CS5100 Foundations of Artificial Intelligence

Module 05 Lesson 9

Introduction to scikit-learn

Overview

Scikit-Learn Features

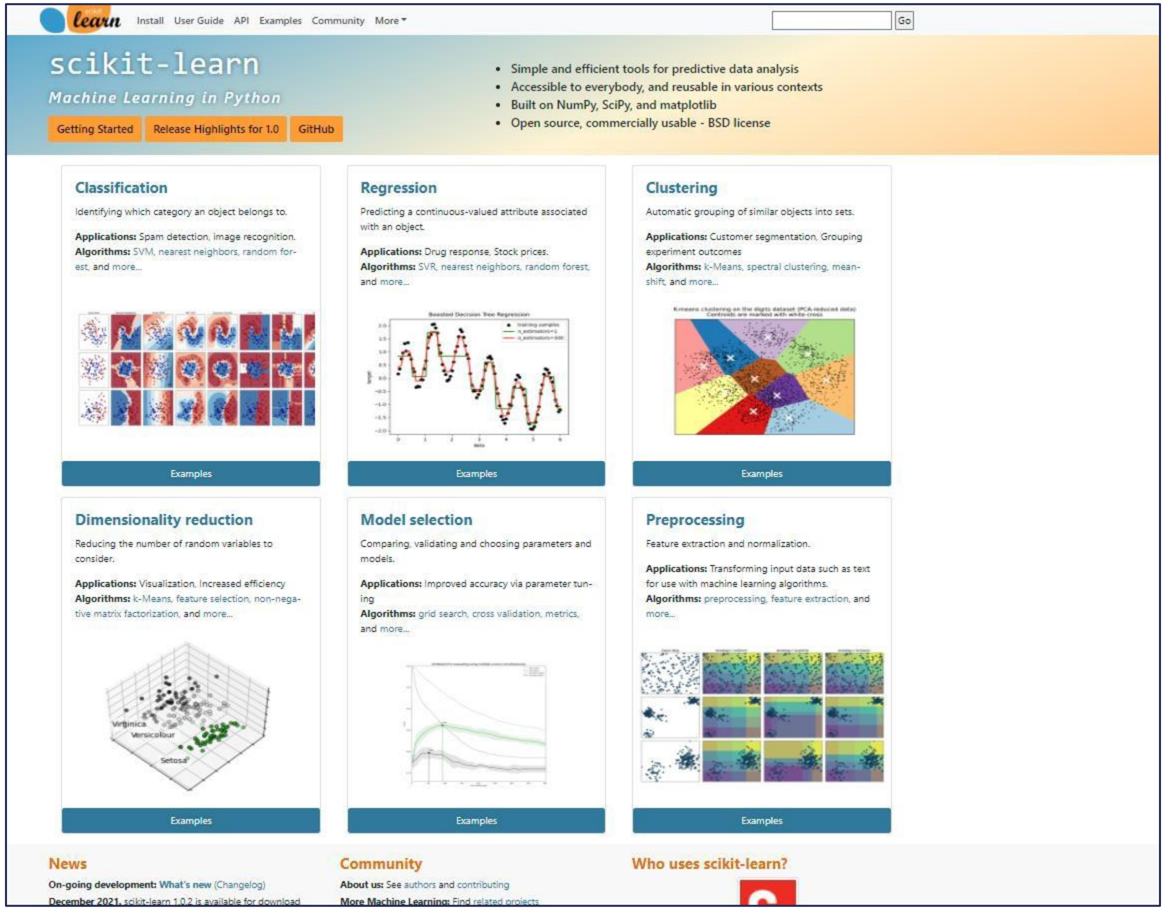
Databases

Sample Application

Metrics

Scikit-Learn ..1

- Scikit-Learn: a Python machine learning library
- Developed by David Coumapeau in 2007, extended by Matthieu Brucher and others.
- Open source, commercially usable
- Tools for data mining and data analysis
- Support from Microsoft, INRIA, nVidia, BNP Paribas, Fujitsu, Columbia Univ, Alfred P Sloan Foundation , Intel, Univ of Sydney, etc.



