Кузовчиков Д.О. +79688384371

Решение тестового задания на вакансию Стажер Data scientist/MLE в компанию «Автомакон»

Ответ на вопрос №1 - EDA

Влияние факторов на целевую переменную (спрос)

Фактор	Пиво	Сигареты	Анальгетики
Марка товара	Есть несколько лидеров среди марок по продажам	Есть один явный лидер, среди остальных зависимость выражена слабее	Есть несколько лидеров среди марок по продажам
Цена	Комфортная цена для покупателя не превышает 14 у.е. Выше этой цены продажи резко падают	Покупатели готовы приобретать товар за любую цену, но уровень с самым высоким спросом - около 2 у.е. При этом есть две ценовые категории, которые можно разделить на "обычные" и "элитные".	Комфортная цена на анальгетики для покупателя не превышает 13 у.е. Выше этой цены продажи значительно падают
Неделя	Видна сильная сезонная зависимость продажи пива. На праздничные дни спрос возрастает, летом в среднем продажи выше, чем зимой. Наблюдается очень резкий спад продаж пива в середине октября	Видна устойчивая тенденция к снижению покупки сигарет. До мая 1991 года наблюдается зависимость продаж от праздников, причем некоторые праздники уменьшают продажу, а некоторые	Наблюдается некоторая сезонность продаж анальгетиков. Летом продажи в среднем ниже, чем зимой. В праздничные недели продажи в среднем выше. В начале было сильное снижение

Фактор	Пиво	Сигареты	Анальгетики
	1995 года. Примерно в это время в США был принят "закон о нулевой терпимости", запрещающий водителям моложе 21 года управлять автомобилем с содержанием алкоголя в крови не менее 0,02%, чтобы воспрепятствовать употреблению алкоголя несовершеннолетними. Также в ходе анализа обнаружилось, что многие популярные марки пива в это время, судя по всему, закрылись, потому что продажи их стали равны нулю.	увеличивают. В мае 1991 года отмечается резкое снижение продаж с дальнейшим сохранением тренда. Скорее всего это связано с тем, что в этом году расходы на рекламу табачных изделий в журналах и газетах достигли рекордно низкого уровня в сравнении с предыдущими периодами. Однако начиная с 1994 года продажи на сигареты в праздничные дни резко увеличились в сравнении с общим трендом, но даже в этих пиках есть снижение.	продаж (май 1990), однако потом продажи вернулись на прежний уровень из-за распространения опиоидных препаратов для лечения хронической боли и активной рекламы фармацевтическими компаниями своих опиоидных препаратов для медицинских работников (начало первой волны эпидемии опиоидов).
Код распродажи	Тип распродажи "бонусная покупка" значительно увеличивает спрос на пиво в сравнении с другими типами	Единственный тип распродажи "снижение цены" для сигарет не привел к увеличению спроса (другие вообще не использовались)	Тип распродажи "бонусная покупка" значительно увеличивает спрос на пиво в сравнении с другими типами, затем идет "снижение цены". Непонятно, что такое "G" (6687 товаров продано с таким кодом)
Размер товара	Выделяются явные лидеры по продажам. Видимо, большинство производителей используют именно эти стандарты или это связано с	Есть один явный лидер. Видимо, большинство производителей используют именно этот стандарт	Выделяются явные лидеры по продажам. Видимо, эти стандарты подходят под самые частые курсы приема анальгетиков

Фактор	Пиво	Сигареты	Анальгетики
	поводом, ставшим причиной покупки		
Доля этнических групп	Нет явной зависимости	Можно заметить, что при увеличении доли этнических групп количество купленных сигарет в магазине в среднем уменьшается (связь слабо выражена)	Нет явной зависимости
Доля пожилых людей	Нет явной зависимости	Есть очень слабо выраженный тренд к снижению спроса на сигареты при увеличении доли пожилых клиентов	Нет явной зависимости
Доля выпускников колледжей	Нет явной зависимости	Нет явной зависимости	Нет явной зависимости

Общее описание данных

Значения факторов имеют слабо нормальное или вовсе ненормальное распределение (согласно тесту Шапиро-Уилка). В целом спрос на представленные товары несбалансированный: есть явные лидеры по продажам. Присуствуют тренд и сезонность в продажах товаров. По частичному сопоставлению продаж в магазинах и доли разных групп клиентов можно предположить их слабую связь или вовсе ее отсутствие.

