

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №41

ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

К.т.н., доц.

Е.Л. Турнецкая

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

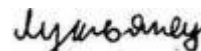
ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

по курсу: Методология и технология проектирования информационных
систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

М320М



П. Е. Лукьянец

подпись, дата

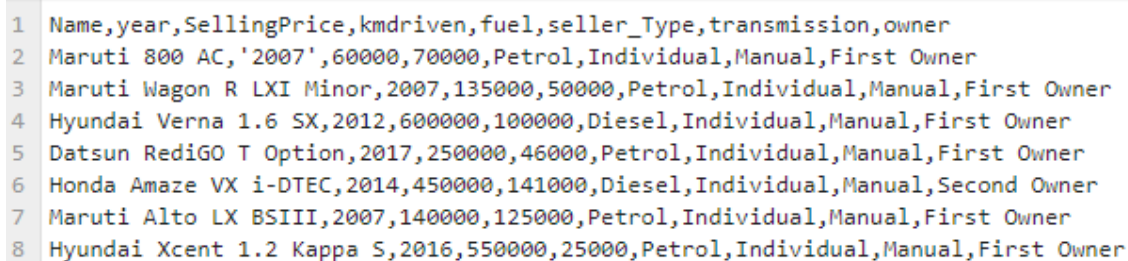
инициалы, фамилия

Санкт-Петербург 2023

Цель работы: осуществить предварительную обработку данных csv-файла, выявить и устранить проблемы в этих данных.

Ход выполнения работы

Для работы был выбран первый датасет из списка под названием «1 auto». В данном датасете представлена информация о продающихся машинах: названии, цене, годе выпуска, трансмиссии и пр. Часть данных этого датасета представлена на рисунке 1.



	Name	year	SellingPrice	kmdriven	fuel	seller_Type	transmission	owner
1	Maruti 800 AC	'2007'	60000	70000	Petrol	Individual	Manual	First Owner
2	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
3	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
4	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
5	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
6	Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner
7	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
8								

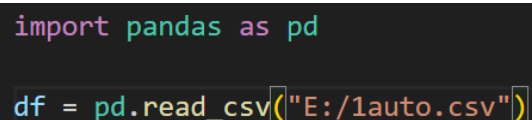
Рисунок 1 – используемый датасет

Для работы с ним использовалась библиотека Pandas.

Работа была выполнена при помощи Visual Studio Code, а также Jupyter Notebook, ссылка на него:

Ссылка на GitHub репозиторий с файлами:

Чтобы начать работу, импортируем библиотеку, а затем считываем CSV файл.

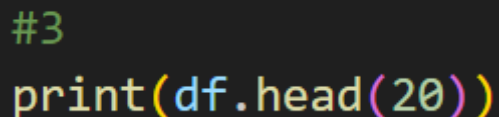


```
import pandas as pd

df = pd.read_csv("E:/1auto.csv")
```

Рисунок 2 – скриншот кода

Выведем первые 20 строк с помощью метода head. На рисунке 3 показан код, а на рисунке 4 – результат его работы.



```
#3
print(df.head(20))
```

