# Handling Numeric Features

- How to classify when features take numeric values?

| Example | Rain Recently (RR) | Rain Today (RT) | Temp (T) | Wind (W) | Sunshine (S) | Swimming |
|---|---|---|---|---|---|---|
| *X0* | Moderate | Moderate | 9 | Light | Some | ??? |

- **Option 1:** Discretise the feature to take fixed number of values. e.g. Temp = {cool, mild, hot}

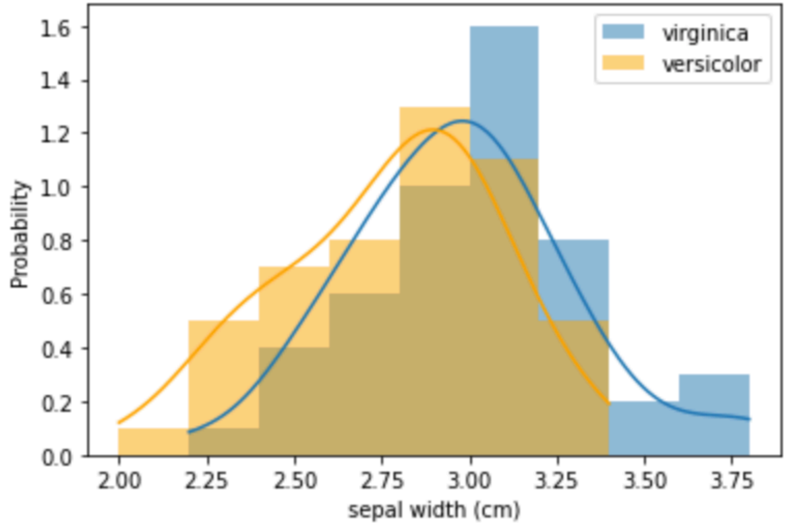
- **Option 2:** Assume that the feature fits to some distribution. e.g. for a Normal Distribution:

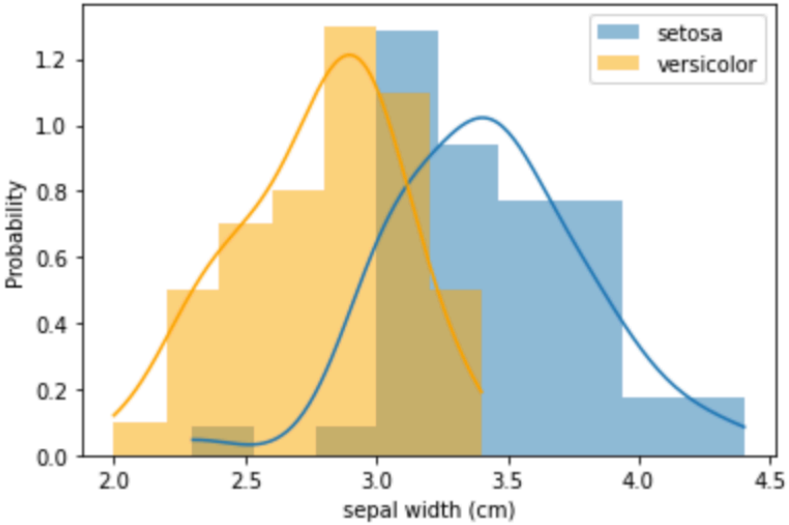
  1. For numeric feature $f_i$, store mean $\mu_i$ and standard deviation $\sigma_i$ for each class $v_j$
  2. When classifying, find the probability that the feature value fits the distribution $N(\mu_i, \sigma_i^2)$

