NAIVE

BAYESIAN

CLASSIFICATION

Bayes Formula

$$\underbrace{P\Big(C|x\Big)}_{} = \underbrace{P\Big(x|C\Big)}_{} \cdot \underbrace{P\Big(C\Big)}_{} / \underbrace{P\Big(x\Big)}_{}$$

$$\text{posterior} = \text{likelihood} \cdot \text{prior}/\text{evidence}$$

- one of main formulae in probability and statistics
- used for classification for labels
- Naive Bayesian classifier

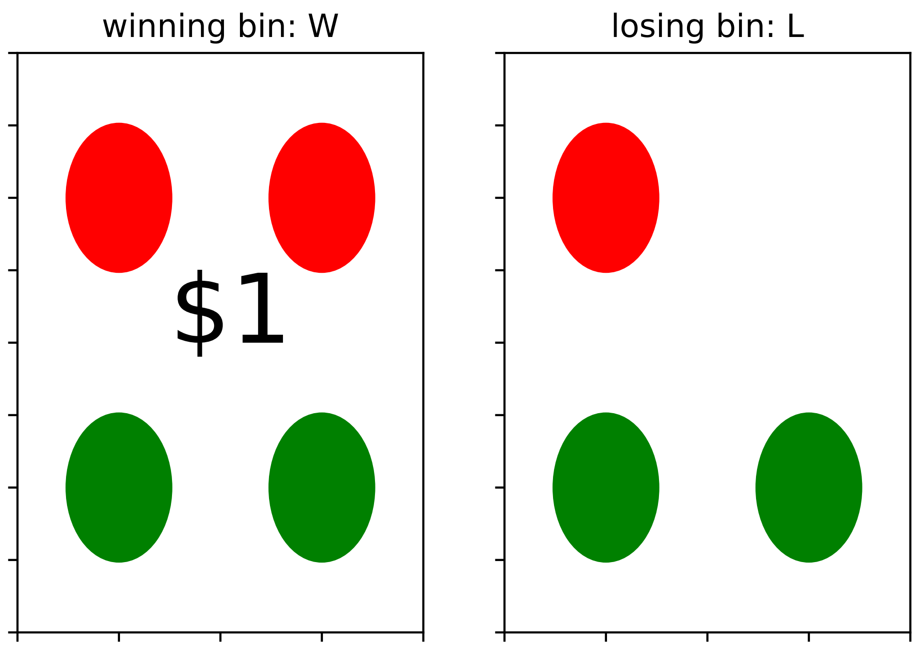
Bayes Formula

$$\underbrace{P\big(C|x\big)}_{} = \underbrace{P\big(x|C\big)}_{} \cdot \underbrace{P\big(C\big)}_{} / \underbrace{P\big(x\big)}_{}$$

$$\text{posterior} = \text{likelihood} \cdot \text{prior/evidence}$$

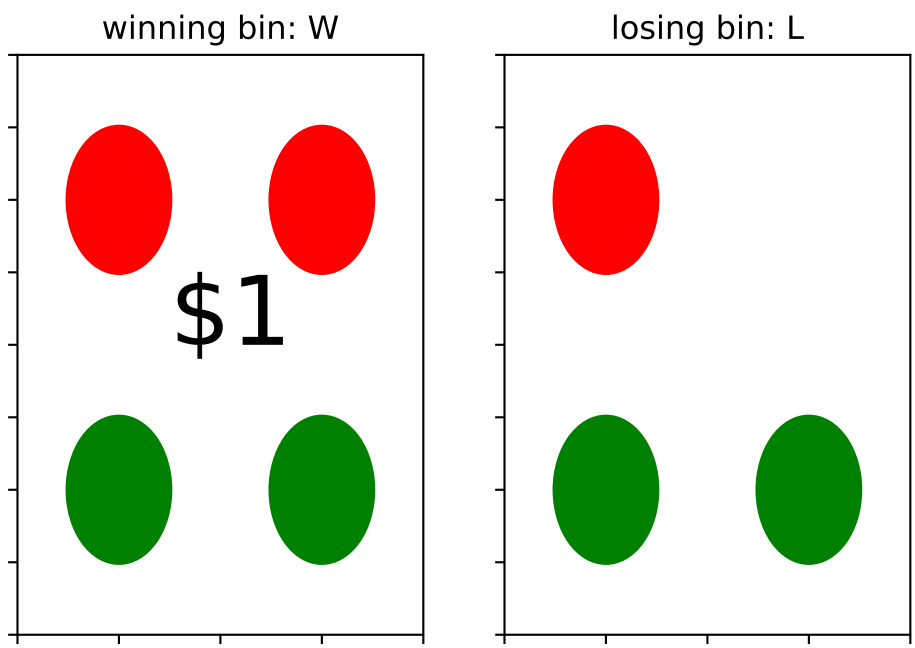
- $P\big(C|x\big)$ - conditional probability of class C given training inputs x
- $P\big(x|C\big)$ - conditional probability of training inputs x given class C
- $P\big(C\big)$ - probability of class C
- $P\big(x\big)$ - probability of the training data x

