Machine Learning for Healthcare

ter.ps/389iweek2

“Big breakthroughs happen when what is suddenly possible meets what is desperately necessary.”

- Thomas Friedman

Challenges

- Healthcare data is unstructured (inconsistent data fields)
- Access is tightly controlled
- Understanding data requires subject matter experts
  - practicing clinicians, researchers, …
  - computer scientists and biologists must collaborate

Opportunities

- Save lives through early detection & treatment
- Empower clinicians to provide more effective care
- Personalize treatment options

…

- Possibilities are endless!

Ethics - key considerations:

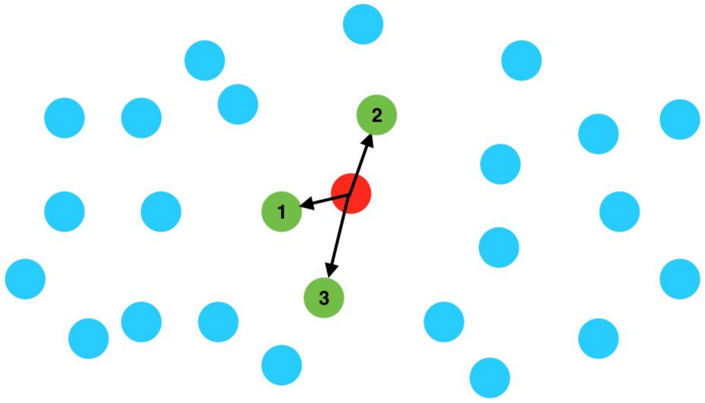
- **Bias** in the data
  - e.g., socio-economic status of subjects
- **Age** of data
  - models may become outdated with new tech & discoveries
- **Quality** of data
  - how was training data labeled?
- **External pressure**
  - motivation of benefactors funding research?

Fundamental ML Algorithms

K-nearest neighbor

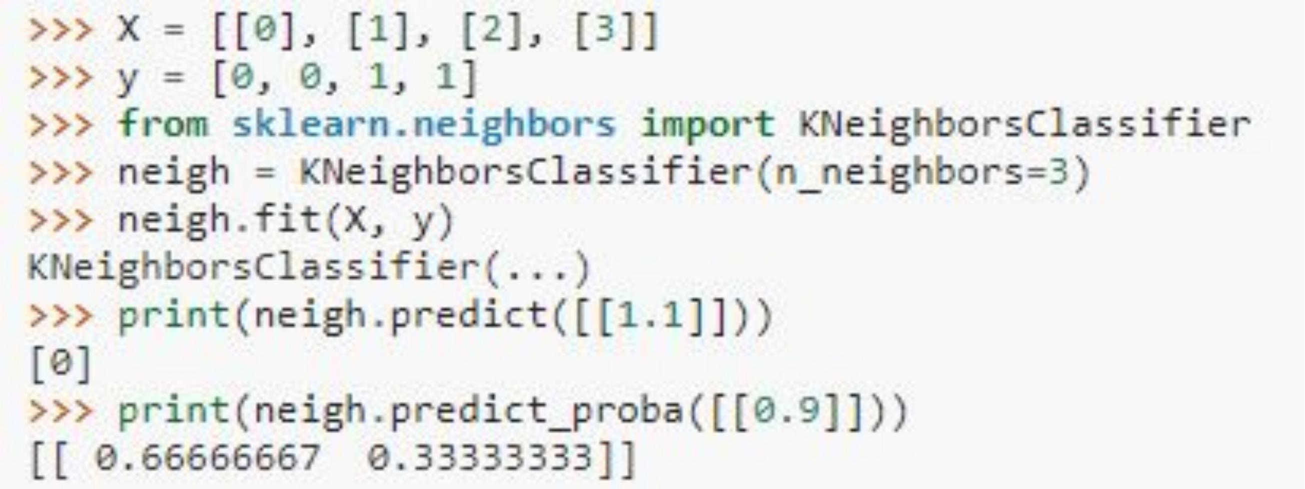
- Uses training set to find K most similar instances ("neighbors") of object to be classified

- Returns most common classification of those K objects - simple!



