ГУАП КАФЕДРА № 41

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

старший преподаватель В. В. Боженко

должность, уч. степень, звание подпись, дата инициалы, фамилия

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

ИССЛЕДОВАТЕЛЬСКИЙ АНАЛИЗ ДАННЫХ

По курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4117 Д. С. Николаев

подпись, дата инициалы, фамилия

Санкт-Петербург 2023

# **Цель работы**

Изучение связи между признаками двумерного набора данных, визуализация данных.

# **Индивидуальное задание**

Вариант 2.

Задание 1: использовать seaborn. По группировке - CATEGORY и количество поездок каждого типа (по цели маршрута - PURPOSE) отфильтровать данные группировки по количеству поездок больше 2 и построить диаграмму.

Задание 2: использовать pandas и plot. По сводной таблице (pivot\_table) -

отобразить среднее количество пройденных миль по каждой цели поездки (PURPOSE). Оставить только маркеры в виде ★ зеленого цвета размеров 18.

Задание 3: использовать matplotlib. Построить круговую диаграмму, которая отображает процент по каждой цели поездки. Уберите из диаграммы количество поездок меньше 5.

# **Ход работы**

Ссылка на [gitHub](https://github.com/TanukiY/DataAnalysis).

Проведена предварительная обработку данных как в 1 лабораторной работе. Загружен датасет через библиотеку Python - pandos. Используем ',' для разделения данных. Код расположен в листинге 1.

Листинг 1 - Код загрузка датасета.

import pandas

df = pandas.read\_csv("drivers2.csv", sep =',')

Затем было выведено на экран названия столбцов с помощью `df.columns`. Результат показан на рисунке 1.

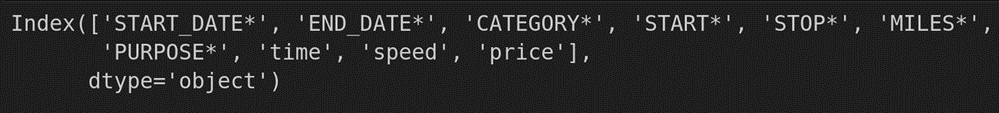


Рисунок 1 — Вывод названия столбцов

Проблема заключается в том, что названия не имеют единого формата. Для этого столбцы были переименованы в едином формате. Код продемонстрирован на листинге 2.

Листинг 2 - Код изменения названий столбцов.

df.columns = ['START\_DATE', 'END\_DATE', 'CATEGORY', 'START', 'STOP', 'MILES', 'PURPOSE\_ROUTE', 'TIME', 'SPEED', 'PRICE']

df.columns

Результат переименования столбцов продемонстрирован на рисунке 2.

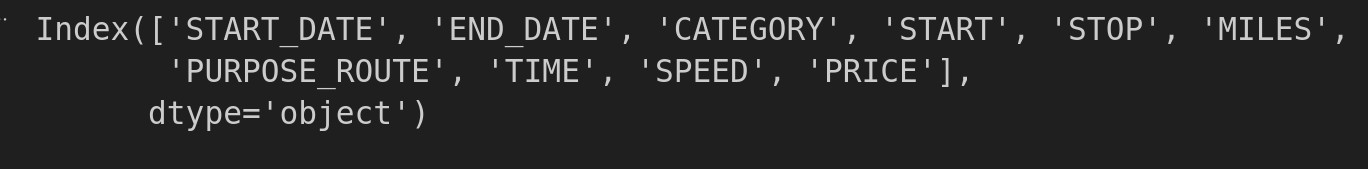


Рисунок 2 — Результат переименования столбцов

Также была проведена проверка на наличие пропусков, используя функцию `df.info()`. Результат данной функции представлен на рисунке 3.

