ECE 20875

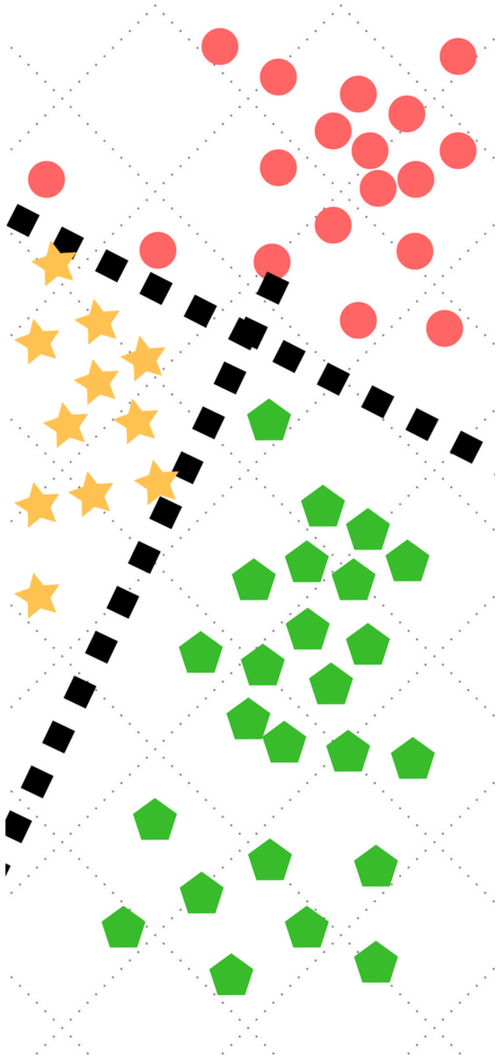
Python for Data Science

Milind Kulkarni and Chris Brinton

classification: naive bayes

what is classification?

- Given a data point, tell me what class it falls into
  - Is this animal a mammal, a bird, a fish, …
  - Is this picture a cat, a horse, a car, …
- Generally, we want to *learn* a classifier
  - Given a bunch of data points: x_1, x_2, x_3, etc. where each data point is *labeled* with its class
  - Classifier should be able to tell which class a new datapoint belongs to
  - Because we start with labeled data, this is an example of **supervised learning**



classification

- Consider datapoints in a *d*-dimensional space, i.e., each data point being defined by *d* features

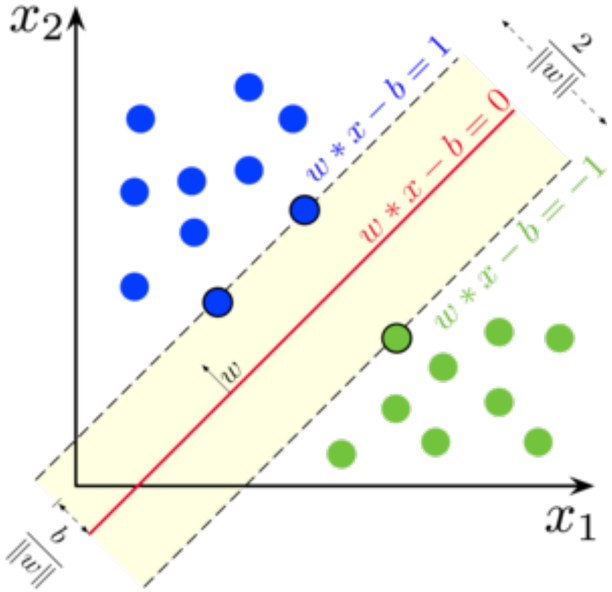
  $a_1, a_2, \ldots, a_d$

- Each datapoint is from one of k possible **classes**

  $C_0, C_1, \ldots, C_{k-1}$

- A classifier is a function $f(a_1, a_2, \ldots, a_d) = C_x$, where $C_x \in \{C_0, C_1, \ldots\}$