Рисунок 3 – скриншот кода

	Name	year	SellingPrice	kmdriven	fuel	seller_Type	transmission	owner
0	Maruti 800 AC	'2007'	60000.0	70000.0	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000.0	50000.0	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000.0	100000.0	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000.0	46000.0	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000.0	141000.0	Diesel	Individual	Manual	Second Owner
5	Maruti Alto LX BSIII	2007	140000.0	125000.0	Petrol	Individual	Manual	First Owner
6	Hyundai Xcent 1.2 Kappa S	2016	550000.0	25000.0	Petrol	Individual	Manual	First Owner
7	Tata Indigo Grand Petrol	2014	240000.0	60000.0	Petrol	Individual	Manual	Second Owner
8	Hyundai Creta 1.6 VTVT S	2015	850000.0	25000.0	Petrol	Individual	Manual	First Owner
9	Maruti Celerio Green VXI	2017	365000.0	78000.0	CNG	Individual	Manual	First Owner
10	Chevrolet Sail 1.2 Base	2015	260000.0	35000.0	Petrol	Individual	Manual	First Owner
11	Tata Indigo Grand Petrol	2014	250000.0	100000.0	Petrol	Individual	Manual	First Owner
12	Toyota Corolla Altis 1.8 VL CVT	2018	1650000.0	25000.0	Petrol	Dealer	Automatic	First Owner
13	Maruti 800 AC	2007	60000.0	70000.0	Petrol	Individual	Manual	First Owner
14	Maruti Wagon R LXI Minor	2007	135000.0	50000.0	Petrol	Individual	Manual	First Owner
15	Hyundai Verna 1.6 SX	2012	600000.0	100000.0	Diesel	Individual	Manual	First Owner
16	Datsun RediGO T Option	2017	250000.0	46000.0	Petrol	Individual	Manual	First Owner
17	Honda Amaze VX i-DTEC	2014	450000.0	141000.0	Diesel	Individual	Manual	Second Owner
18	Maruti Alto LX BSIII	2007	140000.0	125000.0	Petrol	Individual	Manual	First Owner
19	Hyundai Xcent 1.2 Kappa S	2016	550000.0	25000.0	Petrol	Individual	Manual	First Owner

Рисунок 4 – результат вывода

Как уже было сказано, данная таблица содержит информацию о продаваемых автомобилях. Предметная область – автомобили и продажи. Опишем колонки подробнее:

Name – Марка и модель автомобиля

Year – год выпуска

SellingPrice – цена продажи, скорее всего в рублях

Kmdriven – пробег авто

Fuel – тип топлива (бензин, дизель, углеводородный или сжатый газ, электричество)

SellerType – кто продаёт машину (собственник или салон)

Transmission – тип коробки передач (ручная или автоматическая КПП)

Owner – каким по счёту владельцем будет купивший

Теперь с помощью метода «.info» оценим данные. Этот метод возвращает название столбцов, типы данных, количество ненулевых объектов в каждом столбце. Результат работы метода представлен на рисунке 5.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4344 entries, 0 to 4343
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Name             4344 non-null   object
1   year             4344 non-null   object
2   SellingPrice     4342 non-null   float64
3   kmdriven         4343 non-null   float64
4   fuel             4343 non-null   object
5   seller_Type      4344 non-null   object
6   transmission     4344 non-null   object
7   owner            4344 non-null   object
dtypes: float64(2), object(6)
memory usage: 271.6+ KB

```

Рисунок 5 – результат вывода

Теперь выведем на экран названия столбцов с помощью `df.columns`. Как можно увидеть из рисунка 7, некоторые названия столбцов (`SellingPrice`, `kmdriven`, и `seller_Type`) не очень понятны для прочтения, поэтому при помощи метода «`rename`» представленного на рисунке 6, они были переименованы в `selling_price`, `km_driven` и `seller_type`.

```

#6
print(df.columns)
df = df.rename(columns={'SellingPrice': 'selling_price', 'kmdriven':
'km_driven', 'seller_Type': 'seller_type'})

```

Рисунок 6 – скриншот кода

```

Index(['Name', 'year', 'SellingPrice', 'kmdriven', 'fuel', 'seller_Type',
      'transmission', 'owner'],
      dtype='object')

```

Рисунок 7 – результат вывода

Найдём пропуски и устраним их. При помощи метода «`isna`» найдём все пропуски в таблице, а также при помощи `sum` выведем количество пропусков в каждом столбце, результат представлен на верхней половине рисунка 9. Как мы можем увидеть присутствуют пропуски в столбцах цены и пробега. Так как это одни из самых важных столбцов, строки без информации в них не имеют смысла, поэтому их стоит удалить при помощи метода «`dropna`», код представлен на рисунке 8. Проверяем пропуски еще раз и их больше нет.