Scikit-Learn ..2

Provides a large range of learning algorithms, with a Python interface

Built on a stack that includes:

- NumPy: N-dimensional array package
- SciPy: Scientific Computing library
- Matplotlib: 2D/3D plotting
- Pandas: Data structures
- Sympy: Symbolic math
- IPython: Interactive Python console

Uses some C libraries internally (e.g. LibSVM)

Range of Tools

- Classification (SVM, Random forest,…)
- Regression (SVR, Lasso,…)
- Clustering (k-Means, spectral clustering,…)
- Dimensionality Reduction (PCA, feature selection,…)
- Model Selection (grid search, cross validation, metrics…)
- Preprocessing (feature extraction, …)
- Ensemble methods, Supervised Methods, …

Sklearn Datasets

“Built-in” Datasets ..1

- 7.2. Toy datasets
  - 7.2.1. Boston house prices dataset
  - 7.2.2. Iris plants dataset
  - 7.2.3. Diabetes dataset
  - 7.2.4. Optical recognition of handwritten digits dataset
  - 7.2.5. Linnerrud dataset
  - 7.2.6. Wine recognition dataset
  - 7.2.7. Breast cancer wisconsin (diagnostic) dataset

- 7.3. Real world datasets
  - 7.3.1. The Olivetti faces dataset
  - 7.3.2. The 20 newsgroups text dataset
  - 7.3.3. The Labeled Faces in the Wild face recognition dataset
  - 7.3.4. Forest covertypes
  - 7.3.5. RCV1 dataset
  - 7.3.6. Kddcup 99 dataset
  - 7.3.7. California Housing dataset

Datasets ..2

7.2.2. Iris plants dataset

Data Set Characteristics:

Number of Instances:	150 (50 in each of three classes)
Number of Attributes:	4 numeric, predictive attributes and the class

Attribute Information:	• sepal length in cm
	• sepal width in cm
	• petal length in cm
	• petal width in cm
	• **class:**
	    ◦ Iris-Setosa
	    ◦ Iris-Versicolour
	    ◦ Iris-Virginica

Datasets ..3

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-

Iris Versicolor

Iris Versicolor (Linn.) is a perennial herb, found abundantly in swamps and low grounds throughout eastern and central North America, common in Canada, as well as in the United States, liking a loamy or peaty soil. It is not a native of Europe.

Iris Setosa

Iris Setosa, bristle-pointed iris is a species in the genus Iris, it is also in the subgenus of Limniris and in the Iris series Tripetalae. It is a rhizomatous perennial from a wide range across the Arctic sea, including Alaska, Maine, Canada, Russia, northeastern Asia, China, Korea and southwards to Japan.

Iris Virginica

Iris virginica, with the common name Virginia iris, is a perennial species of flowering plant, native to eastern North America. It is common along the coastal plain from Florida to Georgia in the Southeastern United States.

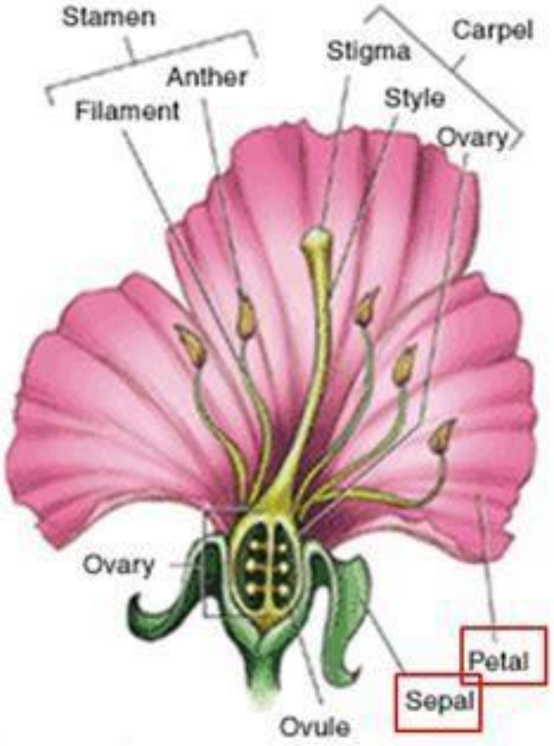
Commonly called Southern blue flag, this is a wetland species of iris which is native primarily to coastal plains from Virginia to Louisiana.

