UNIVERSITY OF PORTSMOUTH

Python for Data Analysis

Modeling in Python – Bayesian and SVM (TB2 - Week 3)

Atefeh Khazaei

atefeh.khazaei@port.ac.uk

What we will learn this week?

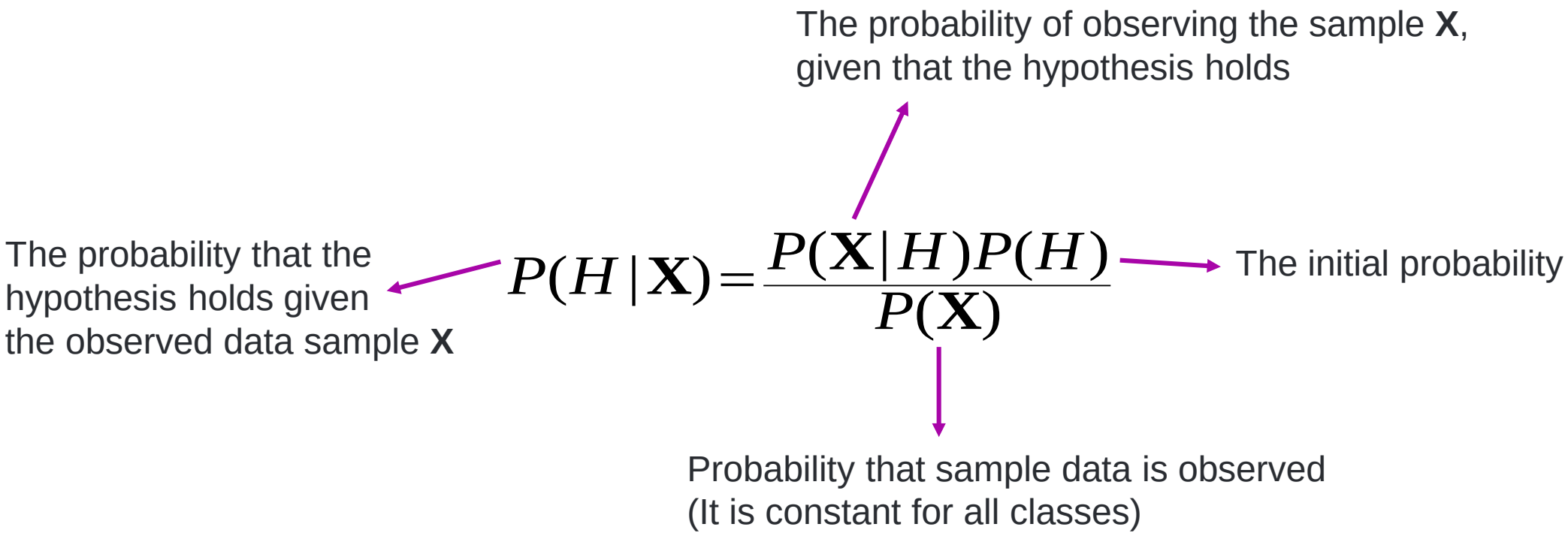
- Previous weeks modelling algorithms
  - K-Nearest Neighbours
  - Decision Tree
  - Random Forest
- This week modelling algorithms
  - Bayesian methods (Naïve Bayes)
  - Support Vector Machines (SVM)

Naïve Bayes

- **A probabilistic method**
  - Predicts class membership probabilities
- **Naïve Bayes classifier**
  - Foundation: Based on Bayes' theorem
  - Performance: Acceptable accuracy and speed when applied to large databases

Naïve Bayes (cont.)

- Bayes' Theorem:



Naïve Bayes (cont.)
Simple Example: One Feature, One Target

	Basketball	Cereal
1	Yes	No
2	Yes	Yes
3	Yes	No
4	No	Yes
5	No	Yes
6	No	No
…	…	…
5000	Yes	Yes

C \ B	YES	NO	
YES	2000	1750	3750
NO	1000	250	1250
	3000	2000	5000

Naïve Bayes (cont.)
Simple Example: One Feature, One Target

C \ B	YES	NO	
YES	2000	1750	3750
NO	1000	250	1250
	3000	2000	5000

Does someone who plays BASKETBALL (X, sample) eats CEREAL (H, Target)?