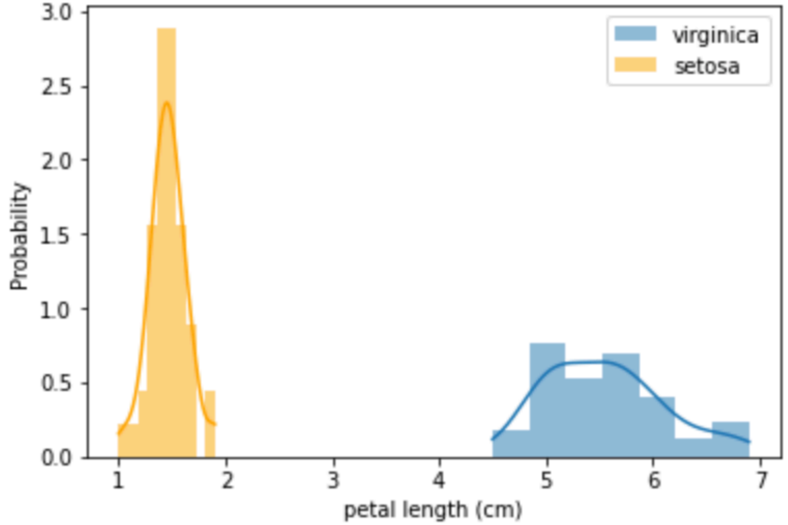
# Handling Numeric Features

- If data is Normal (Gaussian) probability can be calculated as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



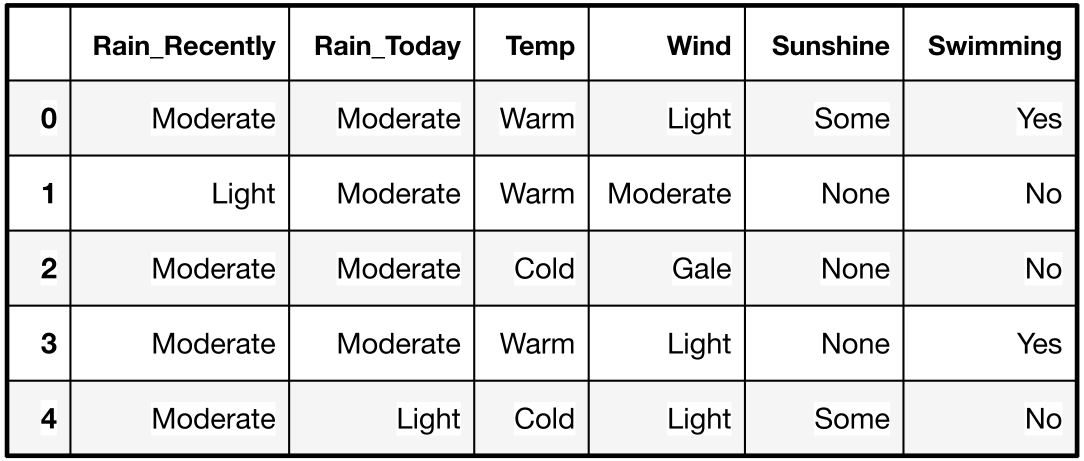




# Naive Bayes in scikit-learn

**Notebook:**
04 Naive Bayes

- Multiple implementations suitable for different types of data.
  - □ `CategoricalNB` will work with categorical data once it is processed using an `OrdinalEncoder`
  - □ `GaussianNB` assumes the numeric features have a Gaussian distribution
  - □ `BernoulliNB` binary data
  - □ `MultinomialNB` count data, e.g. word counts



| | Rain_Recently | Rain_Today | Temp | Wind | Sunshine | Swimming |
|---|---|---|---|---|---|---|
| **0** | Moderate | Moderate | Warm | Light | Some | Yes |
| **1** | Light | Moderate | Warm | Moderate | None | No |
| **2** | Moderate | Moderate | Cold | Gale | None | No |
| **3** | Moderate | Moderate | Warm | Light | None | Yes |
| **4** | Moderate | Light | Cold | Light | Some | No |

