Classifiers on scikit-learn

Big Data Mining Lab.

Naïve Bayes

- "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) **independence assumptions between the features**.
- Given feature vector $x_1, \ldots, x_n$, the probability to be class y is,

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)}$$

$$= \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

Naïve Bayes (cont.)

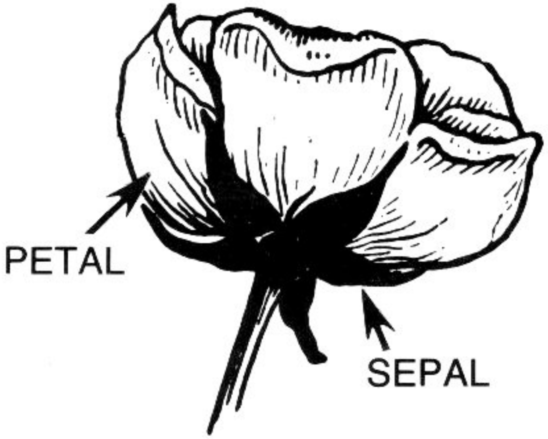
- From given data, $P(x_1, \ldots, x_n)$ is constant

$$P(y|x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

Iris Classification

- Given: features of iris
- Goal: classifying iris



- Data description
  - # of records : 150
  - iris = datasets.load_iris()
  - iris.data : attributes (4-dimension),
  - iris.target : class (1-dimension)

Gaussian Naïve Bayes

- Is based on assumption that the continuous values associated with each class follow to a Gaussian distribution.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right)$$

GNB – example code

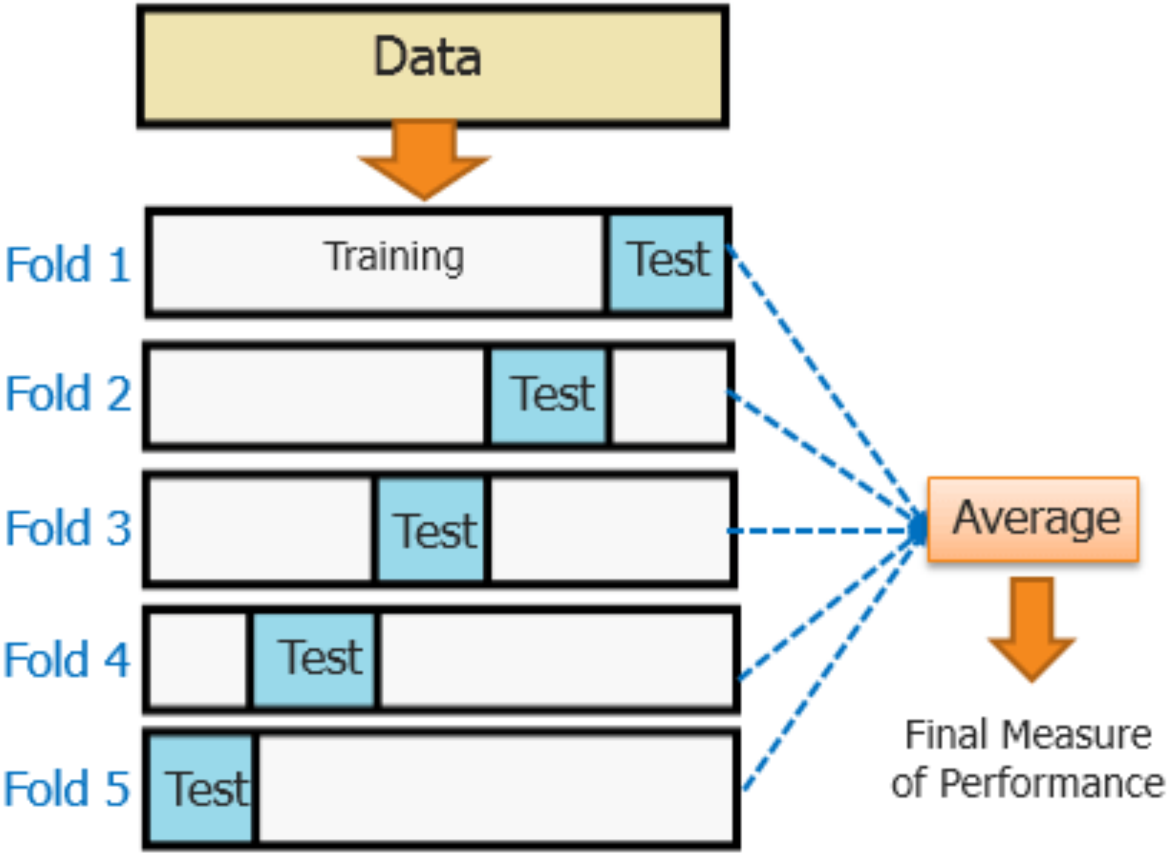
```python
from sklearn import datasets
iris = datasets.load_iris()
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
print("Number of mislabeled points out of a total %d points : %d" % (iris.data.shape[0],(iris.target != y_pred).sum()))
```

