Programowanie Strukturalne

-

Laboratorium 06 - Python lab03

Adam Szajgin s319821 gr. 109

Politechnika Warszawska

13 stycznia 2024

Spis treści

1.1 Temat programu

Poniższy program to proste narzędzie demonstracyjne umożliwiające użytkownikowi wykonywanie różnorodnych działań na niewielkiej bazie danych. Stworzone z myślą o prezentacji różnych funkcji, program oferuje możliwość wyboru operacji na dostępnej bazie danych zarobkowych dorosłych.

1.2 Opis programu

Program przy wykorzystaniu bibliotek matplotlib oraz pandas ma za zadanie umożliwienie użytkownikowi na modelowanie danych oraz ich obrazowanie w prosty sposób. Komunikacja z użytkownikiem odbywa się poprzez CLI (terminal), to tam użytkownik wybiera operacje i przekazuje parametry do późniejszego wywoływania funkcji w back-endzie programu.
Program wczytuję bazę danych w postaci pliku .csv z danymi opisującymi wiersze, kolumny oraz różnego typu zawartością. Istotny jest również fakt, że baza jest niekompletna i niektóre rekordy zostały wypełnione znakiem ” ?” - w ramach oznaczenia, iż informacja ta była nieznana. Poszczególne wiersze zatem potrafią mieć niewypełnioną np.: kolumnę ”occupation”.
Program daje możliwość wykonania czterech operacji; dwóch prezentujących działanie funkcji przetwarzania danych w postaci wykresów (operacje 1 i 2), oraz dwóch dotyczących danych „niepełnych”.

Do wykonania zadania użyłem bazy danych pobranej z repozytorium: https://archive.ics.uci.edu/dataset/2/adult . Jest to ogólno-dostępne repozytorium uniwersytetu UC Irvine. Baza danych została specjalnie wybrana tak, by posiadała dane opisujące wiersze, kolumny jak i różnego typu zawartość – nie zawsze kompletną.

Również program jest wykonany przy użyciu ramek modułu pandas, różnorakich funkcji dekomponujących zadania. Natomiast obsługuje wyjątki poprzez adresowanie problemu z niepełnymi danymi na dwa sposoby udostępnione do wyboru dla użytkownika.

Ten program został utworzony w celu demonstracyjnym, dlatego też wykorzystałem przykładowe dane, aby pokazać działanie różnych funkcji. Nie był optymalizowany pod kątem uniwersalności czy efektywności.

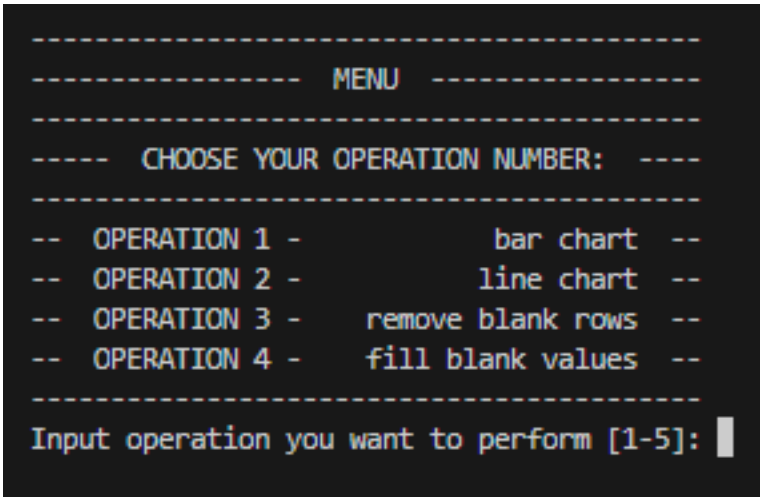
1.3 Prezentacja działania programu

UWAGA: Prezentację działania programu podzieliłem na dwie części; formalną i praktyczną. W formalnej opisuję jak wygląda wejście do programu, itp. W skrócie; mało znaczące z punktu widzenia laboaratorium rzeczy, a dużo znaczące z punktu widzenia ogólno pojętej "dokumentacji". Część praktyczna - odnosi się do wymagań laboratorium.

I CZĘŚĆ FORMALNA

Po uruchomieniu programu, w back-endzie otwierany jest plik csv z zapisami bazy danych.