Ответ на вопрос №2 - Прогнозные модели

Прогнозные модели строились на примере конкретного случайно выбранного товара (поскольку была явная зависимость спроса от марки). В качетсве метрики качества моделей использовались коэффициент детерминации (R2) и средняя квадратичная ошибка (MSE). В качестве критериев значимости результатов использовались критерий Манна-Уитни на тип распределения (насколько распределение прогноза похоже на распределение тестовой выборки) и критерий Левена на дисперсии (насколько набор прогнозов по структуре похож на тестовую выборку). Безусловно, использование этих критериев требует обсуждения, однако на первоначальном этапе можно на них остановиться с учетом эффекта систематической ошибки (сам подход может быть неверным из-за систематических ошибок, однако это не мешает сравнивать результаты подходов для разных данных из-за однородности этих ошибок).

Модель	R2	MSE	Значимость №1 –тест Манна-Уитни	Значимость №2 – тест Левена	Интерпретация
Линейная регрессия	0.122	-	-	-	Идея – линейная зависимость целевой переменной от факторов. Модель показала очень плохие результаты, что объясняется ненормальной природой данных
Метод k ближайших соседей	0.785	45.906	+	+	Идея – прогноз спроса на основе схожих экземпляров (неделя, структура клиентов). Показал хорошие статистически значимые результаты
Градиентный бустинг	0.803	42.115	-	+	Идея – объединить несколько слабых линейных моделей путем итеративного обучения деревьев решений на остатках предыдущего дерева для

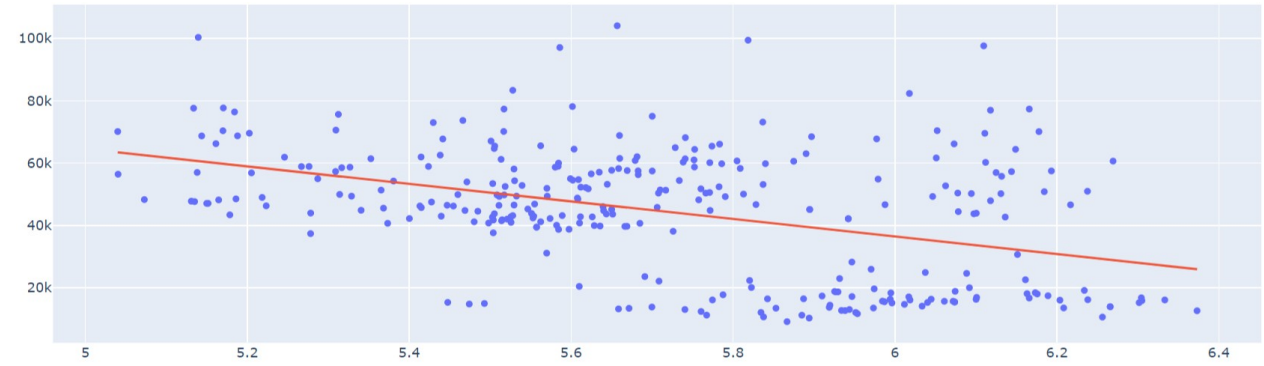
Модель	R2	MSE	Значимость №1 –тест Манна-Уитни	Значимость №2 – тест Левена	Интерпретация
					исправления ошибки предыдущей модели. Показал хорошие результаты, однако их статистическая значимость очень спорная
Случайный лес	0.937	13.483	+	+	Идея – усреднить коллекции решающих деревьев, где каждое дерево обучается на случайной подвыборке обучающих данных. Кроме того, в каждом узле дерева решений для разбиения рассматривается случайная подвыборка признаков. Показал лучшие результаты среди всех, причем статистически значимые
ARIMA	0.127	184361.503	+	-	Идея – линейная

Модель	R2	MSE	Значимость №1 –тест Манна-Уитни	Значимость №2 – тест Левена	Интерпретация
					зависимость текущего состояния от предыдущих. Показал очень плохие результаты, скорее всего из-за значительных перепадов значений. При этом общий тренд целевой переменной был определен моделью верно
ARIMA на экспоненциальном сглаживании	0.523	102978.392	+	-	Идея – линейная зависимость текущего состояния от предыдущих после сглаживания, устраняющего шумовые эффекты и сильные перепады значений. Показал результаты лучше, чем ARIMA, но все равно плохие. При этом общий тренд

