Übungsblatt 5

Abgabe:

bis **28. Januar 2021** um **23:59** via **ecampus**

Aufgabe 5.1: Die *python* Bibliothek *scikit-learn* enthält einige klassische *toy data sets*, die oft zum (ersten) Benchmarking von *machine learning* Algorithmen verwendet werden.

Beim *wine classification* Problem wird versucht, anhand 13-dimensionaler Merkmalsvektoren zu erkennen, um welche von 3 Weinsorten es sich handelt. Dazu stellt das *wine data set* 178 Beobachtungen und entsprechende Klassenlabel bereit. Um diese in eine Designmatrix $\boldsymbol{X} \in \mathbb{R}^{178\times 13}$ und einen Labelvektor $\boldsymbol{y} \in \{0,1,2\}^{178}$ einzulesen, können Sie so vorgehen

```
import numpy as np
from sklearn.datasets import load_wine

wine = load_wine()
matX = wine['data']
vecY = wine['target']
```

In der Vorlesung hatten wir bereits gesehen, dass es *scikit-learn* auch bequem macht, Daten in Trainings- und Testdaten aufzusplitten. Hier können Sie z.B. so vorgehen

```
from sklearn.model_selection import train_test_split

Xtrn, Xtst, ytrn, ytst = train_test_split(matX, vecY, test_size=0.3)
```

Auch über die Möglichkeiten, die *scikit-learn* für die Arbeit mit Entscheidungsbäumen bietet, hatten wir in der Vorlesung bereits gesprochen. Um einen Entscheidungsbaum auf Ihren Traininigsdaten zu trainieren und auf Ihren Testdaten laufen zu lassen bzw. zu evaluieren, können Sie folgendermaßen vorgehen

```
from sklearn.tree import DecisionTreeClassifier

dTree = DecisionTreeClassifier(max_depth=3, criterion='gini')
dTree.fit(Xtrn, ytrn)

pred = dTree.predict(Xtst)
accu = dTree.score(Xtst, ytst)
```

Nutzen Sie dieses *recipe*, um die durchschnittliche *accuracy* in $n = 10$ Trainings-/Testläufen zu berechnen.

Offensichtlich hat die Methode `DecisionTreeClassifier` eine Reihe von Parametern. Lesen Sie die *man pages* auf scikit-learn.org, um sich mit diesen Parametern vertraut zu machen. Untersuchen Sie dann, wie sich unterschiedliche Parametrisierungen Ihres Entscheidungsbaums auf die durchschnittliche *accuracy* auswirken.

Aufgabe 5.2: Entwerfen Sie ein konzeptuelles Datenmodell (ein Entity-Relationship Diagramm) für eine Videosharing Plattform. In dieser *mini-world* gibt es *user*, die *username* und *password hash* haben, sowie *videos*, die einen *title* haben. *User* können *videos* hochladen, anschauen und kommentieren. *Comments* können *replies* auf andere *comments* sein. *Uploads*, *views* und *comments* haben Zeitstempel (*dates*).

Um eine Grafik Ihres Modells zu erstellen, können Sie z.B. das freie online tool draw.io benutzen, das es u.a. erlaubt, ER Diagramme als XML oder PDF Dateien auf Ihrer Festplatte zu speichern.

Aufgabe 5.3: Übertragen Sie Ihr Datenmodell in ein relationales Datenbankschema. Das heißt, überlegen Sie, welche Tabellen, Attribute, Primär- und Fremdschüssel Sie benötigen, um Ihr Modell als relationale Datenbank zu implementieren.

Aufgabe 5.4: Nutzen Sie

```
import sqlite3
```

um Ihre Datenbank als (eine in-memory) SQLite Datenbank in *python* zu implementieren. Erzeugen Sie die entsprechenden Tabellen und Attribute und tragen Sie all diese Fakten in Ihre Datenbank ein:

Folgende *user* haben sich registriert

dingbat	am	2021-01-04	mit pw-hash	#FF12A9
brangdag	am	2021-01-04	mit pw-hash	#614D3E
gigecon	am	2021-01-05	mit pw-hash	#899FF2
nico234	am	2021-01-06	mit pw-hash	#003BC1
senftorte	am	2021-01-06	mit pw-hash	#113B81
honeypot	am	2021-01-06	mit pw-hash	#343321
zoomfish	am	2021-01-06	mit pw-hash	#17EECD

