Machine Learning Bootcamp

Dexter Fichuk - TD Lab

ANACONDA®

https://www.continuum.io/downloads

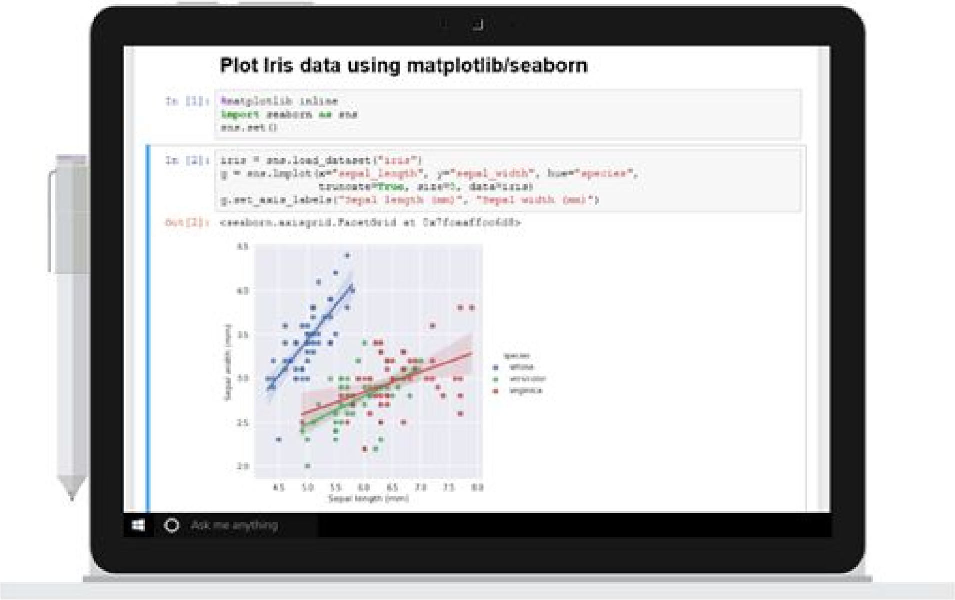
http://tiny.cc/conda

Microsoft Azure Notebooks Preview

Sign In

Libraries What's New Status Help

Now Featuring: Python Data Science Handbook by Jake VanderPlas



Plot Iris data using matplotlib/seaborn

```
In [1]: %matplotlib inline
        import seaborn as sns
        sns.set()

In [2]: iris = sns.load_dataset("iris")
        g = sns.lmplot(x="sepal_length", y="sepal_width", hue="species",
                       truncate=True, size=5, data=iris)
        g.set_axis_labels("Sepal length (mm)", "Sepal width (mm)")

Out[2]: <seaborn.axisgrid.FacetGrid at 0x7fcaaffcc6d8>
```

Interactive coding in your browser

Free, in the cloud, powered by Jupyter

Get Started

Powerful

Numerous Charting

Built for

ML is a practical subset of AI. It involves taking a set of training data and building a model that can either classify new input, or predict the output of it aka predicting the future.

Types of ML

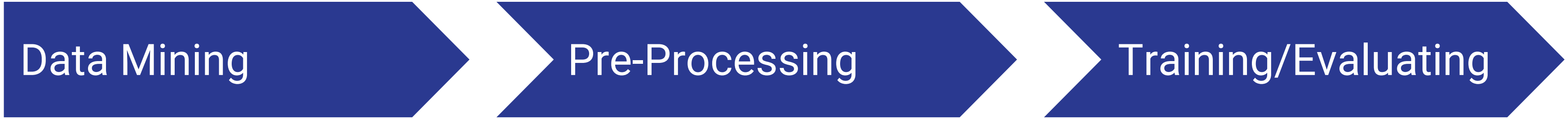
Classification

Classification can if something is true or false (1 or 0), could be classifying a picture as a cat or dog or classifying how old someone is.

Regression

Predicting a value based on the input, could be predicting a credit score, the temperature, stocks, or anything where the there is multiple output options.

The Flow



Data Mining → Pre-Processing → Training/Evaluating

Collecting a Dataset

Mostly doing supervised learning here, meaning that our training set already has outcome labels.

Could also involve creating simulation datasets (transactions, etc.)

Cleaning the Data

Detecting the values/features (columns) that matter, removing ones that don't.

Normalizing/Scaling data

Sometimes plotting different graphs to find trends.

Building a Complete Model

Involves testing different algorithms/hyperparameters to find the highest accuracy for the dataset.

Data-Mining

Data Mining

- Try searching sites like kaggle, open data government sites, and the UCI machine learning repository besides Google.
- If simulating the data, make sure to research reasonable ranges and occurrences of different cases.

