DIHSÜD

FH JOANNEUM
University of Applied Sciences

Data Science Einstieg: Von Excel zu Python

Data Science and Artificial Intelligence

Debora Stickler & Marco Tilli

FFG
Forschung wirkt.

Das Land Steiermark

LAND KÄRNTEN

Vorstellungsrunde

Unser Ziel / Unsere Erwartungen

- „Scheu“ von etwas Neuem nehmen
- Überblick über Python (insbesondere Pandas) verschaffen
- Grundlegende Konzepte in Python/Pandas „Hands-On“ erklärt

Überblick über den Workshop

- Dateiformate Allgemein
- Überblick Excel vs Python
- Python Hands-On
  - Listen in Python
  - FOR-LOOP
  - IF-Anweisung
  - Installation und Laden von Paketen
- Pandas Hands-On
  - Laden von Daten
  - Datenexploration
  - Indexierung, LOC vs ILOC
  - Pivot-Table & GroupBy

Dateiformate

Dateiformate

Text-Datei: *.txt*

- Reiner Text, keine Formatierungen, keine Ansicht/Struktur
- Wenig Speicherplatz

Comma-Separated-Text-Datei: *.csv*

- „Semi"-Reiner Text, Kolumnen durch Kommas abgegrenzt
- Formatierung nur temporär, wenn man es in Excel öffnet
- Nur ein Sheet möglich
- Mix aus Text und Excel

Excel-Datei: *.xlsx*

- Text, Formatierungen, Formeln, …
- Mehrere Sheets
- Mehr Speicherplatz

Dateiformate

.html

„Hypertext Markup Language“ (deutsch: „Hypertext-Auszeichnungssprache“)
Format für Websiten
Text strukturieren und formatieren, Links auf andere Webseiten, Bilder/Videos/Audio
Graphisch

SQL

“Structured Query Language“
Sprache rein für die Kommunikation mit relationalen Datenbanken.
Daten relativ einfach einfügen, verändern oder löschen.

Überblick:
Excel vs. Python

Excel statt Python

Übersicht über (kleine) Datensätze
- Pivot-Tabellen

Schnell & einfach für erste Veranschaulichungen
- Plausibilitätsanalyse

Formatierungen von Daten
- Farbskalen / Bedingte Formatierung
- Ausreißer schneller erkennbar

Interaktion mit nicht-Python-affinen Personen (Excel kennt „jeder“)

-> Excel „Umsaetze“

Python statt Excel

- Python kennt alle Datenformate
- Fehlende Elemente befüllen
- Daten zusammenfassen
- Wiederholende Schritte für …
  - mehrere Kolumnen
  - verschiedene Excel-Sheets
- Speicherplatz/Übersichtlichkeit: Excel braucht alte(n) Spalte(n), Funktion muss in neuer Spalte bereits sein (Python kann „inplace“ arbeiten)
- Zeitaufwand: definiere Funktion für eine Zeile/Zelle & kopiere in alle anderen
- Kombination von Funktionen: in Excel schnell unübersichtlich
- Fehler in Datenaufbereitung können schneller gefunden werden

https://www.jcchouinard.com/pandas-excel/

Terminologie

Excel	Pandas
Worksheet (Tabelle)	Dataframe
Kolumne	Series
Titel	Index
Zeile	Row
Leere Zelle	NaN

EXCEL

Basic Excel-Funktionen

Mathematische Operationen (elementweise)
+, -, *, /, Runden, Absolutbetrag, …, Groß/Kleinbuchstaben

Mathematische Operationen (spalten/zeilenweise)
Summe, Mittelwert, Korrelation, …

Logische Operationen
Wenn, Und/Oder

Daten begutachten

- Filtern, Sortieren
- Duplikate entfernen
- Bedingte Formatierung

Sheet-übergreifend arbeiten

Beispiele: Kombinationen von Funktionen

- SVerweis und „Logische“ Auswahl von Zeilen
- Sortiere und Plotte
- Gruppieren und zusammenfassen

PYTHON

Python Hands-On

https://datalore.jetbrains.com/notebooks

Listen in Python

Listen in Python

- Wenn wir mehrere Werte einer Variablen zuordnen möchten, können wir in Python Listen erstellen
- Listen werden mit eckigen Klammern markiert.
- Wir müssen keine Module laden um mit Listen zu arbeiten.
- Wir können mit verschiedenen Methoden die Listen bearbeiten.
- Eine davon ist len(), die uns die Länge der Liste ausgibt.

```
a = 2 # wir weisen der Variable den Wert 2 zu
b = 4
a+b = __

liste = [1, 3, 4, 5, 1, 1]
liste2 = [‘Python’, ‘ist’, ‘nicht’, ‘schwer’]
# Länge der Liste
len(Liste2)
```