Petals, Sepals…

Petals and Sepals

- **Sepals** – outermost circle of flower parts that encloses a bud before it opens

- **Petals** – brightly colored structure just inside the sepals that attracts insects for pollination



Sample Data

```
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor

6.5,3.0,5.8,2.2,Iris-virginica

7.6,3.0,6.6,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica
7.3,2.9,6.3,1.8,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica
7.2,3.6,6.1,2.5,Iris-virginica
6.5,3.2,5.1,2.0,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica
6.8,3.0,5.5,2.1,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
```

Attribute Information

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
   - -- Iris Setosa
   - -- Iris Versicolor
   - -- Iris Virginica

Sample code

A simple Decision Tree to Classify Iris Plants

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

iris = datasets.load_iris()

# Test on training set -- eeks!
classifier = DecisionTreeClassifier(criterion='entropy')
classifier = classifier.fit(iris.data, iris.target)
print(classifier)

trueVals = iris.target
predictedVals = classifier.predict(iris.data)

print(metrics.classification_report(trueVals, predictedVals))
print(metrics.confusion_matrix(trueVals, predictedVals))
```

Classifier Parameters

```
>>> print(classifier)
DecisionTreeClassifier(class_weight=None, criterion='entropy’,
            max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

Result Metrics

```
>>> print(metrics.classification_report(trueVals, predictedVals))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	50
2	1.00	1.00	1.00	50
avg / total	1.00	1.00	1.00	150

```
>>> print(metrics.confusion_matrix(trueVals, predictedVals))
```

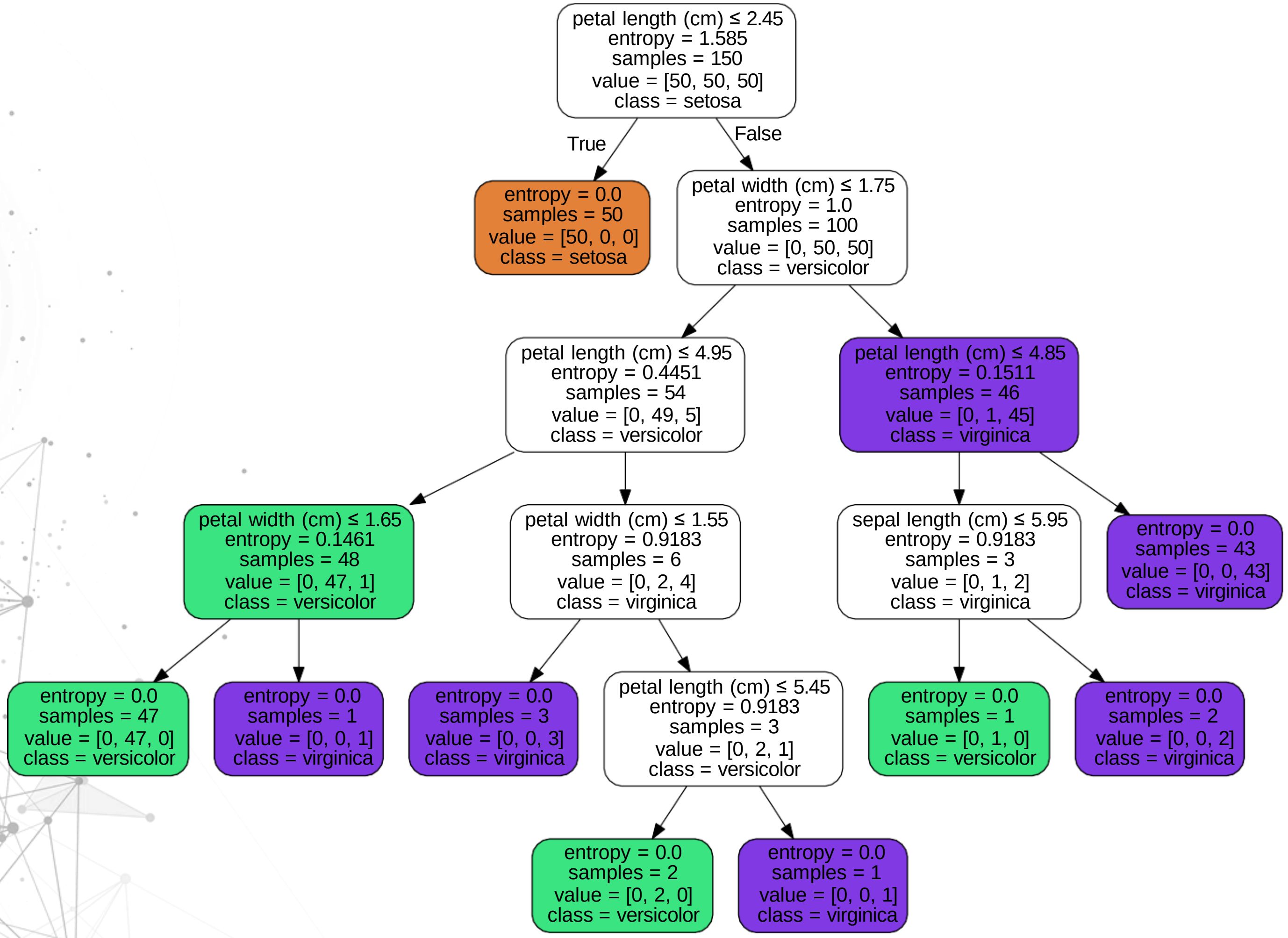
```
[[50  0  0]
 [ 0 50  0]
 [ 0  0 50]]
```

Code to Output Decision Tree