Pre-Processing

Pre-Processing

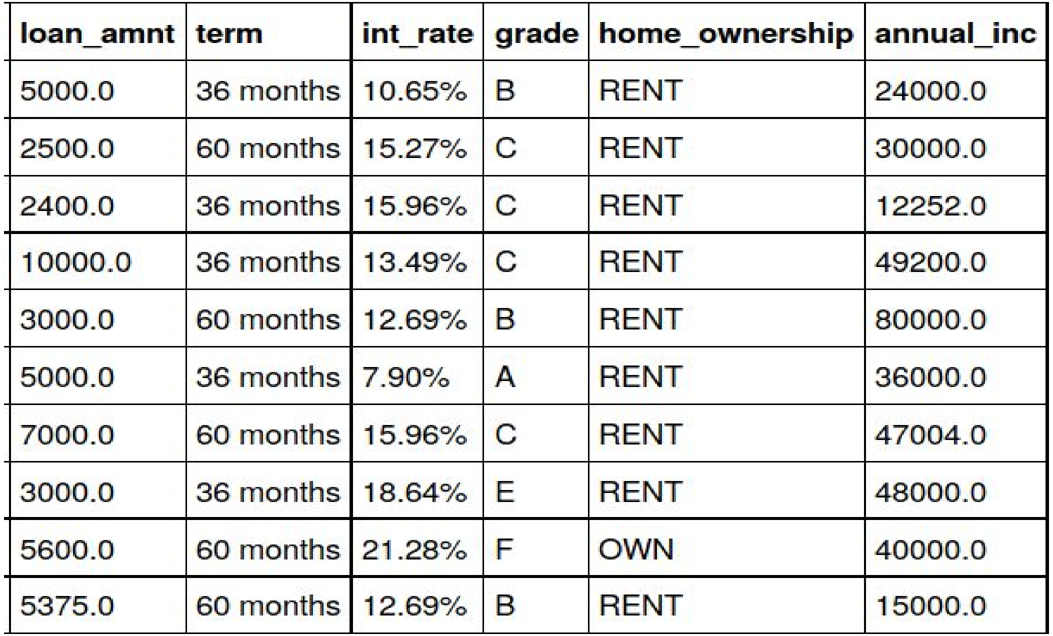
Cleaning your dataset

Involves tasks such as:

- Removing irrelevant features
- Deciding what to do with null entries (replace with column avg., remove row, etc.)
- Scaling inputs and transforming text fields to numerical representations.

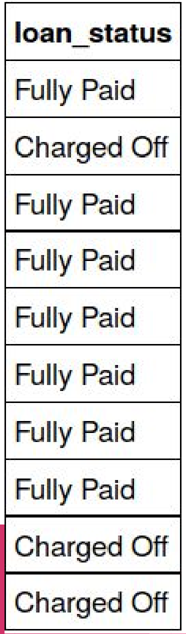
Data Variables

X (input data)



loan_amnt	term	int_rate	grade	home_ownership	annual_inc
5000.0	36 months	10.65%	B	RENT	24000.0
2500.0	60 months	15.27%	C	RENT	30000.0
2400.0	36 months	15.96%	C	RENT	12252.0
10000.0	36 months	13.49%	C	RENT	49200.0
3000.0	60 months	12.69%	B	RENT	80000.0
5000.0	36 months	7.90%	A	RENT	36000.0
7000.0	60 months	15.96%	C	RENT	47004.0
3000.0	36 months	18.64%	E	RENT	48000.0
5600.0	60 months	21.28%	F	OWN	40000.0
5375.0	60 months	12.69%	B	RENT	15000.0

y (output label)



loan_status
Fully Paid
Charged Off
Fully Paid
Fully Paid
Fully Paid
Fully Paid
Fully Paid
Fully Paid
Charged Off
Charged Off

Representing Data

outputs / labels

one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

one feature (column)

Categorical Variables

$$\begin{array}{ccc} \text{“red”} & \text{“green”} & \text{“blue”} \\ \left(\begin{matrix} 1 \\ 0 \\ 0 \end{matrix}\right. & \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} & \left.\begin{matrix} 0 \\ 0 \\ 1 \end{matrix}\right) \end{array}$$

If you mapped:
{red->0, green->1, blue->2}, a linear relationship would be imposed between the values, therefore it is better to perform a categorical transformation on types of text fields that are options, rather than ratings.

A field such as 5-star ratings could be scaled as 0, 0.25, 0.5, 0.75, and 1.

Scaling Inputs

Movie Reviews (/5)

Before Scaling

$$\begin{pmatrix} 1 \\ 3 \\ 5 \\ 2 \end{pmatrix}$$

After Scaling

$$\begin{pmatrix} .2 \\ .6 \\ 1 \\ .4 \end{pmatrix}$$

A field such as 5-star ratings could be scaled as 0, 0.20, 0.4, 0.6, 0.8 and 1.

Whether an input should be scaled is largely dependent on the learning algorithm you’re selecting.

Scaling is great for algorithms such as Neural Networks and SVMs.