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

$$P(Cereal|Basketbal) = \frac{P(Basketbal|Cereal)\,P(Cereal)}{P(Basketbal)} = \frac{\frac{2000}{3750}\,\frac{3750}{5000}}{\alpha} = \frac{0.4}{\alpha}$$

$$P(\overline{Cereal}|Basketbal) = 1 - P(Cereal|Basketbal) = \frac{P(Basketbal|\overline{Cereal})\,P(\overline{Cereal})}{P(Basketbal)} = \frac{\frac{1000}{1250}\,\frac{1250}{5000}}{\alpha} = \frac{0.2}{\alpha}$$

sklearn.naive_bayes.GaussianNB()

- You can find more details in the following link:
  - https://scikit-learn.org/stable/modules/naive_bayes.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- About different types of NB:
  - GaussianNB,
  - MultinomialNB,
  - ComplementNB,
  - BernoulliNB,
  - CategoricalNB

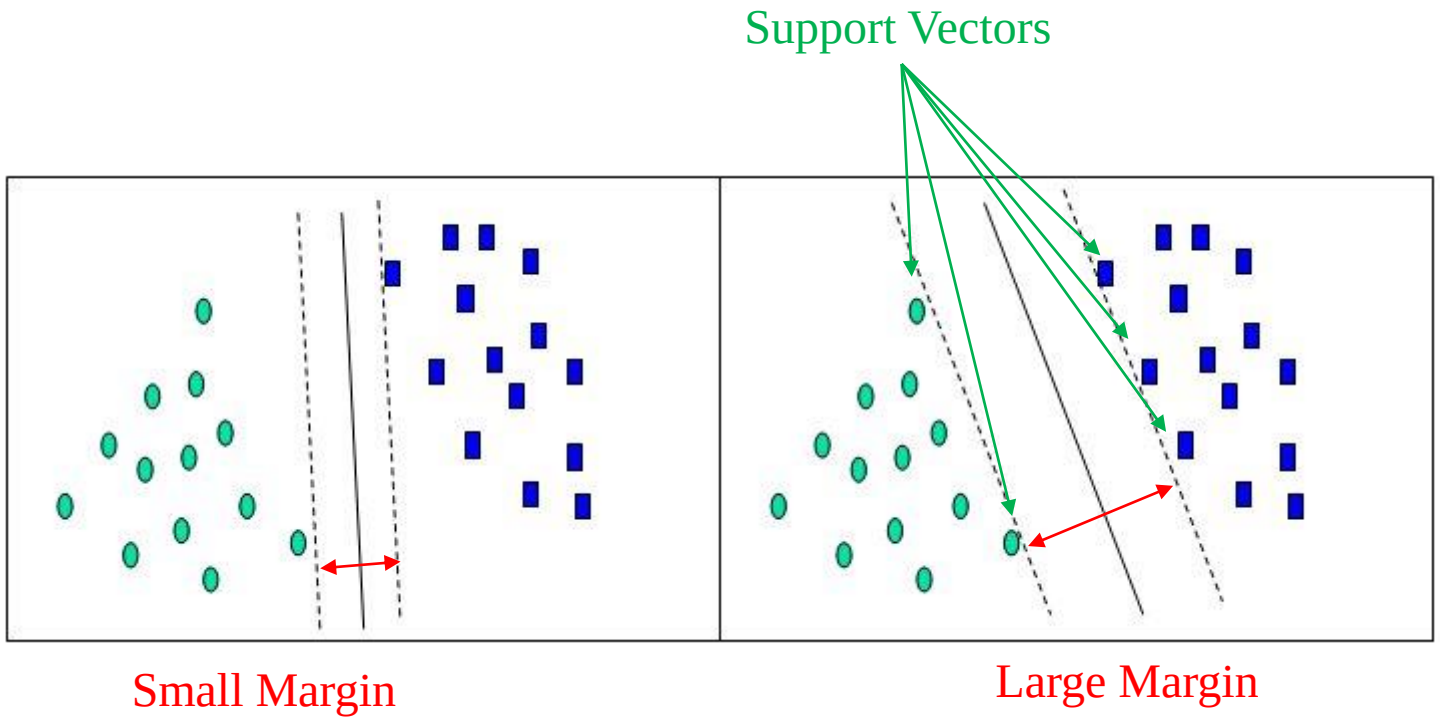
sklearn.naive_bayes.GaussianNB()

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.model_selection import train_test_split

# Training data features, skip the first column 'Survived'
train_features = train_data[:, 1:]

# 'Survived' column values
train_target = train_data[:, 0]

# Split 80-20 train vs test data
train_x, test_x, train_y, test_y = train_test_split(train_features,
                                                    train_target,
                                                    test_size=0.20,
                                                    random_state=0)


clf = GaussianNB()
clf = clf.fit(train_x, train_y)
predict_y = clf.predict(test_x)

from sklearn.metrics import accuracy_score
print ("Accuracy = %.2f" % (accuracy_score(test_y, predict_y)))
```

