

ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

специалист

должность, уч. степень, звание

подпись, дата

В. В. Боженко

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4117

подпись, дата

Д. С. Николаев

инициалы, фамилия

Санкт-Петербург 2023

Цель работы

Осуществить предварительную обработку данных csv-файла, выявить и устранить проблемы в этих данных.

Индивидуальное задание

Вариант 2.

Задание 1 : Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута).

Задание 2 : Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута). Создать датафрейм. Переименовать столбец с количеством в "count". Отсортировать по убыванию столбца "count".

Задание 3: Сводная таблица (pivot_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSEroute). Отсортировать по убыванию столбца MILES. Округлить значение до двух знаков.

Задание 4: Сводная таблица (pivot_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSEroute) - столбцы и каждой категории - строки. Отсортировать по убыванию столбца CATEGORY.

Ход работы

Ссылка на [gitHub](#).

Для начала был загружен датасет через библиотеку Python - pandas. Используем ';' для разделение данных. Код расположен в листинге 1.

Листинг 1 - Код загрузка датасета

```
import pandas
df = pandas.read_csv("drivers.csv", sep = ';')
```

Далее были выведены первые 20 строк через метод df.head(20). На рисунке 1 продемонстрирован результат работы метода.

	START_DATE	END_DATE	CATEGORY*	START	STOP	MILES	PURPOSE	route
0	01.10.2016 19:12	01.10.2016 19:32	Business	Midtown	East Harlem	44963.0	MEETING	
1	01.11.2016 13:32	01.11.2016 13:46	Business	Midtown	Midtown East	45108.0	Meal/Entertain	
2	01.12.2016 12:33	01.12.2016 12:49	Business	Midtown	Hudson Square	45170.0	Meal/Entertain	
3	1.13.2016 15:00	1.13.2016 15:28	Business	Gulfton	Downtown	45149.0	Meeting	
4	1.29.2016 21:21	1.29.2016 21:40	Business	Apex	Cary	45051.0	Meal/Entertain	
5	1.30.2016 18:09	1.30.2016 18:24	Business	Apex	Cary	45112.0	Customer Visit	
6	02.01.2016 12:10	02.01.2016 12:43	Business	Chapel Hill	Cary	45008.0	Customer Visit	
7	02.04.2016 9:37	02.04.2016 10:09	Business	Morrisville	Cary	45116.0	Meal/Entertain	
8	02.07.2016 18:03	02.07.2016 18:17	Business	Apex	Cary	45112.0	Customer Visit	
9	02.07.2016 20:22	02.07.2016 20:40	Business	Morrisville	Cary	44932.0	Meeting	
10	02.09.2016 20:24	02.09.2016 20:40	Business	Morrisville	Cary	44932.0	Meal/Entertain	
11	02.11.2016 20:36	02.11.2016 20:51	Business	Morrisville	Cary	44932.0	Temporary Site	
12	02.12.2016 11:14	02.12.2016 11:35	Business	Morrisville	Raleigh	17.0	Customer Visit	
13	02.12.2016 15:33	02.12.2016 16:06	Business	Morrisville	Cary	45057.0	Customer Visit	
14	2.14.2016 14:46	2.14.2016 15:03	Business	Midtown	Midtown West	2.0	Meeting	
15	2.16.2016 10:31	2.16.2016 10:41	BUSINESS	Colombo	Colombo	45079.0	NaN	
16	2.16.2016 11:32	2.16.2016 12:02	Business	Colombo	Colombo	45050.0	NaN	
17	2.16.2016 12:39	2.16.2016 12:42	Business	Colombo	Colombo	45108.0	NaN	
18	2.16.2016 13:43	2.16.2016 13:55	BUSINESS	Colombo	Colombo	45139.0	Temporary Site	
19	2.16.2016 16:34	2.16.2016 17:10	Business	Colombo	Colombo	6.0	NaN	

Рисунок 1 - Вывод первых 20 элементов

Столбцы:

1. START_DATE и END_DATE: Эти столбцы содержат дату и время начала и завершения поездки. Они указывают, когда началась и завершилась каждая поездка.

2. CATEGORY: Этот столбец обозначает категорию поездки. В данном случае, больше поездок "Business", что указывает на то, что она связана с деловой деятельностью.