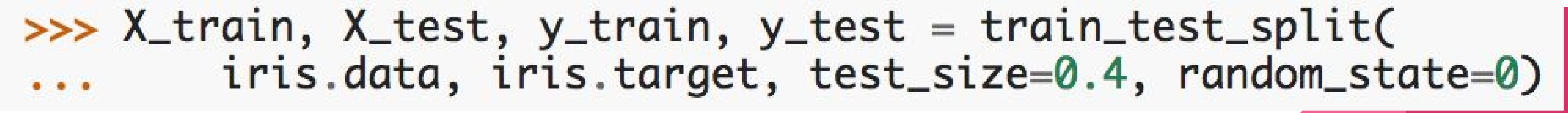
Training A Model

Splitting the Data

Simple Splitting

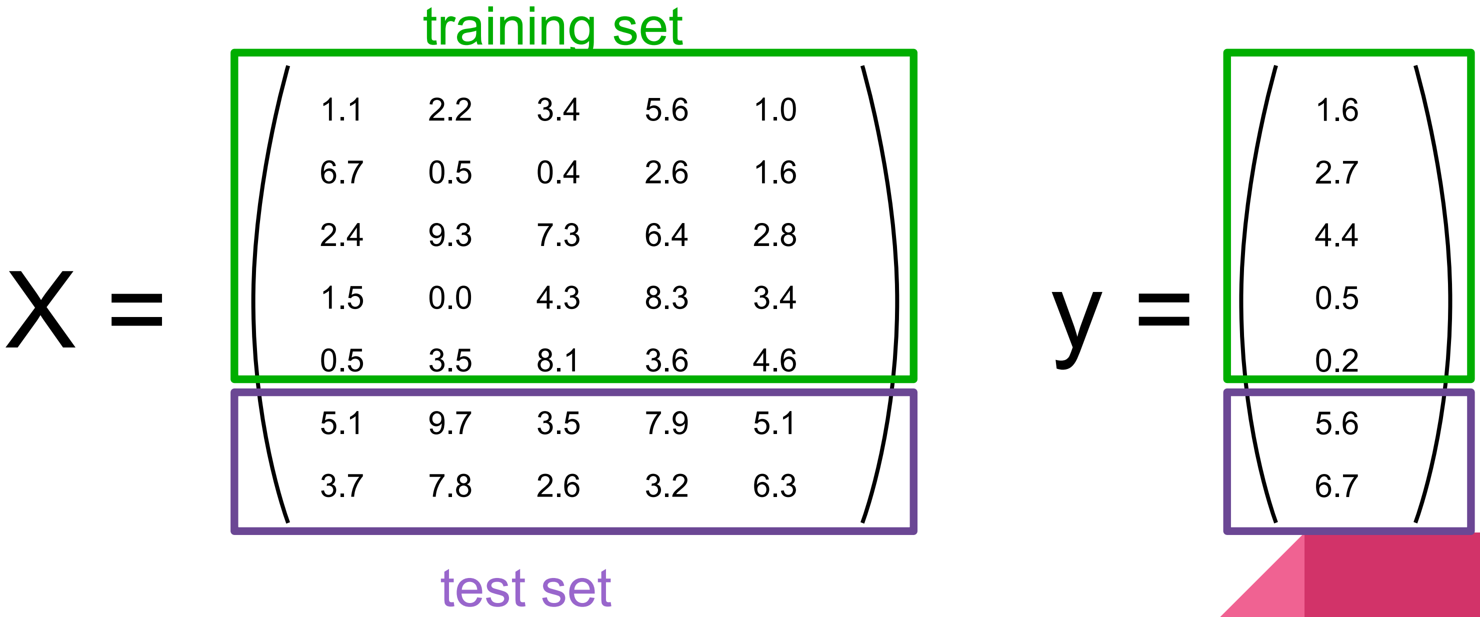
The gold standard of evaluating a model is by testing it on data it has not seen in training. This means taking a percentage out of the training set (typically 10-20%), and running it through the trained model to see it's accuracy.

It's important to set a random state for the split, so you can evaluate your model on the same training set every time, making your results reproducible.



```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     iris.data, iris.target, test_size=0.4, random_state=0)
```