```
# First install graphviz, with e.g. conda install python-graphviz

from sklearn import tree
import graphviz
dot_data = tree.export_graphviz(classifier, out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("irisDT")   # file irisDT.pdf output
```

Iris Decision Tree



Same, with a Train-Test split

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()

# Now with a test-train split
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
test_size=0.33, random_state=5)
classifier = DecisionTreeClassifier(criterion='entropy')
classifier = classifier.fit(X_train, y_train)
print(classifier)

trueVals = y_test
predictedVals = classifier.predict(X_test)

print(metrics.classification_report(trueVals, predictedVals))
print(metrics.confusion_matrix(trueVals, predictedVals))
```

Quick Check

What differences do you expect with the train-test split?

Please move to Next Slide for answers

Quick Check

What differences do you expect with the train-test split?

1. Because we're using only two-thirds of the data to train, we expect lower accuracy.
2. The support numbers will be lower because we're using only a third of the data to test the model we create.
3. Finally, because we're testing on unseen data, we expect the confusion matrix to include some misclassified items.

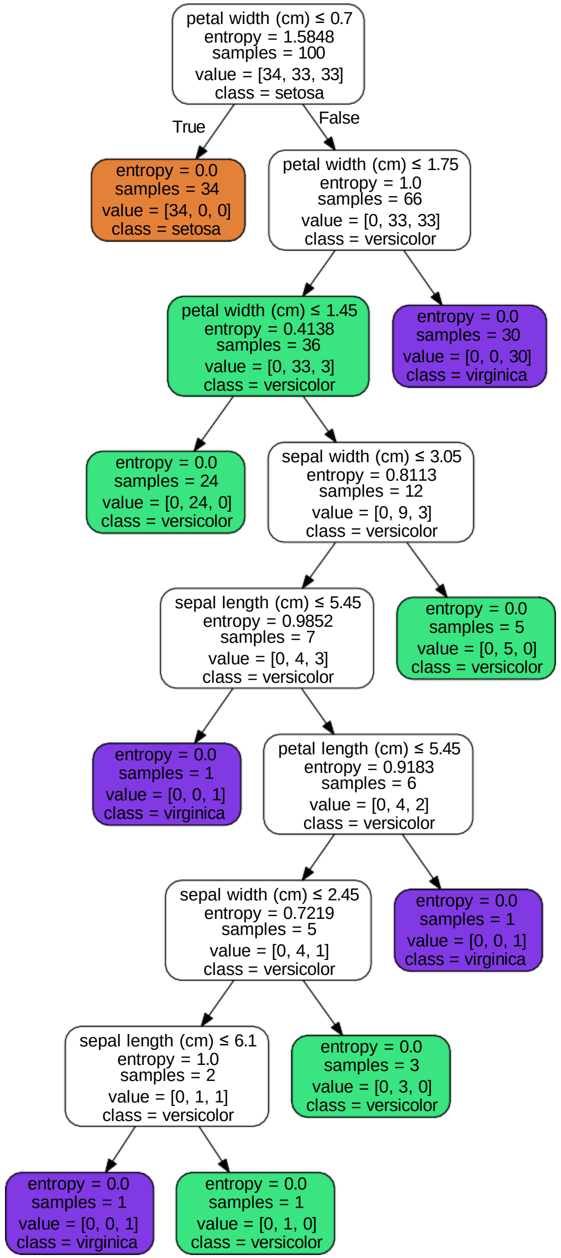
New Result Metrics

```
>>> trueVals = y_test
>>> predictedVals = classifier.predict(X_test)
>>> print(metrics.classification_report(trueVals, predictedVals))
             precision    recall  f1-score   support
          0       1.00      1.00      1.00        16
          1       0.88      0.88      0.88        17
          2       0.88      0.88      0.88        17
avg / total       0.92      0.92      0.92        50

>>> print(metrics.confusion_matrix(trueVals, predictedVals))
[[16  0  0]
 [ 0 15  2]
 [ 0  2 15]]
```

