Naïve Bayes Classifier

Data Set: **iris**

Naïve Bayes Explanation

Data Set: **weather & play**

Source: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained last accessed on 21 September 2019

Basics

Likelihood which is the probability of *predictor* given class

Prior probability of *class*

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Posterior probability of *class (c, target) given* predictor (x, attribute)

Prior probability of *predictor*

Step 1: Make Frequency Table
Step 2: Make Likelihood Table

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

14 Data Points

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Overcast appeared 4 times and all the time Play happened; Rainy appeared 5 times and *three* times Play NOT happened whereas *two* times play happened; Sunny appeared 5 times and *two* times Play NOT happened whereas *three* times play happened. All in all *5 times play NOT happened and 9 times play happened.*

Priory probabilities

Step 3: Calculate Posterior Probability

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Players will play if weather is sunny. Is this correct?

$$P(Yes|Sunny) = \frac{P(Sunny|Yes)P(Yes)}{P(Sunny)}$$

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

$$P(Yes|Sunny) = \frac{P(\frac{3}{9})P(9/14)}{P(5/14)}$$

$$P(Sunny|Yes)$$
$$P(\frac{3}{9})$$

How many times weather was sunny when Play happened? = 3
Total Yes = 9, so, denominator.

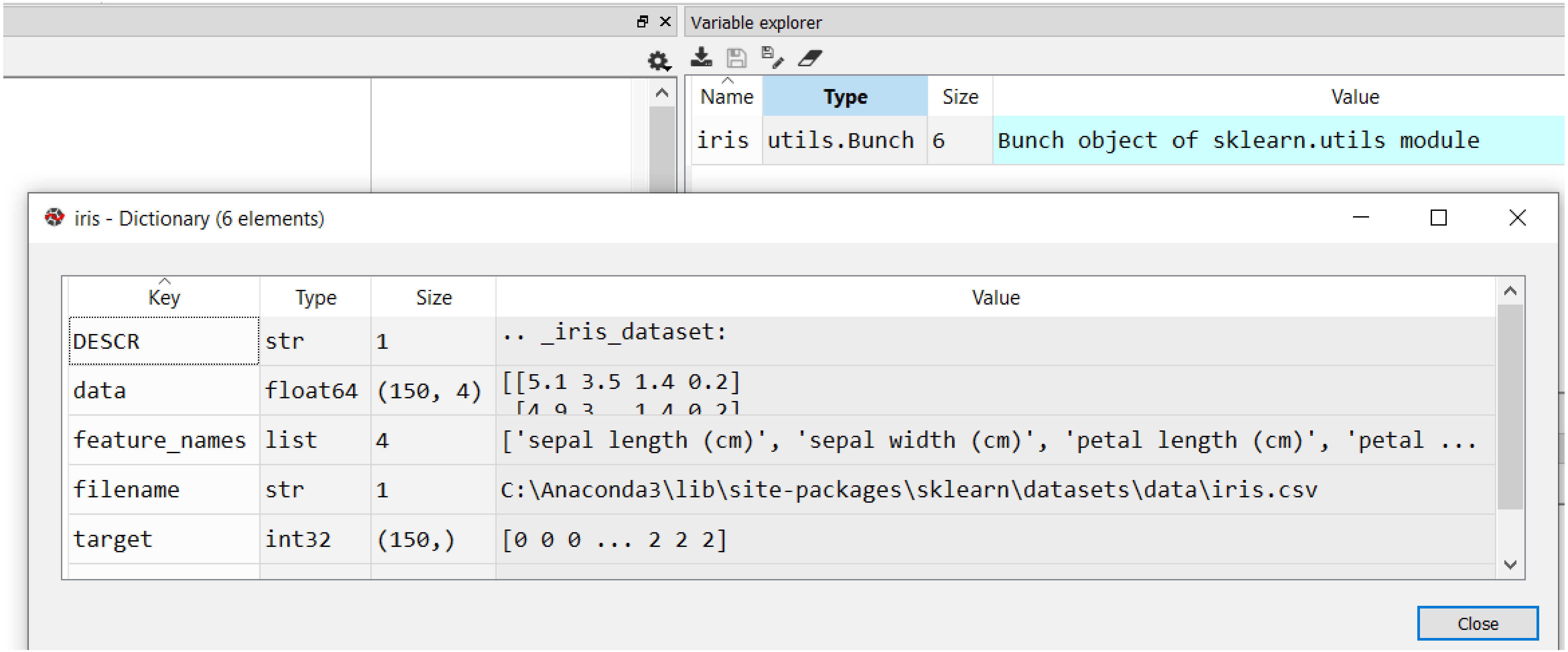
$$P(Yes|Sunny) = \frac{0.33 \times 0.64}{0.36} = 0.60; High\ Probabililty$$

```python
# Jesus is my Saviour!

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
```

```
In [1]: # Jesus is my Saviour!

In [2]: import pandas as pd

In [3]: import numpy as np

In [4]: import matplotlib.pyplot as plt

In [5]: from sklearn.datasets import load_iris

In [6]: from sklearn.model_selection import train_test_split

In [7]: from sklearn.naive_bayes import GaussianNB

In [8]: from sklearn.metrics import confusion_matrix

In [9]: from sklearn import metrics

In [10]: from sklearn.metrics import classification_report
```

```python
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
```


```
In [11]: # load the iris dataset

In [12]: from sklearn.datasets import load_iris

In [13]: iris = load_iris()
```



Variable explorer

Name	Type	Size	Value
iris	utils.Bunch	6	Bunch object of sklearn.utils module

iris - Dictionary (6 elements)