# Categorical Naive Bayes

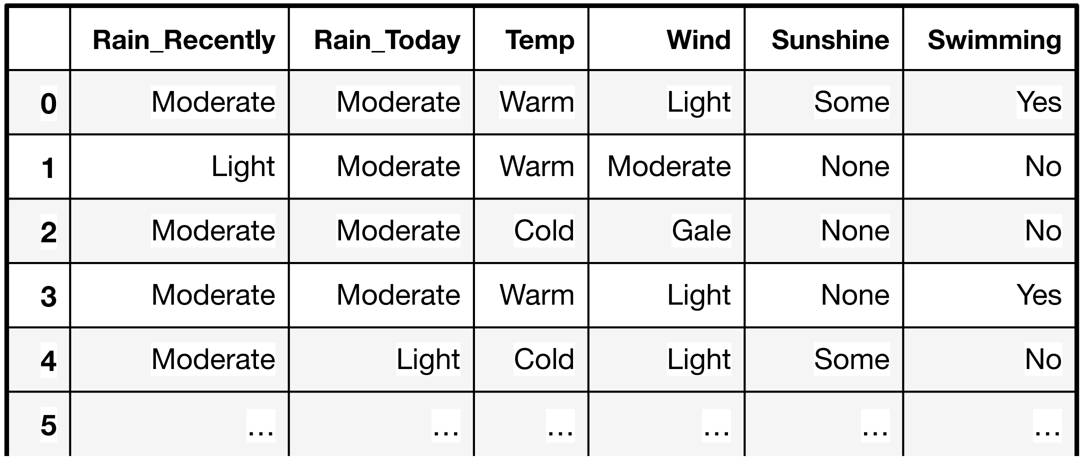
- Use OrdinalEncoder to convert to numbers
- CategoricalNB will treat these as categories

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

swim = pd.read_csv(‘Swimming.csv')
y = swim.pop('Swimming').values

ord_encoder = OrdinalEncoder()
swimOE = ord_encoder.fit_transform(swim)
```

**swim**



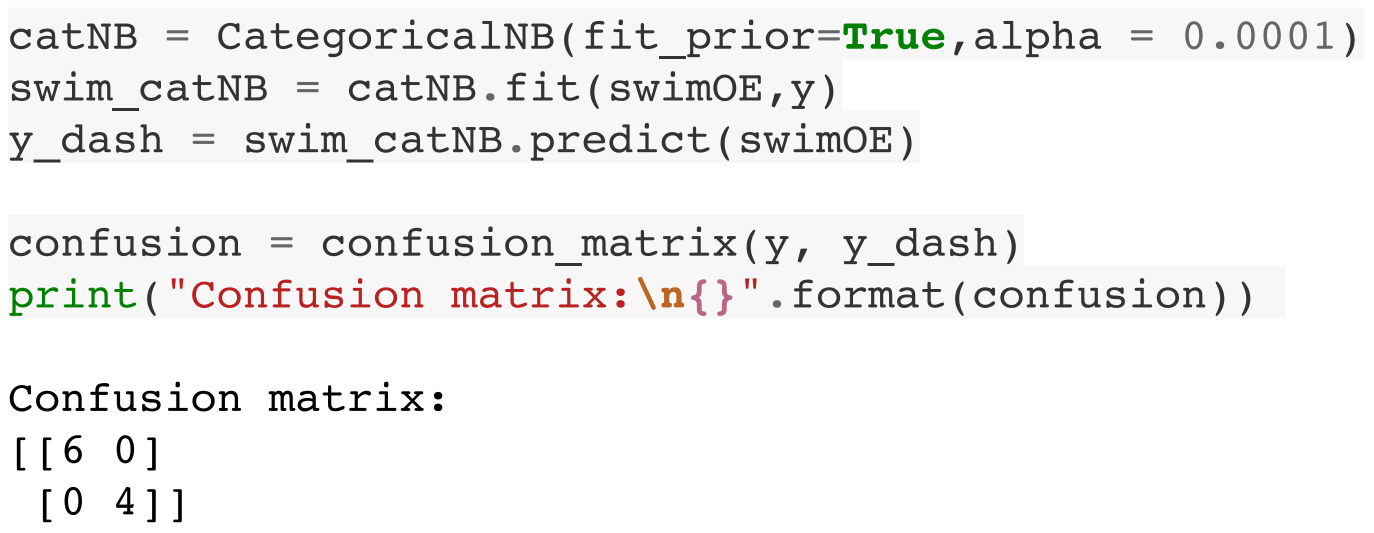
| | Rain_Recently | Rain_Today | Temp | Wind | Sunshine | Swimming |
|---|---|---|---|---|---|---|
| **0** | Moderate | Moderate | Warm | Light | Some | Yes |
| **1** | Light | Moderate | Warm | Moderate | None | No |
| **2** | Moderate | Moderate | Cold | Gale | None | No |
| **3** | Moderate | Moderate | Warm | Light | None | Yes |
| **4** | Moderate | Light | Cold | Light | Some | No |
| **5** | ... | ... | ... | ... | ... | ... |

**swimOE**