Training and Testing Data



training set

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

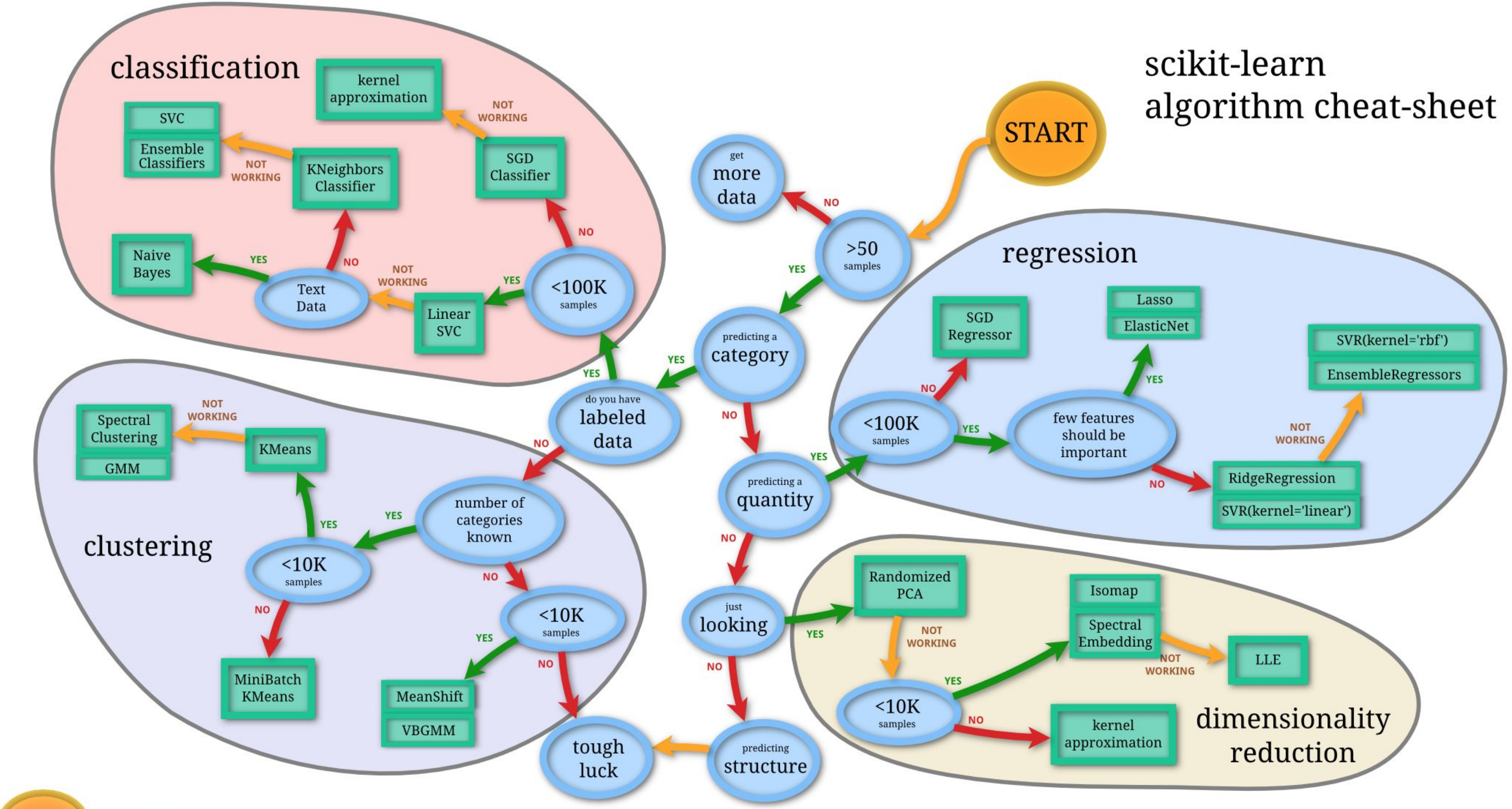
test set

Picking an Algorithm

There are many algorithms to choose from, but lucky for us, Scikit-Learn has a ton built in and can be used mostly interchangeably, meaning that different classifiers can be used in a loop then plotted to compare performance.

Each algorithm has better use cases and could outperform others for a specific task. There is no master algorithm.

Scikit-Learn has a great cheat sheet for picking algorithms.



