Bayes' Theorem

Example : Telecom Customers

- A telecom firm has many customers. Each customer either talks for the duration of more than 100 minutes or less than 100 minutes. The firm has launched a plan for the customers who talk more specially to optimize the amount spent by them on bills.
- Call Centre staff had been instructed to call some customers. In that operation, some customers bought the new plan and others didn't.
- In this case each customer is a record, and the response of interest, Y = {Bought ,Not Bought}, has two classes: C1 = Bought and C2 = Not Bought.

Conditional Probabilities

- A conditional probability of event A given event B [denoted by P(A|B)] represents the chances of event A occurring only under the scenario that event B occurs.
- In the response example, we may be interested in P(bought| Talk Time >=100, gender=Male), also P(bought| Talk Time >=100, gender=Female), as we have gender as additional feature of the customers

BAYES FORMULA

- The Bayes theorem gives us the following formula to compute the probability that the record belongs to class Ci:

$$P(C_i|X_1,\ldots,X_p) = \frac{P(X_1,\ldots,X_p|C_i)P(C_i)}{P(X_1,\ldots,X_p|C_1)P(C_1) + \cdots + P(X_1,\ldots,X_p|C_m)P(C_m)}.$$

Where

Ci : classes of interest

$X_1, X_2, \ldots X_p$: Variables which co-exist with Classes of interest

Example

Talks for more than 100 min? (TT >= 100)	Gender	Response
y	male	not bought
n	male	not bought
n	female	not bought
n	female	not bought
n	male	not bought
n	male	not bought
y	male	bought
y	female	bought
n	female	bought
y	female	bought

Bayes' Formula Calculations

$P(Buy|Male, TT \geq 100)$

$$= \frac{P(Male, TT{\geq}100\ |Buy)\ P(Buy)}{P(Male, TT{\geq}100\ |Buy)\ P(Buy) + P(Male, TT{\geq}100\ |\ Not\ Buy)\ P(\ Not\ Buy)}$$

$$= \frac{P(Male|Buy)P(TT \geq 100|Buy)\ P(Buy)}{P(Male|Buy)P(TT \geq 100|Buy)P(Buy) + P(Male|\ Not\ Buy)P(TT \geq 100|Not\ Buy)P(Not\ Buy)}$$

$$= \frac{\frac{1}{4} \times \frac{3}{4} \times \frac{4}{10}}{\frac{1}{4} \times \frac{3}{4} \times \frac{4}{10} + \frac{4}{6} \times \frac{1}{6} \times \frac{6}{10}}$$

= 0.529

(TT >= 100)	Gender	Response
y	male	not bought
n	male	not bought
n	female	not bought
n	female	not bought
n	male	not bought
n	male	not bought
y	male	bought
y	female	bought
n	female	bought
y	female	bought

Bayes Probabilities

- For the conditional probability of bought behaviors given (TT >= 100) = y, gender = male , the numerator is a multiplication of the proportion of (TT >= 100) = y instances among the bought customers, times the proportion of gender = male instances among the bought customers, times the proportion of bought customers: (3/4)(1/4)(4/10) = 0.075.
- To get the actual probabilities, we must also compute the numerator for the conditional probability of not bought given (TT >= 100) = y, gender = male : (1/6)(4/6)(6/10) = 0.067.
- The denominator is then the sum of these two conditional probabilities (0.075 + 0.067 = 0.14).

Bayes Probabilities

- The conditional probability of bought behaviors given (TT >= 100) = y, gender = male is therefore 0.075/0.14 = 0.53.
- Similarly,
  - P(bought | (TT >= 100) = y, gender = female) = 0.87,
  - P(bought | (TT >= 100) = n, gender = male) = 0.07,
  - P(bought | (TT >= 100) = n, gender = female) = 0.31.