Folgende *videos* wurden hochgeladen

my cat watches videos	am	2021-01-04	von	dingbat
in soviet russia videos watch your cat	am	2021-01-05	von	gigecon
introduction to machine learning (1)	am	2021-01-06	von	senftorte
introduction to machine learning (2)	am	2021-01-06	von	senftorte
honigkuchen backen	am	2021-01-06	von	honeypot

Folgende *videos* wurden angeschaut

```
my cat watches videos                    am  2021-01-05  von  brangdag
my cat watches videos                    am  2021-01-05  von  brangdag
my cat watches videos                    am  2021-01-05  von  brangdag
in soviet russia videos watch your cat   am  2021-01-06  von  brangdag
in soviet russia videos watch your cat   am  2021-01-06  von  dingbat
in soviet russia videos watch your cat   am  2021-01-06  von  honeypot
my cat watches videos                    am  2021-01-06  von  senftorte
honigkuchen backen                       am  2021-01-07  von  nico234
honigkuchen backen                       am  2021-01-07  von  dingbat
```

Folgende *comments* wurden abgegeben

```
LOL             am  2021-01-05  von  brangdag   zu  my cat watches videos
srsly? nobody?  am  2021-01-05  von  brangdag   zu  my cat watches videos
@brangdag nope  am  2021-01-06  von  nico234    zu  my cat watches videos
lecker!         am  2021-01-07  von  gigecon    zu  honigkuchen backen
find ich auch   am  2021-01-07  von  dingbat    zu  honigkuchen backen
```

Folgende *comments* waren *replies*

```
srsly? nobody?  von  brangdag  auf  LOL             von  brangdag
@brangdag nope  von  nico234   auf  srsly? nobody?  von  brangdag
```

Das Erscheinungsbild der obigen Textblöcke deutet darauf hin, dass sie automatisch durch Datenbankabfragen erzeugt wurden. Das ist in der Tat der Fall. Es ist also möglich, eine entsprechende Datenbank zu erstellen und zu benutzen ;-)

Aufgabe 5.5: Wenn Sie Ihre Datenbank mit angemessenen `UNIQUE` constraints konfiguriert haben, sollte es nicht möglich sein, einen weiteren *username* `senftorte` zu registrieren. Probieren Sie aus, was passiert, wenn Sie dennoch ein entsprechendes `INSERT` versuchen.

Schreiben Sie ein `SELECT` Statement, das alle *video titles* und deren *upload dates* zurückliefert, in denen das wort “cat” vorkommt. Ihr Ergebnis könnte so aussehen

```
                              videoTitle uploadDate
------------------------------------- ----------
                   my cat watches videos 2021-01-04
in soviet russia videos watch your cat 2021-01-05
```

Schreiben Sie ein `SELECT` Statement, das zählt, in wie vielen *video titles* das Wort “introduction” vorkommt.

Schreiben Sie ein *SELECT* Statement, das alle *video titles* und *view dates* der von *user* `dingbat` angeschauten *videos* liefert. Ihr Ergebnis könnte so aussehen

```
                                videoTitle   viewDate
------------------------------------------- ----------
in soviet russia videos watch your cat 2021-01-06
                      honigkuchen backen 2021-01-07
```

Schreiben Sie ein *SELECT* Statement, das ermittelt, welche *user* die von `dingbat` angeschauten *videos* wann hochgeladen haben. Ihr Ergebnis könnte so aussehen

```
userName uploadDate
-------- ----------
 gigecon 2021-01-05
honeypot 2021-01-06
```

Schreiben Sie ein *SELECT* Statement, das *username* und *password hash* der *user* ermittel, die die von `dingbat` angeschauten *videos* hochgeladen haben. Ihr Ergebnis könnte so aussehen

```
userName  userPW
-------- -------
 gigecon #899FF2
honeypot #343321
```

Aufgabe 5.6: Denken Sie über das von Ihnen aufgestellte Datenbankschema nach. Was passiert in Ihrer Datenbank aktuell, wenn mehrere Personen ein Video mit identischem Titel hochladen wollen? Ist es für die Anwendung sinnvoll, videoTitle mit einem *UNIQUE* constraint zu versehen? Passen Sie ihr Datenmodell gegebenenfalls an, um `brangdag` und `senftorte` zu ermöglichen, Erfahrungen mit Ihren eigenen Katzen zu teilen:

```
my cat also watches videos  am  2021-01-06  von  brangdag
my cat also watches videos  am  2021-01-06  von  senftorte
```