Chapter 6 - Other Popular Machine Learning Methods

Segment 5 - Naive Bayes Classifiers

```
import numpy as np
import pandas as pd
import urllib
import sklearn

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

Naive Bayes

Using Naive Bayes to predict spam

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data"

import urllib.request

raw_data = urllib.request.urlopen(url)
dataset = np.loadtxt(raw_data, delimiter=',')
print(dataset[0])
```

```
[  0.      0.64    0.64    0.      0.32    0.      0.      0.      0.
   0.      0.      0.64    0.      0.      0.      0.32    0.      1.29
   1.93    0.      0.96    0.      0.      0.      0.      0.      0.
   0.      0.      0.      0.      0.      0.      0.      0.      0.
   0.      0.      0.      0.      0.      0.      0.      0.      0.
```

```
        0.      0.      0.      0.      0.      0.      0.778   0.      0.
        3.756  61.    278.      1.    ]
```

```
X = dataset[:,0:48]

y = dataset[:,-1]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=17)


BernNB = BernoulliNB(binarize=True)
BernNB.fit(X_train, y_train)
print(BernNB)

y_expect = y_test
y_pred = BernNB.predict(X_test)

print(accuracy_score(y_expect, y_pred))
```

```
     BernoulliNB(alpha=1.0, binarize=True, class_prior=None, fit_prior=True)
     0.8577633007600435
```

```
MultiNB = MultinomialNB()
MultiNB.fit(X_train, y_train)
print(MultiNB)


y_pred = MultiNB.predict(X_test)

print(accuracy_score(y_expect, y_pred))
```

```
     MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
     0.8816503800217155
```

```
GausNB = GaussianNB()
GausNB.fit(X_train, y_train)
print(GausNB)


y_pred = GausNB.predict(X_test)

print(accuracy_score(y_expect, y_pred))
```

```
    GaussianNB(priors=None, var_smoothing=1e-09)
    0.8197611292073833
```

```
BernNB = BernoulliNB(binarize=0.1)
BernNB.fit(X_train, y_train)
print(BernNB)

y_expect = y_test
y_pred = BernNB.predict(X_test)

print(accuracy_score(y_expect, y_pred))
```

```
    BernoulliNB(alpha=1.0, binarize=0.1, class_prior=None, fit_prior=True)
    0.9109663409337676
```