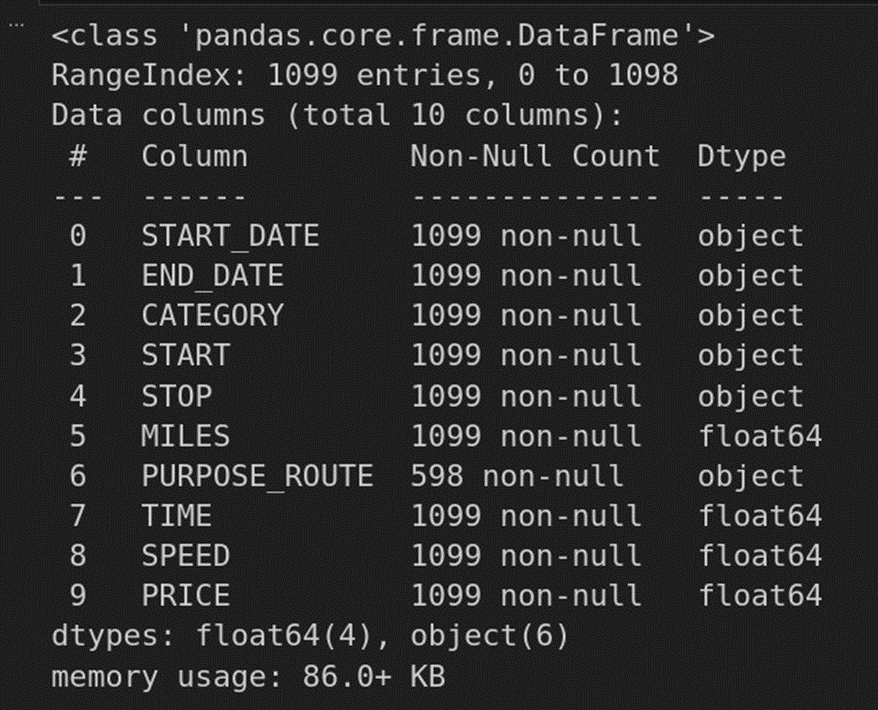


Рисунок 3 — Результат функции info

По данным видно, что пропуски есть только в столбце PURPOSE\_ROUTE. С помощью метода `fillna()` пустые строки были заменены на 'Unknown'. Код представлен на листинге 3.

Листинг 3 - Код замены пустых строк.

df['PURPOSE\_ROUTE'].fillna('Unknown', inplace=True) df.info()

Результат работы функции на рисунке 4.

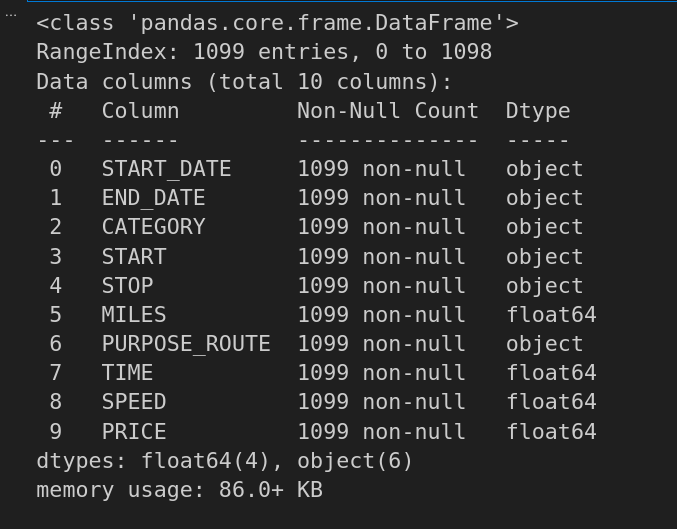


Рисунок 4 — Результат заполнения пустых строк

Далее была проведена проверка на явные дубликаты, с помощью `duplicated()`, и на явные дубликаты, с помощью `unique()`. Ни там, ни там дубликатов не было обнаружено.

Для удобства у столбцов с датой тип данных был изменен на `datetime` (Листинг 4). Листинг 4 – Код изменения типа данных.

df['START\_DATE'] df['END\_DATE']

df.dtypes

= pandas.to\_datetime(df['START\_DATE'])

= pandas.to\_datetime(df['END\_DATE'])

Результат работы функции на рисунке 5.