Listen in Python

- Zugriff auf Elemente mittels eckiger Klammern
- ACHTUNG: INDEX beginnt mit 0, nicht mit 1.
- Wir können ganz leicht Elemente überschreiben
- Mittels del() Operator können wir Elemente aus der Liste löschen.

```
liste2 = [‘Python’, ‘ist’, ‘nicht’, ‘schwer’]

# wir wollen nun das zweite Wort der Liste ausgeben
print(liste2[1]) # ACHTUNG: Index beginnt bei 0!

# Wir können Werte überschreiben:
liste2[2] = ‘sehr’

# … und die Liste um weitere Elemente ergänzen
liste2 = liste2 + [‘meint’ ‘Debora’]

# Löschen von Listenelementen:
del(liste2[2])
```

Aufgabe

- Erstellen Sie die Liste [‘Python’, ‘ist’, ‘nicht’, ‘schwer’] in Python.
- Überschreiben Sie das erste Element mit ‘Excel’
- Fügen Sie ein Wort am Ende der Liste hinzu.
- Lassen Sie sich die Länge der Liste ausgeben.
- Geben sie die ersten zwei Elemente aus.

Zählschleifen und bedingte Konditionen

For-LOOP

Idee: wir wollen über einen fixen Bereich summieren.

→ Wichtiges Programmierparadigma.

Aufgabe:

Gegeben ist eine Liste von Zahlen [1, 4, 1, 9, 10]. Wie können Sie diese geeignet summieren?

FH | J

For-LOOP

```
list = [1, 4, 1, 9, 10]
sum = 0
for i in range(4):
          sum = sum + list[i]
```

Aufgabe

- Gegeben ist eine Liste von Zahlen [1, 4, 1, 9, 10]. Summieren Sie die Zahlen in der Liste!
- Summieren Sie die Zahlen von 1-50 mit Hilfe von Python!

Konditionale Bedinung - IF

Logische Operatoren in Python:

Gleichheit	a==b
Ungleich	a!=b
Kleiner /Kleiner gleich	a<b / a<=b
Größer / Größer gleich	a>b / a>=b
Und-Operator	a & b
Oder-Operator	a | b

FH | J

IF-Anweisung

if Bedingung:

Wahr-Block

Alle Anweisungen werden ausgeführt, wenn die Bedingung wahr ist.

else:

Falsch-Block

Hier können weitere Anweisungen stehen, wenn die Bedingung falsch ist.

```
x = 10
if x > 9:
        print(‘x ist groesser als 9’)
else:
        print(‘x ist kleiner als 9’)
```

FH | J

IF-Anweisung

if Bedingung:

Wahr-Block

Alle Anweisungen werden ausgeführt, wenn die Bedingung wahr ist.

else:

Falsch-Block

Hier können weitere Anweisungen stehen, wenn die Bedingung falsch ist.

```
x = 10
if x < 9:
        print(‘x ist groesser als 9’)
elif x < 5:
        print(‘x ist groesser als 5’)
else:
        print(‘x ist kleiner als 5’)
```

Aufgabe

1. Schreibe einen IF-Else-Ablauf, der Checken soll, ob eine Variable „var“
   - ein Integer ist
   - oder der Boolscher Wert "TRUE" ist
   - oder ein String
   - oder etwas anderes, dann gib den type der Variable zurück
2. Probiere deinen Ablauf mit folgenden Werten aus:
   - var = „Integer“
   - var = „1“
   - var = 1.2
   - var = False
   - var = bool(0)
   - var = bool(1)

Module in Python

Module

- Über Module können zusätzliche Funktionalitäten und Funktionen eingebunden werden.
- Diese müssen zu Beginn der Session geladen werden…
- … sie müssen allerdings nur einmal installiert werden!
- Es können auch nur einzelne Funktionalitäten aus Modulen geladen werden.

Beispiele:

```
import numpy as np
import pandas as pd
from scipy import linalg
```

Ein paar wichtige Packages & Module

- Numpy: (effizientes!) Rechnen mit Vektoren und Matrizen
- Pandas: Arbeiten mit Dataframes (Datenstrukturen und Analyse)
- Matplotlib: Bibliothek zur Erstellung von Plots. Ist den Plots in Matlab nachgebaut
- Scipy: Bibliothek für Scientific Computing

Pandas

Funktionen-Übersicht

- *read_csv(<path>), df.to_csv(<path>)*
- *Daten anschauen:*
  - df.shape, df.head(), df.describe()
  - sum(df.isna()), df.fillna(), df.drop_duplicates(),
  - df.plot(x=.., y=..)
  - df.value_counts()
- Daten wählen
  - df[COL], df.loc
- Daten arbeiten
  - df.sort_values(by=..)
  - df.groupby(by=..)
- Sonstiges
  - df.mean(), df[COL].unique()
  - df.round(<decimal>)

Filterung eines Pandas DataFrames

Wir können die Informationen basierend auf verschiedenen Faktoren filtern basierend auf

1. Konditionen
2. Zeilen-/Spaltenindex
3. Spaltennamen

WICHTIG: Die Indexierung beginnt bei 0!

