Hochschule Bochum
Bochum University
of Applied Sciences

Fachbereich Wirtschaft
Prof. Dr. Christian Bockermann

Data Science 1

Sommersemester 2022

Hausarbeit

Die Prüfungsleistung zum Modul *Data Science 1* findet als Hausarbeit statt. Die Aufgabenstellung zur Hausarbeit finden Sie in diesem Dokument.

Für die Bearbeitung der Aufgabenstellung und die Erstellung Ihrer Hausarbeit steht wieder der Jupyter-Notebook Server zu Verfügung. Die Abgabe der Hausarbeit erfolgt dann als PDF-Export Ihres Jupyter-Notebooks. Das PDF Ihres Notebooks schicken Sie bis zum 29.7. per Mail an: `christian.bockermann@hs-bochum.de`
Andere Formen der Abgabe sind nicht vorgehen.

Als Materialien können Sie sämtliche Unterlagen aus der Vorlesung und den Übungen mit benutzen, im Internet recherchieren oder weitere Bücher/Kurse mit verwenden. Geben Sie bitte bei Verwendung von umfangreicherem Programm-Code aus dem Netz (mehr als 3-4 Zeilen) die Quelle kurz mit an.

Aufgabe 1 (Python Basics)

Der populäre Weg heutzutage Musik zu hören ist immer mehr die Nutzung von Streaming Angeboten der verschiedenen Musik-Plattformen. Die Firma Spotify hat dazu 2018 auch mal einen Datensatz veröffentlicht[1], der Songs und Streaming-Informationen (anonymisiert) enthält.
Wir betrachten dazu die Liste Songs in folgendem Format vorliegt:

```
(titel, album, musiker, laenge)
```

Die **titel** ist der Name des Songs, **album** der Name des Albums und **musiker** der Name des Künstlers, der den Song spielt. Zusätzlich ist in **laenge** noch die Länge des Songs in Millisekunden angegeben.
Auf dem Jupyter Notebook Server steht Ihnen das Modul **spotify** zur Verfügung, das eine Funktion **songs()** enthält, die die Liste der Songs liefert:

```
import spotify

songs = spotify.songList()

songs[98]  #  ('Black Hole Sun', 'Superunknown', 'Soundgarden',
                                    318000)
```

Wie in dem Python Code zu sehen ist, hat beispielsweise der Song am Index 98 der Liste die folgenden Werte:

```
('Black Hole Sun', 'Superunknown', 'Soundgarden', 318000)
```

Das heisst, das Lied *Black Hole Sun* ist von der Gruppe *Soundgarden* aus dem Album *Superunknown* und ist 318000 Millisekunden lang.

Für eine derartige Liste sollen Sie die folgenden Aufgaben lösen:

1. Bestimmen Sie die *verschiedenen* Musiker in der Song-Liste!
   Schreiben Sie dazu eine Funktion `musiker(xs)`, die für die Liste xs von Songs, die Menge der Musiker zurückgibt, die in der Liste enthalten sind.

   Das Ergebnis der Funktion soll also ein Liste oder Menge (**set**) sein, die jeder Musiker nur einmal enthält.

2. Schreiben Sie eine Funktion `songsVonMusiker(xs, musiker)`, die für die Liste **xs** der Songs und den Musiker **musiker**, die Liste der Songs von dem/den Musiker/n **musiker** zurückliefert.

3. Schreiben Sie eine Funktion `anzahlNachMusiker(xs)`, die die Liste **xs** der Songs nach den Musikern gruppiert und für jeden Musiker die Anzahl der Songs berechnet, die dieser Musiker geschrieben hat.

   Die Liste soll also folgendes Format haben:

   ```
   [('Carly Rae Jepsen', 20), ('Young the Giant', 34), ('Bohkeh', 1),...]
   ```

[1] *Spotify Million Playlist DataSet*,
https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge

4. Schreiben Sie eine Funktion `avgLaenge(xs, musiker)`, die für die Liste `xs` und den Musiker `musiker` die durchschnittliche Länge seiner Songs berechnet.

   Geben Sie dazu die Durchschnittliche Länge in Minuten an, also, z.B. 3,5 für die Länge von 3 Minuten und 30 Sekunden.

