Московский государственный технический университет им. Н.Э.Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №2 по
дисциплине
«Методы машинного обучения» на
тему
«Изучение библиотек обработки данных»

Выполнила: Ся Бэйбэй

Группы : ИУ5-21М

Москва — 2022 г.

1. Цель лабораторной работы

Изучить библиотеки обработки данных Pandas и PandaSQL.

2. Задание

Задание состоит из двух частей.

2.1. Часть 1

Требуется выполнить первое демонстрационное задание под названием «Exploratory data analysis with Pandas» со страницы курса mlcourse.ai.

2.2. Часть 2

Требуется выполнить следующие запросы с использованием двух различных библиотек — Pandas и PandaSQL:

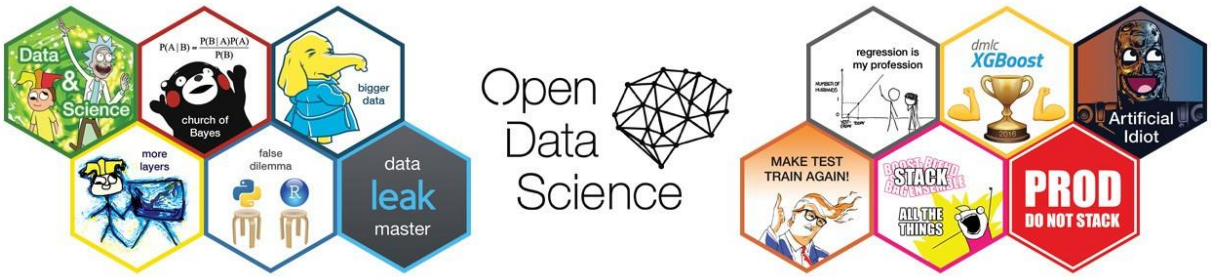
- один произвольный запрос на соединение двух наборов данных,
- один произвольный запрос на группировку набора данных с использованием функций агрегирования.

Также требуется сравнить время выполнения каждого запроса в Pandas и PandaSQL.

3. Ход выполнения работы

3.1. Часть 1

Ниже приведён демонстрационный Jupyter-ноутбук «Exploratory data analysis with Pandas» курса mlcourse.ai (файл `assignment01_pandas_uci_adult.ipynb`). Все пояснения приведены на исходном языке ноутбука — на английском.



mlcourse.ai – Open Machine Learning Course

Author: Yury Kashnitskiy. Translated and edited by Sergey Isaev, Artem Trunov, Anastasia Manokhina, and Yuanyuan Pao

Assignment #1 (demo) Exploratory data analysis with Pandas

In this task you should use Pandas to answer a few questions about the dataset from the traffic police enforcement.

Unique values of all features (for more information, please see the links above):

- `Stop_date`: date.
- `stop_time`: time.
- `driver_gender`: F-female,M-male.
- `driver_age_raw`:date.
- `driver_age`: continuous.
- `driver_race`: `Asian,Black,Hispanic,Other,White`.
- `violation_raw`: `APB, Call for Service, Equipment/Inspection Violation,Motorist Assist /Courtesy,Other Traffic Violation,Registration Violation, Seatbelt Violation,Special Detail/Directed Patrol,Speeding,Suspicious Person,Violation of City/Town Ordinance,Warrant`.
- `violation`: `Equipment,Moving violation,Other,Registration/plates,seat belt,speeding`
- `search_conducted`: `false, true`.
- `Stop_outcome`: `Arrest Driver, Arrest Passenger,Citation,N/D,No Action,Warning`.
- `is_arrested`: `true,false`.
- `stop_duration`: `1,2,0-15min,16-30min,30+min`.
- `drugs_related_stop`: `true,false`.

Importing all required packages:

```
In [11]: import pandas as pd
```

Setting maximum display width for text report [1] & Loading data:

```
In [13]: pd.set_option("display.width", 70)

In [15]: data = pd.read_csv('./police.csv')
         data.head()
```

Out[15]:

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	v
0	2005-01-02	01:55	NaN	M	1985.0	20.0	White	
1	2005-01-18	08:15	NaN	M	1965.0	40.0	White	
2	2005-01-23	23:15	NaN	M	1972.0	33.0	White	
3	2005-02-20	17:15	NaN	M	1986.0	19.0	White	
4	2005-03-14	10:00	NaN	F	1984.0	21.0	White	

data processing:

```
In [19]: data.shape
Out[19]: (91741, 15)

In [18]: data.isnull().sum()
Out[18]: stop_date              0
         stop_time              0
         county_name        91741
         driver_gender       5335
         driver_age_raw      5327
         driver_age          5621
         driver_race         5333
         violation_raw       5333
         violation           5333
         search_conducted       0
         search_type        88545
         stop_outcome        5333
         is_arrested         5333
         stop_duration       5333
         drugs_related_stop     0
         dtype: int64

In [21]: data.drop("county_name", axis = 1, inplace = True)
```