Lab. - 1

- Try the example code.

Cross Validation

- Is a resampling procedure to evaluate models on a limited sample.



Cross Validation – example code

```python
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=0)
y_pred = gnb.fit(X_train,y_train).predict(X_test)
print ("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
```

Lab. - 2

- Try the example code.

Multinomial Naïve Bayes

- Is suitable with discrete features (e.g., word counts).
- Normally requires integer feature counts, but in practice, it works with real numbers (e.g., TF-IDF).

Multinomial Naïve Bayes – example code

```
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=0)
y_pred = mnb.fit(X_train,y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
```

Lab. - 3

- Try the example code.

Decision Tree

```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=0)
y_pred = clf.fit(X_train,y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
```

Lab. - 4

- Try the example code.

Read CSV using pandas

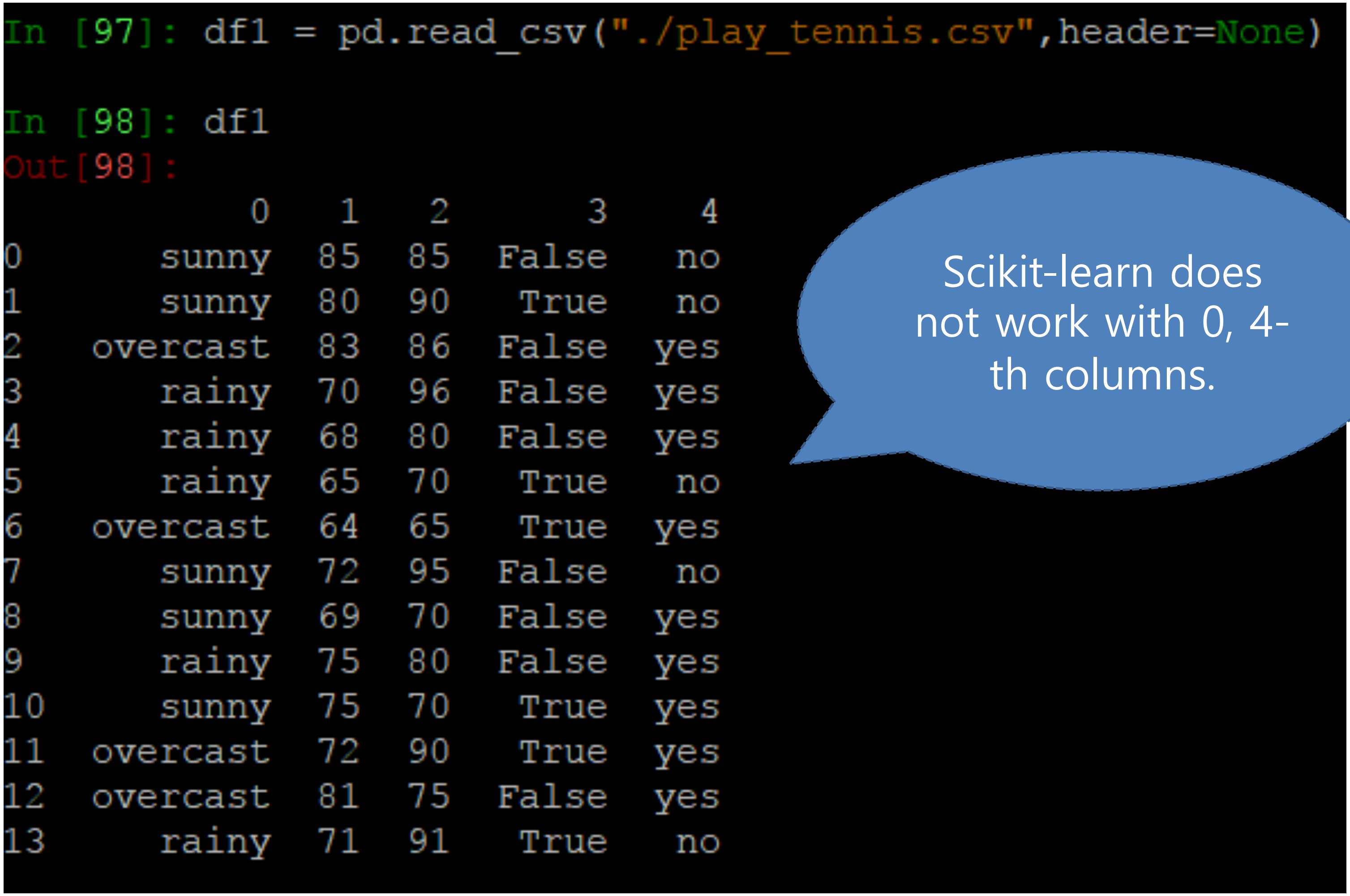
- import pandas as pd

- Read CSV into dataframe with header
- df1 = pd.read_csv("./file")

- Read CSV into Dataframe without header
- df1 = pd.read_csv("./play_tennis.csv",header=None)

Data Preprocess

- Encode categorical data into integer



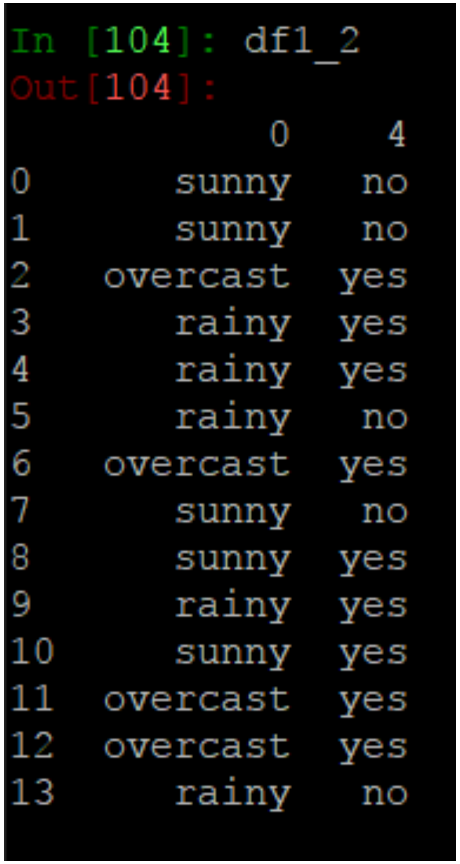
```
In [97]: df1 = pd.read_csv("./play_tennis.csv",header=None)

In [98]: df1
Out[98]:
           0   1   2      3    4
0      sunny  85  85  False   no
1      sunny  80  90   True   no
2   overcast  83  86  False  yes
3      rainy  70  96  False  yes
4      rainy  68  80  False  yes
5      rainy  65  70   True   no
6   overcast  64  65   True  yes
7      sunny  72  95  False   no
8      sunny  69  70  False  yes
9      rainy  75  80  False  yes
10     sunny  75  70   True  yes
11  overcast  72  90   True  yes
12  overcast  81  75  False  yes
13     rainy  71  91   True   no
```

Scikit-learn does not work with 0, 4-th columns.