Bayes Formula Example



- two bins: W and L
- W: 2 red, 2 green tokens, $1
- L: 1 red, 2 green tokens

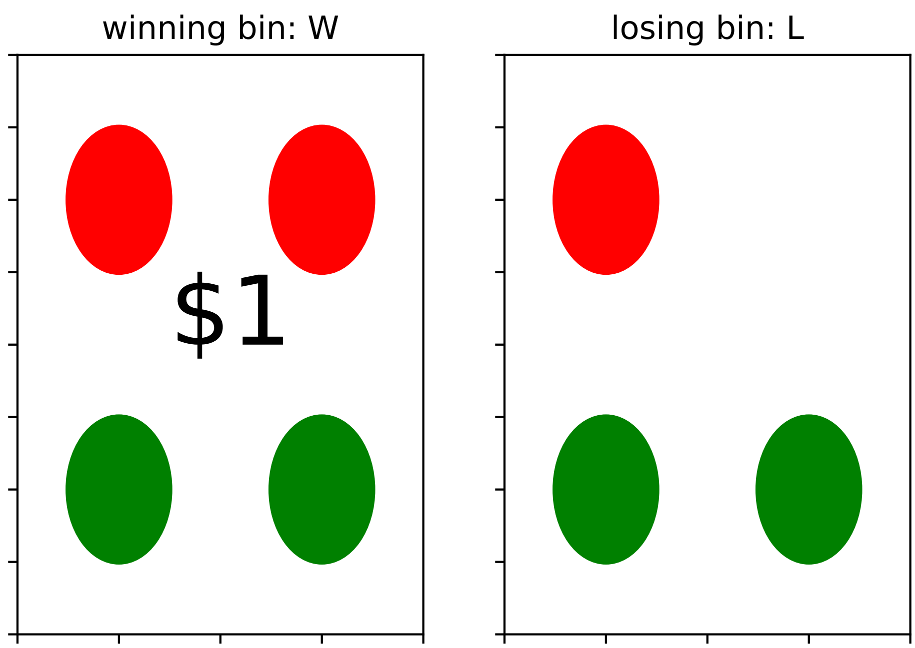
Bayes Formula Example



- equally likely choices
- prior probabilitites:

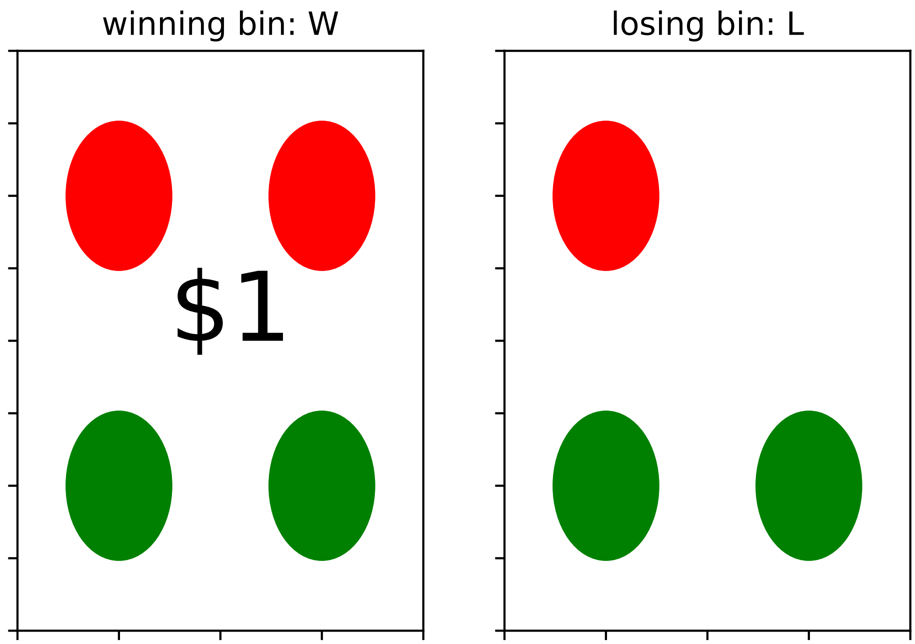
$$P(W) = P(L) = 0.5$$

Bayes Formula Example



- can see one token from a bin
- token is 'green'
- what is $P(W \mid g)$?