Back

Source: https://goo.gl/liKQbr

Parameter Tuning

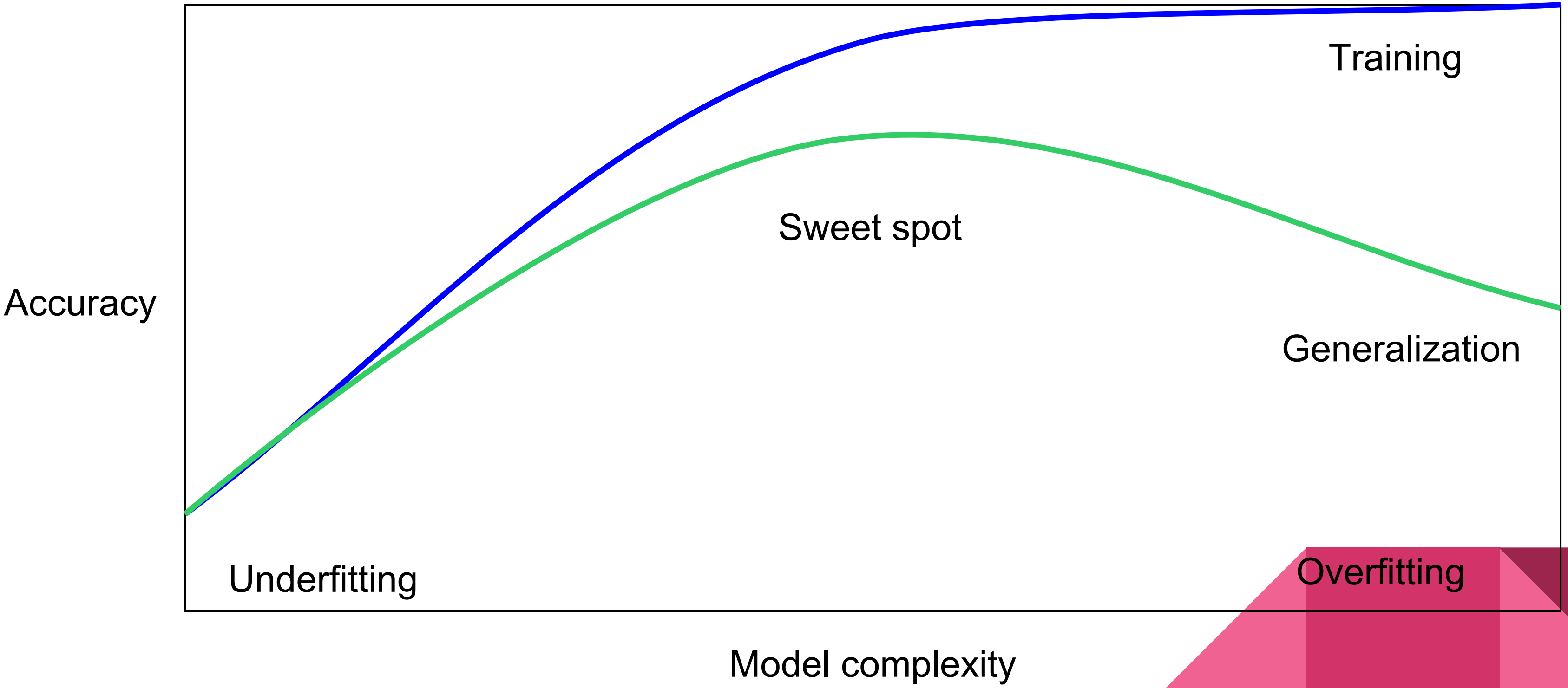
Each Algorithm has a variety of parameters, there are a few ways of finding optimal ones.