New Iris Decision Tree

Note different shape,
different branching checks



Naïve Bayes & scikit-learn

Three variants:

Gaussian NB — features assumed to be in normal distribution

Multinomial NB — used for text problems, where data is represented as word count or tf-idf vectors; incorporates smoothing*

Bernoulli NB — may be multiple features, but all features assumed to be binary-valued

Metrics

Metrics

- Confusion matrix very useful
- Accuracy not enough. Why?
- Prefer Precision and Recall, maybe F1 score
- Support also very useful

Precision and Recall ..1

Say we want to classify web pages as
homepages or not

- In a test set of 1000 pages, say there are 3 homepages
- Our classifier blindly says they are all non-homepages: 99.7% accuracy!
- Need new measures for rare positive events

Precision: fraction of total *guessed* positives which were *really* positive
How accurate are you in your guesses?

Recall: fraction of total *actual positi*ves which were *guessed as positive*
How much of what you should identify do you actually identify?

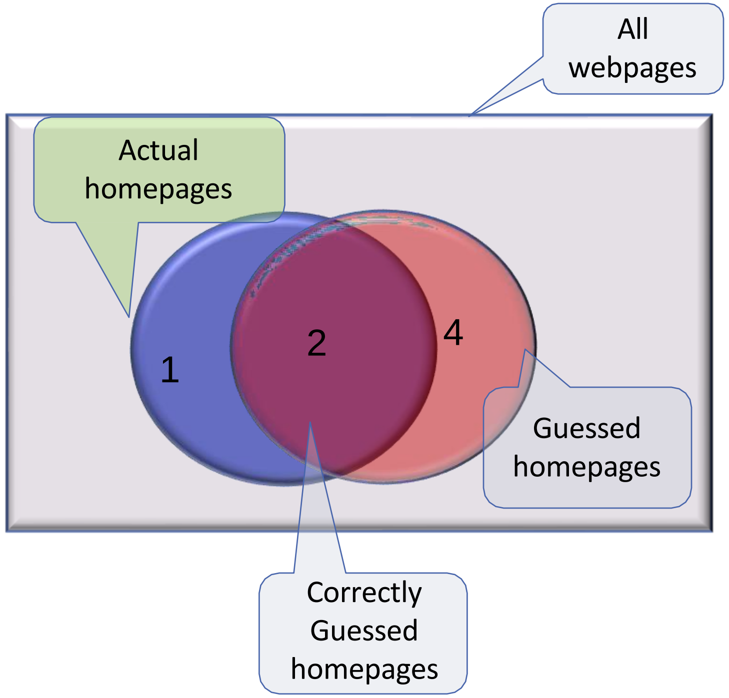
Precision and Recall ..2

Precision: correctly guessed positives / total guessed positives
Recall: correctly guessed positives / total true positives

Say we guess 6 are homepages, of which
2 were actually homepages

We know there are 3 homepages in the test set

- Precision: 2 correct / 6 guessed = 0.33
- Recall: 2 correct / 3 true = 0.67



Precision and Recall ..3

- Which is more important Precision or Recall, in –
  - Face recognition for security?
  - Targeted voter messaging?
- Where would Precision be more important?
- Recall?

Precision, Recall and F-measure

- Precision/recall tradeoff
  - Often, you can trade off precision and recall

F-measure: harmonic mean of p and r

$$F_1 = \frac{2}{1/p + 1/r}$$

= 2 *p * r / (p + r)