- We will focus on learning classifiers for two classes, i.e., $k = 2$



four different strategies

- Like regression, there are many different flavors of classification algorithms
- We will talk about four different, yet common and representative strategies:
  - **Naïve Bayes**: Simple model to use. But is *parametric* — requires assumptions about data
  - **k-nearest neighbor (kNN)**: Very easy model to understand. Expensive model to evaluate. But is *non-parametric* — requires few assumptions about data
  - **Support Vector Machine (SVM)**: Another non-parametric model which is harder to interpret than **kNN** but may have more explanatory power
  - **Neural networks**: Trendy approach! Essentially cascading nonlinear functions together to maximize potential predictive power

naïve bayes

- Basic idea: Each class of data can be described by a distribution (histogram) showing how likely different data points are
  - If I have a new data point *x*, which distribution is it more likely to come from?
- Example: Two classes of cars — **sports cars** and **minivans**
  - Each car can be described by its average speed, and the two classes have different distributions of speeds (e.g., sports cars have a higher average speed than minivans)
  - I see a new speed reading. Is it a **sports car** or a **minivan**?

naïve bayes

- Basic idea: each class of data can be described by a distribution (histogram) showing how likely different data points are
  - If I have a new data point *x*, which distribution is it **more likely** to come from?
- Use the **posterior distributions** (conditional probabilities):

$$P(C_0 \mid x) \textbf{ vs } P(C_1 \mid x)$$

Probability that I am a minivan given that I have average speed *x*

Probability that I am a sports car given that I have average speed *x*

- Problem: How do we actually infer these probabilities?

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(C_0 \mid x) \textbf{ vs } P(C_1 \mid x)$$

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$\frac{P(x \mid C_0)P(C_0)}{P(x)} \overset{?}{>} \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)P(C_0) \overset{?}{>} P(x \mid C_1)P(C_1)$$

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)\frac{P(C_0)}{P(C_1)} \overset{?}{>} P(x \mid C_1)$$

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)\frac{P(C_0)}{P(C_1)} \overset{?}{>} P(x \mid C_1)$$

Likelihood of *x* in Class 0

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)\frac{P(C_0)}{P(C_1)} \overset{?}{>} \boxed{P(x \mid C_1)}$$

Likelihood of *x* in Class 1

bayes' theorem to the rescue

- Bayes' theorem tells us how to compute the posterior probability (which we do not necessarily know) using probabilities we have good estimates for (i.e., prior knowledge)

- We have seen this "trick" before, most recently with Gaussian mixture models

$$P(C_0 \mid x) = \frac{P(x \mid C_0)P(C_0)}{P(x)} \qquad P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)\frac{P(C_0)}{P(C_1)} \overset{?}{>} P(x \mid C_1)$$

How much more common Class 0 Is than Class 1

how does this help?

- Once we've applied Bayes' theorem, we can estimate the necessary probabilities

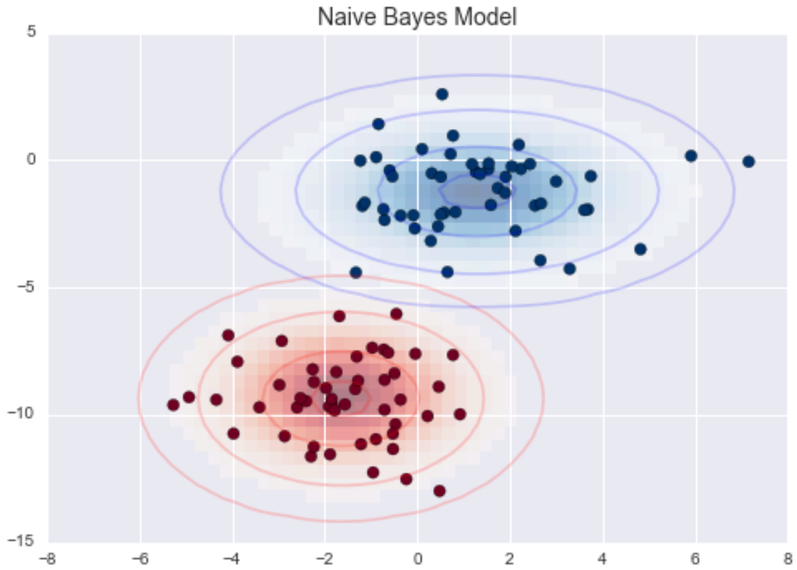
- $P(C_0)/P(C_1)$: Comes from *prior knowledge*