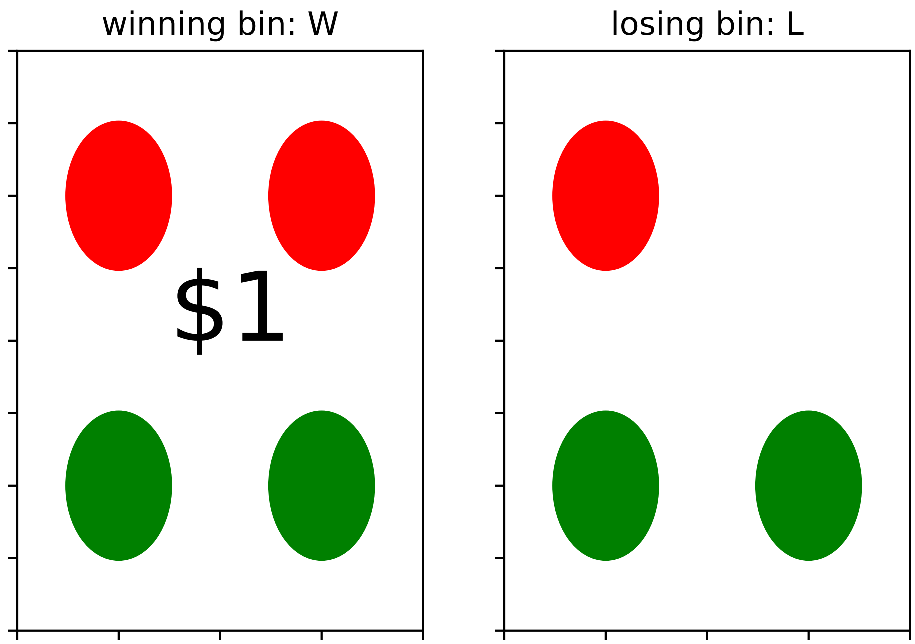
Intuition



- L has a higher percentage of green tokens
- likelihoods:

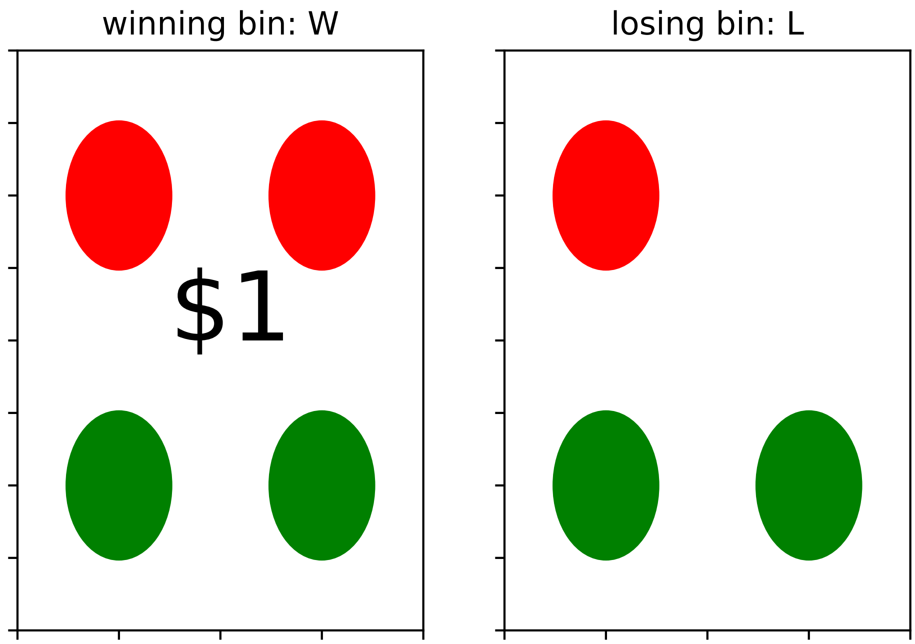
$$P(g|W) = 0.5, \quad P(g|L) = 2/3$$

Intuition



- a green token means more likely that we are looking at bin L

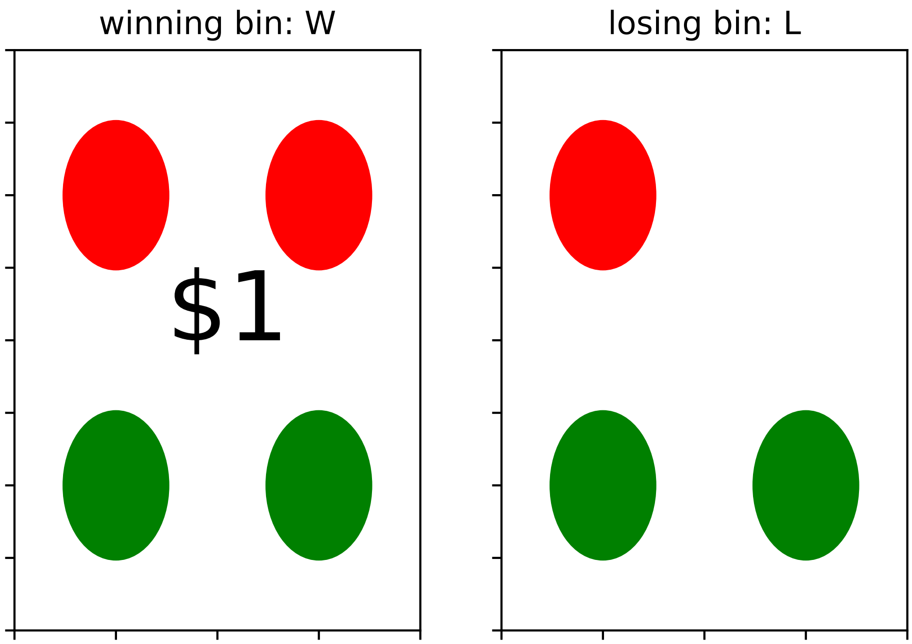
Applying Formula



$$\underbrace{P\Big(W|g\Big)}_{} = \underbrace{P\Big(g|W\Big)}_{} \cdot \underbrace{P\Big(W\Big)}_{} / \underbrace{P\Big(g\Big)}_{}$$

$$\text{posterior} = \text{likelihood} \cdot \text{prior}/\text{evidence}$$

Applying Formula



$$P\Big(W|g\Big) = \frac{P\Big(g|W\Big) \cdot P\Big(W\Big)}{P\Big(g|W\Big)P\Big(W\Big) + P\Big(g|L\Big) \cdot P\Big(L\Big)}$$
$$= \frac{0.5 \cdot 0.5}{0.5 \cdot 0.5 + (2/3) \cdot 0.5} = 0.43$$

Example: "Spam" Mail

- 40% of email is "spam" (S)
- 30% of spam contains "free" keyword (f)
- 1% of no-spam ($\neg S$) contains f keyword
- a new email contains f
- what is $P(s|f)$ - probability it is a spam?

Example: "Spam" Mail (cont'd)

$$
\begin{aligned}
P\Big(S\Big) &= 0.4 \\
P\Big(\neg S\Big) &= 0.6 \\
P\Big(f|s\Big) &= 0.3 \\
P\Big(f|\neg S\Big) &= 0.01 \\
P\Big(s|f\Big) &= ?
\end{aligned}
$$

- prior probability $p\Big(S\Big) = 0.4$
- intuition: $p\Big(S|f\Big) > 0.4$
- why? - we saw f

Computing $P\Big(S|f\Big)$

$$
\begin{aligned}
P\Big(S|f\Big) &= \frac{P\Big(f|S\Big)\cdot P\Big(S\Big)}{P\Big(f\Big)} \\
&= \frac{P\Big(f|S\Big)\cdot P\Big(S\Big)}{P\Big(f|S\Big)\cdot P\Big(S\Big) + P\Big(f|\neg S\Big)\cdot P\Big(\neg S\Big)} \\
&= \frac{0.3\cdot 0.4}{0.3\cdot 0.4 + 0.01\cdot 0.6} \\
&= \frac{0.12}{0.12 + 0.06} \\
&= 2/3
\end{aligned}
$$

- note $P\Big(S|f\Big) > P\Big(S\Big)$

Bayes Formula Example

Day	Weather	Play
1	sunny	**no**
2	rainy	**yes**
3	sunny	**yes**
4	rainy	**no**
5	sunny	**yes**
6	overcast	**yes**
7	sunny	**yes**
8	overcast	**yes**
9	rainy	**no**
10	rainy	**yes**

- assume Weather = sunny
- what is $P(\text{yes} \mid \text{sunny})$?