3. START и STOP: Эти столбцы указывают начальное и конечное местоположение поездки. Они указывают, откуда и куда направлялась поездка.

4. MILES: Этот столбец содержит информацию о количестве миль, пройденных во время поездки. Это может быть полезным для расчета расстояний и затрат на топливо.

5. PURPOSE: Этот столбец описывает цель поездки. Он указывает, для чего была совершена поездка, например, "MEETING" (встреча), "Meal/Entertain" (питание/развлечение), "Customer Visit" (визит к клиенту) и т. д.

Предметная область этой таблицы связана с учетом и анализом служебных поездок. Возможно, она используется для отслеживания затрат на бизнес-поездки, анализа расходов на топливо, определения целей и местоположение поездок, а также для управления служебными маршрутами и встречами.

Далее с помощью команды `df.info()` была выведена информацию о DataFrame (Рисунок 2).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161 entries, 0 to 160
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   START_DATE      161 non-null   object
1   END_DATE        161 non-null   object
2   CATEGORY*       161 non-null   object
3   START           161 non-null   object
4   STOP            161 non-null   object
5   MILES           161 non-null   float64
6   PURPOSEroute    84 non-null    object
dtypes: float64(1), object(6)
memory usage: 8.9+ KB
```

Рисунок 2 - Информация о DataFrame

Результат выполнения метода `info()` показывает следующую информацию о DataFrame:

1. Всего строк: 161.
2. Количество непустых (non-null) значений для каждого столбца.
3. Типы данных для каждого столбца.

Из этой информации видно следующее:

- Столбцы "START_DATE", "END_DATE", "CATEGORY*", "START", "STOP" и "PURPOSEroute" содержат объекты (строки).
- Столбец "MILES" содержит числовые значения с типом данных float64.
- Столбец "PURPOSEroute" имеет некоторые пропущенные значения, так как количество непустых значений меньше общего числа строк.

С помощью команды `df.describe()` были оценены числовые столбцы (Рисунок 3).

MILES	
count	161.000000
mean	37766.519255
std	16614.925558
min	0.800000
25%	44931.000000
50%	45008.000000
75%	45081.000000
max	45177.000000

Рисунок 3 - Числовые столбцы

Результат выполнения метода describe() для числового столбца "MILES" выглядит следующим образом:

- count: Количество непустых (non-null) значений в столбце - 161.
- mean: Среднее значение - 37766.519255.
- std: Стандартное отклонение - 16614.925558.
- min: Минимальное значение - 0.800000.
- 25%: 25-й перцентиль - 44931.000000.
- 50%: Медианное значение (50-й перцентиль) - 45008.000000.
- 75%: 75-й перцентиль - 45081.000000.
- max: Максимальное значение - 45177.000000.

Эти статистические показатели позволяют оценить основные характеристики числового столбца "MILES", такие как среднее значение, разброс данных (стандартное отклонение), минимальное и максимальное значения, а также квантили, что может быть полезно при анализе данных и поиске выбросов.

Далее были выведены на экран названия столбцов с помощью df.columns (Рисунок 4).

```
Index(['START_DATE', 'END_DATE', 'CATEGORY*', 'START', 'STOP', 'MILES',
      'PURPOSEroute'],
      dtype='object')
```

Рисунок 4 - Вывод названия столбцов

Проблема заключается в том, что названия не имеют единого формата. Для этого были переименованы столбцы. Код продемонстрирован в листинге 2, результат — на рисунке 5.

Листинг 2 - Код переименовки столбцов

```
df.columns = ['START_DATE', 'END_DATE', 'CATEGORY',  
'START', 'STOP', 'MILES', 'PURPOSE_ROUTE']  
df.columns
```

```
Index(['START_DATE', 'END_DATE', 'CATEGORY', 'START', 'STOP', 'MILES',  
      'PURPOSE_ROUTE'],  
      dtype='object')
```

Рисунок 5 - Изменения названия столбцов

С помощью метода `fillna()` были заменены пропуски на 'Unknown' (Листинг 3).

Листинг 3 - Код замены пропусков