Key	Type	Size	Value
DESCR	str	1	.. _iris_dataset:
data	float64	(150, 4)	[[5.1 3.5 1.4 0.2]
feature_names	list	4	['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal ...
filename	str	1	C:\Anaconda3\lib\site-packages\sklearn\datasets\data\iris.csv
target	int32	(150,)	[0 0 0 ... 2 2 2]

Close

iris - Dictionary (6 elements)

Key	Type	Size	Value
DESCR	str	1	.. _iris_dataset:
data	float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
feature_names	list	4	['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal ...
filename	str	1	C:\Anaconda3\lib\site-packages\sklearn\datasets\data\iris.csv
target	int32	(150,)	[0 0 0 ... 2 2 2]
target_names	str320	(3,)	ndarray object of numpy module

```
# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
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)
```

```
In [14]: # store the feature matrix (X) and response vector (y)

In [15]: X = iris.data

In [16]: y = iris.target

In [17]: # splitting X and y into training and testing sets

In [18]: from sklearn.model_selection import train_test_split

In [19]: X_train, X_test,y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=1)
```

Variable explorer

Name	Type	Size	Value
X	float64	(150, 4)	[[5.1 3.5 1.4 0.2]
X_test	float64	(60, 4)	[[5.8 4. 1.2 0.2]
X_train	float64	(90, 4)	[[4.8 3.4 1.6 0.2]
iris	utils.Bunch	6	Bunch object of sklearn.utils module
y	int32	(150,)	[0 0 0 ... 2 2 2]
y_test	int32	(60,)	[0 1 1 ... 1 2 1]
y_train	int32	(90,)	[0 2 2 ... 1 2 0]

Variable explorer | File explorer | Help

```python
# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```


```
In [20]: # training the model on training set

In [21]: from sklearn.naive_bayes import GaussianNB

In [22]: gnb = GaussianNB()

In [23]: gnb.fit(X_train, y_train)
Out[23]: GaussianNB(priors=None, var_smoothing=1e-09)
```

sklearn.naive_bayes.GaussianNB

class `sklearn.naive_bayes.` **GaussianNB** (*priors=None*, *var_smoothing=1e-09*) [source]

Gaussian Naive Bayes (GaussianNB)

Can perform online updates to model parameters via **`partial_fit`** method. For details on algorithm used to update feature means and variance online, see Stanford CS tech report STAN-CS-79-773 by Chan, Golub, and LeVeque:

http://i.stanford.edu/pub/cstr/reports/cs/tr/79/773/CS-TR-79-773.pdf

Read more in the User Guide.

Parameters:	**priors : *array-like, shape (n_classes,)*** Prior probabilities of the classes. If specified the priors are not adjusted according to the data. **var_smoothing : *float, optional (default=1e-9)*** Portion of the largest variance of all features that is added to variances for calculation stability.


```
# making predictions on the testing set
y_pred = gnb.predict(X_test)

y_pred
```

```
In [24]: y_pred = gnb.predict(X_test)

In [25]: y_pred
Out[25]:
array([0, 1, 1, 0, 2, 2, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0,
1, 1,
       2, 0, 2, 1, 0, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0,
1, 2,
       1, 2, 0, 0, 0, 1, 0, 0, 2, 2, 2, 2, 2, 1, 2, 1])
```

```python
# comparing actual response values (y_test) with predicted response values (y_pred)
# by manual cross tabulation

pd.crosstab(y_test, y_pred, margins = True)
```


```
In [26]: # comparing actual response values (y_test) with predicted
response values (y_pred)

In [27]: # by manual cross tabulation

In [28]: pd.crosstab(y_test, y_pred, margins = True)
Out[28]:
col_0   0   1   2  All
row_0
0      19   0   0   19
1       0  19   2   21
2       0   1  19   20
All    19  20  21   60
```

```
# by Making the confusion Matrix through sklearn
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
In [29]: # by Making the confusion Matrix through sklearn

In [30]: from sklearn.metrics import confusion_matrix

In [31]: cm = confusion_matrix(y_test, y_pred)

In [32]: cm
Out[32]:
array([[19,  0,  0],
       [ 0, 19,  2],
       [ 0,  1, 19]], dtype=int64)
```

```
# calculating accuracy manually
(19+19+19)/(19+19+19+1+2) # 0.95
```

```
In [28]: pd.crosstab(y_test, y_pred, margins = True)
Out[28]:
col_0   0   1   2  All
row_0
0      19   0   0   19
1       0  19   2   21
2       0   1  19   20
All    19  20  21   60
```


```
In [33]: # calculating accuracy manually

In [34]: (19+19+19)/(19+19+19+1+2) # 0.95
Out[34]: 0.95
```

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


```
In [35]: # accuracy from sklearn

In [36]: from sklearn import metrics

In [37]: print("Gaussian Naive Bayes model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
Gaussian Naive Bayes model accuracy(in %): 95.0
```

```python
# classification report
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
In [28]: pd.crosstab(y_test, y_pred, margins = True)
Out[28]:
col_0    0   1   2  All
row_0
0       19   0   0   19
1        0  19   2   21
2        0   1  19   20
All     19  20  21   60
```

```
In [38]: # classification report

In [39]: from sklearn.metrics import classification_report

In [40]: print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.95      0.90      0.93        21
           2       0.90      0.95      0.93        20

    accuracy                           0.95        60
   macro avg       0.95      0.95      0.95        60
weighted avg       0.95      0.95      0.95        60
```


2 teach is
+
2 touch lives
4 ever