constrained.
- from sklearn.ensemble import RandomForestRegressor

class sklearn.ensemble.RandomForestRegressor(*n_estimators=100, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None*)



The Good, the Bad, the Ugly about Random Forests

The Good:

No scaling needed.

No intense hyperparameter tuning needed.

One of the most used ML techniques.

The Bad:

There can be too many trees, turning the White Box to a Gray Box.

Too much dependence on random states.

Don’t perform well when data has too many dimensions or too many sparse entries.

The Ugly:

Computationally heavy, but can be parallelized.

Maximum features needs to be adjusted for different applications.