sklearn.neighbors.KNeighborsClassifier

Docs

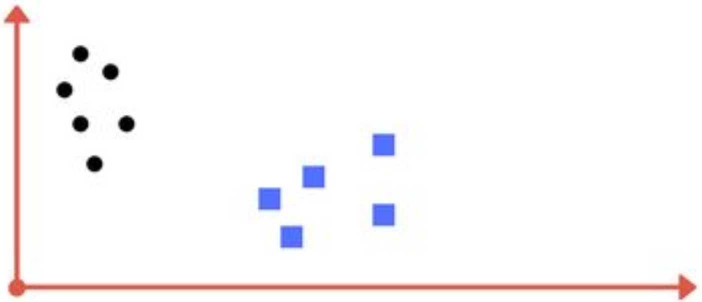


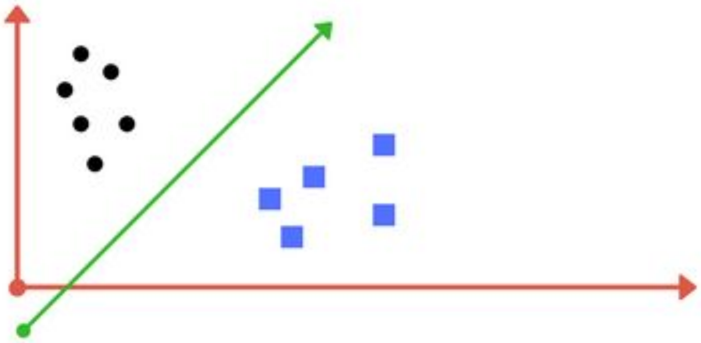
```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[ 0.66666667  0.33333333]]
```

Support vector machines (SVM)

- Given labeled training data, returns optimal hyperplane categorizing it

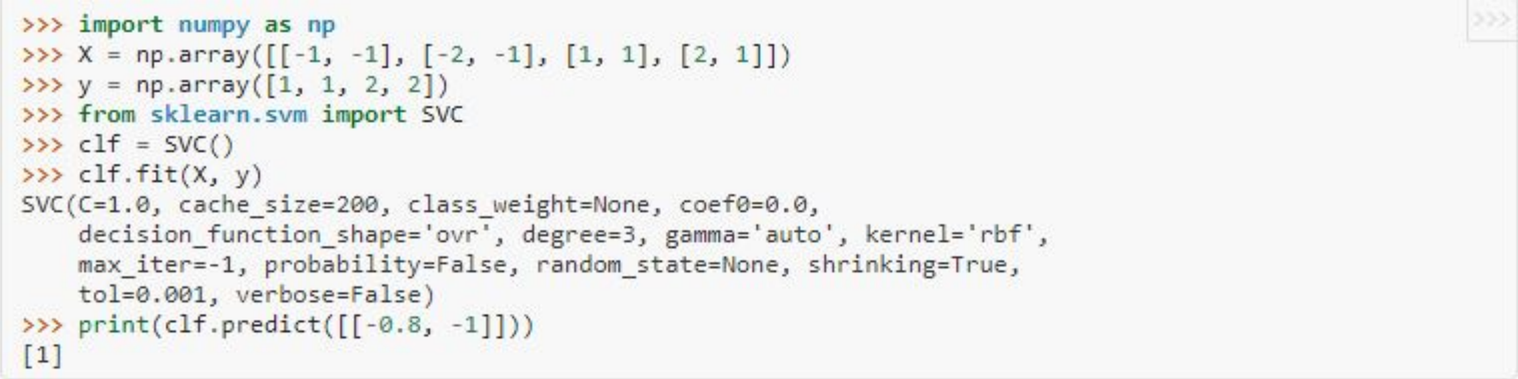
- In 2 dimensions, the “hyperplane” is just a line dividing points on a graph





sklearn.svm.SVC (Support Vector Classifier)