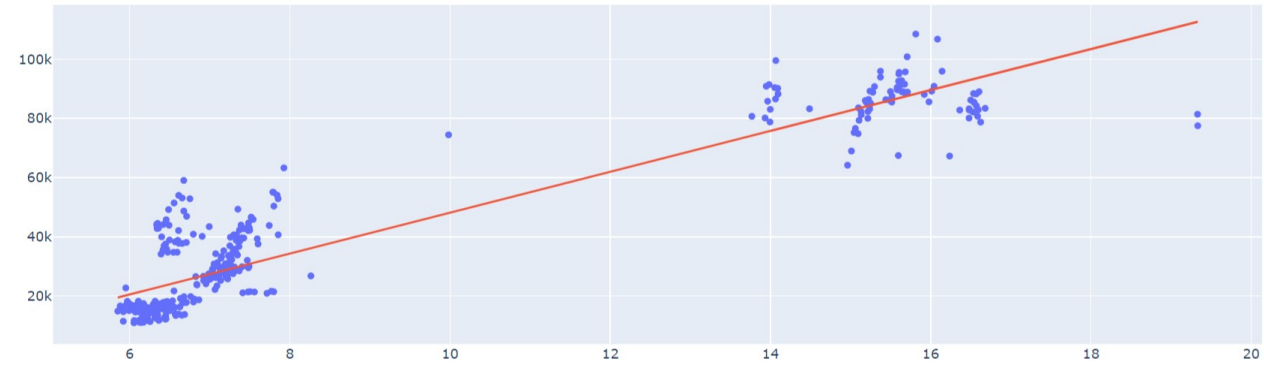
Модель	R2	MSE	Значимость N°1 –тест Манна-Уитни	Значимость N°2 – тест Левена	Интерпретация
					целевой переменной был определен моделью верно

Ответ на вопрос N°3 - Эластичность

Среднее значение эластичности для пива = -6.83 (эластичный спрос, спрос изменяется больше, чем цена: товары имеют замену либо не играют важной роли для потребителя). На графике совокупного спроса от средней цены цены наблюдается тренд на снижение спроса при росте цены

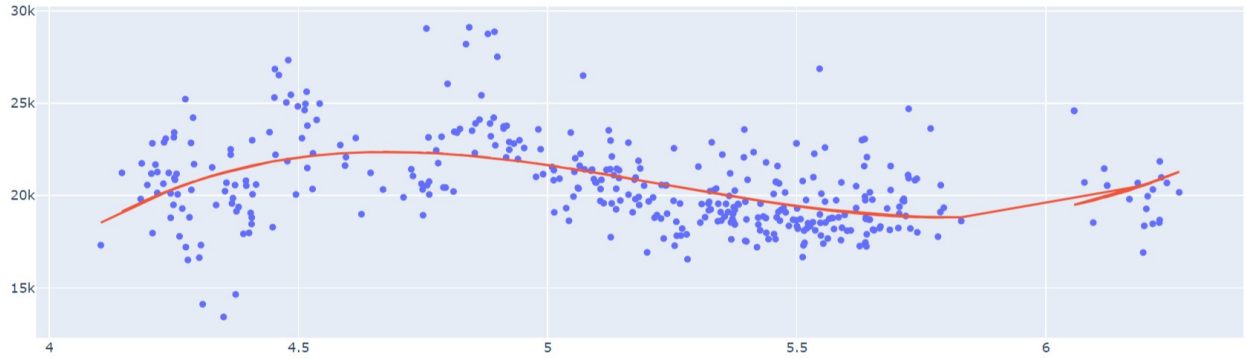


Среднее значение эластичности для сигарет = 92.91 (совершенно эластичный спрос, спрос изменяется на конечную величину при бесконечно малом изменении цены: спрос очень зависим даже от незначительных колебаний цены). На графике совокупного спроса от средней цены цены наблюдается тренд на увеличение спроса при росте цены



Среднее значение эластичности для анальгетиков = 2.94 (эластичный спрос, спрос изменяется больше, чем цена: товары имеют замену либо не играют важной роли для

потребителя, но все же важнее, чем пиво). На графике совокупного спроса от средней цены цены наблюдается колебательный тренд при росте цены



Конец описания результатов анализа.