Data Preprocess (cont.)

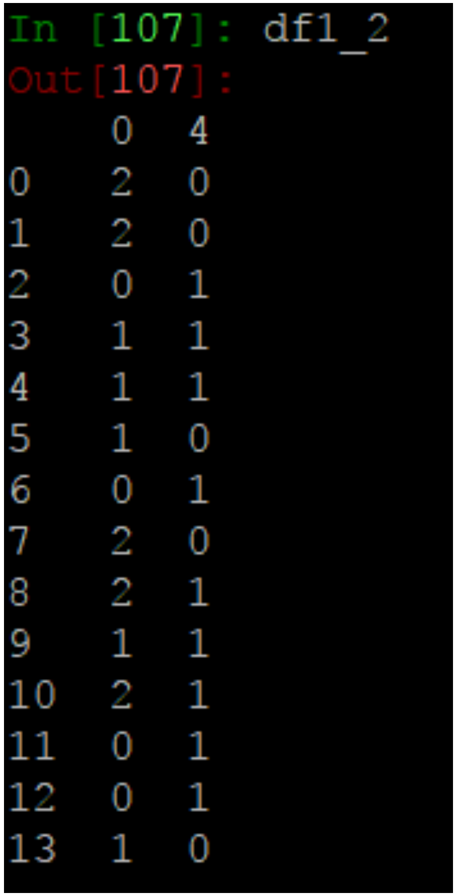
```
from sklearn import preprocessing
df1_2 = df1.select_dtypes(include=[object])
```



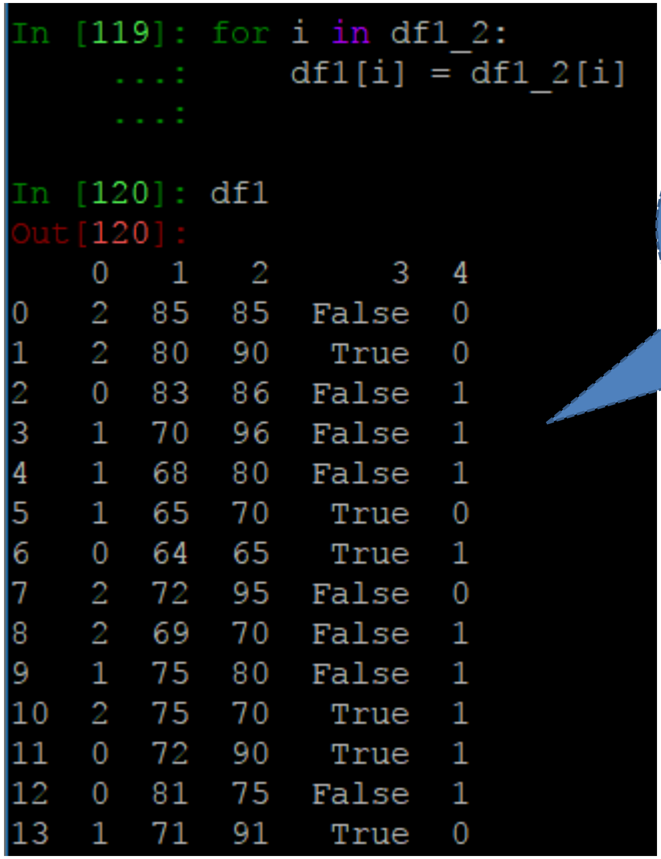
```
In [104]: df1_2
Out[104]:
           0    4
0      sunny   no
1      sunny   no
2   overcast  yes
3      rainy  yes
4      rainy  yes
5      rainy   no
6   overcast  yes
7      sunny   no
8      sunny  yes
9      rainy  yes
10     sunny  yes
11  overcast  yes
12  overcast  yes
13     rainy   no
```