```
array([[2., 2., 1., 1., 1.],
       [1., 2., 1., 2., 0.],
       [2., 2., 0., 0., 0.],
       [2., 2., 1., 1., 0.],
       [2., 1., 0., 1., 1.],
       [0., 1., 0., 2., 1.],
       [1., 1., 0., 2., 1.],
       [2., 2., 0., 0., 1.],
       [0., 0., 1., 2., 0.],
       [1., 1., 0., 1., 1.]])
```

# Train a Naive Bayes model

- Train and test on training data



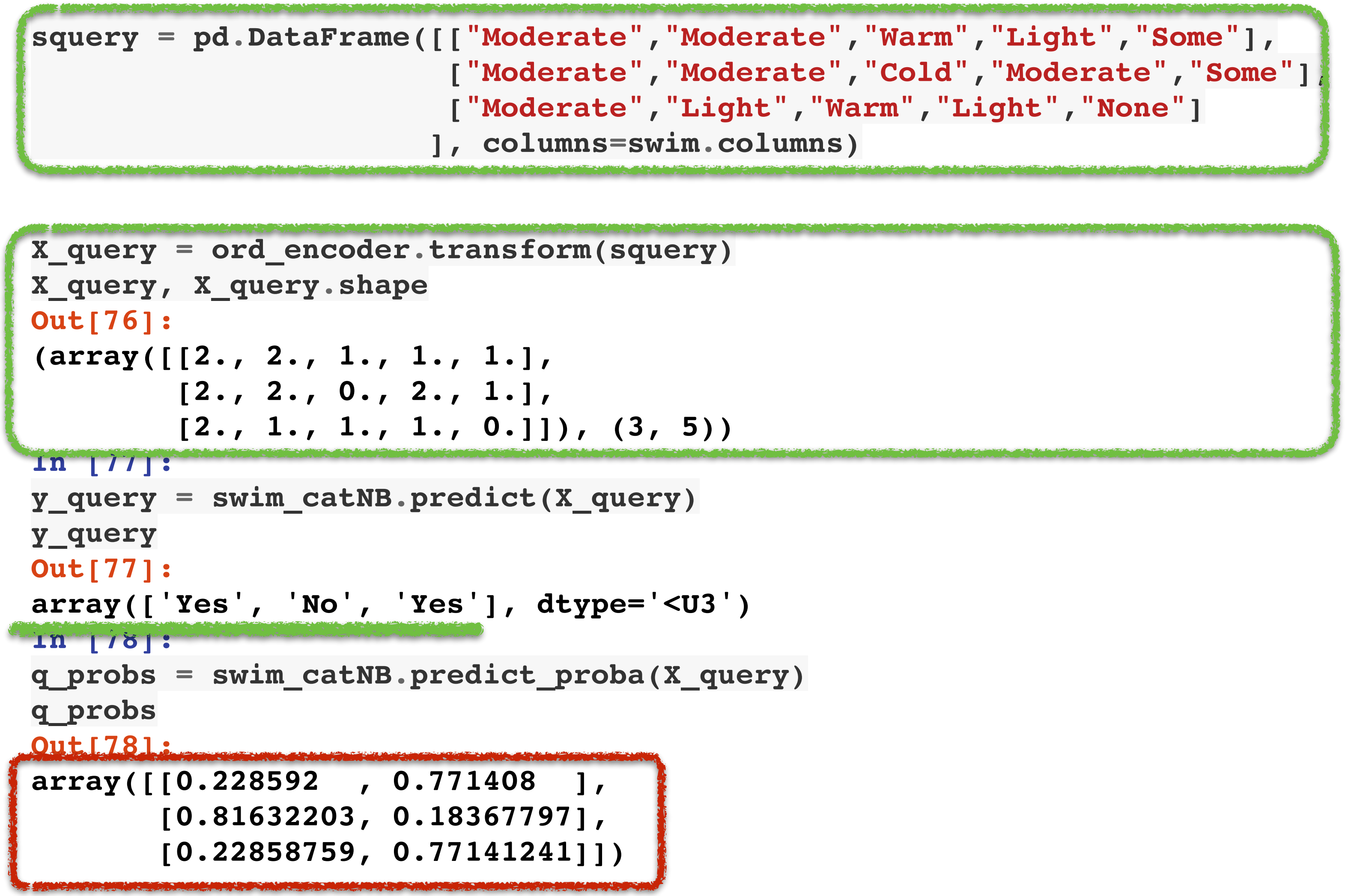
```
catNB = CategoricalNB(fit_prior=True,alpha = 0.0001)
swim_catNB = catNB.fit(swimOE,y)
y_dash = swim_catNB.predict(swimOE)

confusion = confusion_matrix(y, y_dash)
print("Confusion matrix:\n{}".format(confusion))