Naïve Bayes Algorithm

Naïve Bayes

- Naïve Bayes is a classification algorithm
- There are two types of Naïve Bayes Algorithms:
  - Discrete Naïve Bayes: For categorical predictors
  - Kernel Naïve Bayes: For numerical predictors

Discrete Naive Bayes

- In this algorithm, the probability of a record belonging to a certain class is evaluated on the basis of conditional probability calculated using Bayes theorem
- Discrete Naive Bayes works only with predictors that are categorical.
- Numerical predictors must be binned and converted to categorical variables before the application Naive Bayes algorithm
- In Python, package sklearn.naive_bayes supports Descrete Naïve Bayes (MultinomialNB)

Kernel Naïve Bayes

- Kernel Naïve Bayes works with numeric predictors assuming some distribution of the predictors
- It can assume Normal Distribution (Gaussian Naïve Bayes) or any other distribution
- On assuming the distribution, the prior probabilities are calculated
- In Python, package sklearn.naive_bayes supports Gaussian (GaussianNB) Naïve Bayes (assumes Normality of predictors)

Example 1 : Telecom Customers (Discrete NB)

- A telecom firm has many customers. Each customer either talks for the duration of more than 100 minutes or less than 100 minutes. The firm has launched a plan for the customers who talk more specially to optimize the amount spent by them on bills. In this case each customer is a record, and the response of interest, Y = {Bought ,Not Bought}, has two classes that a company can be classified into: C1 = Bought and C2 = Not Bought.
- Apart from talk time we also have information about the gender of the customer

Data

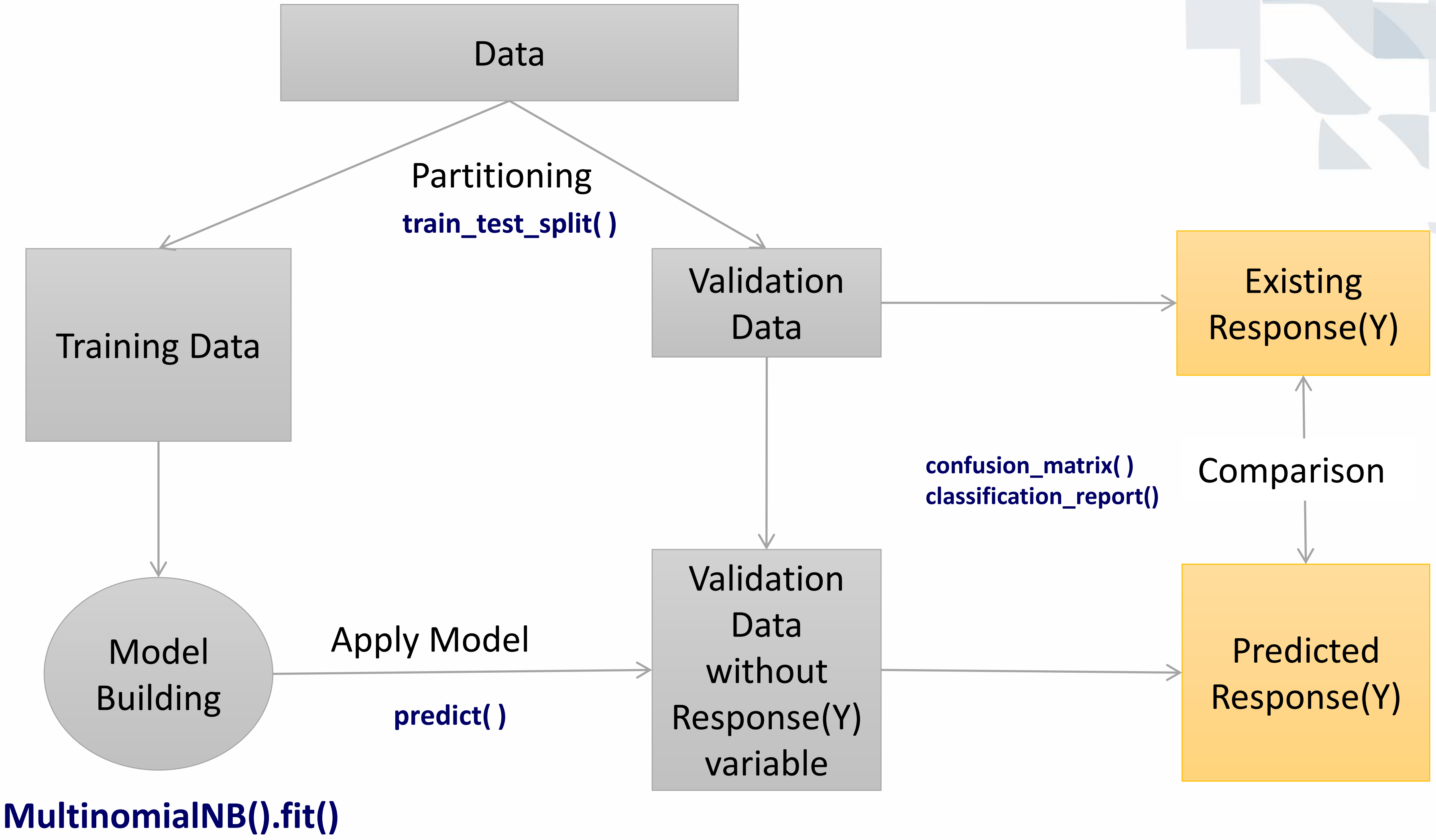
- 150 Observations , 3 variables
- First 8 observations:

telecom - DataFrame

Index	Gender	TT_gt_100	Response
0	F	Y	N
1	M	N	N
2	M	N	N
3	F	Y	Y
4	F	N	N
5	F	N	N
6	F	Y	Y
7	M	Y	Y
8	M	Y	N

Format Resize Background color Column min/max OK Cancel

Naïve Bayes Classifier



Program and Output

```
In [1]: import pandas as pd
   ...:
   ...: telecom = pd.read_csv("G:/Statistics (Python)/Cases/Telecom/Telecom.csv")
   ...:
   ...: dum_telecom = pd.get_dummies(telecom, drop_first=True)
   ...:
   ...: from sklearn.model_selection import train_test_split
   ...: from sklearn.metrics import confusion_matrix
   ...: from sklearn.metrics import classification_report, accuracy_score
   ...: from sklearn.naive_bayes import MultinomialNB
   ...:
   ...: X = dum_telecom.iloc[:,0:2]
   ...: y = dum_telecom.iloc[:,2]
   ...:
   ...: # Create training and test sets
   ...: X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.3,
   ...:                                                     random_state=42,
   ...:                                                     stratify=y)
   ...:
   ...: multinomial = MultinomialNB()
   ...: multinomial.fit(X_train, y_train) # Model Building
   ...:
   ...: y_probs = multinomial.predict_proba(X_test)
   ...: y_pred = multinomial.predict(X_test) # Applying built on test data
   ...:
   ...: print(confusion_matrix(y_test, y_pred))
[[18  4]
 [ 2 21]]
```

