Wykład Specjalizujący

Zadanie 1

Czyszczenie, weryfikacja poprawności oraz reformatowanie danych

Autor:

Jerzy Dębowski 151266

Plik z danymi

Do zadania wykorzystałem plik zawierający dane odnośnie punktualności pociągów różnych przewoźników na terenie Polski z lat 2019-2022. Plik posiada 46 wierszy oraz 23 kolumny (rok, miesiąc oraz 21 nazw przewoźników). Wartości w kolumnach to odpowiednio: rok, miesiąc oraz procentowa punktualność danego przewoźnika. Plik został pobrany ze strony https://dane.gov.pl/pl. Na potrzeby zadania plik został zmodyfikowany w następujący sposób:

- z pierwszego wiersza został usunięty rok (początkowo 2019)
- z drugiego wiersza został usunięty miesiąc (początkowo “luty”)
- z trzeciego wiersza została usunięta wartość w kolumnie "Arriva” (początkowo “97,91 %”)
- w trzecim wierszu w kolumnie “WKD” została przypisana wartość “sto procent” (początkowo ”99,16 %”)

Załadowanie pliku

Do załadowania pliku została użyta funkcja read_table z biblioteki pandas, dla której przekazujemy argumenty, tj. nazwa pliku oraz delimiter. DataFrame został przypisany do atrybutu values:

```
file = FileReader( file_name: 'Punktualność_pasażerska_przewoźnicy_w_2022_r._csv. (1).csv', delimiter: ';')
```

```
1 usage  new *
class FileReader(object):

    new *
    def __init__(self, file_name: str, delimiter: str) -> None:
        df = pd.read_table(file_name, delimiter=delimiter)
        self.values = df
```

Czyszczenie danych

Czyszczenie danych w pliku polega na usunięciu z DataFrame wierszy,w których brak jest informacji o danym roku lub miesiącu, ponieważ w żaden sposób nie możemy zweryfikować okresu, z którego takie dane pochodzą:

```
self.values = self.values.dropna(subset=['rok', 'miesiąc'])
```

Weryfikacja poprawności danych

W celu weryfikacji poprawności danych zostały sprawdzone wszystkie kolumny oprócz kolumny z rokiem oraz miesiącem. Wartości w tych kolumnach powinny mieć schemat: liczba – ewentualny przecinek – ewentualna liczba – znak %. W tym celu zostało stworzone wyrażenie regularne. Dane z kolumn przekazywane są do odpowieniej funkcji, która sprawdza sposób zapisu wartości. Jeżeli wartość nie spełnia wymagania, to jest nadpisywana przez wartość None, która w następnych etapach będzie odpowiednio przetworzona.

```
def _check_values(self) -> None:
    for column in self.values.columns:
        if column not in ['rok', 'miesiąc']:
            self.values[column] = self.values[column].astype(str).apply(process_column)
```

```
def process_column(value) -> Any:
    regex_pattern = r'^[0-9]*([,.][0-9]+)?%?$'
    return value if value and re.match(regex_pattern, value) else None
```

Reformatowanie danych

Reformatowanie danych odbędzie się dla wartości we wszystkich kolumnach oprócz kolumny “rok”. Wartości w kolumnie miesiąc zostaną zamienione z nazwy miesiąca na jego numer, co ułatwi odnoszenie się w systemie do konkretnej daty:

```
def _month_name_to_month_number(self) -> None:
    month_dict = {
        'styczeń': 1,
        'luty': 2,
        'marzec': 3,
        'kwiecień': 4,
        'maj': 5,
        'czerwiec': 6,
        'lipiec': 7,
        'sierpień': 8,
        'wrzesień': 9,
        'październik': 10,
        'listopad': 11,
        'grudzień': 12,
    }
    self.values['miesiąc'] = self.values['miesiąc'].str.casefold().map(month_dict)
```

Wartości w pozostałych kolumnach zostaną zamienione z tekstów w postaci “50%” na postać zmienno-przecinkową 0.5, ponieważ jeżeli chcemy użych tych danych do obliczeń lub prezentacji na wykresach muszą być to dane liczbowe, a nie tekstowe:

```
def _percentage_to_number(self) -> None:
    for column in self.values.columns:
        if column not in ['rok', 'miesiąc']:
            self.values[column] = self.values[column].str.replace(',', '.').str.replace('%', '').astype(float) / 100
```

Ponowne czyszczenie danych

Na końcu w kolumnach, w których wcześniej przypisywaliśmy wartości None, jeżeli nie pasowały one do podanego schematu zapisu przypisujemy średnie wartości z danej kolumny. Będą to w miarę wiarygodne wartości, ponieważ w zbiorze danych jest wystarczająco dużo wpisów.

```
def _fill_na_values(self) -> None:
    for column in self.values.columns:
        if column not in ['rok', 'miesiąc']:
            mean = self.values[column].mean()
            self.values[column].fillna(mean, inplace=True)
```

Podsumowanie

Pierwsze 5 wierszy przed obróbką:

	rok	miesiąc	Arriva	...	SKPL Cargo	UBB	WKD
0	NaN	styczeń	96,50%	...	83,06%	98,09%	99,64%
1	2019.0	NaN	98,26%	...	91,07%	99,45%	99,06%
2	2019.0	marzec	NaN	...	98,39%	98,92%	sto procent
3	2019.0	kwiecień	96,82%	...	93,33%	98,29%	99,20%
4	2019.0	maj	96,45%	...	88,24%	97,92%	99,96%

Pierwsze 5 wierszy po obróbce:

	rok	miesiąc	Arriva	...	SKPL Cargo	UBB	WKD
2	2019.0	3	0.9583	...	0.9839	0.9892	0.994876
3	2019.0	4	0.9682	...	0.9333	0.9829	0.992000
4	2019.0	5	0.9645	...	0.8824	0.9792	0.999600
5	2019.0	6	0.9334	...	0.9367	0.9599	0.998700
6	2019.0	7	0.9589	...	0.8961	0.9613	0.997900

Można zauważyć, że wiersze, w których brakowało roku lub miesiąca zostały usunięte, w wierszu gdzie nie było wartości w kolumnie “Arriva” została przypisana średnia o wartości 0.9583, a w kolumnie “WKD”, w której wartość nie pasowała do schematu została przypisana średnia 0.994876. Ponadto nazwy miesięcy zostały zamienione na ich numery, a wartości procentowe w postaci tekstu zostały zastapione wartościami zmienno-przecinkowymi. Po wszystkich wymienionych wyżej operacjach nasz plik z danymi jest gotowy do załadowania i pracy na nim.