5. Schreiben Sie zwei Funktionen `meisteAlben(xs)` und `meisteSongs(xs)` die jeweils die Namen der Musiker mit den meisten Alben und den meisten Songs berechnen.

   (Sofern es mehrere Musiker mit der gleichen maximalen Anzahl von Songs/Alben gibt, sollen die Liste aller dieser Musiker zurückgegeben werden. Ihre Funktion müssen also Listen zurückgeben.)

Aufgabe 2 (Pandas und Statistiken)

Die Datei `Kurse/DataScience1/data/music-streams.csv` enthält (anonymisierte) Informationen über die Streaming Sessions von verschiedenen Benutzern. Die Datei ist ein minimaler Auszug aus dem Million-Song Datensatz von Spotify.
Der Datensatz ist im wesentlichen selbsterklärend. In der folgenden Tabelle sind einige der wichtigsten Spalten exemplarisch dargestellt:

User	SongID	Date	HourOfDay	Skip1	Skip2	Skip3	NotSkipped	ReasonEnd
9697	5b643c54	2018-07-15	14	True	True	True	False	fwdbtn
9512	64ccf2f5	2018-07-15	11	False	False	False	True	trackdone
6874	ca9aeca6	2018-07-15	11	False	True	True	False	fwdbtn
867	0f123593	2018-07-15	3	False	False	True	False	backbtn
4403	f3c3c11b	2018-07-15	15	True	True	True	False	logout
3138	2d25e854	2018-07-15	14	True	True	True	False	fwdbtn
8463	dcc1ff17	2018-07-15	4	False	True	True	False	trackdone
5372	3a8222f8	2018-07-14	20	False	False	True	False	endplay

Die Spalte `HourOfDay` gibt die Stunde des Tages an, zu der ein Song abgespielt wurde.
Die `ReasonEnd`-Spalte enthält den Grund, warum das Anhören eines Songs beendet wurde.
Zur Erklärung der `Skip`-Spalten:

- `Skip1`: Der Song wurde ganz kurz angehört
- `Skip2`: Der Song wurde ein wenig angehört
- `Skip3`: Der Song wurde größtenteils angehört
- `NotSkipped`: Der Song wurde vollständig gehört.

In der Tabelle kann eine SongID natürlich mehrfach auftauchen, sofern ein Song z.B. mehrmals angehört wird. Im Folgenden bezeichnen wir mit *Stream* jeweils das Anhören eines Songs und mit Song den eindeutigen Musik-Titel.
D.h. *die Anzahl der Songs an Tag X*, bezieht sich auf die Anzahl der Musik-Titel, wobei jede SongID nur einmal gezählt wird. Die *Anzahl der Streams an Tag X* hingegen ist die Anzahl aller Abspielungen – auch mehrfach – der verschiedenen Songs an dem Tag X.

Die Aufgaben:

1. Zunächst sollen ein paar generelle Informationen berechnet werden:
   - Wieviele User, wieviele Songs, wieviele Streams enthält der Datensatz?
   - Über welchen Zeitraum erstrecken sich die Daten?
   - Wie oft wurden Songs vollständig angehört?
   - Wie oft wurden Songs nur ganz kurz angespielt?

2. Erstellen Sie einen Plot, der die Anzahl der vollständig abgespielten Streams pro Tag zeigt.

3. Teilen Sie den Tagesverlauf in disjunkte 4-Stundenblöcke auf (0-4 Uhr, 4-8 Uhr, usw.). In welchen Tagesblöcken werden die meisten Streams abgespielt?

   Tipp: Es lohnt sich, dafür eine neue Spalte zu berechnen, die Sie aus der Spalte *HourOfDay* ableiten können.

4. Gibt es in den 4-Stundenblöcken welche, in denen Songs häufiger ganz gehört werden? Berechnen Sie für jeden Zeitblock die Anzahl der kurz/wenig/größtenteils/ganz gehörten Streams.

   Tipp: Wenn Sie die `Skip`-Spalten von True/False auf die Werte 1 und 0 abbilden, können Sie diese Spalten einfach aufsummieren um auf die jeweilige Anzahl zu kommen.

5. Plotten Sie die Anzahl der kurz/wenig/größtenteils/ganz gehörten Streams nach Tageszeit (HourOfDay).

6. Ermitteln Sie die Anzahl der Benutzer nach Tageszeit, die die Streaming-Plattform benutzen. Erstellen Sie einen Plot dazu.

