# Lecture 2

```
>> load fisheriris
>> whos meas
  Name        Size            Bytes  Class     Attributes

  meas      150x4             4800  double

>> whos species
  Name          Size            Bytes  Class    Attributes

  species     150x1            18100  cell

>> species(1)

ans =

  1×1 cell array

    {'setosa'}

>> meas(1)

ans =

    5.1000

>> meas(1,:)

ans =

    5.1000    3.5000    1.4000    0.2000

>> Mdl = fitcnb (X,Y);
>> Mdl

Mdl =

  ClassificationNaiveBayes
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: {1×3 cell}
           ScoreTransform: 'none'
          NumObservations: 150
        DistributionNames: {1×4 cell}
    DistributionParameters: {3×4 cell}


  Properties, Methods

>> Mdl.ClassNames

>> species(51)

ans =

  1×1 cell array

    {'versicolor'}

>> species(100)

ans =

  1×1 cell array

    {'versicolor'}

>> species(101)

ans =

  1×1 cell array

    {'virginica'}
```

```
ans =

  3×1 cell array

    {'setosa'    }
    {'versicolor'}
    {'virginica' }
```

```
>> resubLoss(Mdl)

ans =

    0.0400
```

# resubLoss

Resubstitution classification loss for naive Bayes classifier

- predict - Classify observations using naive Bayes classifier

```
>> label = predict(Mdl, X);
>> label(1:5)

ans =

  5×1 cell array

    {'setosa'}
    {'setosa'}
    {'setosa'}
    {'setosa'}
    {'setosa'}

>> label(51:55)

ans =

  5×1 cell array

    {'versicolor'}
    {'versicolor'}
    {'virginica' }
    {'versicolor'}
    {'versicolor'}

>> Y(51:55)

ans =

  5×1 cell array

    {'versicolor'}
    {'versicolor'}
    {'versicolor'}
    {'versicolor'}
    {'versicolor'}
```

```
>> diff = strcmp(label, Y);
>> length( find(diff == 0) )

ans =

     6

>> 6/150

ans =

    0.0400
```

```
import numpy as np
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
from sklearn.datasets import load_iris
train_samples = load_iris()
X = train_samples.data
Y = train_samples.target
clf.fit(X, Y)
clf.score(X, Y)
# Verify the classification accuracy
Y_pred = clf.predict(X)
num_correct = np.sum(Y == Y_pred)
num_sample = np.size(Y_pred)
num_correct/num_sample
```