**Naive Bayes:** simple boundary, apply Bayes’ theorem with the “naive” assumption of independence between every pair of features

1. efficient

<http://scikit-learn.org/stable/modules/naive_bayes.html>

**sensitivity**: Pr (positive| have it)

**specitivity**: Pr (negative| do not have it)

**Support Vector Machine (SVM)**: maximize the margin (distance to the nearest point) -> robustness

**Kernel trick:** not separable -> separable

SVM to prevent over-fitting:

1. Kernel
2. C: control trade-off between smoothing decision boundary and classifying training points correctly
3. gamma

Linear kernel only gives a linear decision boundary.

**Entropy**: how a decision tree decides where to split the data

1. measure of impurity in a bunch of examples
2. find the threshold where the split of data is as pure as possible

**Information Gain:** entropy (parent) - weighted average of entropy (children)

1. **Decision tree maximizes information gain**
2. at the start of the data, entropy (parent) is 1

**Algorithms:**

1. K-nearest neighbor
2. random forest (ensemble method) -> fit classifier to sub-tree and average result
3. boosted decision tree (adaboost) (ensemble method) -> fit a base classifier and then adjust to fit the mis-classified data

**More data > Fine-Tuned Algorithm!!**

**Types of data:**

1. numerical
2. categorical number (limited number of discrete values)
3. time-series (temporal, data, time)
4. text data

**Outlier Rejection:**

1. train all the data
2. remove the data with highest residual error -> 10%
3. train the data gain
4. maybe repeat 1 to 3

Feature Scaling: , prone to outlier

Bag of Words: take the words and count the frequency

Stop words (low-information words occur frequently)

**Feature Selection**

1. select best features
2. add new features

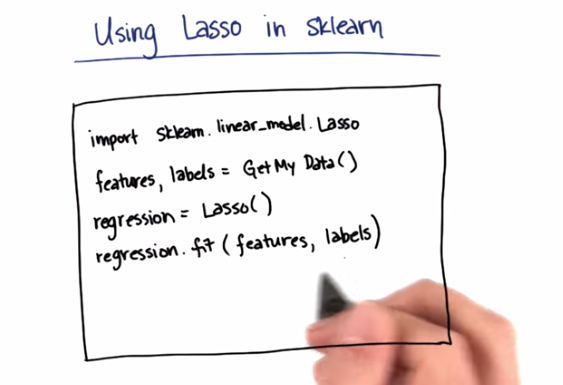
Univariate Feature Selection: treats each feature independently and asks how much power it gives you in classifying or regressing.

sklearn -> feature selection

from sklearn.feature\_selection import \*\*\*

Lasso Regression: min SSE + lambda \* |\beta|

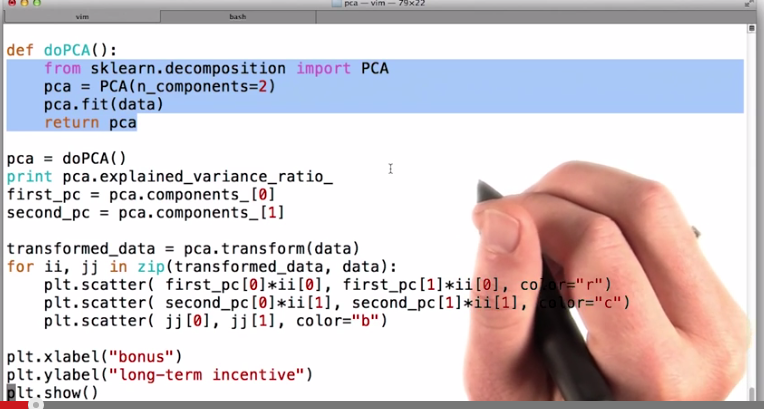
(|\beta| is # of features)



Difference between Feature Selection and PCA: PCA makes composite features, while feature selection eliminates some features.

**variance**: the spread of a data distribution

Principal Component: the direction with the largest variance



K-fold cross validation:

1. run K times
2. choose one as test set, the others as train sets

Accuracy is not good for skewed class (only very few data points are of interest)

Confusion Matrix:

(2 by 2 matrix, actual positive/negative v.s. predict positive/negative) -> for skewed class

1. **false alarm**: true negative, predict positive

(n by n matrix, where n is # of features)

1. **recall**: if true, the probability of identifying it to be true
2. **precision**: if predicted to be true, the probability of it to be true

**positive for actual, true or false for prediction**