Scikit-learn
Things to Explore

- Pipeline
- TfidfVectorizer (for word and character n-grams)
- Gridsearch
- …

Further Information

Home page: http://scikit-learn.org/stable/

Installation: http://scikit-learn.org/stable/install.html

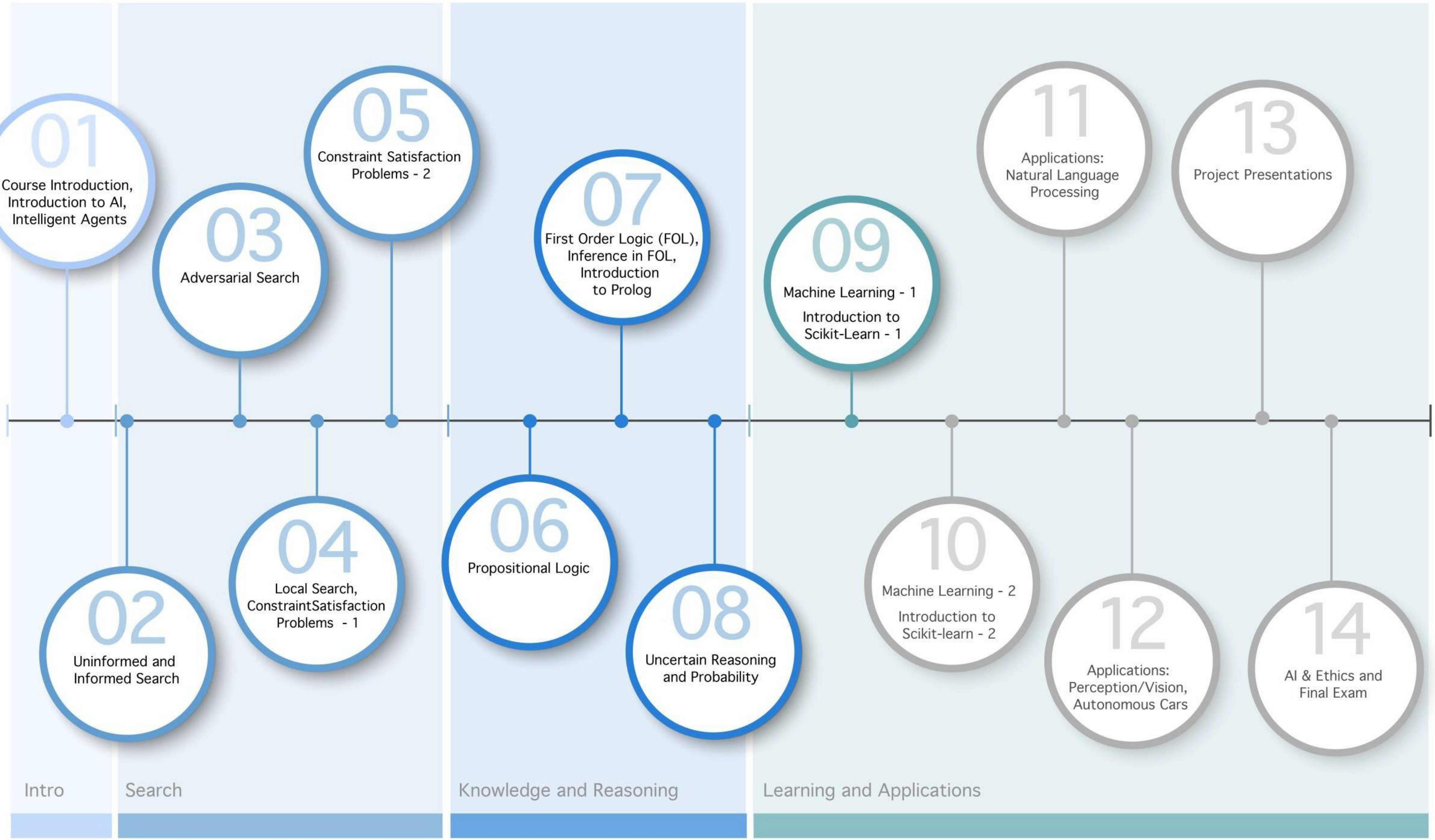
Quick Start: http://scikit-learn.org/stable/tutorial/basic/tutorial.html

Tutorials: http://scikit-learn.org/stable/tutorial/index.html

Examples: http://scikit-learn.org/stable/auto_examples/index.html

Github: https://github.com/scikit-learn

User Guide (PDF): http://scikit-learn.org/stable/downloads/scikit-learn-docs.pdf


01
Course Introduction, Introduction to AI, Intelligent Agents
02
Uninformed and Informed Search
03
Adversarial Search
04
Local Search, ConstraintSatisfaction Problems - 1
05
Constraint Satisfaction Problems - 2
06
Propositional Logic
07
First Order Logic (FOL), Inference in FOL, Introduction to Prolog
08
Uncertain Reasoning and Probability
09
Machine Learning - 1
Introduction to Scikit-Learn - 1
10
Machine Learning - 2
Introduction to Scikit-learn - 2
11
Applications: Natural Language Processing
12
Applications: Perception/Vision, Autonomous Cars
13
Project Presentations
14
AI & Ethics and Final Exam
Intro
Search
Knowledge and Reasoning
Learning and Applications