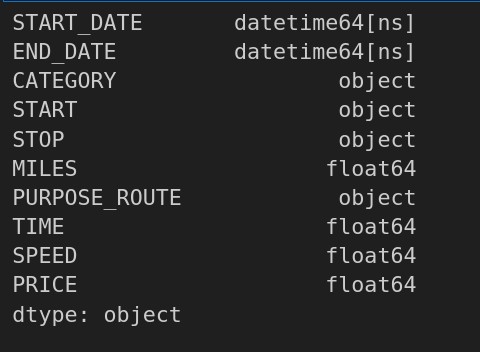


Рисунок 5 — Результат изменения типа данных

Далее была проделана работа по визуализации данных.

Для начала была подключена библиотека `matplotlib` - одна из наиболее популярных библиотек для визуализации данных. В данной случае достаточно будет импортировать модуль `pyplot`, который содержит все необходимые компоненты для построения графиков.

Была построена диаграмма рассеивания (`scatter`), которая помогает обнаружить взаимосвязи между данными (например, определить связь роста и веса, связь стоимости жилья от площади и т.д.). Код представлен на листинге 5.

import matplotlib.pyplot as plt

plt.scatter(df['TIME'], df['PRICE'], s=5, color='green')

Листинг 5 — Код создания диаграммы рассеивания.

Результат рисования графика на рисунке 6.

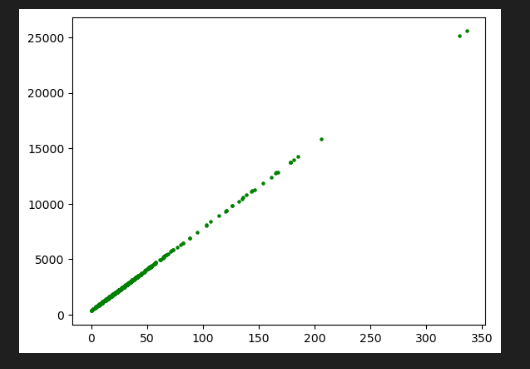


Рисунок 6 — Диаграмма рассеивания

Использовался столбец - время в пути (TIME). С помощью данного столбца легче будет строить графики, которые в дальнейшем можно анализировать. Так был построен график, который показывает зависимость времени поездки от ее цены. Из него видно, что зависимость линейная. Получается цена строится от времени в пути.

Для дальнейшего анализа была построена матрица диаграмм рассеивания - диаграммы для нескольких признаков набора данных, которая отражает попарные взаимосвязи величин.

Для построения нескольких попарных двумерных распределений в наборе данных можно воспользоваться библиотекой `seaborn`, кототая содержит метод `pairplot()`. Данные графики были построены для места количества миль, времени, скорости и цены. Ниже представлен листинг 6.

import seaborn as sns

sns.pairplot(df[['MILES', 'TIME','SPEED', 'PRICE']], hue

= 'TIME')

Листинг 6 — Код создания матрица диаграмм рассеивания.

Результат рисования графика на рисунке 7.

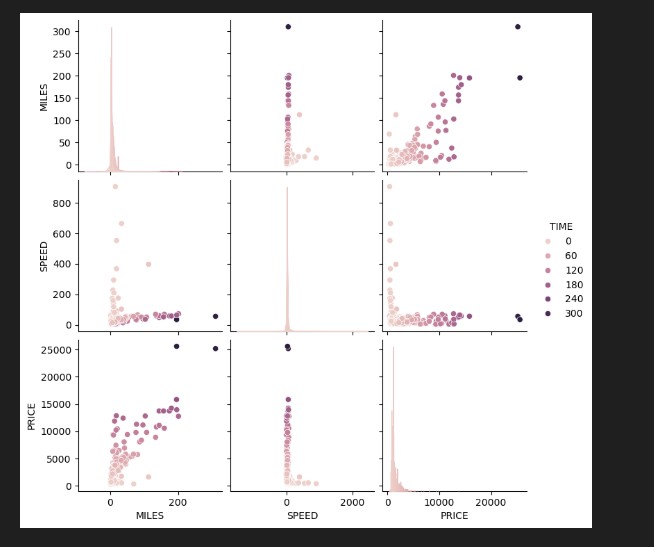


Рисунок 7 - Матрица диаграмм рассеивания

Данный график показывает зависимость скорости, пройденных миль, времени и цены от друг друга по времени.

