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 CSV souboru stažena a jsou k dispozici na stránce: ehw.fit.vutbr.cz/izv/data.zip. Popis dat naleznete na stránce https://ehw.fit.vutbr.cz/izv. 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.

Odevzdávání a hodnocení

Soubor `analysis.py` odevzdejte do 9. 12. 2022. 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 2 body.

Úkol 1: Načtení dat (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

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

Funkcionalita

- Funkce načte data obsažená v ZIP souboru získaného z adresy http://ehw.fit.vutbr.cz/izv/data.zip . Argument `filename` určuje cestu k souboru. Soubor nestahujte z webu.
- Tento soubor obsahuje ZIP soubory pro každý rok, kdy jsou data rozdělená v rámci krajů (čísla krajů a zkratky naleznete v popisu datového souboru).
- Vaším úkolem je postupně projít všechny potřebné soubory a roky a načíst data do jednoho DataFrame.
- Přidejte jeden sloupec *region* obsahující třípísmennou zkratku daného regionu.
- 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, …) + sloupec *region*.

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()`.
- 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*)
- S výjimkou sloupce region reprezentujte vhodné sloupce pomocí kategorického datového typu. Měli byste se dostat pod velikost v paměti 0.5 GB. Sloupec *region* je vhodné ponechat v původní podobě pro lepší práci s figure-level funkcemi Seaborn.
- Sloupce obsahující čísla s desetinnou čárkou reprezentujte jako float (zejména slouce *d* a *e*).
- Odstraňte duplikované záznamy dle identifikačního čísla (*p1*).
- Při povoleném výpisu (`verbose == True`) spočítejte kompletní (hlubokou) velikost všech sloupců v datovém rámci před a po vaší úpravě a vypište na standardní výstup pomocí funkce print následující 2 řádky
  `orig_size=X MB`
  `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 viditelnosti (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

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

Funkcionalita

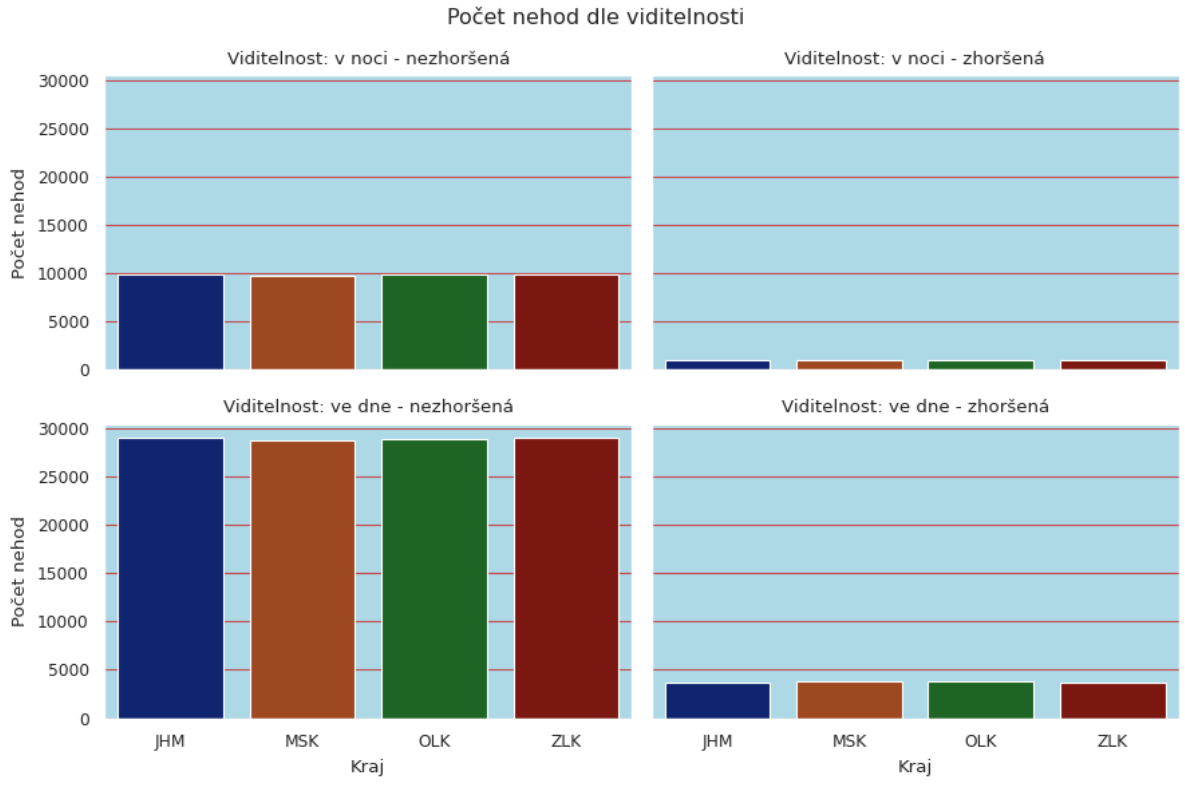
Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud argument `show_figure` je **`True`**. Argument `df` odpovídá DataFrame jenž je výstupem funkce `parse_data`. Graf se bude skládat z 4 podgrafů uspořádaných do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