Python Code

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import \
                        LabelEncoder
from sklearn.naive_bayes import \
                        MultinomialNB

df = pd.DataFrame(
   {"Day":[1,2,3,4,5,6,7,8,9,10],
    "Weather":["sunny","rainy",
               "sunny","rainy",
               "sunny","overcast",
               "sunny","overcast",
               "rainy","rainy"],
    "Play":["no","yes","yes",
            "no","yes", "yes",
            "yes","yes",
            "no","yes"]},
    columns=["Day", "Weather",
             "Play"] )
```

Computing Priors

Day	Weather	Play
1	sunny	**no**
2	rainy	**yes**
3	sunny	**yes**
4	rainy	**no**
5	sunny	**yes**
6	overcast	**yes**
7	sunny	**yes**
8	overcast	**yes**
9	rainy	**no**
10	rainy	**yes**

$$P(\text{yes}) = 7/10$$
$$P(\text{no}) = 3/10$$

- unconditional probabilities

Computing Priors in Python

```
>> df_play = df["Play"].value_counts()
>> df_play
yes    7
no     3
Name: Play, dtype: int64
```

$$P(\text{yes}) = 7/10$$
$$P(\text{no}) = 3/10$$

Computing Evidence

Day	Weather	Play
1	sunny	**no**
2	rainy	**yes**
3	sunny	**yes**
4	rainy	**no**
5	sunny	**yes**
6	overcast	**yes**
7	sunny	**yes**
8	overcast	**yes**
9	rainy	**no**
10	rainy	**yes**

$$P(\text{sunny}) = 4/10$$
$$P(\text{rainy}) = 4/10$$
$$P(\text{overcast}) = 2/10$$

Computing Evidence in Python

```
>> df_weather = df.groupby(["Weather",
                              "Play"]).size()
>> df_weather
Weather   Play
overcast  yes     2
rainy     no      2
          yes     2
sunny     no      1
          yes     3
dtype: int64
```

$$P(\text{sunny}) = 4/10$$
$$P(\text{rainy}) = 4/10$$
$$P(\text{overcast}) = 2/10$$

- unconditional probabilities

Computing Likelihoods

Day	Weather	Play
1	sunny	**no**
2	rainy	**yes**
3	sunny	**yes**
4	rainy	**no**
5	sunny	**yes**
6	overcast	**yes**
7	sunny	**yes**
8	overcast	**yes**
9	rainy	**no**
10	rainy	**yes**

$$P(\text{sunny} \mid \text{yes}) = 3/7$$

- conditional probabilities

Computing Likelihood in Python

```
>> df_weather = df.groupby(["Weather",
                             "Play"]).size()
>> df_weather
Weather   Play
overcast  yes     2
rainy     no      2
          yes     2
sunny     no      1
          yes     3
dtype: int64
```

$$P(\text{sunny} \mid \text{yes}) = 3/7$$

Applying the Formula

- likelihood: $P(\text{sunny}|\ \text{yes}) = \frac{3}{7}$
- prior: $P(\text{yes}) = \frac{7}{10}$
- evidence: $P(\text{sunny}) = \frac{4}{10}$

$$
\begin{aligned}
P(\text{yes} \mid \text{sunny}) &= \frac{P(\text{sunny} \mid \text{yes}) P(\text{yes})}{P(\text{sunny})} \\
&= \frac{3}{7} \cdot \frac{7}{10} / \frac{4}{10} \\
&= \frac{3}{4}
\end{aligned}
$$

Naive Bayesian Classification

- assume multiple features $x_1, \ldots, x_n$ with class label C
- want to compute $P\Big(C|x_1, \ldots x_n\Big)$
- how? use the Bayes formula:

$$P\Big(C|x_1, \ldots x_n\Big) = \frac{P\Big(x_1, \ldots, x_n|C\Big) \cdot P\Big(C\Big)}{P\Big(x\Big)}$$

$$\propto P\Big(x_1, \ldots, x_n|C\Big) \cdot P\Big(C\Big)$$

- Q: where is difficulty?
- A: joint probability $P\Big(x_1, \ldots, x_n|C\Big)$

Naive Bayesian Classification (cont'd)

- assume feature independence

$$P(x_1, \ldots, x_n | C) = P(x_1|C) \cdots P(x_n|C)$$

- Bayes simplifies to

$$P(C|x_1, \ldots x_n) \propto \left[P(x_1|C) \cdots P(x_n|C)\right] P(C)$$

- assign $x^* = (a_1, \ldots, a_n)$ to class C^* with maximium value $P(C^*|x^*)$

$$C^* = \text{argmax}_C \left[P(a_1|C) \cdots P(a_n|C)\right] P(x^*)$$

Naive Bayesian in SkLearn

- three classifiers:

(a) *GaussianNB*: features are continuous and follow normal distribution

(b) *MultinomialNB*: features are frequencies

(c) *BernouilliNB*: features are boolean

Naive Bayesian Example

- additional features: wind and temperature

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- $x^* =$ (sunny, cold, low)?

Python Code