```
df['PURPOSE_ROUTE'].fillna('Unknown', inplace=True)
```

Результат работы данного кода продемонстрирован на рисунке 6.

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 161 entries, 0 to 160  
Data columns (total 7 columns):  
#   Column          Non-Null Count  Dtype  
---  ---  
0   START_DATE      161 non-null   object  
1   END_DATE        161 non-null   object  
2   CATEGORY        161 non-null   object  
3   START           161 non-null   object  
4   STOP            161 non-null   object  
5   MILES           161 non-null   float64  
6   PURPOSE_ROUTE   161 non-null   object  
dtypes: float64(1), object(6)  
memory usage: 8.9+ KB
```

Рисунок 6 - Замена пропусков

Далее была проведена проверка на явные дубликаты с помощью метода `df[df.duplicated()]` (Рисунок 7).

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE_ROUTE
159	7.26.2016 22:31	7.26.2016 22:39	Business	Morrisville	Cary	45048.0	Meal/Entertain
160	7.26.2016 22:31	7.26.2016 22:39	Business	Morrisville	Cary	45048.0	Meal/Entertain

Рисунок 7 - Проверка на дубликаты

На рисунке 7 видно, что есть еще две строки идентичные другой, избавимся от них с помощью `drop_duplicates(inplace=True)`, после проверим на дубликаты еще раз. На рисунке 8 видно, что больше дубликатов нет.

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE_ROUTE
--	------------	----------	----------	-------	------	-------	---------------

Рисунок 8 - Результат удаление дубликатов

Проверка на неявные дубликаты (различные написания одного и того же) при помощи функции `df['CATEGORY'].unique()` для столбцов с текстовым типом данных (Рисунок 9).

Для начала проверен столбец CATEGORY.

```
df['CATEGORY'].unique()
array(['Business', 'BUSINESS', 'Personal'], dtype=object)
```

Рисунок 9 - Просмотр на наличие уникальных значение

На рисунке 9 видим, что Business пишется в двух вариациях. С помощью `replace()` значения были изменены на общий вид (Листинг 4).

Листинг 4 - Код замены значений.

```
df['CATEGORY'] = df['CATEGORY'].replace('BUSINESS',
'Business')
```

Результат работы продемонстрирован на рисунке 10.

```
df['CATEGORY'] = df['CATEGORY'].replace('BUSINESS', 'Business')
df['CATEGORY'].unique()
array(['Business', 'Personal'], dtype=object)
```

Рисунок 10 - Замена неявных дубликатов

Тоже самое было проделано и для столбца PURPOSE_ROUTE, другие не рассматривались, так как это или дата, или название (Рисунок 11).

```

df['PURPOSE_ROUTE'].unique()
array(['MEETING', 'Meal/Entertain', 'Meeting', 'Customer Visit',
       'Temporary Site', 'Unknown', 'Moving'], dtype=object)

df['PURPOSE_ROUTE'] = df['PURPOSE_ROUTE'].replace('MEETING', 'Meeting')
df['PURPOSE_ROUTE'].unique()
array(['Meeting', 'Meal/Entertain', 'Customer Visit', 'Temporary Site',
       'Unknown', 'Moving'], dtype=object)

```

Рисунок 11 - Замена неявных дубликатов для другого столбца

У столбцов с датой был изменен тип данных на datetime (Листинг 5).

Листинг 5 - Код изменения типа данных

```

df['START_DATE'] = pandas.to_datetime(df['START_DATE'],
format='%m.%d.%Y %H:%M')
df['END_DATE'] = pandas.to_datetime(df['END_DATE'],
format='%m.%d.%Y %H:%M')
df.dtypes

```

Результат работы продемонстрирован на рисунке 12.

```

...  START_DATE      datetime64[ns]
     END_DATE      datetime64[ns]
     CATEGORY      object
     START         object
     STOP          object
     MILES         float64
     PURPOSE_ROUTE object
     dtype: object

```

Рисунок 12 - Изменение типов данных

Далее были выполнены индивидуальные задания.

Задание 1

Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута). Код продемонстрирован в листинге 6.