- GridSearch
- RandomSearch
- Hyperopt

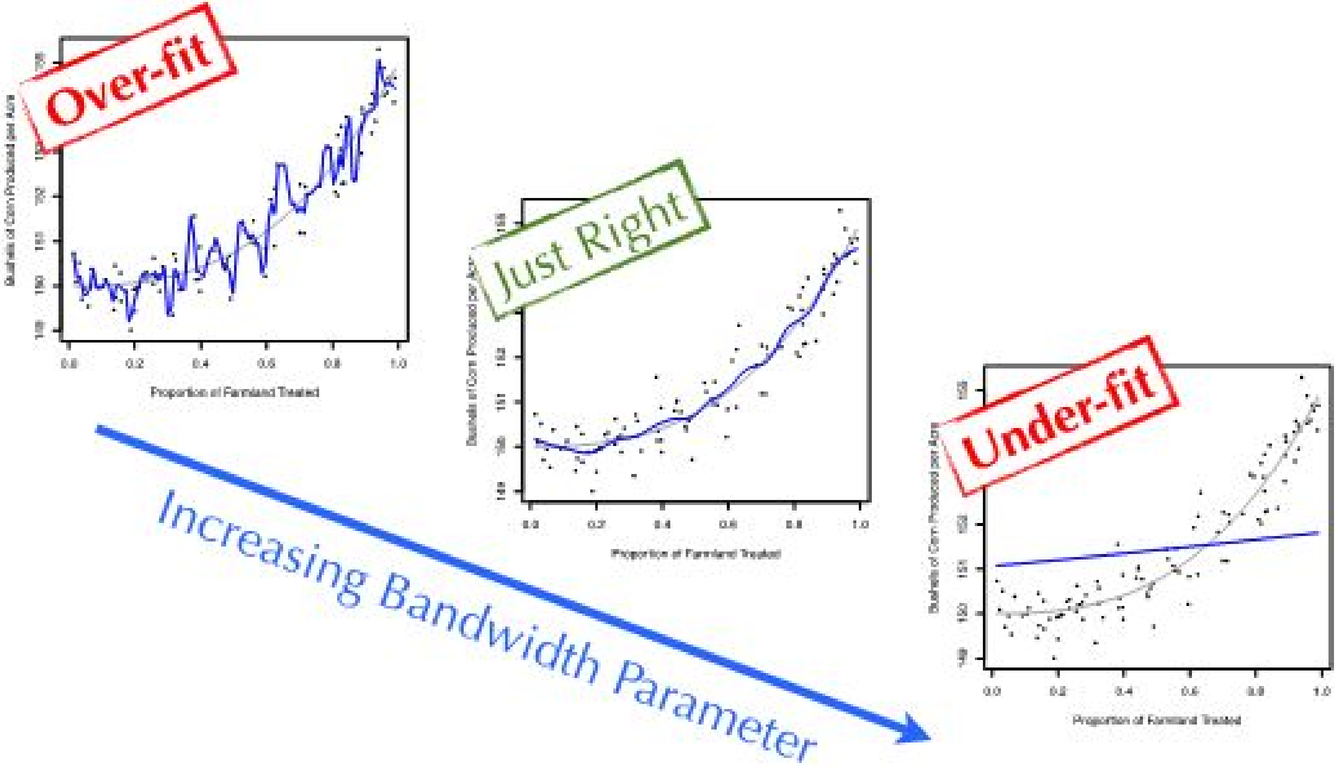
My Fave Alg’s

- Gradient Boosting (XGBoost, LightGBM)
- Random Forests
- Genetic Algorithms
- Support Vector Machines
- Multi-Layer Perceptron (NN)
- Neural Networks (NNs)
- Convolutional Neural Networks

Overfitting and Underfitting



Overfitting and Underfitting

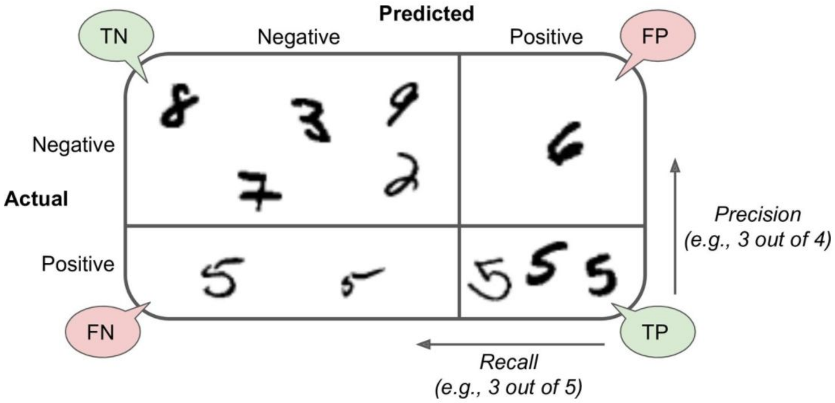


Recall vs Precision

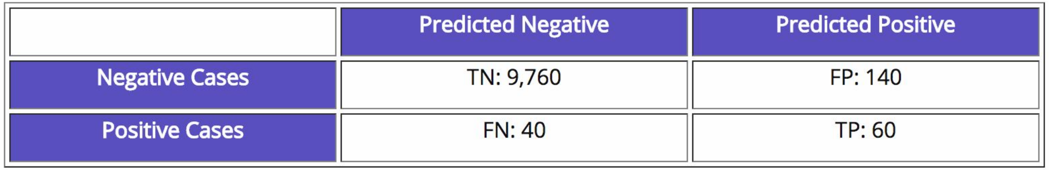
What percent of your predictions were correct?
The "accuracy" was (9,760+60) out of 10,000 = 98.2%

What percent of the positive cases did you catch?
The "recall" was 60 out of 100 = 60%

What percent of positive predictions were correct?
The "precision" was 60 out of 200 = 30%