По графику можно сказать, что скорость никак не связана с другими данными, так как в целом скорость зависит только от ситуации на дороге (скоростной режим).

Также видно, что количество миль также зависит от стоимости, но уже более хаотично. А так как время линейно зависимо от количества миль, то и время от стоимости зависит также как и время от количества миль.

Корреляция может быть измерена с помощью коэффициента корреляции, такого как коэффициент Пирсона. Коэффициент Пирсона находят с помощью метода `corr()`. Результат и код продемонстрированы на листинге 7, рисунке 8.

df2 = df[['MILES', 'SPEED', 'TIME', 'PRICE']]

df2.corr()

Листинг 7 — Код нахождения корреляции.

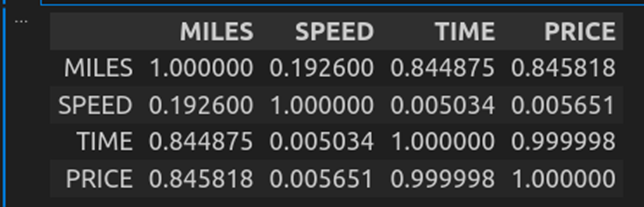


Рисунок 8 — Результат нахождения корреляции

По корреляции уже точно можно сказать, что время линейно зависит от цены. Также видно, что время и цена зависят от количества миль почти линейно. А вот скорость почти не зависит от других столбцов.

Также можно вычислить коэффициент корреляции между двумя конкретными столбцами, используя `df2['TIME'].corr(df2['PRICE']) `.

Для расчета ковариации используется функция `cov()` библиотеки `numpy`. Код представлен на листинге 8.

Листинг 8 — Код нахождения ковариации.

import numpy as np

df\_cov = df[df['SPEED']<1000]

data = np.array([df['TIME'], df['MILES'], df['PRICE'], df['SPEED']])

cov = np.cov(data,bias=True) print(cov)

На рисунке 9 представлен результат листинга 8.

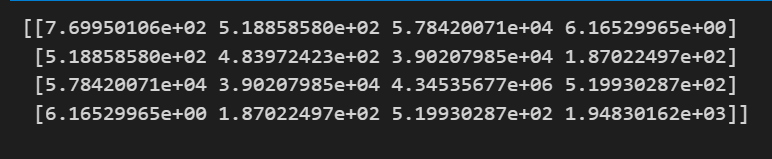


Рисунок 9 — Результат нахождения корреляции

Ковариация положительна, что значит, что при увеличении одного значения - увеличивается и другое и наоборот. Для поля speed взяты значения меньше 1000 так как присутствуют значения inf (бесконечность).

Также был была построена тепловая карта корреляции (`heatmap`). Матрица оформляется в цветовой палитре, так яркие цвета - положительные коэффициенты, а темные - отрицательные. `imshow` позволяет создавать двумерные картинки, используя цветовую карту `summer`. Для наименования столбцов и строк использовались методы `xticks` и `yticks`. Код представлен на листинге 9.

Листинг 9 — Код построения тепловой карты.

df2 = df[['MILES', 'TIME', 'SPEED', 'PRICE']]

corr = df2.corr() plt.imshow(corr, cmap='summer') plt.colorbar()

plt.xticks(range(len(corr.columns)), corr.columns)

plt.yticks(range(len(corr.columns)), corr.columns)

На рисунке 10 представлен результат листинга 9.