1. **How many men and women (sex feature) are represented in this dataset?**

```
In [20]: data["driver_gender"].value_counts()
Out[20]: M    62895
         F    23511
         Name: driver_gender, dtype: int64
```

What is the average age (age feature) of women?

```
In [22]: data[data["driver_gender"] == "F"]["driver_age"].mean()
Out[22]: 32.60739856801909
```

2. **What is the percentage of the white people (driver_race)?**

```
In [28]: print("{0:%}".format(round((data[data["driver_race"]=="White"].shape[0]/data.shape[0]),2)))
         68.000000%
```

4-5. What are the mean and standard deviation of age for those who violated the law, because they speeded or carried equipment?

```
In [47]: ages1 = data[data["violation"] == "Speeding"]["driver_age"]
         ages2 = data[data["violation"] == "Equipment"]["driver_age"]
         print("Speeding : = {0} ± {1} years".format(ages1.mean(), ages1.std()))
         print("Equipment : = {0} ± {1} years".format(ages2.mean(), ages2.std()))

         Speeding : = 33.53009656541428 ± 12.821847021840082 years
         Equipment : = 31.781502680112656 ± 11.400899609138921 years
```

6. **Is it true that people who is arrested have the results(Arrest Driver, Arrest Passenger) after checked?**

In [96]:
```
out_result = set(["Arrest Driver", "Arrest Passenger"])
def out_resulted(e):return e in out_result

data[data["is_arrested"] == "True"]["stop_outcome"].map(out_resulted).all()
```

Out[96]: True

7. Display age statistics for each race (race feature) and each gender (sex feature).Use groupby() and describe().Find the maximum age of different race.

In [72]:
```
data.groupby(["driver_race", "driver_gender"])["driver_age"].describe()
```

Out[72]:

		count	mean	std	min	25%	50%	75%	max
driver_race	driver_gender								
Asian	F	511.0	31.996086	10.847245	17.0	23.00	29.0	38.5	75.0
	M	1742.0	34.044202	11.886878	17.0	24.25	31.0	42.0	82.0
Black	F	2577.0	31.008925	10.216993	17.0	23.00	28.0	37.0	78.0
	M	9620.0	33.673493	11.748992	15.0	24.00	31.0	41.0	85.0
Hispanic	F	1867.0	30.660418	10.007572	16.0	23.00	28.0	36.0	71.0
	M	7610.0	31.787385	10.682480	15.0	23.00	29.0	38.0	76.0
Other	F	26.0	30.923077	11.520150	20.0	22.25	26.5	35.0	60.0
	M	213.0	34.990610	11.567681	18.0	25.00	33.0	42.0	74.0
White	F	18483.0	33.046205	12.592247	15.0	23.00	29.0	42.0	99.0
	M	43464.0	35.228350	13.470667	15.0	24.00	32.0	45.0	94.0

8. Count the number of Female with/without the "serch_type", which is started with "Incident to Arrest" .

:
```
data["ince_rest"] = data["search_type"].str.startswith("Incident to Arrest")

data[(data["driver_gender"] == "M")]["ince_rest"].value_counts()
```

[47]:
```
False    1419
True     1306
Name: ince_rest, dtype: int64
```

9. What is the minimum age of people, who are checked by police? How many people in this age? And the percentage of white people?

In [77]:
```
m = data["driver_age"].min()
print("Minimum is {} age.".format(m))
people = data[data["driver_age"] == m]
c = people.shape[0]
print("{0} people with age {1} checked by polic.".format(c,m))
s = people[people["driver_race"] == "White"].shape[0]
print("{0:%} whilte people.".format(s / c))
```

```
Minimum is 15.0 age.
5 people with age 15.0 checked by polic.
60.000000% whilte people.
```

10. Separately count the number of people who is arrested or not, although they violated the laws. Separately count the average age of people who is arrested or not, although they violated the laws.