  - Given all cars, how much more common are minivans than sports cars?

  - If we have no prior knowledge, we may assume that both classes are equally likely, i.e., $P(C_0)/P(C_1) = 1$

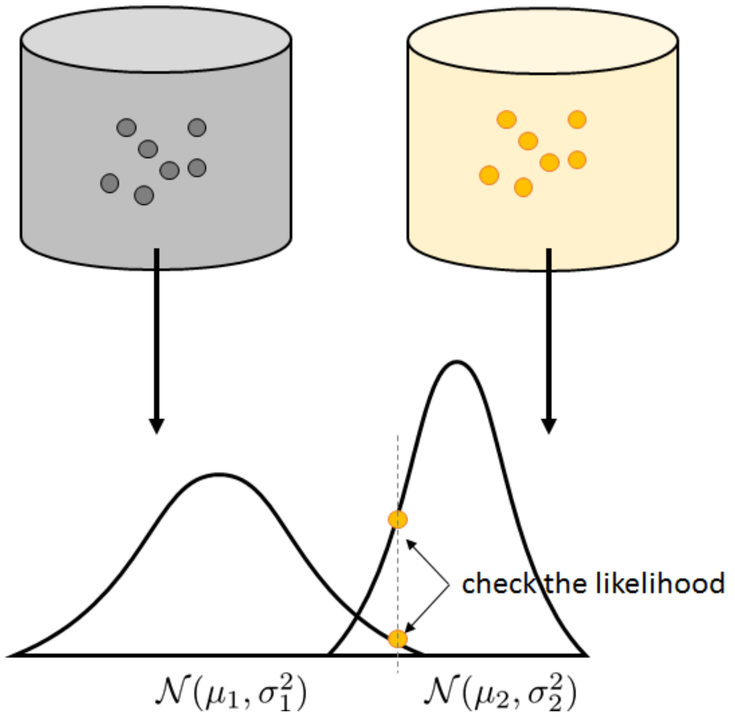
- $P(x \mid C_0)$ & $P(x \mid C_1)$: Comes from estimates of distributions

  - Estimate the distributions of the two classes given the labeled data!



