Druhá část projektu - základní analýza dat

Cílem této částu projektu je provést základní analýzu datasetu Statistika nehodovosti Policie ČR. Data pro jednotlivé roky jsou v XLS souboru stažena a jsou k dispozici na stránce: https://ehw.fit.vutbr.cz/izv/data_23_24.zip . Popis dat naleznete na stránce https://ehw.fit.vutbr.cz/izv. Ačkoliv se jedná o soubory s příponou XLS, ve skutečnosti jsou to HTML soubory. Tento formát je dán daty z veřejných zdrojů.

Řešení se skládá ze 5 úkolů, přičemž výstupem každého úkolu bude jedna funkce implementovaná v jazyce Python.

Cíl

Cílem je vytvořit kód, který vizualizuje tři různé závislosti v datech. Kód bude součástí jednoho souboru `analysis.py`, jehož **kostru naleznete v elearningu** v IS VUT. Předpokládá se, že budete **primárně pracovat s knihovnami Pandas a Seaborn** + je dovolené využít všechny knihovny zmiňované během přednášek. Matplotlib používejte pouze pro doladění vizuální podoby. Doporučený postup řešení je pouze nápovědou, můžete samozřejmě problém řešit jiným způsobem, pokud to povede ke stejnému výsledku bez nárustu komplikovanosti.

Odevzdávání a hodnocení

Soubor `analysis.py` odevzdejte do 1. 12. 2024. Hodnotit se bude zejména:

- správnost výsledků
- vizuální zpracování grafů
- kvalita kódu
    - efektivita implementace (nebude hodnocena rychlost, ale bude kontrolováno, zda nějakým způsobem řádově nezvyšujete složitost)
    - korektní práce s Pandas a Seaborn
    - přehlednost kódu
    - dodržení standardů a zvyklostí pro práci s jazykem Python (PEP8)
    - dokumentace kódu

Celkem lze získat až 20 bodů, přičemž k zápočtu je nutné získat z této části minimálně 1 bod.

Úkol 1: Načtení dat (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

```
def load_data(filename : str, ds : str) -> pd.DataFrame:
```

Funkcionalita

- Funkce načte data obsažená v ZIP souboru získaného z adresy https://ehw.fit.vutbr.cz/izv/data_23_24.zip. Argument `filename` určuje cestu k souboru. Argument ds určuje název xls souboru (bez "l" prefixu - t.j. např "nehody" či "nasledky"). Soubor nestahujte z webu.
- Tento soubor obsahuje XLS soubory, které jsou však ve skutečnosti HTML soubory s kódováním cp1250.
- Vaším úkolem je postupně projít všechny (oba) potřebné soubory a spojit je dohromady - funkce musí umí načíst oba dva, ale samostatně.
- Není dovoleno vytvářet pomocné soubory.
- Data ve sloupcích nemodifikujte (zpracování dat bude součástí funkce `parse_data`).
- Názvy sloupců musí odpovídat názvům specifikovaném v popisu datového souboru (t.j. p1, p36, p37, …). Smažte nepojmenované sloupce (obsahují vždy NaN).

Tipy: pracujte s třídou zipfile.ZipFile(), při načítání dat přes Pandas berte v úvahu fakt, že data jsou v kódování `cp1250`.

Upozornění: Další skripty budou využívat vaši implementaci načítání datového rámce. Bez této implementace není možné hodnotit další úkoly**.**

Úkol 2: Formátování a čištění dat (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

```
def parse_data(df : pd.DataFrame,
               verbose : bool = False) -> pd.DataFrame:
```