Požadavky:

1) Vyberte si čtyři různé kraje.
2) Pracujte se sloupcem *p19*, rozlišujte pouze viditelnost ve dne / v noci a zhoršenou / nezhoršenou.
3) Nastavte správně titulky jednotlivých podgrafů.
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 *p21* 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: Druh srážky jedoucích vozidel (až 4 body)

Signatura funkce, která bude odpovídat tomuto úkolu:

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

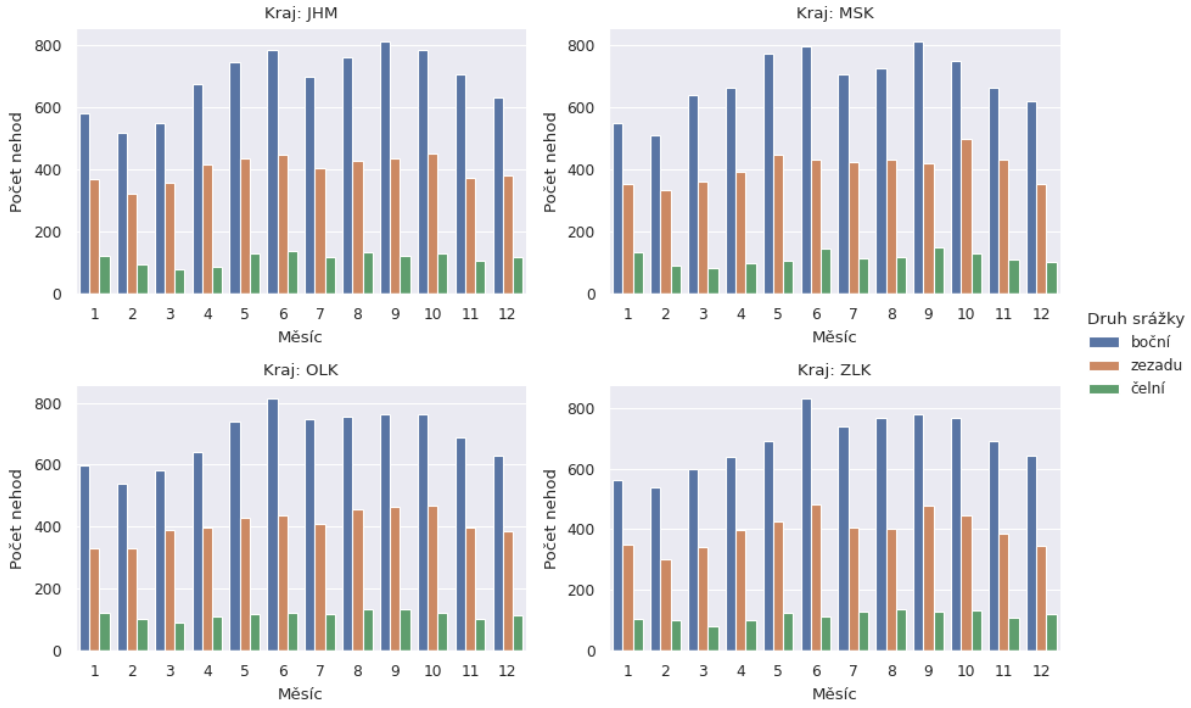
Funkcionalita

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`. Pro čtyři vámi vybrané kraje znázorněte počet nehod jedoucích vozidel *(p7 != 0)*. Určite počty nehod podle směru obou vozidel.