Natomiast na front-endzie aplikacji użytkownik napotyka ekran "WELCOME" a następnie ekran "MENU" wyboru operacji na bazie:



Rys. 1. "Menu" programu

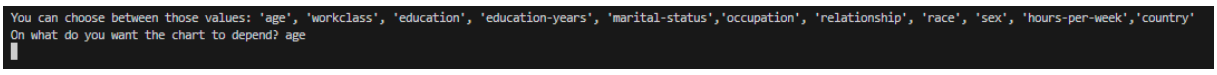
I CZĘŚĆ PRAKTYCZNA:

WAŻNE: Komunikacja z użytkownikiem realizowana jest poprzez CLI i wpisywanie odpowiednich numerów operacji 1-5. Pierwsze dwie funkcje, są przykładami funkcji przetwarzania danych w postaci wykresów. Natomiast, dwie pozostałe funkcje dotyczą danych „niepełnych".

1. Pierwsza operacja - "bar chart"

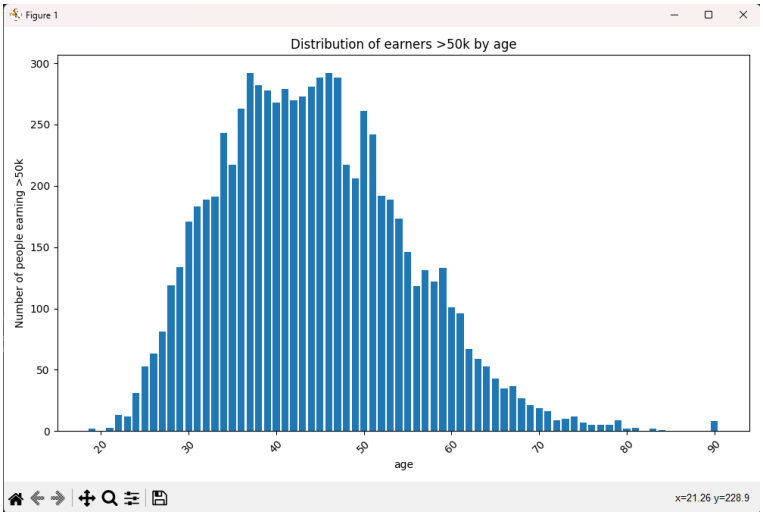
   Operacja ta polega na wykreśleniu wykresu słupkowego ilości osób zarabiających powyżej ¿50k$ w zależności od wybranej zmiennej.

   Po wybraniu tej operacji użytkownik napotyka wybór:



Rys. 2. Symulacja wyboru opcji: "bar chart"

W tym przypadku, wybrałem opcję 'age' by zwizualizować zależność zarobków od wieku osób w bazie danych. Tak prezentuje się uzyskany wykres:



Rys. 3. Uzyskany wykres z operacji 1 - zależność zarobków od wieku

Oraz kod odpowiedzialny za powyższe operacje:

```
def plot_earnings_by_attribute(data, attribute):
    high_earners = data[data["income"] == " >50K"]

    grouped_data = high_earners.groupby(attribute).size().reset_index(name="count")

    if grouped_data.empty:
        print(f"No data available for attribute: {attribute}")
        return

    plt.figure(figsize=(10, 6))
    plt.bar(grouped_data[attribute], grouped_data["count"])
    plt.title(f"Distribution of earners >50k by {attribute}")
    plt.xlabel(attribute)
    plt.ylabel("Number of people earning >50k")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

Rys. 4. funkcja wywoływana odpowiedzialna za wykreślanie wykresu słupkowego

```python
while True:
    disp_welcome()
    op_choice = disp_menu()

    match op_choice:
        case 1:
            print(
                "You can choose between those values: 'age', 'workclass', 'education',
            )
            x_axis = input("On what do you want the chart to depend? ").lower()
            plot_earnings_by_attribute(df, x_axis)

            if end_option():
                continue
```

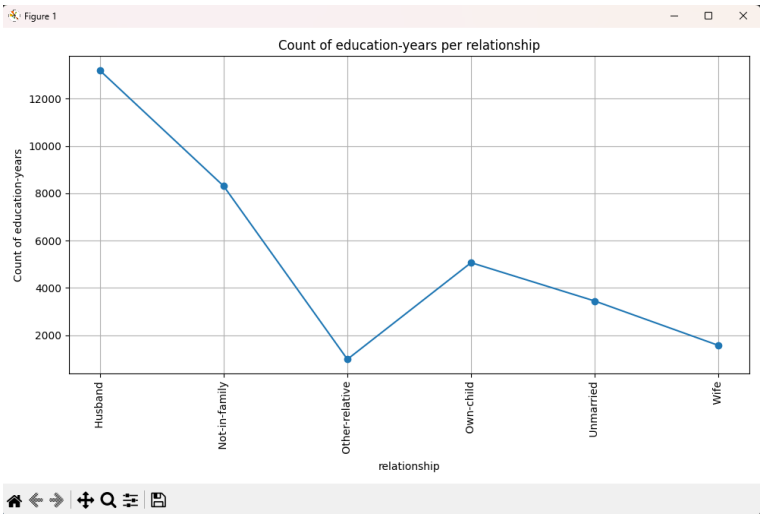
Rys. 5. Wykonanie operacji 1 w switch case

2. Również prezentację graficzną danych zawarłem w operacji nr. 2:
   Po wybraniu tej operacji użytkownik napotyka wybór:

```
You can choose between those values: 'age', 'workclass', 'education', 'education-years', 'marital-status','occupation', 'relationship', 'race', 'sex', 'hours-per-week','country'
On what do you want the chart to depend? relationship
What youd like to be the depending value? education-years
```

Rys. 6. Symulacja wyboru opcji: ”line chart”

W tym przypadku, wybrałem opcję ’relationship’ by zwizualizować zależność ’education-years’ od statusu marytalnego w bazie danych. Tak prezentuje się uzyskany wykres:



Rys. 7. Uzyskany wykres z operacji 2 - zależność zarobków od wieku

Oraz kod odpowiedzialny za powyższe operacje:

```python
def plot_line(dataframe, column_x, column_y):
    grouped = dataframe.groupby(column_x)[column_y].count().reset_index()

    plt.figure(figsize=(10, 6))
    plt.plot(grouped[column_x], grouped[column_y], marker="o")
    plt.xlabel(column_x)
    plt.ylabel(f"Count of {column_y}")
    plt.title(f"Count of {column_y} per {column_x}")
    plt.xticks(rotation=90)
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Rys. 8. funkcja wywoływana odpowiedzialna za wykreślanie wykresu liniowego

```python
case 2:
    print(
        "You can choose between those values: 'age', 'workclass', 'education',
    )
    x_axis = input("On what do you want the chart to depend? ").lower()
    y_axis = input("What youd like to be the depending value? ").lower()
    plot_line(df, x_axis, y_axis)
    if end_option():
        continue
```

Rys. 9. Wykonanie operacji 2 w switch case

Poniższe dwie operacje natomiast przedstawiają "radzenie" sobie z pustymi danymi.

3. Operacja 3 polega na usunięciu wszystkich wierszy które posiadają pusty rekord w jakiejkolwiek z kolumn Po wybraniu tej operacji użytkownik napotyka wybór:

```
Youre about to remove all rows with at least one blank value at any column, do you want to proceed [P] or exit [E]? p
Number of rows: 32561
Number of rows: 30162
Do you want to exit (E) or return to the menu (M)?:
```

Rys. 10. Symulacja wyboru opcji: "remove blank rows"

Użytkownik zostaje poinformowany o tym, że za chwile zostaną usunięte wszystkie wiersze które posiadają min. 1 rekord pusty. Po wybraniu opcji "p" (od proceed -ang.) wyświetlana jest obecna ilość wierszy, uaktywniana funckcja do usuwania i następnie ponownie wyświetlana jest ilość wierszy bazy danych.

Oraz kod odpowiedzialny za powyższe operacje:

```python
def remov_blank_rows(df):
    mask = (df == " ?").any(axis=1)
    df_cleaned = df[~mask]
    return df_cleaned


def check_records_number(df):
    row_count = len(df)
    print("Number of rows:", row_count)
```

Rys. 11. funkcje wywoływane odpowiedzialne za wykreślanie wykresu liniowego

```python
case 3:
    choice = input(
        "Youre about to remove all rows with at least one blank value at any column, do you want t
    ).lower()
    if choice == "p":
        curr_records = check_records_number(df)
        df_cleaned = remov_blank_rows(df)
        check_records_number(df_cleaned)
    else:
        continue

    if end_option():
        continue
```

Rys. 12. Wykonanie operacji 3 w switch case

4. Operacja 4 polega natomiast na wypełnieniu pustych rekordów przypadkowym rekordem z zestawu wszystkich możliwych rekordów kolumny w bazie danych.

   Po wybraniu tej operacji od razu następuje wydrukowanie wszystkich różnych opcji kolumny ”occupation”, jak możemy zaobserwować, znajduje się tam ” ?” - czyli rekord pusty. Natomiast od razu później drukowane są wszystkie możliwe opcje kolumny po wypełnieniu rekordów pustych jakąś inną opcją, tam już nie widzimy ” ?” - rekordu pustego!

```
[' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']
[' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support'
 ' Protective-serv' ' Priv-house-serv' ' Armed-Forces']
Do you want to exit (E) or return to the menu (M)?:
```

Rys. 13. Symulacja wyboru opcji: "fill blank values"

Osobiście uważam, że ten sposób manipulacji danymi jest nienajlepszy jako, że przekłamuje nam statystki. By lepiej odzwierciedlać prawdę należałoby wprowadzić element "wagi" do wyboru losowego (tzn. najczesciej wystepujaca opcja powinna wypadac czesciej). Jednak jest to sposób który pozwala nam eliminować w jakiś sposób rekordy puste.