Unterschied loc vs. iloc

- Mit loc können wir Spaltennamen direkt ansprechen

- iloc gibt Zeilen (oder Spalten) an einer bestimmten Position mit gegebenen Index zurück (erlaubt nur Integer (also ganzzahlige Werte) als Input, keine Strings (also Buchstabenketten/Wörter).

Aufgabe

Wir verwenden die Datei movies2000.csv.

1. Laden Sie die Datei, und speicheren Sie sie als pandas-DataFrame unter den Namen „movies2000“.
2. Wir wollen „nur“ die folgenden Spalten behalten:
   Title, Year, Language, Country, Content Rating, Duration, Budget, GrossEarnings, Director, IMDB Score
3. Geben Sie die ersten 10 Zeilen aus, um einen Überblick zu erhalten.
4. Wie viele leere (Tipp: .isna) Felder haben die ersten paar Kolumnen?
5. Sortieren Sie das DataFrame nach dem Budget der Filme.
6. Kreieren Sie ein Sub-DataFrame „movies_german“, dass nur Filme in deutscher Sprache beinhaltet. Löschen Sie danach die Spalte „Language“.
7. Berechnen Sie für die deutschen Filme den Gewinn (= GrossEarnings – Budget) und speicheren Sie den Wert in eine neue Spalte „Profit“.

Gruppieren & Pivot-Table in Pandas

- GroupBy
- Pivot Table

https://builtin.com/data-science/pandas-pivot-tables

Wiederholung

Excel → Python (Pandas)

	Excel	Python
Math. Operationen, Wenn-Operationen	Braucht die alten Spalten und neue Funktion in jeder Zelle der neuen Spalte	Elementweise, Spaltenweise oder Tabellenweise möglich, entweder als neue Instanz oder „inplace“ (siehe dann Anwendung)
Daten Summary	Leiste anklicken	*df.describe()*
Daten verschieben	Ausschneiden, einfügen, auf Funktionen/Formeln achten	- *merge()*, *concat()* - *df.loc[-1] = df.loc[0]* *df.drop(0, inplace=True)*
Text umschreiben	Zelle händisch umschreiben	df.iloc[i,j]: best. Elem. Zugreifen
Daten-Typ ändern	Schaltfläche (wenn überhaupt möglich)	*df[COL].astype(<TYP>)*
Filtern	Schaltfläche	z.B.: *df.loc[df.COL > 100]*
Sortieren	Schaltfläche (oder neue Spalte)	z.B.: *df.sort_values(by='COL')*
Schnelles Visualisieren	Tabelle markieren, Schaltfläche	*df. plot(x=idx, y=['COL1', 'COL2'], kind=bar/hist/line/…)*

Excel → Python (Pandas)

	Excel	Python
Kopiere Zellinhalt und ersetze bestimmten Teil	Ersetzen(Alter Text; Erstes Zeichen; Anzahl Zeichen; Neuer Text)	*df[COL].str.replace(old, new)*
Sortiert eine Spalte in der Reihenfolge einer anderen	SortierenNach(COL2; COL1)	*df.sort_values(by='COL1')[COL2]*
Teilergebnis von gefilterten Daten	Teilergebnis(Bereich)	[<Logical>] + Operation z.B. df[df[COL1] < 10].sum() Achtung: (df[COL1] < 10).sum() zählt Anz. Der Zeilen, die Bedingung erfüllen
Durchsucht eine Tabelle nach einem Kriterium/ Suchtext und gibt den zugehörigen Wert zurück	SVerweis(Suchtext; Suchfeld; Spalte)	df[df[<Suchfeld>] == <Suchtext>][<Spalte>]
Gibt aus Tabelle den Wert aus Zeile&Spalte zurück	Index(Suchfeld; Zeile; Spalte)	df.iloc[Zeile, Spalte] (hier: df ist Suchfeld)
Ähnlich zu SVerweis, gibt aber Position zurück UND braucht geordnete Spalte (sonst Fehler „#NV“)	Vergleich(Suchzahl; Suchfeld)	df.index[<Suchfeld> == <Suchzahl>][0]
Gibt dem Index entsprechendes Element aus Liste zurück – oft für Kombinationen genutzt	Wahl(Index; Liste)	<Liste>[<Index>]

Vielen Dank für Ihre Aufmerksamkeit!