Applied Text Mining in Python

Naïve Bayes Classifier

Case study: Classifying text search queries

- Suppose you are interested in classifying search queries in three classes: Entertainment, Computer Science, Zoology

- Most common class of the three is Entertainment.

Case study: Classifying text search queries

- Suppose the query is “Python”
  - Python, the snake (Zoology)
  - Python, the programming language (Computer Science)
  - Python, as in Monty Python (Entertainment)

- Most common class, given “Python”, is Zoology.

Case study: Classifying text search queries

- Suppose the query is "Python download"
- Most probable class, given "Python download", is Computer Science.

Probabilistic Model

- Update the likelihood of the class given new information
- Prior Probability: Pr(y = Entertainment), Pr(y = CS), Pr(y=Zoology)
- Posterior probability: Pr(y = Entertainment|x = “Python”)

Bayes’ Rule

- $$\text{Posterior probability} = \frac{\text{Prior probability x Likelihood}}{\text{Evidence}}$$

- $$\Pr(y \mid X) = \frac{\Pr(y) \times \Pr(X \mid y)}{\Pr(X)}$$

Naïve Bayes Classification

- $$\Pr(y=\text{CS}|\text{“Python”}) = \frac{\Pr(y=\text{CS}) \text{ x } \Pr(\text{“Python”} \mid y=\text{CS})}{\Pr(\text{“Python”})}$$

- $$\Pr(y=\text{Zoology}|\text{“Python”}) = \frac{\Pr(y=\text{Zoology}) \text{ x } \Pr(\text{“Python”} \mid y=\text{Zoology})}{\Pr(\text{“Python”})}$$

- $\Pr(y=\text{CS} \mid \text{“Python”}) > \Pr(y=\text{Zoology} \mid \text{“Python”})$ ⇨

Naïve Bayes Classification

- $\Pr(y \mid X) = \dfrac{\Pr(y) \times \Pr(X \mid y)}{\Pr(X)}$

$$y^* = \underset{y}{\operatorname{argmax}} \Pr(y \mid X) = \underset{y}{\operatorname{argmax}} \Pr(y) \times \Pr(X \mid y)$$

- Naïve assumption: Given the class label, features are assumed to be independent of each other

$$y^* = \underset{y}{\operatorname{argmax}} \Pr(y \mid X) = \underset{y}{\operatorname{argmax}} \Pr(y) \times \prod_{i=1}^{n} \Pr(x_i \mid y)$$

Naïve Bayes Classifier

$$y^* = \underset{y}{\text{argmax}}\ \Pr(y \mid X) = \underset{y}{\text{argmax}}\ \Pr(y) \times \prod_{i=1}^{n} \Pr(x_i \mid y)$$

Query: “Python download”

$$y^* = \underset{y}{\text{argmax}}\ \Pr(y) \times \Pr(\text{“Python”} \mid y) \times \Pr(\text{“download”} \mid y)$$

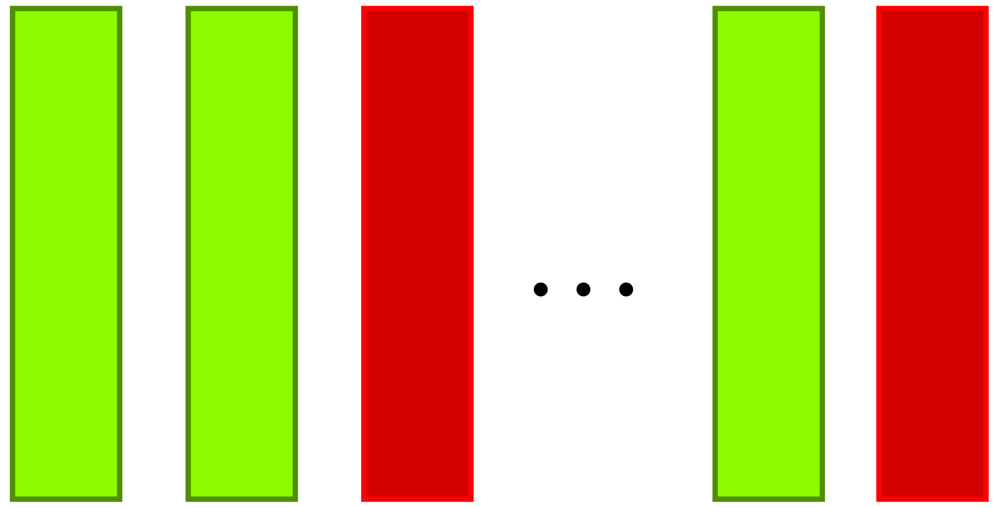
Naïve Bayes: What are the parameters?

- Prior probabilities: Pr(y) for all y in Y
- Likelihood: Pr(x_i | y) for all features x_i and labels y in Y

- If there are 3 classes ($|Y| = 3$) and 100 features in X, how many parameters does naïve Bayes models have?

Naïve Bayes: Learning parameters

- Prior probabilities: Pr(y) for all y in Y
  - Remember training data?



  - Count the number of instances in each class
  - If there are N instances in all, and n out of those are labeled as class y ⇒ Pr

Naïve Bayes: Learning parameters

- Likelihood: $Pr(x_i \mid y)$ for all features x_i and labels y in Y
  - Count how many times feature x_i appears in instances labeled as class y
  - If there are p instances of class y, and x_i appears in k of those, $Pr(x_i \mid y) = k / p$

Naïve Bayes: Smoothing

- What happens if $\Pr(x_i \mid y) = 0$?
  - Feature x_i never occurs in documents labeled y
  - But then, the posterior probability $\Pr(y \mid x_i)$ will be 0!!
- Instead, smooth the parameters
- Laplace smoothing or Additive smoothing: Add a dummy count
  - $\Pr(x_i \mid y) = (k+1) / (p+n)$; where n is number of features

Take Home Concepts

- Naïve Bayes is a probabilistic model
- Naïve, because it assumes features are independent of each other, given the class label
- For text classification problems, naïve Bayes models typically provide very strong baselines
- Simple model, easy to learn parameters