Confusion matrix:
[[6 0]
 [0 4]]
```

# Test with other data

- Set up a dataframe and then do an OrdinalTransform



Query Dataframe

```
squery = pd.DataFrame([["Moderate","Moderate","Warm","Light","Some"],
                       ["Moderate","Moderate","Cold","Moderate","Some"],
                       ["Moderate","Light","Warm","Light","None"]
                      ], columns=swim.columns)
```

OrdinalEncoder Format

```
X_query = ord_encoder.transform(squery)
X_query, X_query.shape
Out[76]:
(array([[2., 2., 1., 1., 1.],
        [2., 2., 0., 2., 1.],
        [2., 1., 1., 1., 0.]]), (3, 5))
In [77]:
y_query = swim_catNB.predict(X_query)
y_query
Out[77]:
array(['Yes', 'No', 'Yes'], dtype='<U3')
In [78]:
q_probs = swim_catNB.predict_proba(X_query)
q_probs
Out[78]:
array([[0.228592  , 0.771408  ],
       [0.81632203, 0.18367797],
       [0.22858759, 0.77141241]])
```

Predictions

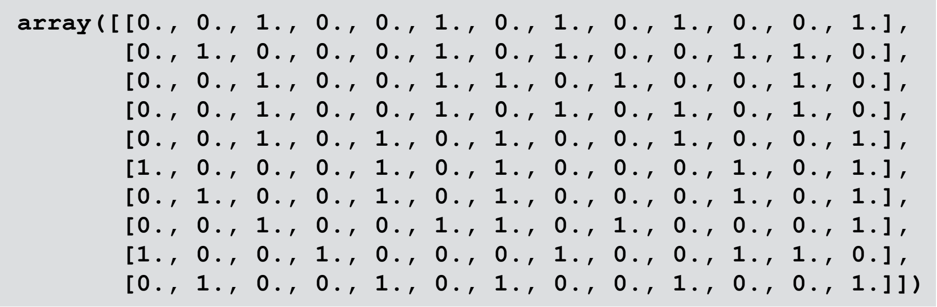
Probabilities

# One-Hot Encoding

- One-Hot encode the other data

```
onehot_encoder = OneHotEncoder(sparse=False)
swimOH = onehot_encoder.fit_transform(swim)
swimOH
```

  - □ swimOH is a numpy array