estimating distribution

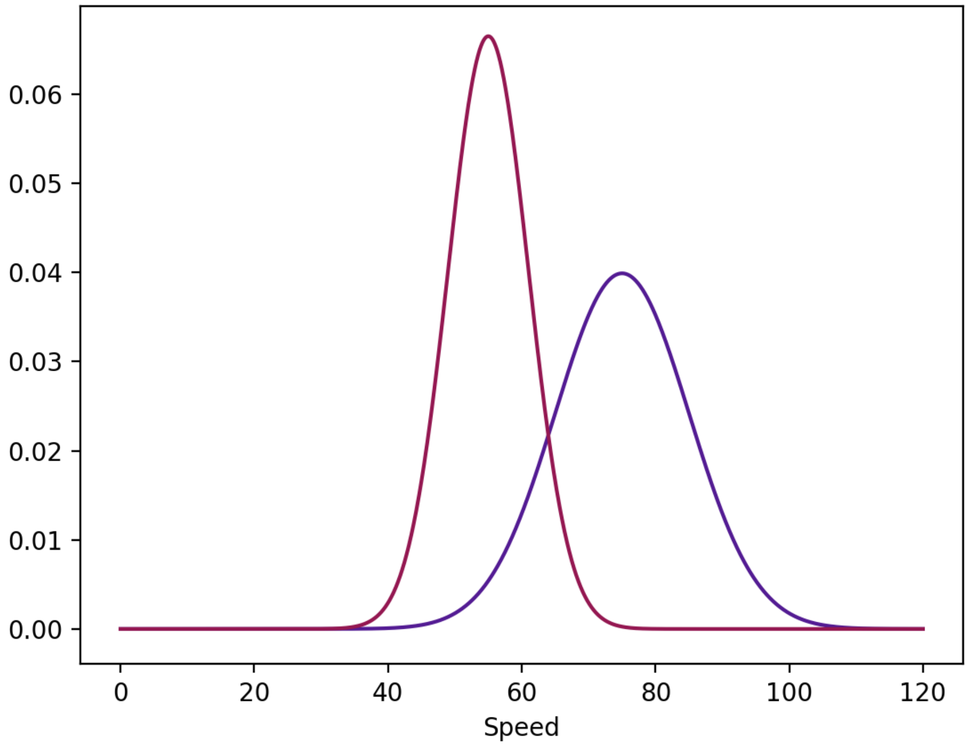
- Can always fall back on empirical distributions
  - Use datasets to build histograms, use histograms as estimators (recall the histogram lecture)
- Parametric is more common with Naive Bayes
  - Use some prior knowledge to choose a model for your data, estimate the parameters of that model
- Common choice: **Gaussian Naive Bayes**
  - Estimate mean and variance from data sets
  - Given a Gaussian, can directly “read off” likelihoods given *x*



naïve bayes with gaussian

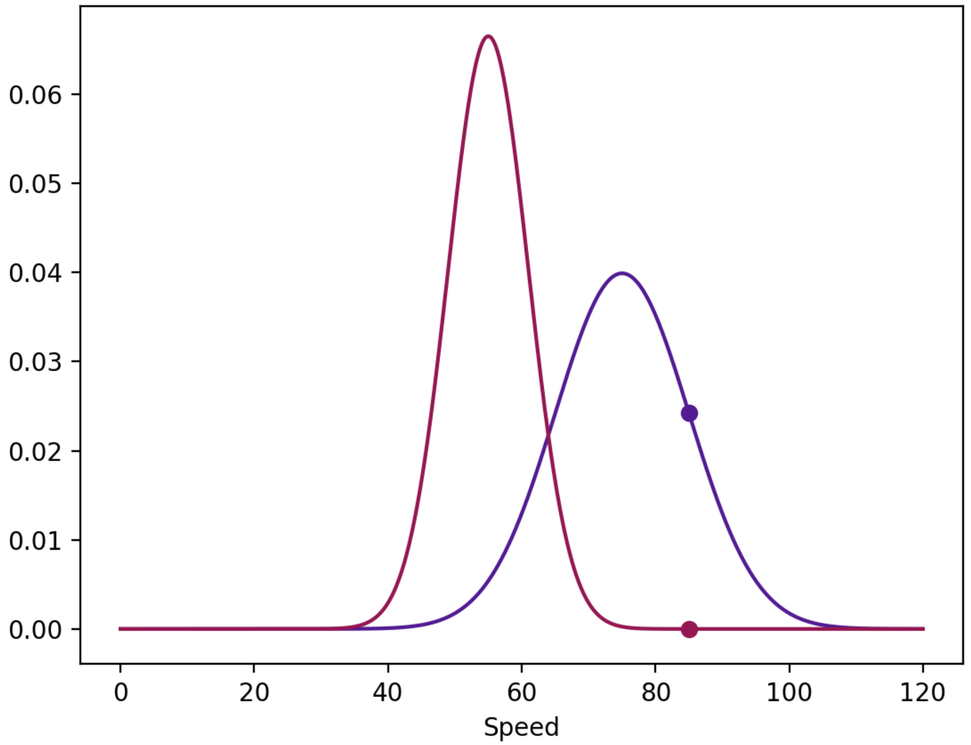
- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed



naïve bayes with gaussian

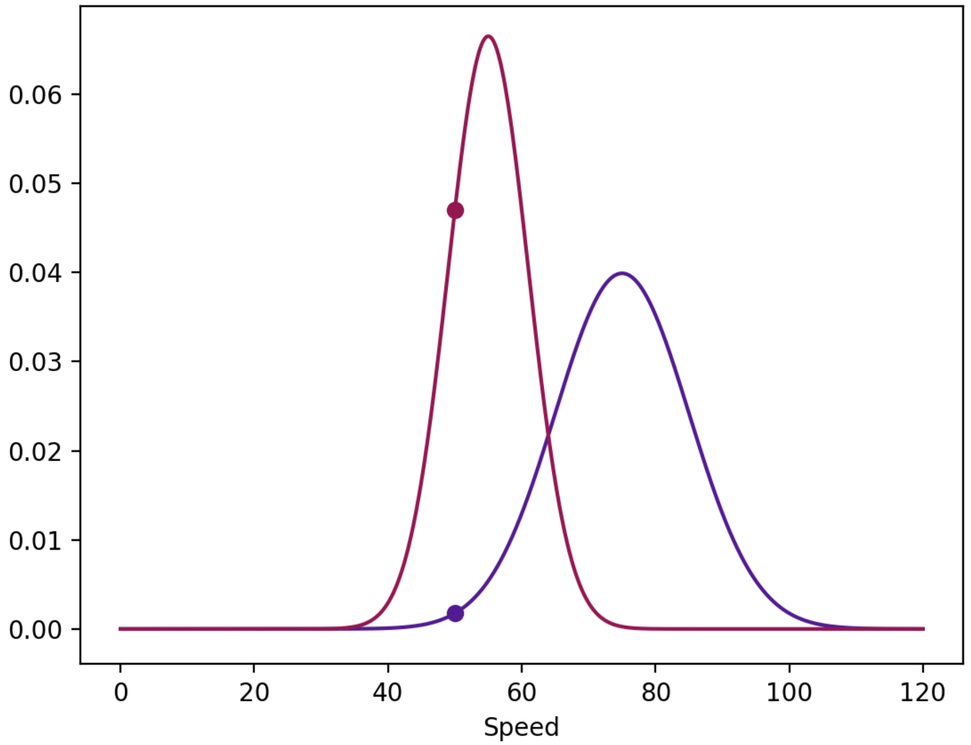
- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data
- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed
- If speed is high, more likely to be **sports car**



Note: here we are assuming that P(minivan) = P(sports car)

naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data
- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed
- If speed is low, more likely to be **minivan**