Funkcionalita

- Funkce dostane na vstup DataFrame získaný voláním funkce `load_data()` s parametrem ds=”nehody”.
- Vytvoří se nový DataFrame.
- Funkce vytvoří v DataFrame sloupec *date*, který bude ve formátu pro reprezentaci data (berte v potaz pouze datum, t.j. sloupec *p2a*)
- Vytvořte sloupec region podle následující tabulky:
  `{0: "PHA", 1: "STC", 2: "JHC", 3: "PLK", 4: "ULK", 5: "HKK",6: "JHM", 7: "MSK", 14: "OLK", 15: "ZLK", 16: "VYS", 17: "PAK", 18: "LBK", 19: "KVK"}`
- Odstraňte duplikované záznamy dle identifikačního čísla (*p1*).
- Při povoleném výpisu (`verbose == True`) spočítejte kompletní (hlubokou) velikost všech sloupců v datovém rámci po vaší úpravě a vypište na standardní výstup pomocí funkce print následující řádek
  `new_size=X MB`
  Čísla vypisujte na 1 desetinné místo a počítejte, že 1 MB = 10^6 B.

Tipy: Převod do formátu data provedete funkcí `pd.to_datetime`

Upozornění: Další skripty budou využívat vaši implementaci parsování datového rámce. Bez této implementace není možné hodnotit další úkoly.

Úkol 3: Počty nehod podle stavu vozovky (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

```
def plot_state(df: pd.DataFrame, fig_location: str = None,
               show_figure: bool = False):
```

Funkcionalita

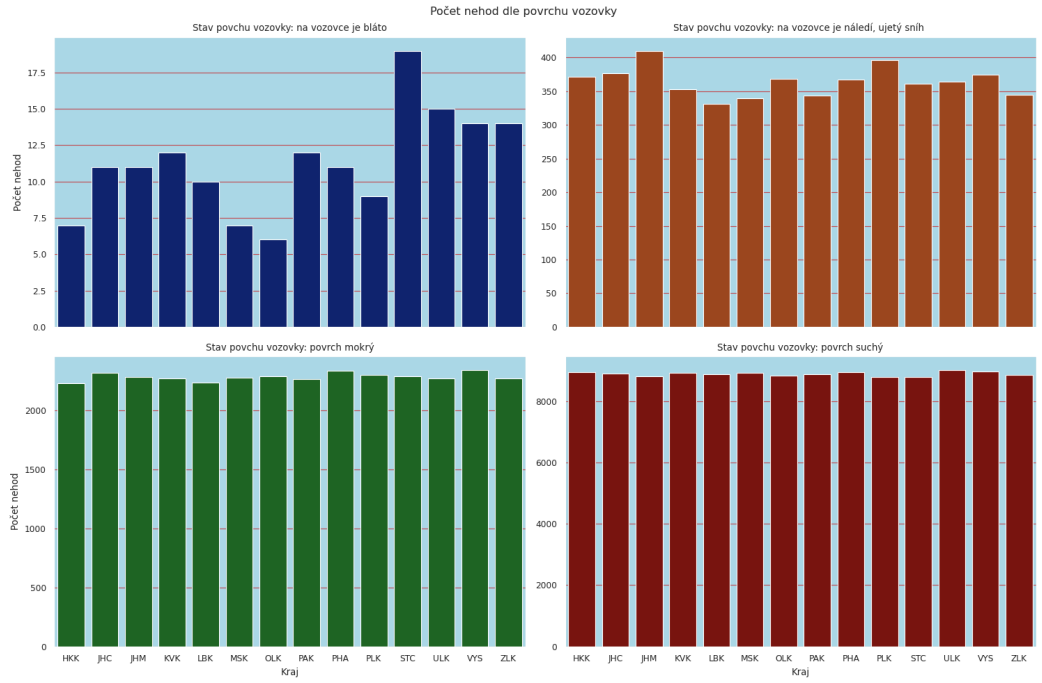
Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud argument `show_figure` je **`True`**. Argument `df` odpovídá DataFrame jenž je výstupem funkce `parse_data` s parametrem ds="nehody". Graf se bude skládat z 4 podgrafů uspořádaných do mřížky o dvou řádcích a dvou sloupcích.