```
import pandas as pd
from sklearn.preprocessing import \
                    LabelEncoder
from sklearn.naive_bayes import \
                    MultinomialNB
df = pd.DataFrame(
{"Day":[1,2,3,4,5,6,7,8,9,10],
"Weather":["sunny","rainy","sunny",
      "rainy","sunny","overcast","sunny",
      "overcast","rainy","rainy"],
"Temperature":["hot","mild","cold","cold",
  "cold","mild","hot","hot","hot","mild"],
"Wind":["low","high","low","high","high",
       "low","low","high", "high", "low"],
"Play": ["no","yes","yes","no","yes",
          "yes","yes","yes","no","yes"]},
columns=["Day", "Weather","Temperature",
           "Wind","Play"] )
```

- $x^* = (\text{sunny, cold, low})$?

Naive Bayesian Example (cont'd)

- x* = (sunny, cold, low)?
- need a label $C*$ to maximize

$$P(\text{sunny}|C*) \cdot P(\text{cold}|C*) \cdot P(\text{low}|C*)$$

- need conditional probabilities
  1. weather: $P(\text{sunny}|C)$
  2. temperature: $P(\text{cold}|C)$
  3. wind: $P(\text{low}|C)$
- compute during learning phase

Cond. Probability: Weather

Weather	Play = **yes**	Play = **no**
sunny	3/7	1/3
overcast	2/7	0
rainy	2/7	2/3

$$P(\text{sunny} \mid \text{yes}) = 3/7$$
$$P(\text{sunny} \mid \text{no}) = 1/3$$

Cond. Probability: Weather (Python)

```
>> df_weather = data.groupby(["Weather",
                               "Play"]).size()
>> df_weather
Weather   Play
overcast  yes     2
rainy     no      2
          yes     2
sunny     no      1
          yes     3
dtype: int64
```

$$P(\text{sunny} \mid \text{yes}) = 3/7$$

$$P(\text{sunny} \mid \text{no}) = 1/3$$

Cond. Probability: Temperature

Temperature	Play = **yes**	Play = **no**
cold	2/7	1/3
mild	3/7	0
hot	2/7	2/3

$$P(\text{cold} \mid \text{yes}) = 2/7$$
$$P(\text{cold} \mid \text{no}) = 1/3$$

Cond. Probability: Temperature (Python)

```
>> df_temp = data.groupby(["Temperature",
                           "Play"]).size()
>> df_temp
Temperature  Play
cold         no      1
             yes     2
hot          no      2
             yes     2
mild         yes     3
dtype: int64
```

$$P(\text{cold} \mid \text{yes}) = 2/7$$
$$P(\text{cold} \mid \text{no}) = 1/3$$

Cond. Probability: Wind

Wind	Play = **yes**	Play = **no**
high	3/7	2/3
low	4/7	1/3

$$P(\text{low} \mid \text{yes}) = 4/7$$
$$P(\text{low} \mid \text{no}) = 1/3$$

Cond. Probability: Wind (Python)