```
Accuracy = 0.81
```

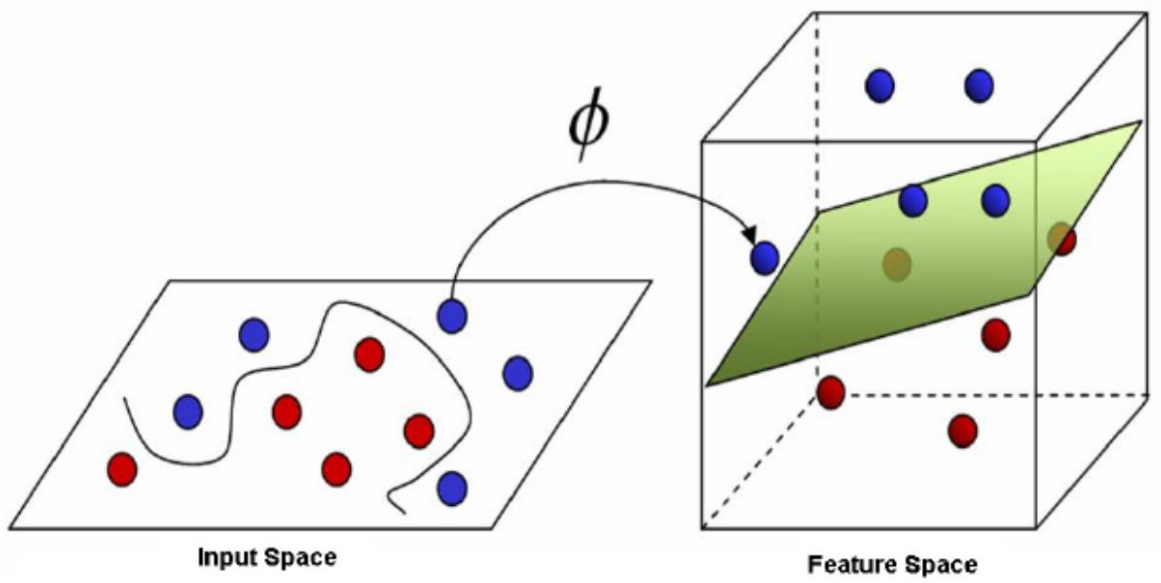
Support Vector Machines (SVM)

- Support Vector Machine constructs a hyperplane in a high-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.



Support Vector Machines (SVM) (Cont.)

- With an appropriate nonlinear mapping to a sufficiently high dimension (**kernel** trick), data from two classes can always be separated by a hyper plane.
- With the new dimension, SVM searches for the optimal linear decision boundary.



sklearn.svm.SVC()

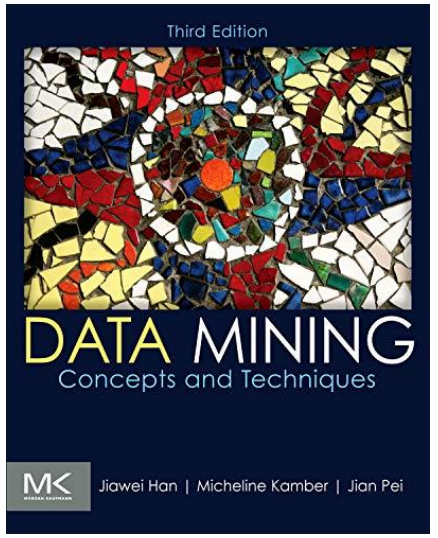
- You can find more details in the following link:
  - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- About different parameters of SVM:
  - kernel (linear, poly, rbf, sigmoid, precomputed)
  - degree
  - gamma
  - …

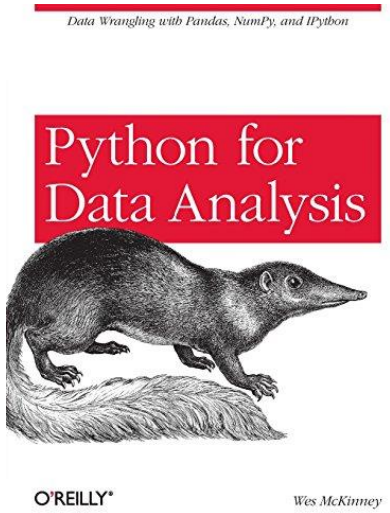
sklearn.svm.SVC()

```
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.model_selection import train_test_split

# Training data features, skip the first column 'Survived'
train_features = train_data[:, 1:]

# 'Survived' column values
train_target = train_data[:, 0]

# Split 80-20 train vs test data
train_x, test_x, train_y, test_y = train_test_split(train_features,
                                                    train_target,
                                                    test_size=0.20,
                                                    random_state=0)


clf = SVC(kernel='linear')
clf = clf.fit(train_x, train_y)
predict_y = clf.predict(test_x)

from sklearn.metrics import accuracy_score
print ("Accuracy = %.2f" % (accuracy_score(test_y, predict_y)))
```

References & More Resources

- References:
  - McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and Ipython*, O'Reilly Media, Inc., 2012.
  - Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.





Practical Session

- Revise the Titanic Case Study (Last session of TB1) and build some **Naïve Bayesian** and **SVM** models for Titanic. Try **different parameters** for these models and **compare them together**.