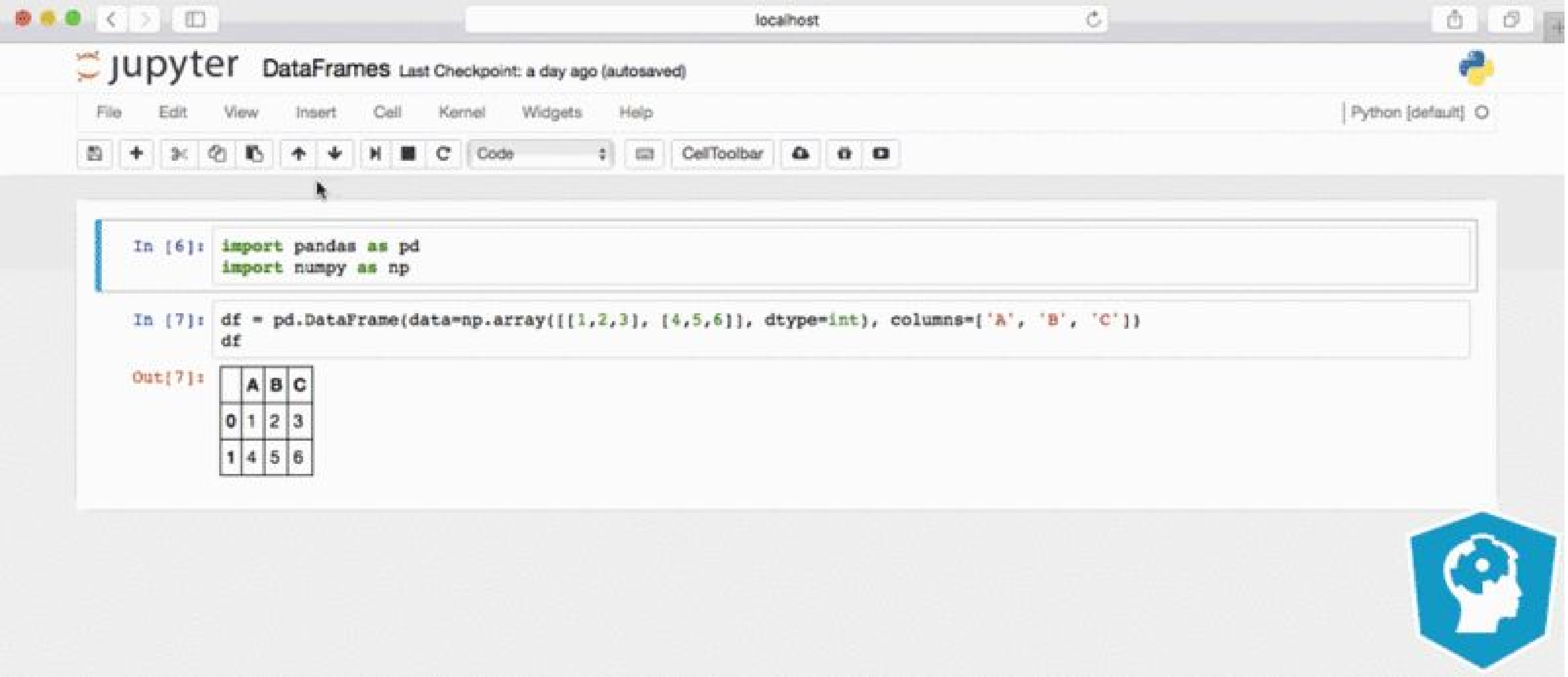


$$\text{recall} = \frac{TP}{TP + FN} \quad \text{precision} = \frac{TP}{TP + FP}$$



	Predicted Negative	Predicted Positive
Negative Cases	TN: 9,760	FP: 140
Positive Cases	FN: 40	TP: 60

Confusion Matrix



```
In [6]: import pandas as pd
        import numpy as np
```

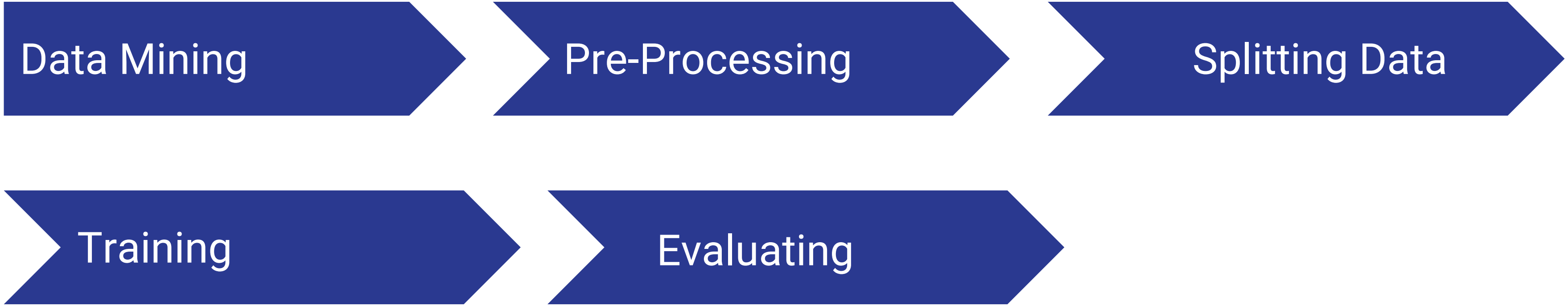
```
In [7]: df = pd.DataFrame(data=np.array([[1,2,3], [4,5,6]], dtype=int), columns=['A', 'B', 'C'])
        df
```

Out[7]:

	A	B	C
0	1	2	3
1	4	5	6

Jupyter Notebook Use

Recap



Data Mining

Pre-Processing

Splitting Data

Training

Evaluating

Python Issues

- Try to stick with using Python 3.5 over 3.6 and 2.7 as it has the best compatibility.
- You'll find lot's of code made for 2.7 that has to be modified for Python > 3.
- If you have a package issue, first try installing it again, then just Google away trying different versions.