Požadavky a doporučený postup:

1) Druh srážky (p7) konvertujte jako text.
2) Přes `dt` accessor zjistěte měsíc, kdy se stala nehoda.
3) Data správně agregujte pomocí `groupby` a vykreslete vhodným figure-level grafem.
4) Upravte zobrazení tak, aby se grafy a popisky nepřekrývaly.
5) Graf upravte tak, aby popisky na osách, titulky atd. dával smysl.

Příklad výstupu: (použita náhodná data)



Úkol 5: Následky nehod v čase (až 5 bodů)

Signatura funkce, která bude odpovídat tomuto úkolu:

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

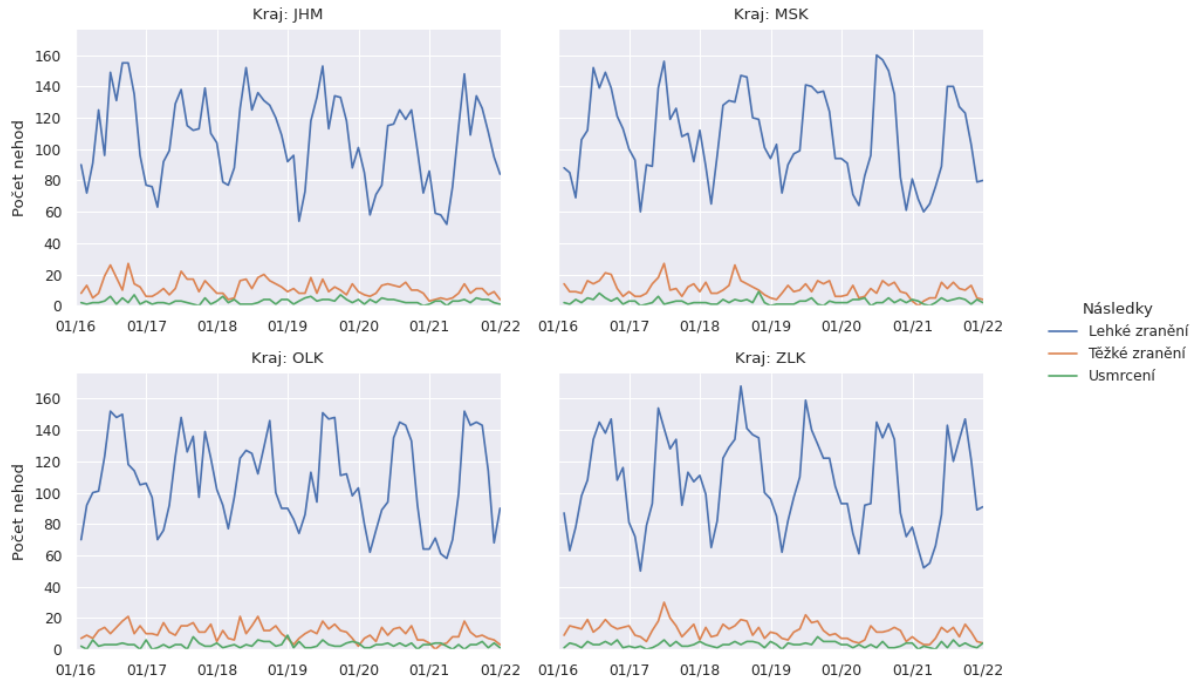
Funkcionalita

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`. Pro čtyři vámi vybrané kraje pro různé měsíce vykreslete čárový graf, který bude zobrazovat pro jednotlivé měsíce (osa X- sloupec *date*) počet nehod podle následků (s tím, že počítáte ten nejhorší následek).

Požadavky a doporučený postup:

1. Vyberte čtyři kraje a vyfiltrujte všechny nehody.
2. Na základě hodnot ze sloupců *p13a, p13b, p13c* určete ke každé nehodě textovou reprezentaci následky.
3. Transformujte tabulku tak, aby pro každý den a region byl v každém sloupci odpovídajícím následkům 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. 2016 do 1. 1. 2022.
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.