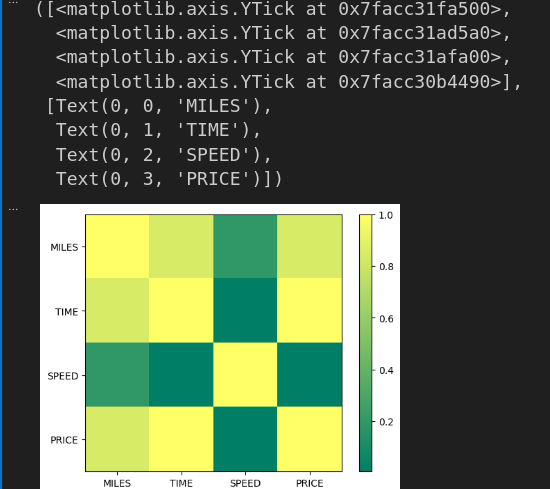


Рисунок 10 — Результат построения тепловой карты

По данному графику можно сказать, что количество миль, время и цена зависят друг от друга почти линейно (светлее), а вот скорость не зависит от других значений (темнее).

Задание 1:

Для начала была проведена группировка данных функцией `groupby`, а с помощью `size` считает кол-во вхождений. Далее создается таблица с фильтрацией количества вхождений больше двух. Функция `barplot` строит столбчатую диаграмму, где по оси X отображаются различные цели маршрута (PURPOSE\_ROUTE), по оси Y отображается количество поездок, а цвет столбцов указывает на категорию (CATEGORY). Код представлен на листинге 10, а результат на рисунке 11.

Листинг 10 — Код задания 1.

grouped\_data = df.groupby(['CATEGORY', 'PURPOSE\_ROUTE']).size().reset\_index(name='count')

filtered\_data = grouped\_data[grouped\_data['count'] > 2] plt.subplots(figsize=(20, 6))

sns.barplot(x='PURPOSE\_ROUTE',

y='count', hue='CATEGORY',

data=filtered\_data

)

plt.title('Диаграмма количества поездок по целям и категориям')

plt.xlabel(xlabel = 'Цель поездки') plt.ylabel(ylabel = 'Количество поездок')

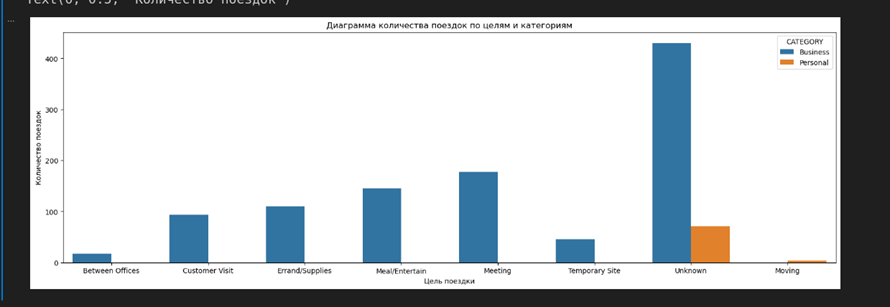


Рисунок 11 — Результат задания 1

Этот график позволяет наглядно сравнить количество поездок для различных целей маршрута в разных категориях. Визуализация данных в виде столбчатой диаграммы делает анализ более понятным и удобным для интерпретации.

Задание 2:

Сначала создаем сводную таблицу функцией `pivot\_table`, с сркдним значением `MILES`, которые группируются по `PURPOSE\_ROUTE`. Затем строим график по условию, звездочку рисуем символом - '\*', а размер указываем 180. Сетку же добавляем при помощи `grid()`. Код представлен на листинге 11, а результат на рисунке 12.

Листинг 11 — Код задания 2.

pivot\_table = pandas.pivot\_table(df, values='MILES', index='PURPOSE\_ROUTE', aggfunc='mean')

pivot\_table.plot( style='\*',

figsize=(20, grid = True, markersize =

xticks = list

xlabel='Цель

6),

18,

(range(0,11)),

поездки',

ylabel='Кол-во миль',

title='График среднего количество пройденных миль по каждой цели поездки',

)