```
>> df_wind = data.groupby(["Wind",
                           "Play"]).size()
>> df_wind
Wind  Play
high  no      2
      yes     3
low   no      1
      yes     4
dtype: int64
```

$$P(\text{low} \mid \text{yes}) = 4/7$$
$$P(\text{low} \mid \text{no}) = 1/3$$

Computing Label

- $x^* =$ (sunny, cold, low)?

$$
\begin{aligned}
P(\text{yes}|x^*) =& P(\text{sunny}|\text{yes})P(\text{cold}|\text{yes}) \\
& \cdot P(\text{low}|\text{yes})P(\text{yes}) \\
=& 3/7 \cdot 2/7 \cdot 4/7 \cdot 0.7 = 0.049
\end{aligned}
$$

$$
\begin{aligned}
P(\text{no}|x^*) =& P(\text{sunny}|\text{no})P(\text{cold}|\text{no}) \\
& \cdot P(\text{low}|\text{no})P(\text{no}) \\
=& 1/3 \cdot 1/3 \cdot 1/3 \cdot 0.3 = 0.011
\end{aligned}
$$

- $P(\text{yes}|x^*) > P(\text{no}|x^*)$
- therefore, $x^* \mapsto$ yes

Python Code

```python
import numpy as np
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import LabelEncoder

data = pd.DataFrame(
        {'Day':          [1,2,3,4,5,6,7,8,9,10],
        'Weather':       ['sunny','rainy','sunny','rainy',
                          'sunny','overcast','sunny','overcast',
                          'rainy','rainy'],
        'Temperature': ['hot', 'mild', 'cold','cold','cold',
                          'mild','hot','hot', 'hot','mild'],
        'Wind':          ['low','high','low','high','high',
                          'low','low', 'high','high','low'],
        'Play':          ['no', 'yes','yes','no','yes',
                          'yes','yes','yes','no','yes']},
        columns = ['Day','Weather','Temperature','Wind','Play'])

input_data = data[['Weather', 'Temperature', 'Wind']]
dummies = [pd.get_dummies(data[c]) for c in input_data.columns]
binary_data = pd.concat(dummies, axis=1)
X = binary_data[0:10].values
le = LabelEncoder()
Y = le.fit_transform(data['Play'].values)
NB_classifier = MultinomialNB().fit(X, Y)
new_instance = np.asmatrix([0,0,1,1,0,0,0,1])
prediction = NB_classifier.predict(new_instance)
```

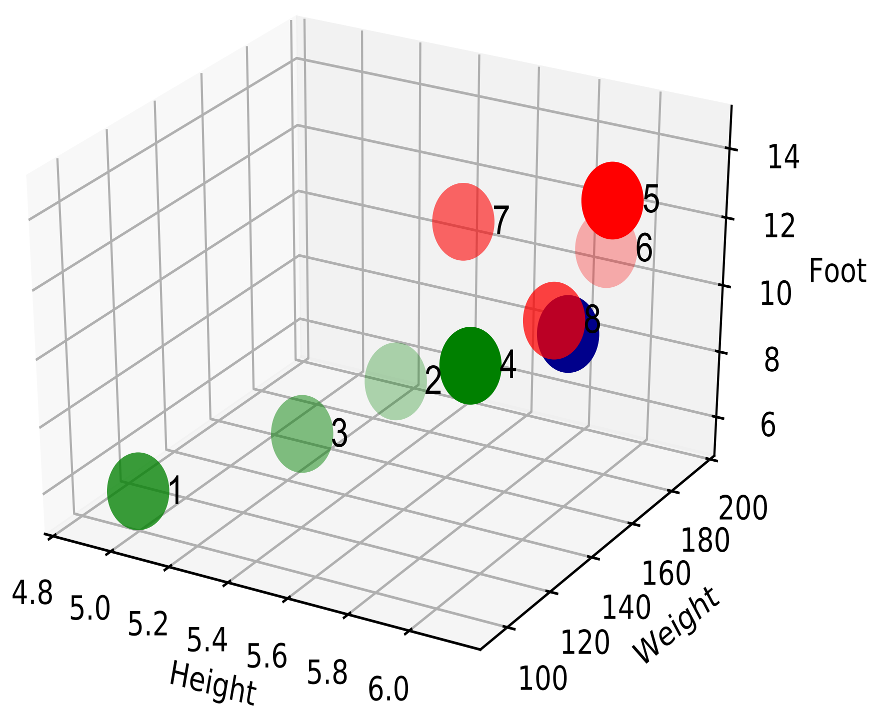
```
ipdb> prediction[0]
1
```

Naive Bayesian: Continuous Case

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

- $P(\text{green}) = P(\text{red}) = 0.5$
- $(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto ?$

Naive Bayesian: Continuous Case (cont'd)



$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto ?$$

Computing the Label

- assume H, W, F independent

$$P\left(C|\, x^*\right) = \frac{P\left(x^*|C\right)P\left(C\right)}{P\left(x^*\right)}$$

$$= \frac{P\left(H|C\right)P\left(W|C\right)P\left(F|C\right)P\left(C\right)}{P\left(x^*\right)}$$

- term $P\left(x^*\right)$ same for each class
- assign C^* to x^* if for $C \neq C^*$:

$$P\left(H|C^*\right)P\left(W|C^*\right)P\left(F|C^*\right)P\left(C^*\right)$$

$$>$$

$$P\left(H|C\right)P\left(W|C\right)P\left(F|C\right)P\left(C\right)$$

Computing the Parameters

- assume Gaussian

label C	height H		weight W		foot F	
	μ	σ	μ	σ	μ	σ
green	5.39	0.27	132.5	20.46	7.5	1.12
red	5.86	0.16	176.25	9.60	11.5	1.12

Python Code for Probabilities