Oraz kod odpowiedzialny za powyższe operacje:

```python
def fill_blanks(df, column):
    unique_values = df[column].unique()

    def other_values(unique_values):
        other_values = [value for value in unique_values if value != " ?"]
        return other_values

    non_question_mark_values = other_values(unique_values)

    for index, value in df[column].items():
        if value == " ?":
            random_filler = random.choice(non_question_mark_values)
            df.at[index, column] = random_filler

    return df
```

Rys. 14. funkcje wywoływane odpowiedzialne za wykreślanie wykresu liniowego

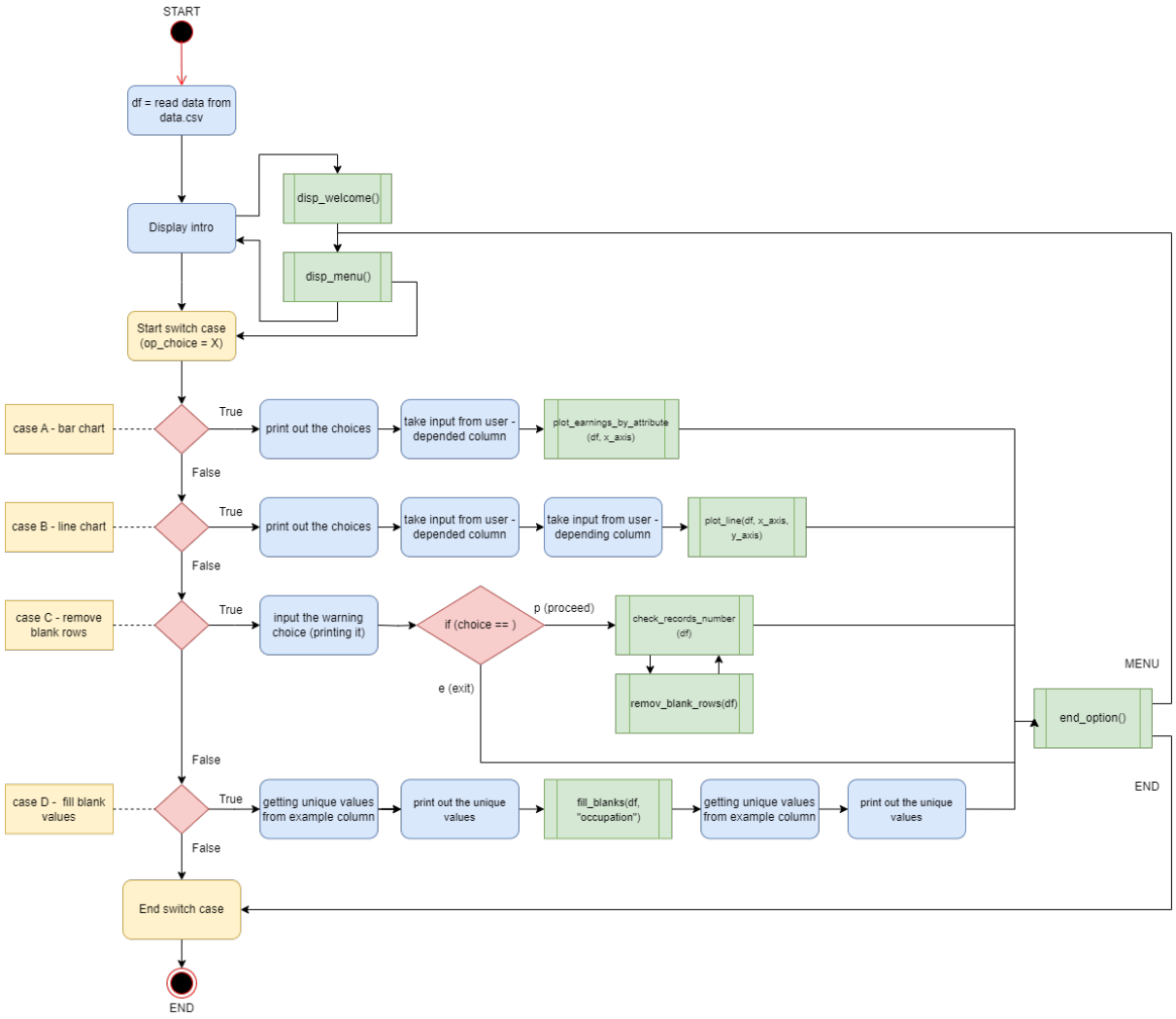
```
case 4:
    unique_values = df["occupation"].unique()
    print(unique_values)
    df_fixed = fill_blanks(df, "occupation")
    unique_values = df_fixed["occupation"].unique()
    print(unique_values)

    if end_option():
        continue
```