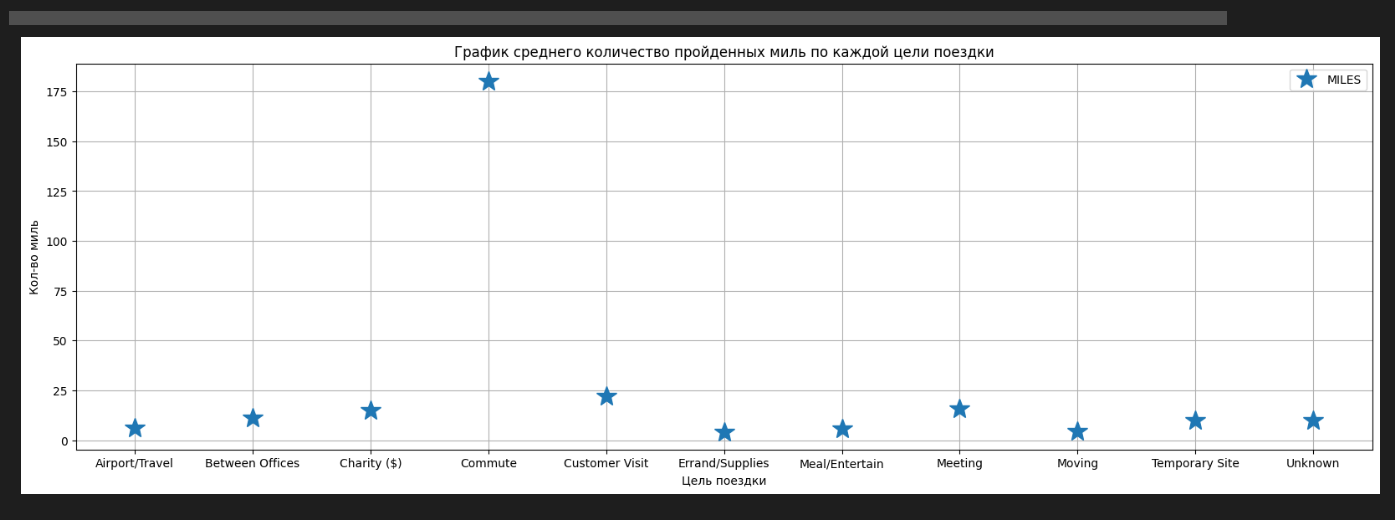


Рисунок 12 — Результат задания 2

Этот график позволяет визуализировать и сравнить средние расстояния, пройденные для различных целей поездки. Зеленые звезды на графике представляют каждую уникальную цель поездки, и их вертикальное положение отражает среднее количество миль для каждой цели. Такой график позволяет быстро определить различия в пройденных расстояниях между разными целями поездки.

Задание 3:

Создаем сводную таблицу и фильтруем ее значения аналогична предыдущим заданиям. Далее строим круговую диаграмму, у которой указываем формат отображения процентов. Код представлен на листинге 12, а результат на рисунке 13.

Листинг 12 — Код задания 3.

pivot\_table = pandas.pivot\_table(df, values='MILES', index='PURPOSE\_ROUTE', aggfunc='count')

filtered\_pivot\_table = pivot\_table[pivot\_table['MILES']

>= 5]

plt.figure(figsize=(8, 8)) plt.pie(

filtered\_pivot\_table['MILES'], labels=filtered\_pivot\_table.index, autopct='%1.1f%%'

)

plt.legend(title='PURPOSE\_ROUTE', loc='lower right') plt.title('Распределение целей поездки')