Docs



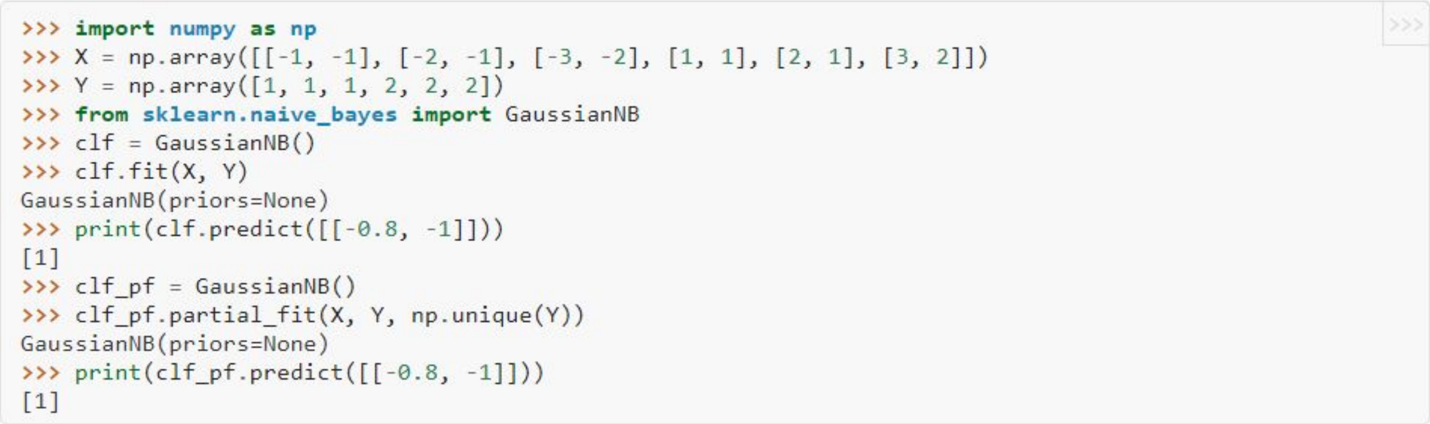
```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
>>> y = np.array([1, 1, 2, 2])
>>> from sklearn.svm import SVC
>>> clf = SVC()
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

Naive Bayes classifiers

- Classifiers based on Bayes' Theorem
- Assumes each feature is independent and equally weighted
- Finds probability of classification given feature values

sklearn.naive_bayes.GaussianNB

Docs

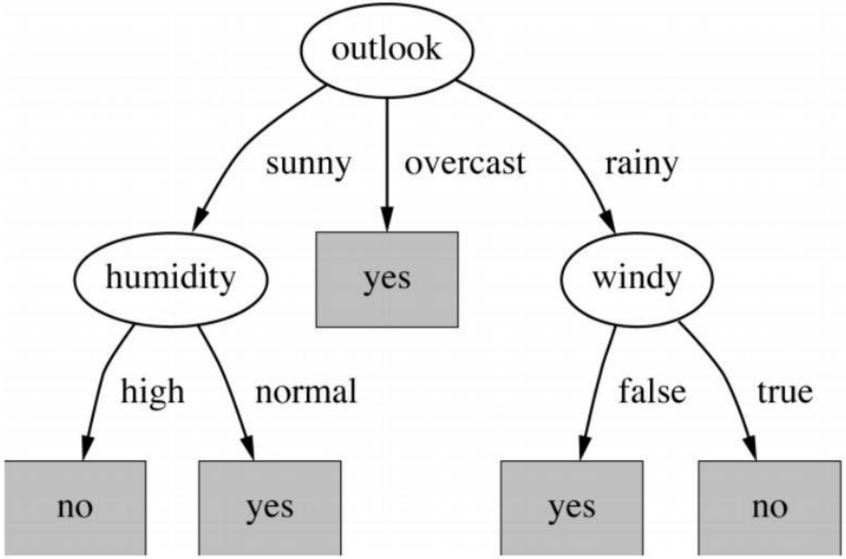


```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> Y = np.array([1, 1, 1, 2, 2, 2])
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
GaussianNB(priors=None)
>>> print(clf.predict([[-0.8, -1]]))
[1]
>>> clf_pf = GaussianNB()
>>> clf_pf.partial_fit(X, Y, np.unique(Y))
GaussianNB(priors=None)
>>> print(clf_pf.predict([[-0.8, -1]]))
[1]
```

Decision trees

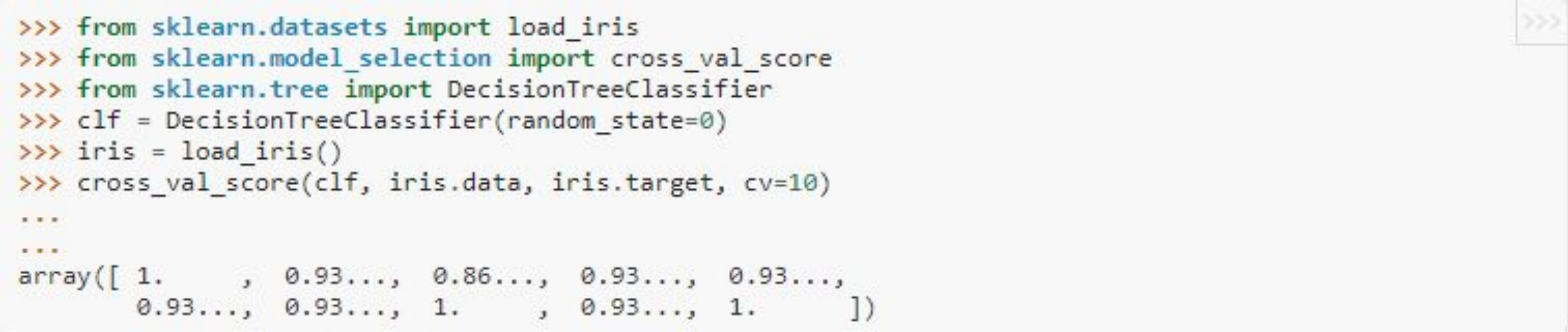
Tree where:

- ○ Each node : feature
- ○ Each branch : decision/rule
- ○ Each leaf : outcome



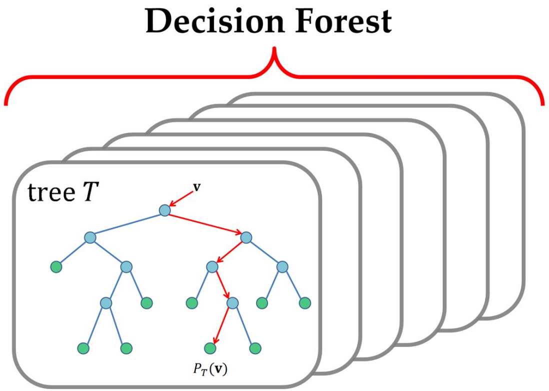
sklearn.tree.DecisionTreeClassifier

Docs