```
#7
print(df.isna())
print(df.isna().sum())
df = df.dropna(subset=['celling_price', 'km_driven'])
print(df.isna().sum())
print(df.isna())
```

Рисунок 8 – скриншот кода

```
dtype: object
Name      year  celling_price  km_driven  fuel  seller_type  transmission  owner
0      False  False           False      False  False      False      False  False
1      False  False           False      False  False      False      False  False
2      False  False           False      False  False      False      False  False
3      False  False           False      False  False      False      False  False
4      False  False           False      False  False      False      False  False
...      ...   ...           ...        ...   ...        ...        ...   ...
4339   False  False           False      False  False      False      False  False
4340   False  False           False      False  False      False      False  False
4341   False  False           False      False  False      False      False  False
4342   False  False           True       True    False      False      False  False
4343   False  False           True       False   True       False      False  False

[4344 rows x 8 columns]
Name      0
year      0
celling_price  2
km_driven    1
fuel         1
seller_type   0
transmission  0
owner        0
dtype: int64
Name      0
year      0
celling_price  0
km_driven    0
fuel         0
seller_type   0
transmission  0
owner        0
dtype: int64
Name      year  celling_price  km_driven  fuel  seller_type  transmission  owner
0      False  False           False      False  False      False      False  False
1      False  False           False      False  False      False      False  False
2      False  False           False      False  False      False      False  False
3      False  False           False      False  False      False      False  False
4      False  False           False      False  False      False      False  False
...      ...   ...           ...        ...   ...        ...        ...   ...
4337   False  False           False      False  False      False      False  False
4338   False  False           False      False  False      False      False  False
4339   False  False           False      False  False      False      False  False
4340   False  False           False      False  False      False      False  False
4341   False  False           False      False  False      False      False  False

[4342 rows x 8 columns]
```

Рисунок 9 – результат вывода

Проверим данные на наличие дубликатов. Так как уникальным значением в данном наборе является только индекс, то удалить можно лишь

полностью повторяющие строки. Их 764 (см. рис. 11). При помощи метода «drop_duplicates» (см. рис. 10) удаляем дубликаты и проверяем заново.

```
#8
print(df.duplicated().sum())
df = df.drop_duplicates().reset_index()
print(df.duplicated().sum())
df.info()
```

Рисунок 10 – скриншот кода

```
764
0
```

Рисунок 11 – результат вывода

Однако после проверки всех столбцов было выявлено, что в столбце года выпуска 2007 год имеет два вида написания. При помощи метода «replace» (см. рис. 12) удаляем дубликаты и проверяем заново (см. рис. 13).

```
print(df["year"].unique())
df['year'] = df['year'].replace("2007'", "2007")
print(df["year"].unique())
```

Рисунок 12 – скриншот кода

```
["'2007'" '2007' '2012' '2017' '2014' '2016' '2015' '2018' '2019' '2013'
'2011' '2010' '2009' '2006' '1996' '2005' '2008' '2004' '1998' '2003'
'2002' '2020' '2000' '1999' '2001' '1995' '1997' '1992']

['2007' '2012' '2017' '2014' '2016' '2015' '2018' '2019' '2013' '2011'
'2010' '2009' '2006' '1996' '2005' '2008' '2004' '1998' '2003' '2002'
'2020' '2000' '1999' '2001' '1995' '1997' '1992']
```

Рисунок 13 – результат вывода

Проверим все ли типы данных соответствуют действительности. Все столбцы, кроме года выпуска соответствуют своему типу. Поэтому при помощи метода «to_datetime» (см. рис. 14) изменяем тип на временной и проверяем результат (см. рис. 15).

