Metodika analýzy dat: Od základů po aplikace metod strojového učení

Pandas

Jakub Steinbach, Jan Vrba

Ústav matematiky, informatiky a kybernetiky
VŠCHT

1.10.2024

Obsah slajdů I

Pandas

Pandas - základní funkcionality

- slouží jako užitečný nástroj pro datovou analýzu
- hlavním prvkem knihovny jsou *Series* a *Data Frames*, jež reprezentují datový sloupec, respektive datovou tabulku
- umožňuje rychlé hledání skrz data a jejich následné čištění, agregaci dat podle zvolených kategorií, zobrazení základních statistických údajů, filtrování dat, slučování tabulek...
- načítání i ukládání souborů různých formátů (TXT, CSV, JSON, PICKLE, HTML, XLSX...)
- obsahuje základní vizualizaci dat do bodového, sloupcového či plošného grafu
- logika práce s knihovnou hodně připomíná práci v Excelu
- kompletní přehled funkcí viz dokumentace `https://pandas.pydata.org/docs/`

Pandas - Data Frame

```
import pandas as pd

# Data - Columns structure
data = [["Jakub", "Novak", 20],
        ["Karel", "Prochazka", 32],
        ["Jan", "Novotny", 18]]
column_names = ["Name", "Surname", "Age"]
df1 = pd.DataFrame(data=data, columns=column_names)
print(df1)

# Dictionary
data = {"Name": ["Jakub", "Karel", "Jan", "Tomas", "Radek", "Milan"],
        "Surname": ["Novak", "Prochazka", "Novotny", "Cerveny", "Jerman",
                    "Varecka"],
        "Age": [20, 32, 18, 45, 23, 56]}
df2 = pd.DataFrame(data)
print(df2)

# First n rows (default five)
print(df2.head())

# Last n rows (default five)
print(df2.tail())
```

Pandas - základní informace o datech

Pandas umožňuje přes funkce describe, info, mean, mode, median a další zobrazit přehled základních statistických údajů o tabulce. Pokud chceme použít tyto funkce na jednotlivé sloupce, můžeme k nim přistoupit přes tečkovou notaci.

```
# Information on the columns and their data types
print(df2.info())

# Basic statistical information of columns with numeric values
print(df2.describe(), end="\n\n")

# Other statistical information - you can approach
print(df2.Age.mode(), end="\n\n")
print(df2.Age.median(), end="\n\n")
```

Pandas - vrácení částí tabulek

Pro vrácení či změny části Data Frame se využívají lokátory loc a iloc. Zároveň lze tabulku indexovat podobně jako jiné kontejnerové proměnné přes hranaté závorky.

```
# Accessing whole columns
print(df2[["Name", "Surname"]])

# Using single square brackets for a single column will return
# Series instead of a table
print(type(df2[["Name"]]), type(df2["Name"]))
```

Pandas - Data Frame

```
# Accessing whole columns
print(df2[["Name", "Surname"]])
# Using single square brackets for a single column will return
# Series instead of a table
print(type(df2[["Name"]]), type(df2["Name"]))
# Locate by index and column names
print("Printing an entire row...")
print(df2.loc[5], end="\n\n")
print("Printing an entire column...")
print(df2.loc[:, "Surname"], end="\n\n")
print("Printing a specific cell...")
print(df2.loc[5, "Surname"], end="\n\n")
# Locate by numerical index
print("Printing an entire row...")
print(df2.iloc[5], end="\n\n")
print("Printing multiple rows...")
print(df2.iloc[3:5,], end="\n\n")
print("Printing an entire column...")
print(df2.iloc[:, 1], end="\n\n")
print("Printing a specific cell...")
print(df2.iloc[5, 1], end="\n\n")
# Changing indices to different values to show difference between the two
import numpy as np
df2.index = np.linspace(20, 25, 6, dtype="int")
print("The table with different indices")
print(df2.head(), end="\n\n")
print(f"Accessing by loc: {df2.loc[20, 'Surname']}, by iloc: {df2.iloc[0, 1]}",
      end="\n\n")
```

Pandas - odstranění řádků a sloupců

Pomocí lokátorů lze tabulky zvětšovat i zmenšovat, tedy přidávat i ubírat řádky a sloupce. Pro mazání sloupců a řádků lze také využít funkci drop. Pro spojení dvou tabulek můžeme využít metodu concat.

```
# Resetting back the indices
df2.reset_index(drop=True, inplace=True)

# Adding a new row and column using locators
new_row = ["Radka", "Dvorakova", 25]
df2.loc[df2.shape[0]] = new_row
print(df2.tail(), end="\n\n")

df2["Vocation"] = np.nan
df2.loc[1, "Vocation"] = "ucitel"
print(df2.head(), end="\n\n")

# Deleting a row using drop function
print("Dropped row...")
print(df2.drop(3), end="\n\n")
print("Dropped column...")
df2.drop("Surname", axis=1, inplace=True)
print(df2, end="\n\n")

# Concatenating two tables
df3 = pd.concat([df2, df2])
print(df3, end="\n\n")
```

Pandas - sloučení sloupců dvou tabulek

Pokud máme dvě tabulky se společným identifikátorem, můžeme je spojit pomocí metody merge.

```
# Merging two tables
data3 = {"Name": ["Karel", "Jan", "Pavel"],
         "Age": [20, 25, 30]}
data4 = {"Vocation": ["ucitel", "pravnik", "vyrobce bublifuku"]}
id = ["ID001", "ID002", "ID003"]
data3["id"] = id
data4["id"] = id

df3 = pd.DataFrame(data3)
df4 = pd.DataFrame(data4)
print(df3, end="\n\n")
print(df4, end="\n\n")
print(df3.merge(df4, on="id", how="left"), end="\n\n")
```