Požadavky:

1) Pracujte se sloupcem *p16*, rozlišujte pouze stavy označené číslem 1 až 6. Je vhodné využít náhrady - používejte pojmenování z první části definice, kategorie 1 a 2 jsou stejné, 5 a 6 také.
2) Nastavte správně titulky jednotlivých podgrafů.
3) Upravte zobrazení tak, aby se grafy a popisky nepřekrývaly.
4) Graf upravte tak, aby popisky na osách, titulky atd. dávaly smysl. Dle zásad dobré vizualizace (viz přednáška 4) zvolte vhodnou barvu a vhodný styl. U podgrafu nastavte vlastní pozadí.

Tip: nejdříve je vhodné si nahradit celočíselné hodnoty ve sloupci *p16* vhodnými řetězci a vytvořit nějaký *pomocný* sloupec, který potom budete sčítat při agregaci `groupby`. Vhodným figure-level grafem pak můžete tato data vizualizovat.

Příklad výstupu: (použita náhodná data a záměrně zcela nevhodná grafická forma)



Úkol 4: Alkohol a následky v jednotlivých krajích (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

```
def plot_alcohol(df: pd.DataFrame, df_consequences : pd.DataFrame,
                 fig_location: str = None, show_figure: bool = False):
```

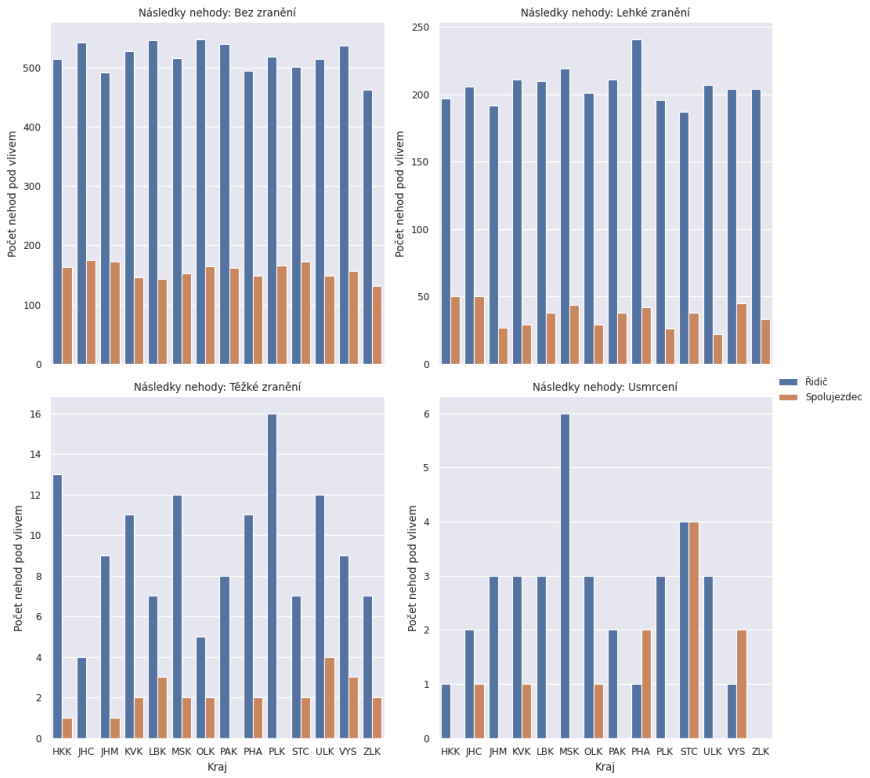
Funkcionalita

Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud `show_figure` je `True`. Argument `df` odpovídá DataFrame jenž je výstupem funkce `parse_data` s parametrem ds="nehody" a `df_consequences` s parametrem ds="nasledky". Jednotlivé dataframy je potřeba spojit podle indexu *p1* (vazba 1:n). Započítavejte pouze nehody pod vlivem (*p11* >= *3*). Dále pro jednotlivé druhy zranění (*p59g*) zobrazte samostatný graf, kolik bylo následků pro řidiče či spolujezdce (všechny) (*p59a*) v jednotlivých krajích. Nehody, kdy čas není znám, nezapočítávejte.