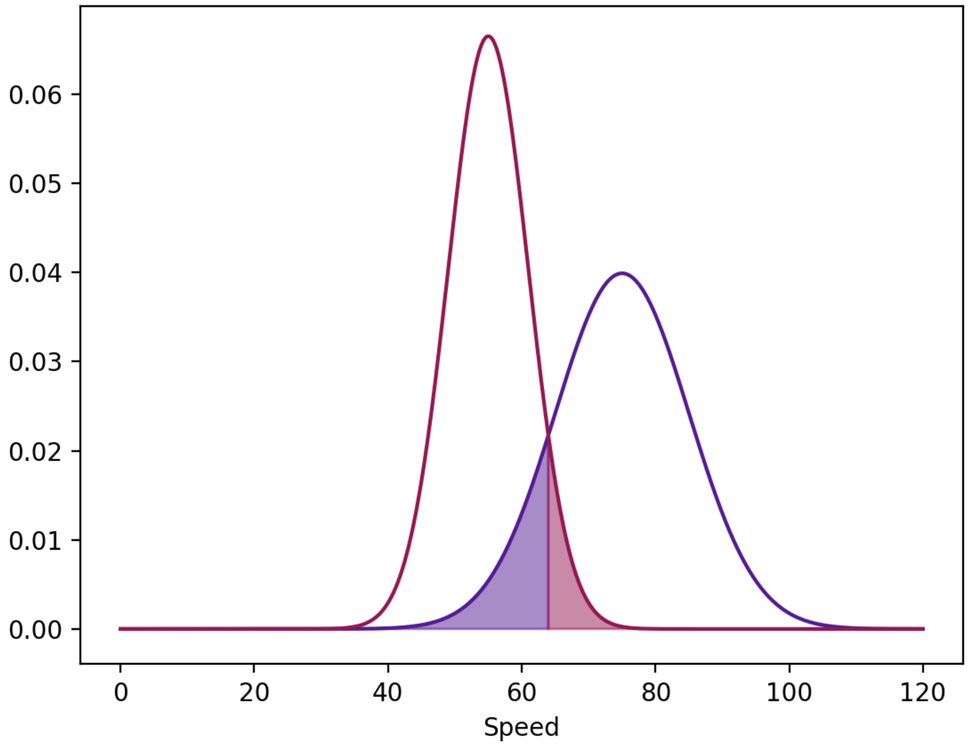
Note: here we are assuming that P(minivan) = P(sports car)

naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed

- Note that it is possible to misclassify!



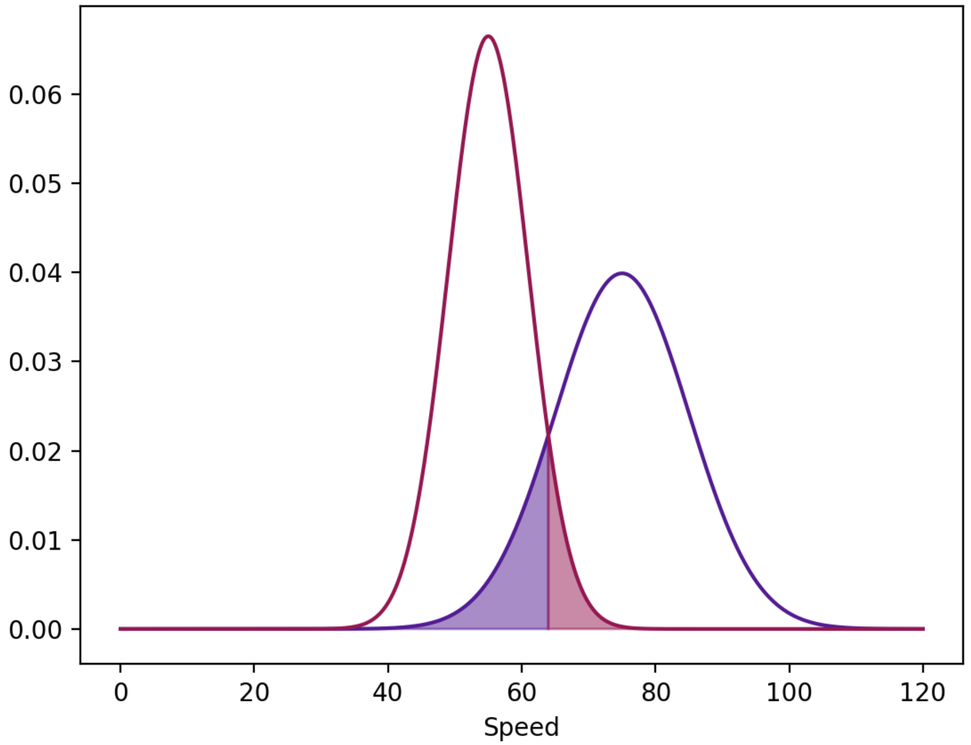
Note: here we are assuming that P(minivan) = P(sports car)

naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed

- Note that it is possible to misclassify!