Pandas - ukládání a nahrávání dat

Pro ukládání dat existují metody pojmenované *to_* a pro nahrávání pojmenované *read_*. Zároveň pandas umožňuje měnit Data Frame na jiné typy proměnné. Níže jsou uvedené pouze příklady funkcí, výčet není úplný.

```
# Saving data
# To csv (and other text formats)
df2.to_csv("data.csv", index=False)
# To Excel (openpyxl library required)
df2.to_excel("data.xlsx", index=False)
# To HTML
df2.to_html("data.htm")
# To JSON
df2.to_json("data.json")
# To PICKLE
df2.to_pickle("data.pickle")
# To a dictionary
print(df2.to_dict(), end="\n\n")
# Loading data
df5 = pd.read_csv("data.csv")
print(df5.head())
```

Explanatory Data Analysis (EDA)

EDA - úvod

EDA je proces zpracování neznámého datasetu za účelem popisu jednotlivých datových řad a potenciálních vztahů mezi jednotlivými kategoriemi. Obvykle je jeho součástí předzpracování, jež má za úkol data očistit o chybné hodnoty.
Příklady některých kroků budou ukázány na datasetu stažitelným zde: `https://web.vscht.cz/~steinbaj/python_intro/diabetes_modified.htm`.

EDA - úvod

Knihovna Pandas obsahuje velké množství nástrojů právě pro EDA včetně jednoduchého grafického zobrazení. Zároveň ji lze dobře kombinovat s knihovnou Seaborn, jež umožňuje složitější vizualizaci. Seaborn je založena na knihovně matplotlib, nicméně je tvořena tak, aby výsledné grafy měly uspokojivou grafickou podobu bez dlouhého přenastavování různých vlastností grafu.
Více o Seaborn včetně příkladů naleznete v dokumentaci:
documentation:`https://seaborn.pydata.org/tutorial.html`

EDA - Načtení a inspekce dat

```
# Imports
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

# Data import
url = "https://web.vscht.cz/~steinbaj/python_intro/diabetes_modified.htm"
data = pd.read_html(url)[0]

# Basic information on data types
print(data.info(verbose=True, show_counts=True, memory_usage=True), end="\n\n")

# Basic statistical information
print(data.describe(percentiles=[0.1, 0.25, 0.3, 0.5, 0.7, 0.75, 0.9]).T,
      end="\n\n")

# Checking null values
print(data.isnull().sum(), end="\n\n")

# Deleting null values
data_null_dropped = data.copy().dropna()
print(f"Dataset␣without␣nulls␣has␣{data_null_dropped.shape[0]}␣rows.")

# Alternatively, filling nan values with a mean
mean_BMI = data["BMI"].mean()
data_null_replaced = data.copy().fillna(mean_BMI)
print(f"Dataset␣with␣replaced␣nulls␣has␣{data_null_replaced.shape[0]}␣rows.")
```

EDA - Hledání souvislostí mezi sloupci

```
# Check counts of outcome
data_null_replaced["Outcome"].value_counts()
plt.figure()
data_null_replaced["Outcome"].value_counts().plot.bar()

# Correlation matrix
data_corr = data_null_replaced.corr(numeric_only=True)
print(data_corr, end="\n\n")

# Selecting BMI and Pregnancies to check relation with Outcome
# Dividing both columns into bins
data_null_replaced["BMI_cat"] = pd.cut(x=data_null_replaced.BMI,
                                       bins=[0, 15, 25, 99],
                                       labels=["low", "norm", "high"],
                                       include_lowest=True)
data_null_replaced["Pregnancies_cat"] = pd.cut(x=data_null_replaced.Pregnancies,
                                               bins=[0, 0.99, 4, 99],
                                               labels=["none", "norm", "high"],
                                               include_lowest=True)

# Pivot table
print(pd.pivot_table(data_null_replaced, values="Outcome", index="BMI_cat",
                     columns=["Pregnancies_cat"], aggfunc=np.mean), end="\n\n")

# Visualization through grouping
print(data_null_replaced.groupby(["Pregnancies_cat", "BMI_cat"])[["Outcome"]].mean()
      end="\n\n")

# Show created charts
plt.show() # Always move this to the end.
```

EDA - Vizualizace hodnot pomocí Seaborn

```
# Correlation matrix with heatmap in Seaborn
plt.figure()
sns.heatmap(data_corr, annot=True, cmap="crest")

# Histogram plots of all featured columns
columns = list(data_null_replaced.columns)
columns.remove("Outcome")
for col in columns:
    plt.figure()
    sns.histplot(data_null_replaced, x=col, stat="density", hue="Outcome",
                 element="step")

# Bivariate histogram showing relation between BMI, Pregnancies and Outcome
plt.figure()
sns.histplot(data_null_replaced, x="BMI", y="Pregnancies", hue="Outcome",
             element="step", cbar=True)
# Doing the same but with relations plot
plt.figure()
sns.relplot(data_null_replaced, x="Glucose", y="BloodPressure",
            hue="Outcome")

# Showing box plots of each column to see outliers
columns.remove("Pregnancies_cat")
columns.remove("BMI_cat")
for col in columns:
    plt.figure()
    sns.boxplot(data_null_replaced, x=col)
plt.show() # Always move this to the end.
```