```
#9
df['year'] = pd.to_datetime(df['year'], format='%Y')
df.info()
print(df["year"].unique())
```

Рисунок 14 – скриншот кода

```

---  -----  -----  -----
0   index      3578 non-null  int64
1   Name       3578 non-null  object
2   year       3578 non-null  datetime64[ns]
3   selling_price 3578 non-null  float64
4   km_driven  3578 non-null  float64
5   fuel       3578 non-null  object
6   seller_type 3578 non-null  object
7   transmission 3578 non-null  object
8   owner      3578 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(5)
memory usage: 251.7+ KB
['2007-01-01T00:00:00.000000000' '2012-01-01T00:00:00.000000000'
 '2017-01-01T00:00:00.000000000' '2014-01-01T00:00:00.000000000'
 '2016-01-01T00:00:00.000000000' '2015-01-01T00:00:00.000000000'
 '2018-01-01T00:00:00.000000000' '2019-01-01T00:00:00.000000000'
 '2013-01-01T00:00:00.000000000' '2011-01-01T00:00:00.000000000'
 '2010-01-01T00:00:00.000000000' '2009-01-01T00:00:00.000000000'
 '2006-01-01T00:00:00.000000000' '1996-01-01T00:00:00.000000000'
 '2005-01-01T00:00:00.000000000' '2008-01-01T00:00:00.000000000'
 '2004-01-01T00:00:00.000000000' '1998-01-01T00:00:00.000000000'
 '2003-01-01T00:00:00.000000000' '2002-01-01T00:00:00.000000000'
 '2020-01-01T00:00:00.000000000' '2000-01-01T00:00:00.000000000'
 '1999-01-01T00:00:00.000000000' '2001-01-01T00:00:00.000000000'
 '1995-01-01T00:00:00.000000000' '1997-01-01T00:00:00.000000000'
 '1992-01-01T00:00:00.000000000']

```

Рисунок 15 – результат вывода

Создадим сводную таблицу при помощи метода «data_pivot», код представлен на рисунке 16. Индексацию возьмём по году производства машины, а колонки по трансмиссии (механика или автомат). Подсчёт будет по сумме стоимости. Таким образом, получится таблица, показывающая, на какой год и с какую коробку передач приходится больше всего суммарная стоимость машин. Это машины 17-18 года с автоматической коробкой передач. Это можно увидеть на рисунке 17.

```

#10
data_pivot = df.pivot_table(index=['year'], columns='transmission', values='selling_price', aggfunc='sum')
print(data_pivot)

```

Рисунок 16 – скриншот кода

transmission	Automatic	Manual
year		
1992-01-01	NaN	50000.0
1995-01-01	NaN	95000.0
1996-01-01	NaN	450000.0
1997-01-01	79000.0	200000.0
1998-01-01	1000000.0	486000.0
1999-01-01	NaN	665000.0
2000-01-01	NaN	978000.0
2001-01-01	NaN	1752999.0
2002-01-01	NaN	1550000.0
2003-01-01	95000.0	1821000.0
2004-01-01	350000.0	4464499.0
2005-01-01	NaN	7004108.0
2006-01-01	2855000.0	11483997.0
2007-01-01	2030000.0	17367999.0
2008-01-01	2080000.0	20839194.0
2009-01-01	5963000.0	33830994.0
2010-01-01	11320000.0	46552683.0
2011-01-01	13368999.0	59749989.0
2012-01-01	25383000.0	102837985.0
2013-01-01	26830000.0	102217793.0
2014-01-01	32548999.0	132075986.0
2015-01-01	24685999.0	144650986.0
2016-01-01	40837999.0	139812987.0
2017-01-01	82063999.0	174385986.0
2018-01-01	77316000.0	164707990.0
2019-01-01	43638998.0	96549997.0
2020-01-01	5340000.0	30877998.0

Рисунок 17 – результат вывода

Вывод: Таким образом, в ходе выполнения лабораторной работы был выбран и описан выбранный датасет про продаваемые автомобили, изучен интерфейс и возможности Jupyter Notebook, изучены базовые функции библиотеки Pandas и разработана программа, которая считывает данные, выводит о них информацию, удаляет дубликаты, пропуски, изменяет тип данных и создаёт сводную таблицу.