Далее - код реализации анализа

```
import pandas as pd
import numpy as np
import plotly
import plotly.graph_objs as go
import plotly.express as px
from plotly.subplots import make_subplots
import statsmodels.api as sm
```

Чтение файлов

```
df_beer_sales = pd.read_parquet('beer_sales_data.parquet')
df_cig_sales = pd.read_parquet('cig_sales_data.parquet')
df_ana_sales = pd.read_parquet('ana_sales_data.parquet')

df_beer_upc = pd.read_parquet('beer_upc.parquet')
df_cig_upc = pd.read_parquet('cig_upc.parquet')
df_ana_upc = pd.read_parquet('ana_upc.parquet')

df_demographic = pd.read_parquet('demographic_data.parquet')
```

Заполнение пустых ячеек средним значением в колонке

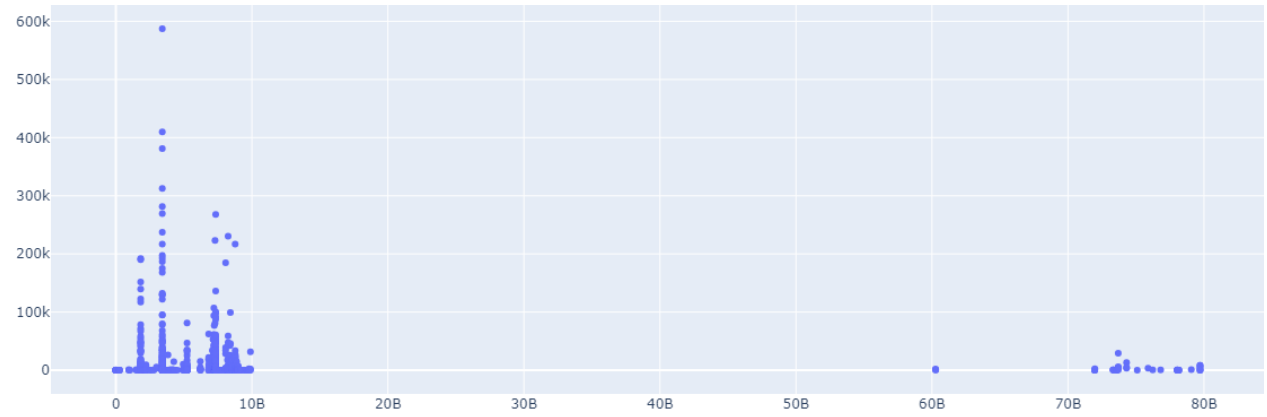
```
for i in df_demographic.columns:
    mean = df_demographic[i].mean()
    df_demographic[i] = df_demographic[i].fillna(mean)
df_demographic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 107 entries, 1 to 107
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   store     107 non-null    float64
 1   age60     107 non-null    float32
 2   age9      107 non-null    float32
 3   educ      107 non-null    float32
 4   ethnic    107 non-null    float32
 5   income    107 non-null    float32
 6   hhlarge   107 non-null    float32
 7   workwom   107 non-null    float32
 8   hval150   107 non-null    float32
 9   sstrdist  107 non-null    float32
 10  sstrvol   107 non-null    float32
 11  cpdist5   107 non-null    float32
 12  cpwvol5   107 non-null    float32
dtypes: float32(12), float64(1)
memory usage: 6.7 KB
```