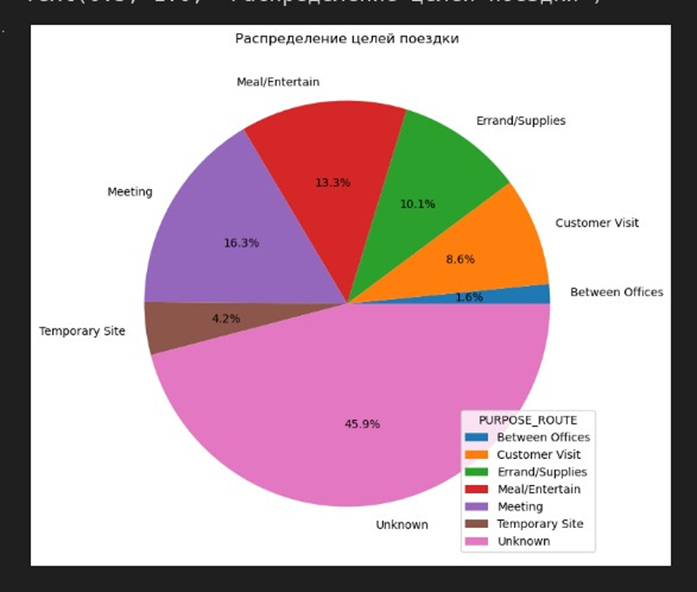


Рисунок 13 — Результат задания 3

Этот график позволяет визуализировать процентное распределение целей поездки для категорий, где количество поездок составляет 5 и более. Круговая диаграмма наглядно показывает, какие цели поездки являются более распространенными среди данных с учетом заданного условия (поездки >= 5).

Далее были выполнены дополнительные задания.

Дополнительное задание 1 - Из цели поездки выбрать топ 3 по количеству и построить boxplot по цели и времени.

Для начала была создана таблица `df\_top\_3`, в которой находятся 3 самых встречаемых цели поездки. С помощью функции `value\_counts` ведется подсчет количества упоминаний, с помощью `value\_counts(3)`, берутся 3 самых высоких значений. Далее строится таблица `filtered\_df`, которая содержит только данные с целями поездки из

`df\_top\_3`, с этим помогает функция `isin`, которая берет только вхождения. Далее строится `boxplot` по цели поездки и времени. Код представлен на листинге 13, а результат на рисунке 14.

df\_top\_3 = df['PURPOSE\_ROUTE'].value\_counts().nlargest(3).index

filtered\_df = df[df['PURPOSE\_ROUTE'].isin(df\_top\_3)] plt.figure(figsize=(12, 6)) sns.boxplot(x='PURPOSE\_ROUTE', y='TIME',

data=filtered\_df, order=df\_top\_3) plt.xlabel('Цель поездки') plt.ylabel('Время в пути')

plt.title('Boxplot времени в пути для топ 3 целей поездки')

plt.xticks(rotation=45) plt.grid(True)

Листинг 13 — Код дополнительного задания 1.

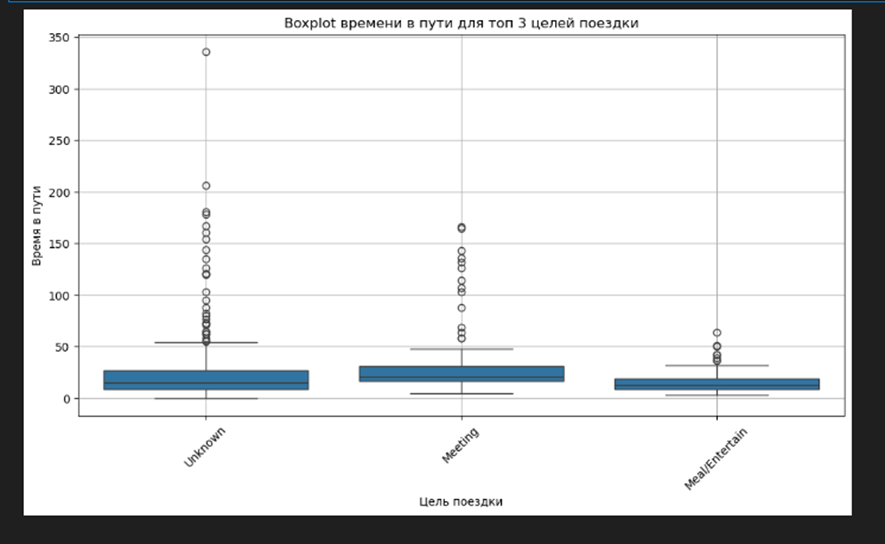


Рисунок 14 — Результат дополнительного задания 1

Анализируя этот график, можно сделать выводы о том, как различные цели поездки влияют на время в пути и наличие выбросов, что может указывать на необычные или аномальные значения времени в пути для определенных целей. Из графика видно, что неизвестный тип поездки имеет большее время в сравнении с остальными.