Листинг 6 - Код 1 задания

```
df.groupby([ 'CATEGORY', 'PURPOSE_ROUTE' ] )  
[ 'PURPOSE_ROUTE' ].count( )
```

Данный код подсчитывает с какой целью часто ездят, исходя из категории. Видно, что в категории деловых поездок, наиболее популярная цель поездки - обед, а в категории личных поездок - встреча (не считая пустых ячеек). Результат работы продемонстрирован на рисунке 13.

```
... CATEGORY PURPOSE_ROUTE  
Business Customer Visit      30  
          Meal/Entertain     34  
          Meeting             13  
          Temporary Site      4  
          Unknown             67  
Personal  Moving              1  
          Unknown             10  
Name: PURPOSE_ROUTE, dtype: int64
```

Рисунок 13 - Результат задания 1

Этот анализ помогает понять, какие категории и цели маршрутов наиболее распространены или часто встречаются в данных, что может быть полезной информацией для принятия бизнес-решений или дальнейшего анализа.

Задание 2

Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута). Создать датафрейм. Переименовать столбец с количеством в “count”. Отсортировать по убыванию столбца “count”. Код продемонстрирован в листинге 7.

Листинг 7 - Код задания 2

```
df1 = df.groupby([ 'CATEGORY', 'PURPOSE_ROUTE' ] )  
[ 'PURPOSE_ROUTE' ].count().reset_index(name  
='count').sort_values('count', ascending=False)  
df1
```

Создаем группировку как в предыдущем задании, столбец с получаемым значением называем - count и применяем функцию sort_values для данного столбца по убыванию. Результат работы продемонстрирован на рисунке 14.

	CATEGORY	PURPOSE_ROUTE	count
4	Business	Unknown	67
1	Business	Meal/Entertain	34
0	Business	Customer Visit	30
2	Business	Meeting	13
6	Personal	Unknown	10
3	Business	Temporary Site	4
5	Personal	Moving	1

Рисунок 14 - Результат задания 2

Из выведенных данных можно заметить, что очень часто встречается неизвестная цель поездки, для более грамотного учета данных, следует обязать заполнять эти данные.

Анализ таких данных может помочь определить, какие цели маршрута наиболее популярны или наименее популярны в каждой из категорий поездок. Например, эти данные могут быть полезны для бизнес-аналитики при определении, на какие виды мероприятий или услуг следует сосредотачивать усилия или ресурсы.

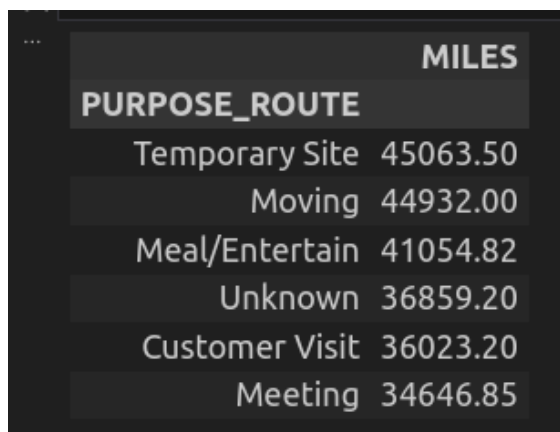
Задание 3

Сводная таблица (pivot_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSE_ROUTE). Отсортировать по убыванию столбца MILES. Округлить значение до двух знаков. Код продемонстрирован в листинге 8.

Листинг 8 - Код задания 3

```
df_pivot = pandas.pivot_table(df, values='MILES',
index='PURPOSE_ROUTE', aggfunc='mean')
df_pivot['MILES'] = df_pivot['MILES'].round(2)
df_pivot.sort_values(by='MILES', ascending=False,
inplace=True)
df_pivot
```

Для начала создадим сводную таблицу, используя - pivot_table. Далее округлим столбец MILES до двух знаков после запятой и отсортируем этот же столбец по убыванию. Результат работы продемонстрирован на рисунке 15.



PURPOSE_ROUTE	MILES
Temporary Site	45063.50
Moving	44932.00
Meal/Entertain	41054.82
Unknown	36859.20
Customer Visit	36023.20
Meeting	34646.85

Рисунок 15 - Результат задания 3

Самая большая средняя длина поездки (около 45063.50 миль) наблюдается для цели маршрута "Temporary Site". Это может указывать на длительные поездки, связанные с временными рабочими объектами или местоположениями.

Анализ средней длины поездок по целям маршрута может быть полезным при определении того, какие типы поездок являются наиболее долгими или короткими, что может иметь значение для планирования и оценки ресурсов или стоимости поездок.