Зависимость спроса от марки товара

Пиво

```
df = df_beer_sales.groupby('upc', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df['upc'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



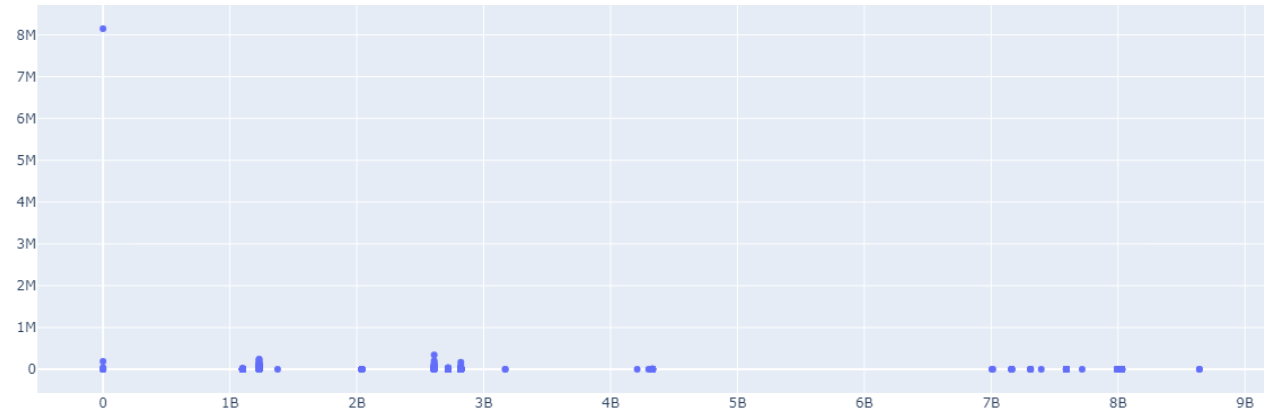
```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(15)
```

	upc	move
225	3410057306	587346
198	3410017505	409707
197	3410017306	381193
226	3410057505	312773
200	3410017528	281763
230	3410057602	269156
457	7336011301	267763
228	3410057528	237320
587	8248812345	230638
442	7289000011	223387
159	3410000354	217100
653	8769210012	216928
187	3410010505	197315
190	3410015306	192498
63	1820000784	191887

Видим сильную зависимость спроса от марки пива

Сигареты

```
df = df_cig_sales.groupby('upc', as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['upc'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(15)
```

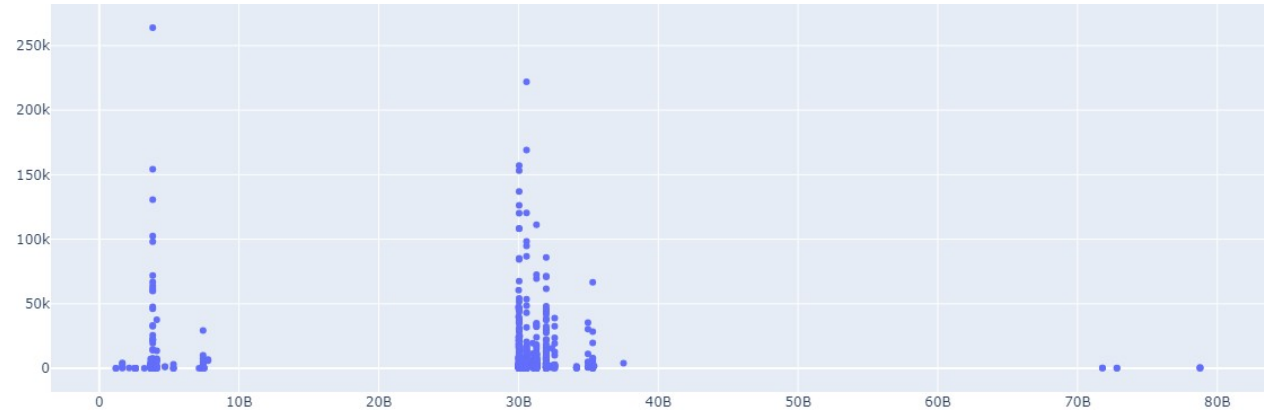
	upc	move
6	197	8151062

```
340  2610000304   348713
70   1230000024   240977
60   1230000010   230954
348  2610000364   203803
61   1230000011   190573
3           194   190371
589  2820000869   171784
378  2610000644   156900
71   1230000025   147434
68   1230000020   145161
63   1230000013   133462
72   1230000026   131508
64   1230000014   112624
346  2610000354   112397
```

Видим явного лидера спроса среди марок сигарет, среди остальных зависимость выражена слабее

Анальгетики

```
df = df_ana_sales.groupby('upc', as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['upc'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(15)
```