Rys. 15. Wykonanie operacji 4 w switch case

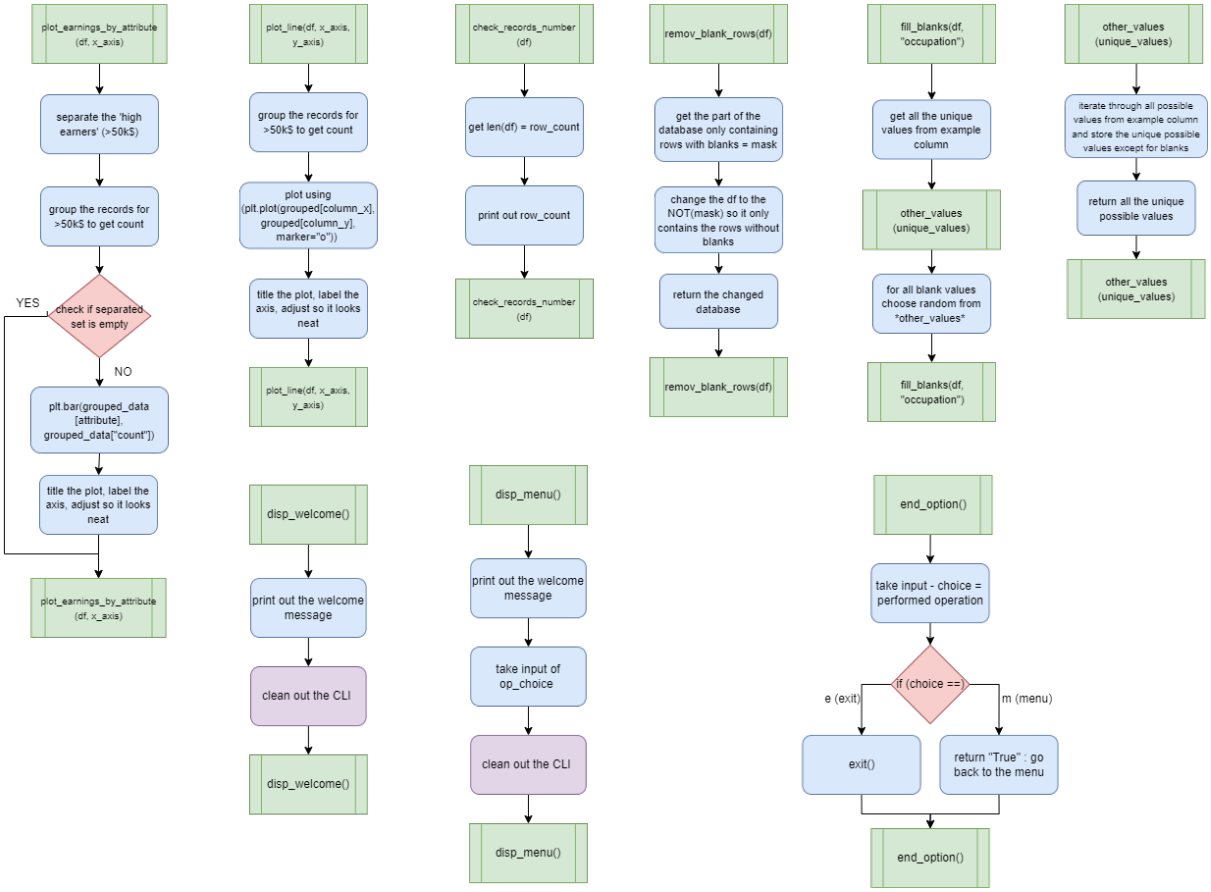
1.3 Algorytm działania w standardzie UML

Poniżej prezentuje diagram w standardzie UML programu. Pierwszy diagram prezentuje schemat działania strukturalnego programu, natomiast drugi prezentuje działanie każdej z użytych funkcji. Do wykonania diagramu został użyty program draw.io.



Rys. 16. diagram programu w standardzie UML

Oraz diagram funkcji użytych w programie:



Rys. 17. diagramy funkcji używanych w programie w standardzie UML