In [80]:
```
pd.crosstab(data["violation"], data["is_arrested"])
```

Out[80]:

violation \ is_arrested	False	True
Equipment	10385	635
Moving violation	15314	910
Other	3978	339
Registration/plates	3098	334
Seat belt	2878	74
Speeding	47826	637

w

In [82]:
```
p=pd.crosstab(data["violation"], data["is_arrested"],values=data['driver_age'], aggfunc="mean")
p
```

Out[82]:

violation \ is_arrested	False	True
Equipment	31.819900	31.154331
Moving violation	36.298683	33.114664
Other	40.110709	32.973373
Registration/plates	32.964759	31.305389
Seat belt	32.242877	30.783784
Speeding	33.571914	30.397174

In [84]:
```
p.loc["Speeding"]
```

Out[84]:
```
is_arrested
False    33.571914
True     30.397174
Name: Speeding, dtype: float64
```

3.2. Часть 2

Импортируем `pandasql`:

```
15]: from pandasql import sqldf pysqldf =
     lambda q: sqldf(q, globals())
```

Для выполнения данного задания возьмём два набора данных из исходных данных[2]:

```
In [23]: item=pd.read_csv("./items.csv")
         sales = pd.read_csv("./sales_train.csv")
```

Посмотрим на эти наборы данных:

```
In [24]: item.dtypes
```

```
Out[24]: item_id              int64
         item_category_id     int64
         dtype: object
```

```
In [25]: sales.dtypes
```

```
Out[25]: date               object
         date_block_num      int64
         shop_id             int64
         item_id             int64
         item_price        float64
         item_cnt_day        int64
         dtype: object
```

Объединим эти наборы данных различными способами, проверяя время их выполнения [3,4,5]:

```
In [26]: sales.merge(item[["item_id","item_category_id"]],on="item_id").head()
```

Out[26]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day	item_category_id
0	12.01.2013	0	25	19	28.0	1	40
1	02.01.2013	0	19	27	2499.0	1	19
2	09.01.2013	0	26	27	2499.0	1	19
3	11.01.2013	0	2	27	2499.0	1	19
4	12.01.2013	0	1	27	1890.0	1	19

```
In [28]: %%timeit
         sales.merge(item[["item_id","item_category_id"]],on="item_id")
```

```
5.4 ms ± 250 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [30]:
```
pysqldf("""SELECT s.date, s.date_block_num, s.shop_id, s.item_id,
s.item_id, s.item_id, s.item_cnt_day, t.item_category_id FROM sales AS s JOIN item AS t
ON s.item_id = t.item_id""").head()
```

Out[30]:

	date	date_block_num	shop_id	item_id	item_id	item_id	item_cnt_day	item_category_id
0	12.01.2013	0	25	19	19	19	1	40
1	02.01.2013	0	19	27	27	27	1	19
2	04.01.2013	0	28	29	29	29	1	23
3	09.01.2013	0	26	27	27	27	1	19
4	11.01.2013	0	2	27	27	27	1	19

In [31]:
```
%%timeit
pysqldf("""SELECT s.date, s.date_block_num, s.shop_id, s.item_id,
s.item_id, s.item_id, s.item_cnt_day, t.item_category_id FROM sales AS s JOIN item AS t
ON s.item_id = t.item_id""")
```

```
27.1 ms ± 609 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Видно, что `pandasql` в 5 раз медленнее, чем `pandas`.

Сгруппируем набор данных с использованием функций агрегирования различными способами:

In [32]:
```
sales.groupby("shop_id")["item_price"].mean().head()
```

Out[32]:
```
shop_id
1    1890.0
2    2499.0
3     249.0
4     398.0
5    1298.0
Name: item_price, dtype: float64
```

In [33]:
```
%%timeit
sales.groupby("shop_id")["item_price"].mean().head()
```

```
893 µs ± 12.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [34]:
```
pysqldf("""SELECT shop_id, AVG(item_price)
FROM sales
GROUP BY shop_id""").head()
```

Out[34]:

	shop_id	AVG(item_price)
0	1	1890.0
1	2	2499.0
2	3	249.0
3	4	398.0
4	5	1298.0

In [35]:
```
%%timeit
pysqldf("""SELECT shop_id, AVG(item_price)
FROM sales
GROUP BY shop_id""").head()
```

```
13.6 ms ± 334 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Здесь разница уже более чем в 15 раз. Таким образом для таких простых запросов проще использовать Pandas.

Список литературы

[1] Analysis Best Practices [Electronic resource] // Kaggle. — 2019. — Access mode: https://www.kaggle.com/melihkanbay/analysis-best-practices/data (online;accessed: 15.03.2022).

[2] You are my Sunshine [Electronic resource] // Kaggle. — 2017. — Access mode: https://www.kaggle.com/gummy2021/visualize-and-analyze-a-russian-it-store/data?select=shops.csv (online; accessed: 15.03.2022).

[3] yhat/pandasql: sqldf for pandas [Electronic resource] // GitHub. — 2017. — Access mode: https://github.com/yhat/pandasql (online; accessed: 22.02.2019).

[4] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/ stable/ (online; accessed: 15.03.2022).

[5] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 15.03.2022)