Naïve Bayes:

Supervised Classification: Given a bunch of labeled examples, the network learns what features to pay attention to, and how to classify new examples.

Features and Labels: Input extracted features, and try to produce labels (simple: 2 labels = classification) When you only have 2 features, you can use a scatterplot to visually analyze data. Machine Learning Algorithms then define the “Decision Surface” that lies between the two different classes. On one side of the decision surface, they will predict one label, on the other side, they will predict the other label.

Linear Decision Surfaces: (Linear Regression)

Using Gaussian Naïve Bayes to create a Decision Surface:

sklearn.naive\_bayes.GaussianNB

import, create a classifier, train (fit) the classifier on your inputs and labels, then get a prediction (predict) on a new point.

Calculating Accuracy: Number of points classified correctly over the total number of points in the test set. There are 2 ways to do this:

1. Use predictions you made in your test set and compare them to labels in your test set manually
2. Or use score functionality from sklearn.naive\_bayes.GaussianNB

Code:

Given features\_train, labels\_train, features\_test, labels\_test:

from sklearn.naive\_bayes import GaussianNB

clf = GaussianNB()

clf.fit(features\_train, labels\_train)

clf.predict(features\_test)

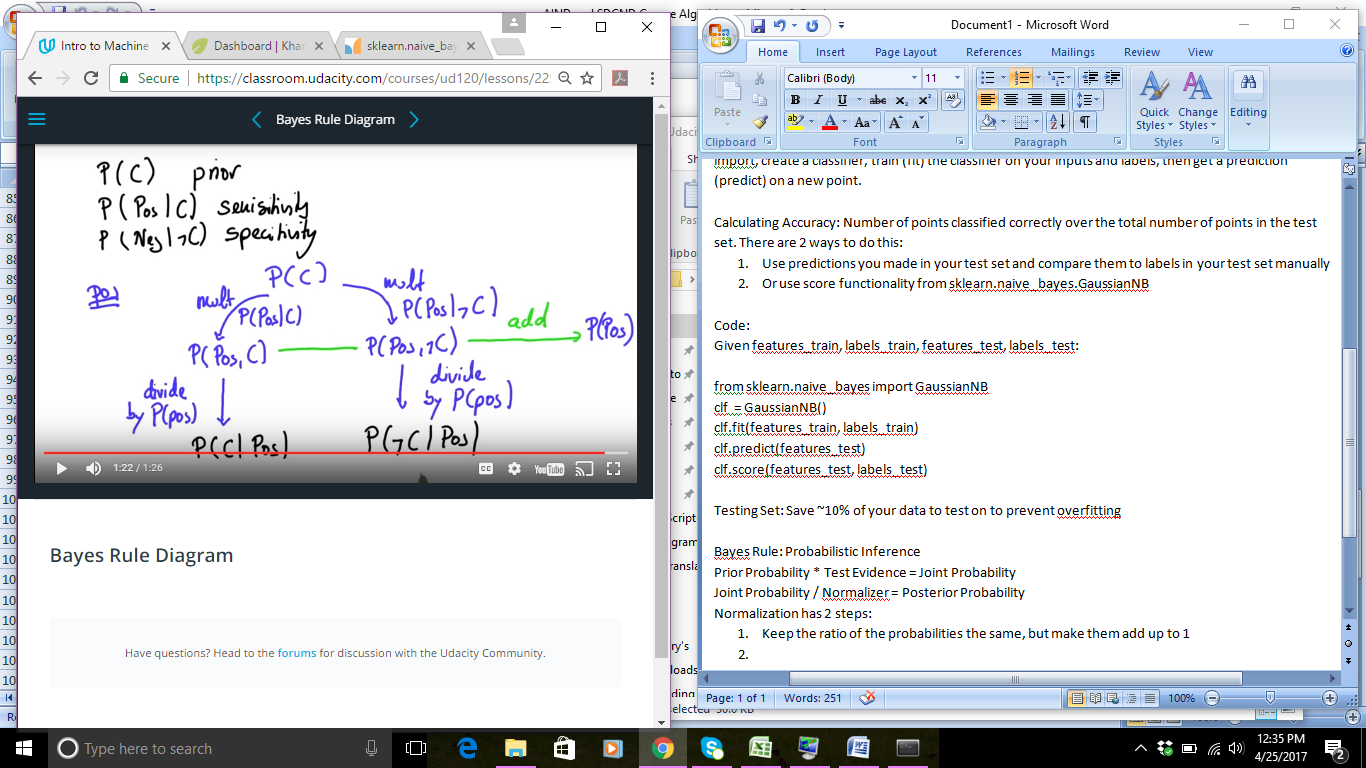
clf.score(features\_test, labels\_test)

Testing Set: Save ~10% of your data to test on to prevent overfitting

Bayes Rule: Probabilistic Inference

Prior Probability \* Test Evidence (can have multiple points) = Joint Probability

Joint Probability / Normalizer = Posterior Probability

Cancer Probability Example:

P(C) = Probablility of Cancer Prior to test = 1%

P (Pos|C) = Probability of a positive test, given you have Cancer (Sensitivity) = 90%

P (Neg| >C) = Probability of a negative test, given you don’t have Cancer (Specificity) = 90%

Text Learning: Learning from documents – Naïve Bayes

Lets you identify from a text source, how to classify it. It’s called Naïve because it ignores word order, it just looks at word frequencies (based on probabilities)

Things that you should think about when choosing a supervised classification algorithm:

Naïve Bayes:

Pros: Easy to implement, simple to run, efficient (even with large feature bases)

Cons: It can “break”, phrases that encompass multiple words don’t work well (when word order matters)