Задание 4

Сводная таблица (pivot_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSE_ROUTE) - столбцы и каждой категории - строки. Отсортировать по убыванию столбца CATEGORY. Код продемонстрирован в листинге 9.

Листинг 9 - Код задания 4

```
df_pivot2 = pandas.pivot_table(df,
columns='PURPOSE_ROUTE', values='MILES',
index='CATEGORY', aggfunc='mean')
df_pivot2.sort_values(by='CATEGORY', ascending=False)
df_pivot2
```

Продолжаем ту же самую работу, что и в прошлом задании, но теперь в pivot_table передается массив столбцов. Сортировка же для символьных типов данных идет по ASCII таблице (по алфавитному порядку). Результат работы продемонстрирован на рисунке 16.

PURPOSE_ROUTE	Customer Visit	Meal/Entertain	Meeting	Moving	Temporary Site	Unknown
CATEGORY						
Business	36023.196667	41054.823529	34646.846154	NaN	45063.5	36982.219403
Personal	NaN	NaN	NaN	44932.0	NaN	36035.000000

Рисунок 16 - Результат задания 4

Дополнительное задание

Добавить столбец с длительностью поездки, выбрать только бизнес класс. Составить сводную таблицу с целью маршрута и уго длительностью. Код продемонстрирован в листинге 10.

Листинг 10 - Код дополнительного задания

```
df_pivot3 = df[df["CATEGORY"] == "Business"].copy()
df_pivot3["TIME_IN_ROAD"] = (df_pivot3["END_DATE"] -
df_pivot3["START_DATE"]).dt.total_seconds() / 60
df_pivot3 = pandas.pivot_table(df_pivot3,
values='TIME_IN_ROAD', index='PURPOSE_ROUTE',
aggfunc='mean')
df_pivot3["TIME_IN_ROAD"] =
df_pivot3["TIME_IN_ROAD"].round(2)
df_pivot3
```

Результат работы продемонстрирован на рисунке 17.

	TIME_IN_ROAD
PURPOSE_ROUTE	
Customer Visit	19.40
Meal/Entertain	14.74
Meeting	25.62
Temporary Site	24.75
Unknown	22.18

Рисунок 16 - Результат задания 4

Самое короткое среднее время в пути наблюдается для цели маршрута "Meal/Entertain" (14.74 минут). Это может означать, что поездки, связанные с обедами или развлечениями, обычно не требуют много времени на перемещение.

Анализ таких данных может быть полезным при планировании или оценке времени, затрачиваемого на деловые поездки, и может помочь определить, какие типы поездок часто требуют большего времени в пути.

Вывод

В данной лабораторной работе были рассмотрены основные операции с данными с использованием библиотеки `pandas` в Python. А также использование `jupyter` для оформления. Вот ключевые моменты и шаги, выполненные в рамках лабораторной работы:

- Загрузка данных: Данные были предоставлены в формате текстового файла или CSV. Мы использовали библиотеку `pandas` для загрузки данных в `DataFrame` - удобную структуру данных для анализа и манипуляций с данными.

- Изучение данных: Мы использовали методы `.head()`, `.tail()`, `.info()`, и `.describe()` для первичного изучения данных. Эти методы позволяют нам получить представление о данных, включая типы данных, наличие пропусков и основные статистические показатели.

- Обработка данных: Мы рассмотрели различные методы обработки данных, такие как удаление или замена пропусков с использованием `.dropna()` и `.fillna()`, а также удаление дубликатов с использованием `.drop_duplicates()`.

- Создание сводных таблиц: Мы использовали метод `pivot_table` для создания сводных таблиц, позволяющих агрегировать данные и анализировать их в разных срезах. Мы выполнили сортировку сводной таблицы по убыванию.

- Округление значений: Мы использовали метод `.round()` для округления числовых значений в сводной таблице до двух знаков после запятой.

Эти шаги представляют собой основные операции по обработке и анализу данных с использованием библиотеки `pandas`. Все они могут быть адаптированы и расширены для работы с реальными данными и решения конкретных задач анализа данных. Важно иметь понимание о том, как использовать эти инструменты для эффективной обработки и анализа данных в Python.