   Hinweis: Hier ist die Anzahl der eindeutigen Benutzer pro HourOfDay gemeint, also ohne dass Duplikate mehrfach gezählt werden.

7. Welche Benutzer haben die meisten Streams vollständig gehört? Selektieren Sie aus den Daten alle Benutzer, die zu den Benutzern mit den meisten Streams gehören.

8. Berechnen Sie für jeden Benutzer den Anteil der Streams, die der Benutzer vollständig gehört hat (anteilig von allen Streams des Benutzers). Wieviele Benutzer haben mindestens 50/75/90 % ihrer Streams komplett gehört?

Aufgabe 3 (Modell-Training)

Eine spannende Lernaufgabe bei Streaming-Diensten ist die Empfehlung von Titeln für die Benutzer. Die Datei `Kurse/DataScience1/data/song-features.csv` enthält einen Datensatz von Songs, für die eine Reihe von Merkmalen aus dem Ton-Daten extrahiert wurden.
Da gibt es z.B. Song-Merkmale wie *dyn_range_mean*, *beat_strength* und andere, die die Musik charakterisieren sollen. Zusätzliche Informationen sind das Erscheinungsjahr, die Länge usw.
Der Datensatz ist in zwei Klassen aufgeteilt, die in der Spalte `label` vermerkt sind. Das sind die Klassen `HIGH` und `LOW`. Songs der Klasse `HIGH` wurden deutlich öfter vollständig gespielt (kaum "*skips*") als Songs der Klasse `LOW` und eignen sich somit besser für die Empfehlungen.
Ihre Aufgabe ist nun, ein Modell zu trainieren, dass für einen Song anhand der musikalischen Merkmale entscheide, ob es eher ein Song der Klasse `HIGH` oder der Klasse `LOW` ist.

1. Laden Sie die Daten in einen DataFrame und geben Sie die Anzahl der Datensätze, sowie die Anzahl der Songs der beiden Klassen an.

   Betrachten Sie die Spalten und Datentypen der Spalten und überlegen Sie sich, welche Spalten Sie für ein Vorhersagemodell verwenden wollen.

2. Wie hoch ist der Fehler eines Modells, das immer nur die Klasse `HIGH` vorhersagt?

3. Trainieren Sie ein Entscheidungsbaum-Modell auf dem vollständigen Datensatz. Welchen Trainingsfehler erreicht ihr Modell?

4. Teilen Sie die Daten in Trainings- und Test-Daten auf und verwenden Sie dabei 80% der Daten zum Training und den restlichen Teil für das Testen.

5. Bestimmen Sie den Parameter `max_depth`, der auf den Daten für ein Entscheidungsbaum-Modell den besten Generalisierungsfehler liefert. Testen Sie für die Bestimmung des Parameters `max_depth` die Werte im Bereich von 2 bis 20.

   Erzeugen Sie dazu einen DataFrame, der den Parameter `max_depth`, den Trainings- und den Test-Fehler enthält.

6. Erstellen Sie einen Plot mit dem Parameter `max_depth` auf der x-Achse, der den Trainings- und Test-Fehler für die verschiedenen Werte von `max_depth` zeigt. Für welches `max_depth` bekommen Sie das beste Modell?

7. Erweitern Sie Ihren Code so, dass für jeden Parameterwert `max_depth` mehrfache Train/Test Splits durchgeführt werden. Das bedeutet, jeder Parameterwert soll mit 5 verschiedenen Train/Test Splits evaluiert werden. Die Ergebnisse sollen dann wieder in einem DataFrame zusammengeführt werden.

   Berechnen Sie aus dem resultierenden DataFrame dann die durchschnittlichen Trainings- und Test-Fehler für jeden Parameterwert und erstellen Sie einen Plot dazu.

Tipp: Dadurch verlängert sich natürlich auch die Laufzeit des Programmes auf das 5-fache. Für die Entwicklung ist es daher hilfreich, sich vielleicht auf z.B. nur 4 Parameterwerte (2 bis 5) und jeweils 2 Train/Tests Splits zu beschränken. Diese Werte können Sie ja dann für den finalen Lauf anpassen.