Note: here we are assuming that P(minivan) = P(sports car)

naïve bayes in Python

- The sklearn.naive_bayes library (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes)
- Two main types of interest
  - Gaussian: from sklearn.naive_bayes import GaussianNB
  - Bernoulli: from sklearn.naive_bayes import BernoulliNB

```
from sklearn.datasets import load_iris
iris = load_iris()

X = iris.data
y = iris.target

from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.4, random_state=1)

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in
%):", metrics.accuracy_score(y_test, y_pred)*100)
```

what about higher dimensions?

- Suppose we have *d* features for each datapoint instead of just one
- Can use a **multivariable Gaussian**
  - Instead of defining in terms of mean and variance, define in terms of mean (of each dimension) and **covariance matrix** (cov command in numpy computes covariance)
  - Mean is *d* x 1 vector μ, covariance is *d* x *d* matrix Σ with determinant $|\Sigma|$

$$\mathcal{N}(\mathbf{x}\,|\,\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d\,|\,\Sigma\,|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

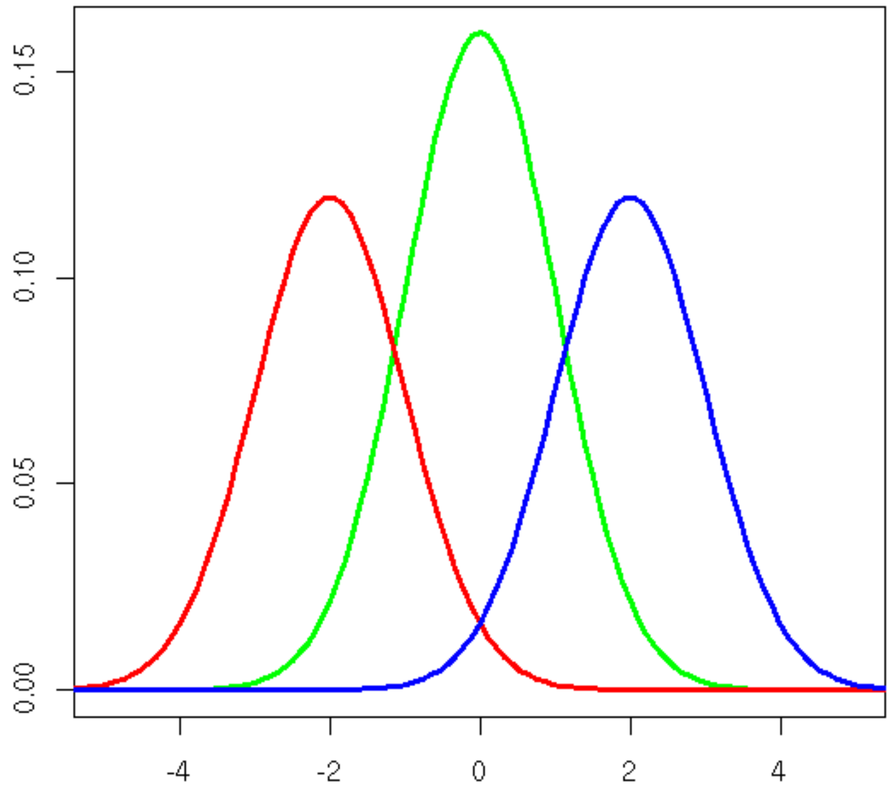
See: scipy.stats.multivariate_normal

what about multiple classes?

- We just compare more cases
- For k classes, the predicted class for datapoint x is

$$\arg\max_{i=0,\ldots,k-1} P(x \mid C_i)P(C_i)$$

- Issue with more classes is we have less data with which to infer each class' $P(x \mid C_i)$



pros vs cons

+ Easy to build classifier (once you have a model)

+ Easy to compute likelihood

+ Robust to outliers (don’t shift distribution much)

+ Good for missing data (distribution lets you estimate behaviors of data points you don’t have)

- Need to choose a model for the data (get it wrong, classifier will not be reliable)

- Need prior knowledge to build classifier (relative likelihood of classes)