Дополнительное задание 2 - Построить любой hexbin график.

Для примера построен hexbin-график для цены и времени в пути. Количество плит - 10, палитра - красная. Код представлен на листинге 14, а результат на рисунке 15.

Листинг 14 — Код дополнительного задания 2.

plt.figure(figsize=(8, 6)) plt.hexbin(df['TIME'], df['PRICE'], gridsize=10,

cmap='Reds') plt.colorbar(label='Количество точек') plt.xlabel('Время в пути') plt.ylabel('Количество миль')

plt.title('Hexbin-график для времени в пути и количества миль')

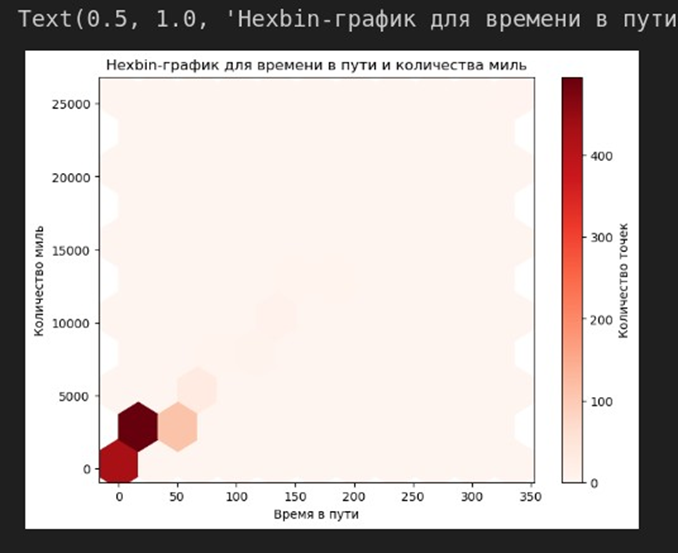


Рисунок 15 — Результат дополнительного задания 2

График получился такой маленький, так как большинство поездок по этим параметрам сосредоточено в левом нижнем углу. Следовательно, можно сделать вывод, что больше 75% поездок совершены до 5000 миль и до 50 минут.

# **Вывод**

Представленный анализ данных о поездках содержит ценную информацию о взаимосвязи различных факторов в контексте поездок. Исходные данные позволяют понять множество аспектов, таких как влияние времени в пути на цену, зависимость цены от пройденного расстояния и цели поездки, а также как эти параметры связаны между собой.

1. Время в пути и Цена: Анализ диаграммы рассеивания и коэффициента корреляции (Пирсона) подтверждает линейную зависимость между временем в пути и ценой поездки. Это означает, что чем больше времени занимает поездка, тем выше ее стоимость.

2. Пройденное расстояние и Цена: Количество миль имеет хаотичное влияние на цену, но все же можно выделить тенденцию к увеличению цены с увеличением пройденного расстояния. Это также видно из коэффициента корреляции, который указывает на слабую, но существующую положительную связь между этими двумя параметрами.

3. Цель поездки и Количество миль: Круговая диаграмма и столбчатые диаграммы позволяют увидеть, какие цели поездки являются более популярными и как они связаны с количеством пройденных миль. Например, можно увидеть, что цели "Деловая поездка" и "Личная поездка" чаще всего связаны с большими расстояниями, что может быть связано с поездками между городами или регионами.

4. Скорость и другие параметры: Скорость поездки практически не имеет корреляции с другими параметрами, что подтверждает логику, что она в основном зависит от текущей ситуации на дороге и ограничений скорости.

В целом, эти выводы могут быть полезны для компании и ее клиентов. Компании могут адаптировать свои цены в зависимости от времени в пути и пройденного расстояния, а также учитывать популярность различных целей поездки при разработке стратегий ценообразования.