Data Preprocess (cont.)

```
le = preprocessing.LabelEncoder()
df1_2 = df1_2.apply(le.fit_transform)
```



```
In [107]: df1_2
Out[107]:
    0  4
0   2  0
1   2  0
2   0  1
3   1  1
4   1  1
5   1  0
6   0  1
7   2  0
8   2  1
9   1  1
10  2  1
11  0  1
12  0  1
13  1  0
```



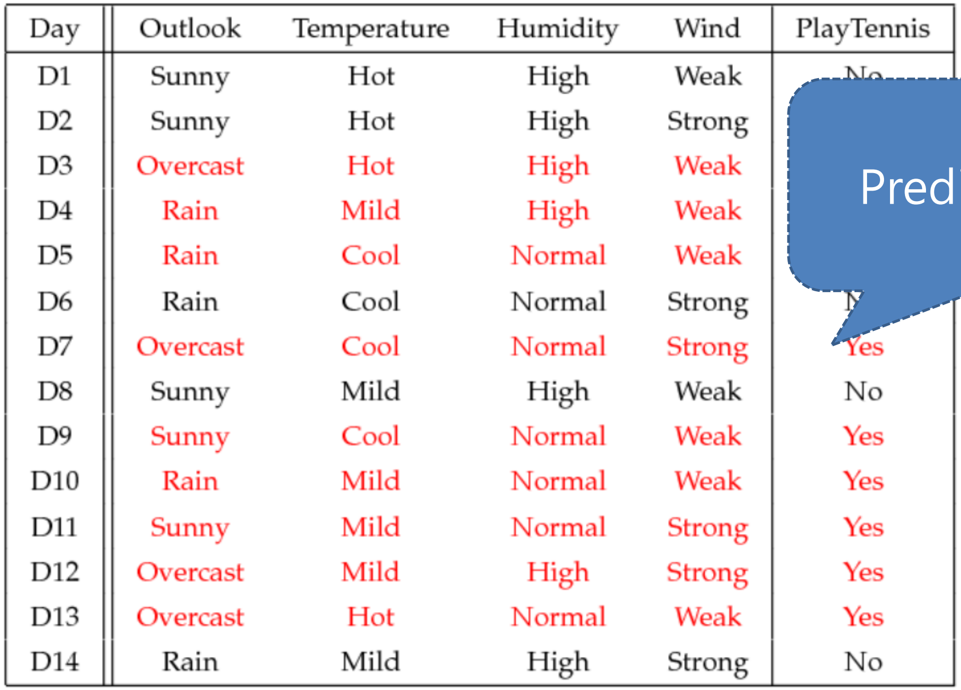
```
In [119]: for i in df1_2:
     ...:     df1[i] = df1_2[i]
     ...:

In [120]: df1
Out[120]:
    0   1   2      3  4
0   2  85  85  False  0
1   2  80  90   True  0
2   0  83  86  False  1
3   1  70  96  False  1
4   1  68  80  False  1
5   1  65  70   True  0
6   0  64  65   True  1
7   2  72  95  False  0
8   2  69  70  False  1
9   1  75  80  False  1
10  2  75  70   True  1
11  0  72  90   True  1
12  0  81  75  False  1
13  1  71  91   True  0
```

Encoded!

Lab. - 5

- Apply classifier on play_tennis.csv, play_tennis_test.csv and obtain results.



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	[illegible]
D3	Overcast	Hot	High	Weak	[illegible]
D4	Rain	Mild	High	Weak	[illegible]
D5	Rain	Cool	Normal	Weak	[illegible]
D6	Rain	Cool	Normal	Strong	N
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Predict classes