```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.tree import DecisionTreeClassifier
>>> clf = DecisionTreeClassifier(random_state=0)
>>> iris = load_iris()
>>> cross_val_score(clf, iris.data, iris.target, cv=10)
...
...
array([ 1.     ,  0.93...,  0.86...,  0.93...,  0.93...,
        0.93...,  0.93...,  1.     ,  0.93...,  1.      ])
```

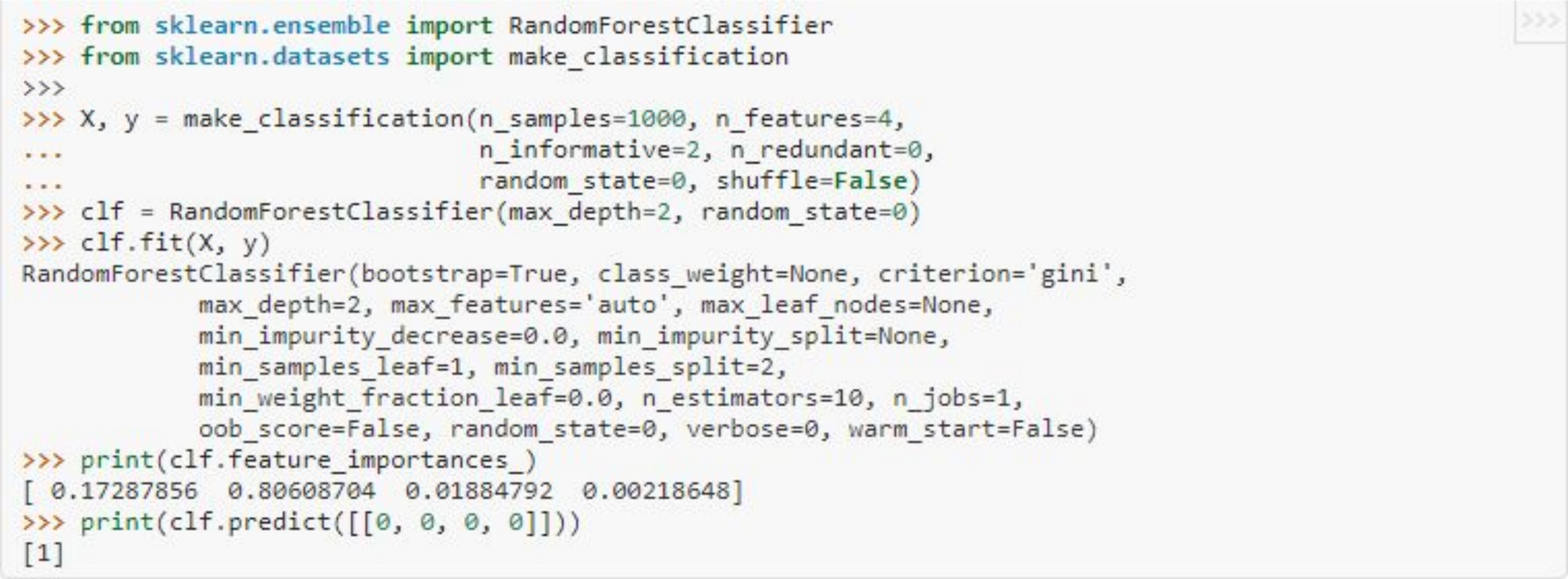
Random forest classifiers



- Ensemble algorithm
  - combines multiple algorithms to classify data
- Creates set of decision trees from randomly chosen subset of data
- Chooses final classification from winning result of all votes

sklearn.ensemble.RandomForestClassifier

Docs



```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>>
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
>>> print(clf.feature_importances_)
[ 0.17287856  0.80608704  0.01884792  0.00218648]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

Summary

- Data bias and quality are super important
- Ethics is tricky but crucial, esp. in healthcare
- Lots of ML algorithms to be applied to exciting challenges in healthcare

Visual Comparison of different algorithms

- http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

Readings + Additional Resources

- https://www.techemergence.com/machine-learning-in-pharma-medicine/