```
array([[0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1.],
       [0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0.],
       [0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0., 1.],
       [1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1.],
       [0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1.],
       [0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1.],
       [1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0.],
       [0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 1.]])
```

- Train and test on training data

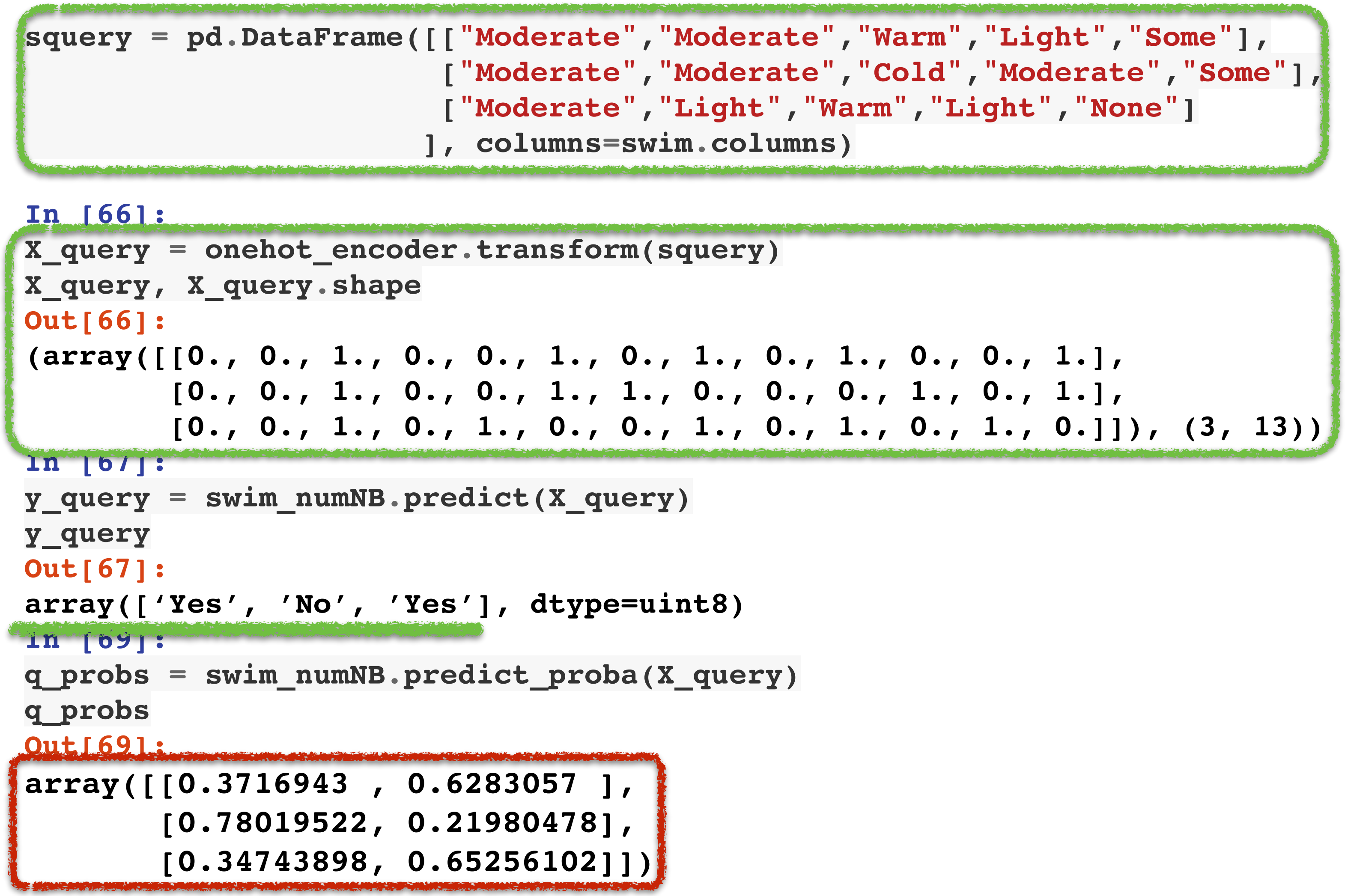
```
mnb = MultinomialNB()
swim_numNB = mnb.fit(swimOH,y)
y_dash = swim_numNB.predict(swimOH)

confusion = confusion_matrix(y, y_dash)
print("Confusion matrix:\n{}".format(confusion))
```

```
Confusion matrix:
[[6 0]
 [1 3]]
```

# Test with other data

- Set up a dataframe and then One-Hot



Query Dataframe

```
squery = pd.DataFrame([["Moderate","Moderate","Warm","Light","Some"],
                       ["Moderate","Moderate","Cold","Moderate","Some"],
                       ["Moderate","Light","Warm","Light","None"]
                      ], columns=swim.columns)
```

OneHot Format

```
In [66]:
X_query = onehot_encoder.transform(squery)
X_query, X_query.shape
Out[66]:
(array([[0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1.],
        [0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1.],
        [0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0.]]), (3, 13))
```

Predictions

```
In [67]:
y_query = swim_numNB.predict(X_query)
y_query
Out[67]:
array([‘Yes’, ’No’, ’Yes’], dtype=uint8)
```

Probabilities

```
In [69]:
q_probs = swim_numNB.predict_proba(X_query)
q_probs
Out[69]:
array([[0.3716943 , 0.6283057 ],
       [0.78019522, 0.21980478],
       [0.34743898, 0.65256102]])
```