Požadavky a doporučený postup:

1) Spojte dataframy
2) Určete podle *p59a,* zda byl zraněn spolujezdec (!=1) či řidič (1)
3) Určete si vhodným způsobem, zda byl v nehodě přítomný alkohol.
4) Určete úroveň zranění.
5) Data správně agregujte pomocí `groupby` a vykreslete vhodným figure-level grafem.
6) Upravte zobrazení tak, aby se grafy a popisky nepřekrývaly.
7) Graf upravte tak, aby popisky na osách, titulky atd. dával smysl.

Příklad výstupu: (použita náhodná data)



Úkol 5: Druh nehody v čase (až 5 bodů)

Signatura funkce, která bude odpovídat tomuto úkolu:

```
def plot_type(df: pd.DataFrame, fig_location: str = None,
              show_figure: bool = False):
```

Funkcionalita

Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud `show_figure` je **`True`**. Argument `df` odpovídá DataFrame jenž je výstupem funkce `parse_data` s parametrem ds="nehody". Pro čtyři vámi vybrané kraje pro různé měsíce vykreslete čárový graf, který bude zobrazovat pro jednotlivé měsíce (osa X- sloupec *date*) počet nehod podle druhu srážky (*p6*). Uvažujte pouze srážky, ne nehody či jiné.

Požadavky a doporučený postup:

1. Vyberte čtyři kraje a vyfiltrujte všechny nehody.
2. Na základě druhu nehody (p6) určete textovou hodnotu.
3. Transformujte tabulku tak, aby pro každý den a region byl v každém sloupci odpovídajícím srážkám počet nehod (`pivot_table`)
4. Pro každý kraj proveďte podvzorkování na úroveň měsíců a převeďte správně do *stacked formátu*.
5. Vykreslete čárový graf a omezte osu X od 1. 1. 2023 do 1. 10. 2024.
6. Vykreslete patřičné grafy upravené tak, aby popisky na osách, titulky atd. dávaly smysl.

Příklad výstupu: (použita náhodná data)

Poznámky k implementaci

Soubor, který vytvoříte, bude při hodnocení importovaný a budou volány jednotlivé funkce. Mimo tyto funkce, část importů a dokumentační řetězce nepište žádný funkční kód. Blok na konci souboru ohraničený podmínkou

```
if __name__ == "__main__":
    pass
```

naopak můžete upravit libovolně pro testovací účely. Dále můžete přidat další funkce (pokud budete potřebovat), pro názvy těchto funkcí použijte prefix "_".

Stručnou dokumentaci všech částí (souboru a funkcí) uveďte přímo v odevzdaných souborech. Respektuje konvenci pro formátování kódu PEP 257 [PEP 257 -- Docstring Conventions] a PEP 8 [PEP 8 -- Style Guide for Python Code].

Grafy by měly splňovat všechny náležitosti, které u grafu očekáváme, měl by být přehledný a jeho velikost by měla být taková, aby se dal čitelně použít v šířce A4 (t.j. cca 18 cm). Toto omezení není úplně striktní, ale negenerujte grafy, které by byly přes celý monitor.

Grafy v zadání jsou pouze ukázkové. Data byla randomizována a vaše výsledky budou vypadat jinak. Není nutné ani chtěné, aby grafy vizuálně vypadaly stejně - vlastní invence směrem k větší přehlednosti a hezčímu vzhledu se cení! Co není přímo specifikováno v zadání můžete vyřešit podle svého uvážení.

Doporučený postup není nutné dodržet. Důležité je však to, aby výsledky odpovídaly zadání (včetně podvýběru dat a podobně). U argumentů funkcí `fig_location` můžete počítat s tím, že adresář, kam se mají data ukládat, již existuje.

Dotazy a připomínky

Dotazy a připomínky směřujte na fórum v IS VUT případně na mail mrazek@fit.vutbr.cz.