Setup

First download and install Anaconda (Python 3.6 Version), then run the following from your terminal

```
$ conda create -n ml python=3.5
$ source activate ml
$ pip install scikit-learn numpy pandas tensorflow keras jupyter matplotlib
$ pip uninstall theano
$ conda install -c anaconda python.app=1.2
$ jupyter notebook
```

Practice

https://kukuruku.co/post/introduction-to-machine-learning-with-python-andscikit-learn/

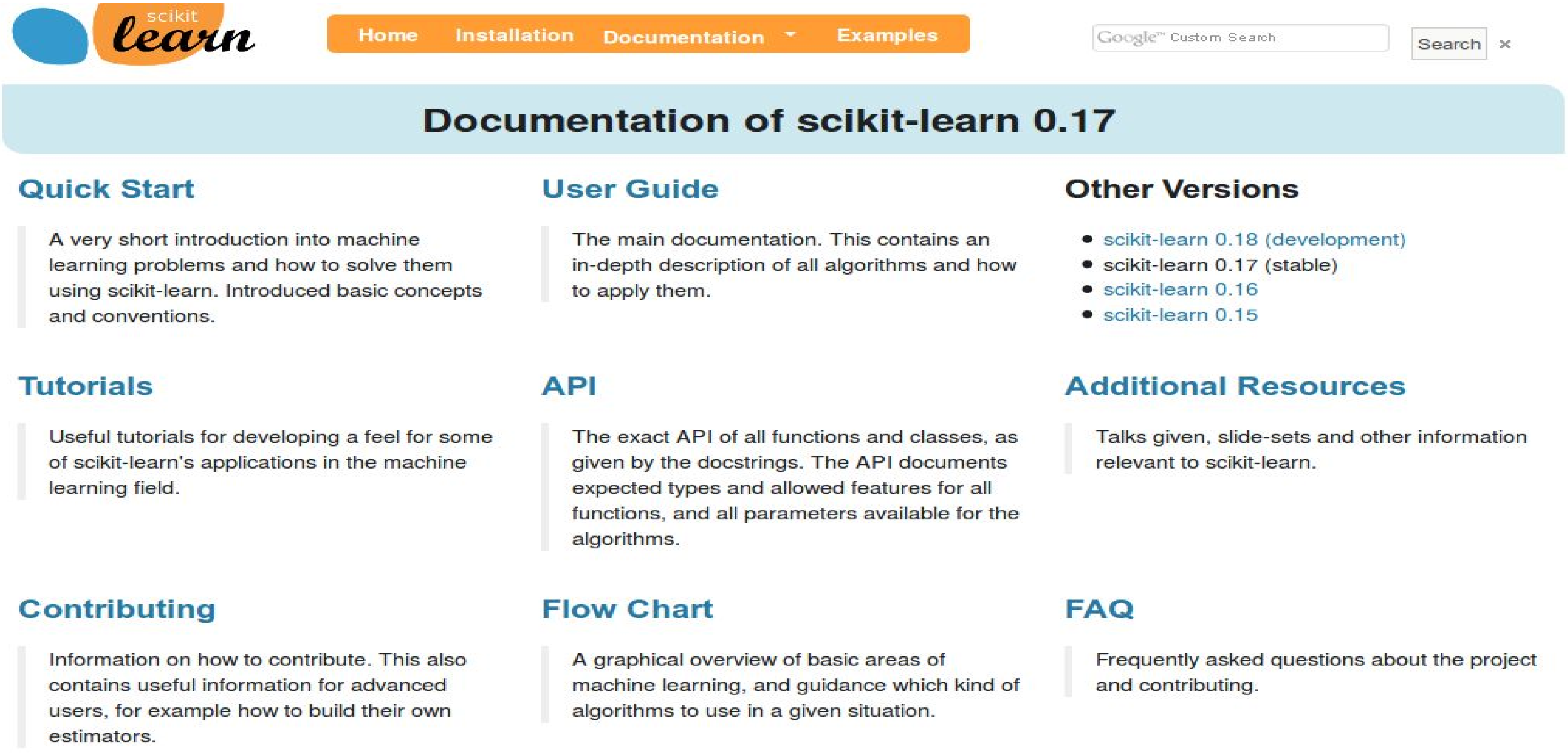
https://github.com/amueller/quick-ml-intro

http://machinelearningmastery.com/python-machine-learning-mini-course/

What to try?

If you’ve done the basics, try out some of these!

- Implement a Voting Classifier
- Implement a Bagging Classifier
- Create a Stacked model of multiple algorithms
- Implement a GridSearch, Randomized Search, or HyperOpt to find the optimal parameters
- Look into Cross Validation to get a more true accuracy of your model generalized.



Documentation of scikit-learn 0.17

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- scikit-learn 0.18 (development)
- scikit-learn 0.17 (stable)
- scikit-learn 0.16
- scikit-learn 0.15

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Contributing

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

http://scikit-learn.org/