Evaluation

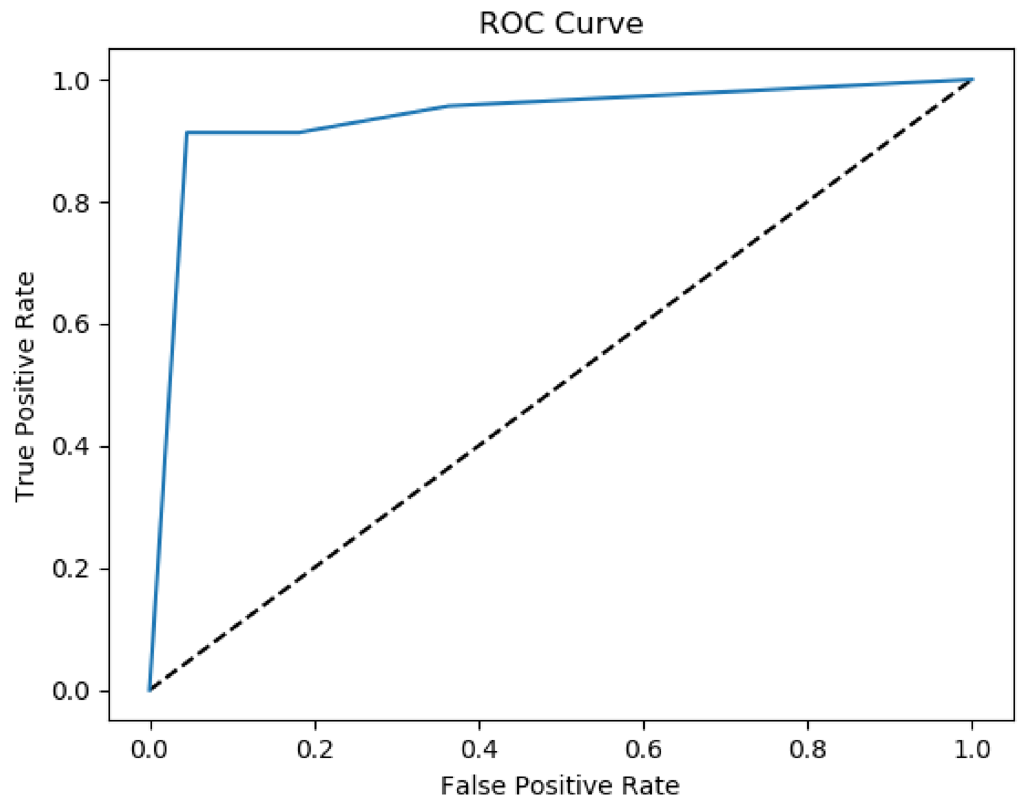
```
In [2]: print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

           0       0.90      0.82      0.86        22
           1       0.84      0.91      0.87        23

    accuracy                           0.87        45
   macro avg       0.87      0.87      0.87        45
weighted avg       0.87      0.87      0.87        45
```

```
In [3]: print(accuracy_score(y_test, y_pred))
0.8666666666666667
```

```
In [5]: roc_auc_score(y_test, y_pred_prob)
Out[5]: 0.9377470355731224
```



Example 2: Predicting Defaulters (Gaussian NB)

- Data Set Details:
  - **default** : A categorical variable with levels No and Yes indicating whether the customer defaulted on their debt
  - **student** : A categorical variable with levels No and Yes indicating whether the customer is a student
  - **balance** : The average balance that the customer has remaining on their credit card after making their monthly payment (Numeric Variable)
  - **income** : Income of customer (Numeric Variable)

- Source: http://www-bcf.usc.edu/~gareth/ISL/

Sane's STATS Academy of Statistics

Data

- 4 variables and 10,000 observations

Default - DataFrame

Index	default	student	balance	income
0	No	No	729.526	44361.6
1	No	Yes	817.18	12106.1
2	No	No	1073.55	31767.1
3	No	No	529.251	35704.5
4	No	No	785.656	38463.5
5	No	Yes	919.589	7491.56
6	No	No	825.513	24905.2
7	No	Yes	808.668	17600.5
8	No	No	1161.06	37468.5
9	No	No	0	29275.3
10	No	Yes	0	21871.1
11	No	Yes	1220.58	13268.6
12	No	No	237.045	28251.7
13	No	No	606.742	44994.6

Program and Output

```
In [10]: import pandas as pd
    ...:
    ...: Default = pd.read_csv("F:/Python Material/ML with Python/Datasets/Default.csv")
    ...: dum_Default = pd.get_dummies(Default, drop_first=True)
    ...:
    ...: from sklearn.model_selection import train_test_split
    ...: from sklearn.metrics import confusion_matrix, classification_report
    ...: from sklearn.naive_bayes import GaussianNB
    ...:
    ...: X = dum_Default.iloc[:,[0,1,3]]
    ...: y = dum_Default.iloc[:,2]

In [11]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state=42)
    ...:
    ...: gaussian = GaussianNB()
    ...: y_pred = gaussian.fit(X_train, y_train).predict(X_test)
    ...:
    ...: print(confusion_matrix(y_test, y_pred))
    ...: print(classification_report(y_test, y_pred))
[[3840   25]
 [ 102   33]]
             precision    recall  f1-score   support

          0       0.97      0.99      0.98      3865
          1       0.57      0.24      0.34       135

avg / total       0.96      0.97      0.96      4000
```

Questions?