Hochschule Bochum
Bochum University
of Applied Sciences

Fachbereich Wirtschaft
Prof. Dr. Christian Bockermann

Data Science

Wintersemester 2021/2022

Übungsblatt 6

Auf diesem Übungsblatt sollen zunächst ein paar einfache Aufgaben zu den Themen *Funktionen* und *Listen* und dazu dienen, einige Python und Pandas Grundlagen weiter zu vertiefen. In den weiteren Aufgaben geht es dann um die Normalisierung und Modellierung von Daten mit Python.

Aufgabe 1 (Python Basics)

Zur Erinnerung: der Modulo Operator `%` liefert den Rest der ganzzahligen Division zurück, d.h. `9 % 4 ergibt 1` (da: $9 = 2 \cdot 4 + 1$). Wir können damit eine Zahl auf Teilbarkeit durch eine andere Zahl testen:

```
if x % 5 == 0:
   print( "x ist durch 5 teilbar!")
```

1. Definieren Sie eine Funktion `teiler(x)`, die für eine ganze Zahl `x` die Liste der Zahlen von 1 bis `(x-1)` zurückliefert, durch die `x` teilbar ist.

2. Schreiben Sie eine weitere Funktion `ist_prim(x)`, die überprüft, ob die Zahl `x` eine Primzahl ist.

3. Erzeugen Sie eine Liste `primes_30` der Primzahlen zwischen 1 und 30.

Aufgabe 2 (Z-Normalisierung)

Im Python Modul `datascience` gibt es die Funktion `ball_daten()`, die den Datensatz der verschiedenen Sportbälle als DataFrame erzeugt.

1. Rufen Sie die Funktion `ball_daten()` auf um einen DataFrame `data` zu erzeugen.

   Ermitteln Sie Mittelwert und Standardabweichung für die Attribute *Umfang* und *Gewicht*. Schreiben Sie dazu die Funktionen `smean(s)` und `sdev(s)`, die für ein Series Objekt den Mittelwert bzw. die Standardabweichung berechnen.

2. Definieren Sie eine Funktion `z_norm_series(s)`, die für ein Series Objekt `s` eine z-normalisierte Series zurückgibt (vgl. Foliensatz 6, Folie 27).

3. Schreiben Sie eine Funktion `z_norm_df(d)`, die für einen DataFrame `d` einen DataFrame zurückgibt, bei dem jede Spalte z-normalisiert ist.

   Überprüfen Sie ihr Ergebnis, in dem Sie sich die Ausgabe von `data.describe()` nach erfolgter Anwendung ihrer Funktion `z_norm_df(..)` anschauen.

Aufgabe 3 (Klassifikation mit Entscheidungsbäumen)

Vielleicht können Sie sich an die Aufgabe aus der Wirtschaftsinformatik erinnern, bei der für ein Paket anhand des Volumens (Länge, Breite, Höhe) und der Entfernung festgelegt werden sollte, welcher Dienstleister für den Transport des Paketes benutzt werden soll.

Die Daten hatte das Format wie in folgender Tabelle angegeben:

Länge (cm)	**Breite** (cm)	**Höhe** (cm)	**Entfernung** (km)	**Dienstleister**
100	100	40	20	WPS
110	100	100	60	UPS
100	210	100	70	WPS
50	100	70	89	DHL

Diese Aufgabe kann man auch als *Klassifikationsproblem* auffassen, bei der das Attribut `Dienstleister` auf Grundlage der Versandangaben (Paketmaße und Entfernung) vorhergesagt werden soll.
In der Datei `Kurse/DataScience1/data/versand-training.csv` finden Sie einen großen Datensatz mit Versandangaben und dem jeweils zugehörigen Dienstleister.

1. Laden Sie den Datensatz in einen DataFrame.

2. Wie sind die Werte für Länge, Breite, Höhe und Entfernung verteilt? Bestimmen Sie jeweils den Mittelwert und die Standardabweichung!

3. Trainieren Sie mit *SciKit-Learn* einen Entscheidungsbaum auf dem Datensatz (siehe Foliensatz 5, ab Folie 46).

   Hinweis: Es gibt für die Versanddaten bereits einen Trainings- und Test-Datensatz. Sie können daher die Aufteilung in Train/Test Daten (Schritt 2 in dem Folienbeispiel) in diesem Fall überspringen.

   Lassen Sie sich den Entscheidungsbaum anzeigen (`datascience.show_tree`) und vergleichen Sie den Baum mit den Regeln, die für die Wahl des Dienstleisters gelten sollen (Foliensatz *Folien-TutorialDay1*, Folie 46).

4. Testen Sie ihr Modell auf dem Datensatz

   `Kurse/DataScience1/data/versand-test.csv`

   und schauen Sie sich mit SciKit-Learn den *Classification Report* an.

   Welche Klasse wird am besten vorhergesagt?

Aufgabe 4 * (Klassifikation mit k-NN)

Betrachten Sie nochmal den Datensatz aus der letzten Aufgabe 3.

1. Trainieren Sie ein k-NN Modell auf dem Datensatz und nutzen Sie dafür $k = 2$. (Vgl. Foliensatz 6, Folie 12)

2. Nutzen Sie die `sklearn.metrics` Funktion `accuracy_score` um den Trainigsfehler für das Modell mit $k = 2$ zu berechnen:

```
from sklearn.metrics import accuracy_score

y_pred = m.predict(X_train)
train_error = 1 - accuracy_score(y_train, y_pred)
```

3. Schreiben Sie eine Funktion `knn_errors(X, y)`, die für die $k \in \{2, \ldots, 6\}$ die Trainingsfehler berechnet. Die Funktion soll eine Liste von Paaren (Tupeln) der mit `k` und dem dazu ermittelten Trainingsfehler zurückliefern, z.B.:

```
[ (2, 0.874), (3, 0.849), (4, 0.519), ...]
```

4. Erstellen Sie aus der Ergebnis-Liste ihrer Funktion `knn_errors(X, y)` einen DataFrame.