Lecture “Advanced Data Analytics”

Problem Set 3

Simon Scheidegger

Exercise 1

k-nearest neighbors

Below, you are given a data set on shop customers. The feature “Channel” indicates for ever customer which is the best way to provide him advertisements of new products.

Age	m/f	Sales	Channel
20	f	10	E-mail
30	m	90	Phone
40	m	70	Post
60	f	100	Phone
20	m	30	E-mail
30	f	40	E-mail
70	m	80	Post
20	f	110	Phone
60	m	80	Post
40	f	20	E-mail

a) Encode the non-numerical features such that they can be used for classification, and introduce a normalization for the data values.

b) Predict with k-nearest neighbors over which “channel” a new customer (list below) should be targeted. Use k = 3, and the Euclidean Distance as metric.
Code the algorithm up yourself, i.e., do not use any off-the-shelve code.

Age	m/f	Sales	Channel
30	m	50	?
90	f	100	?

Exercise 2

Naïve Bayes (In this entire exercise, code the algorithm up yourself.)

You are provided with the following documents d1 to d6 from different categories out of news a news paper.

Document	Content	Category
d1	ball goal cart goal	Sports
d2	theater cart drama	Culture
d3	drama strategy decision drama	Politics
d4	theater ball	Culture
d5	ball goal player strategy	Sports
d6	theater cart opera	Culture

(a) Represent the documents as bag of words in a Table, as done in the lecture.
(b) Compute the Class probabilities P[c] as well as the probabilities of the words given the class---that is to say, P[w|c], as introduced in the lecture.
(c) Based on the training set, classify the documents given below:

Document	Content	Category
d7	ball player strategy	?
d8	theater cart decision	?

Exercise 3

Naïve Bayes – practice with sklearn

a) One place where multinomial naive Bayes is often used is in text classification, where the features are related to word counts or frequencies within the documents to be classified. We discussed the extraction of such features from text in Feature Engineering; here we will use the sparse word count features from the 20 Newsgroups corpus to show how we might classify these short documents into categories.

Import and inspect the this dataset in sklearn by typing

```
from sklearn.datasets import fetch_20newsgroups
data = fetch_20newsgroups()
data.target_names
```

a) Select just a few of these categories, and download the training and testing set:

```
categories = ['talk.religion.misc', 'soc.religion.christian', 'sci.space', 'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories)
```

b) Look at the representative entries from the data, e.g., by typing

```
print(train.data[5])
```

c) In order to use this data for machine learning, we need to be able to convert the content of each string into a vector of numbers. For this we will use the TF-IDF vectorizer, and create a pipeline that attaches it to a multinomial naive Bayes classifier:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

d) With this pipeline, apply the model to the training data, and predict labels for the test data.

e) Now that we have predicted the labels for the test data, we can evaluate them to learn about the performance of the estimator.
Thus, produce the confusion matrix between the true and predicted labels for the test data.

f) Evidently, even this very simple classifier can successfully separate space talk from computer talk, but it gets confused between talk about religion and talk about Christianity. This is perhaps an expected area of confusion!

The very cool thing here is that we now have the tools to determine the category for any string, using the predict() method of this pipeline.

Here's a quick utility function that will return the prediction for a single string:

```
def predict_category(s, train=train, model=model):
    pred = model.predict([s])
    return train.target_names[pred[0]]
```

Use this utility to return you a prediction for the string "***determining the screen resolution***".

Exercise 4

Decision Trees
You are provided a dataset movies. The feature "roles" indicates how many different characters occur during the film. The feature "Audio-book" indicates whether there exists an Audio-book accompanying the movie

Roles	Duration [min]	Audiobook	Genre
5	80	no	Action
15	120	yes	Drama
15	100	yes	Action
20	80	no	Drama
5	80	no	Action

(a) Create a decision tree to classify with respect to. Genre. To do so, use "information gain" as criterion for generating the tree.
(b) Classify the subsequent two movies based on this tree:

Roles	Duration [min]	Audiobook	Genre
12	115	yes	?
2	180	yes	?

Do not use an existing library.

Exercise 5

Classification with Sklearn:

You are given the data set

x1 = [1, 5, 1.5, 8, 1, 9]
x2 = [2, 8, 1.8, 8, 0.6, 11]

a) Generate a scatter plot of this data.
How many “clusters” of data can you see?

Now, of course, we can see with our own eyes how these groups should be divided, though exactly where we might draw the dividing line might be debated. To find this line, use the following two classification methods from sklearn.

b) Perceptron
c) kNN

To do so, assume that you are given the following training targets:

y = [0,1,0,1,0,1]

d) make a prediction for the point [0.58,0.76] for the two classifiers.