```
import pandas as pd

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
    'Label': ['green', 'green', 'green', 'green',
              'red', 'red', 'red', 'red'],
    'Height': [5, 5.5, 5.33, 5.75,
           6.00, 5.92,  5.58, 5.92],
    'Weight': [100, 150, 130, 150,
           180, 190, 170, 165],
    'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
    columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

df_1 = data[data['Label']=='green']
h_mu_1    = df_1['Height'].values.mean()
h_sigma_1 = df_1['Height'].values.std()
w_mu_1    = df_1['Weight'].values.mean()
w_sigma_1 = df_1['Weight'].values.std()
f_mu_1    = df_1['Foot'].values.mean()
f_sigma_1 = df_1['Foot'].values.std()

df_2 = data[data['Label']=='red']
h_mu_2    = df_2['Height'].values.mean()
h_sigma_2 = df_2['Height'].values.std()
w_mu_2    = df_2['Weight'].values.mean()
w_sigma_2 = df_2['Weight'].values.std()
f_mu_2    = df_2['Foot'].values.mean()
f_sigma_2 = df_2['Foot'].values.std()
```

Python Code for Cond. Probabilities

```
from scipy.stats import norm

h, w, f = 6, 160, 10
p_green = 0.5; p_red = 0.5

prob_h_green = norm.pdf((h - h_mu_1)/h_sigma_1)
prob_w_green = norm.pdf((w - w_mu_1)/w_sigma_1)
prob_f_green = norm.pdf((f - f_mu_1)/f_sigma_1)

prob_h_red = norm.pdf((h - h_mu_2)/h_sigma_2)
prob_w_red = norm.pdf((w - w_mu_2)/w_sigma_2)
prob_f_red = norm.pdf((f - f_mu_2)/f_sigma_2)

# unnormalized probabilities
posterior_red = p_red *prob_h_red *prob_w_red *prob_f_red
posterior_green = p_green *prob_h_green *prob_w_green *prob_f_green

normalized_red = posterior_red /(posterior_red + posterior_green)
normalized_green = posterior_green /(posterior_red + posterior_green)
```

```
ipdb> normalized_red
0.96
```

$$x^* = (H = 6, W = 160, F = 10) \mapsto \text{red}$$

Python Code for Naive Bayesian

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values

NB_classifier = GaussianNB().fit(X, Y)
new_instance = np.asmatrix(np.asmatrix([6, 160, 10])
prediction = NB_classifier.predict(new_instance)
```

```
ipdb> prediction[0]
red
```

Naive Bayesian: IRIS

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

url = r'https://archive.ics.uci.edu/ml/' + \
          r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                            'petal-length', 'petal-width']
data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                         'petal-length', 'petal-width', 'Class'])

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
                                     test_size=0.5,random_state=3)
NB_classifier = GaussianNB().fit(X_train, Y_train)
prediction = NB_classifier.predict(X_test)
error_rate = np.mean(prediction != Y_test)
```

```
ipdb> error_rate
0.04
```

Concepts Check:

(a) conditional probability

(b) Bayes formula

(c) prior and posterior probabilities

(d) feature independence assumption

(e) naive bayesian classification

(f) discrete cases

(g) continuous case