```
             upc    move
39    3828161001  263922
282  30573015020  221888
285  30573015030  169168
179  30045044907  157062
```

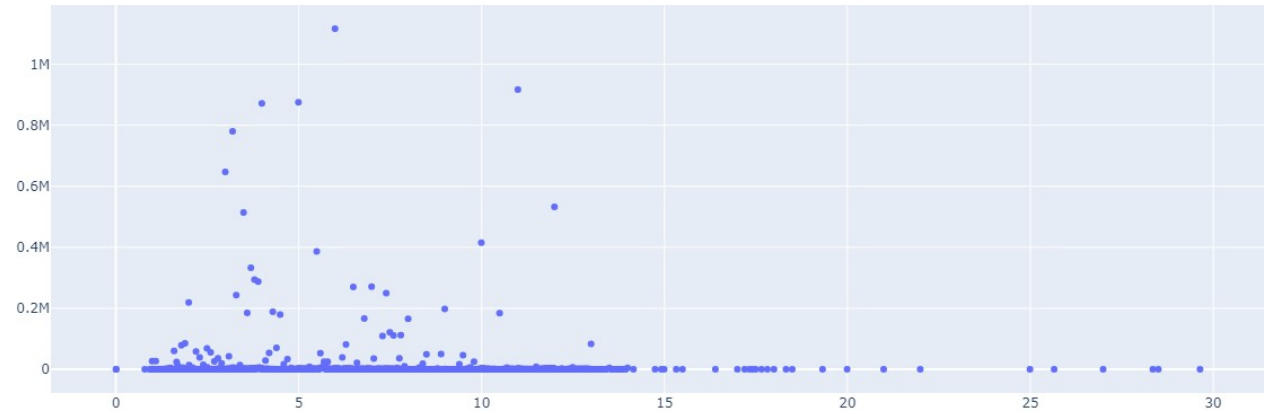
```
46    3828161037  154212
180  30045044909  153167
178  30045044905  136969
57    3828161073  130757
192  30045046824  126223
296  30573016020  120409
194  30045046850  120198
391  31284310112  111212
224  30045049960  108357
225  30045049968  108061
48    3828161041  102377
```

Видим сильную зависимость спроса от марки анальгетиков

Зависимость спроса от цены

Пиво

```
df = df_beer_sales.groupby('price', as_index=False).agg({'move':
'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['price'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

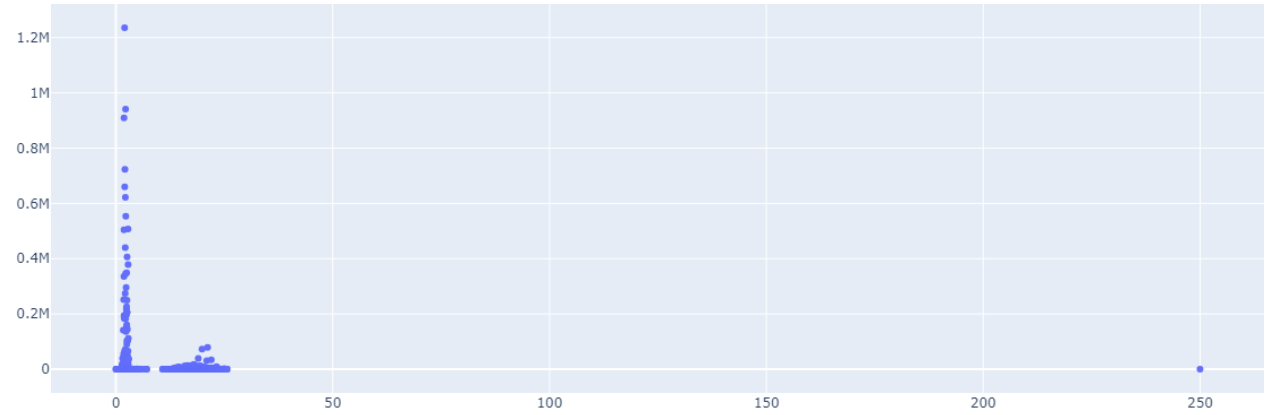


На основании диаграммы рассеяния можно сделать вывод, что комфортная цена на пиво для покупателя не превышает 14 у.е. Выше этой цены продажи резко падают

Сигареты

```
df = df_cig_sales.groupby('price', as_index=False).agg({'move':
'sum'})
fig = go.Figure()
```

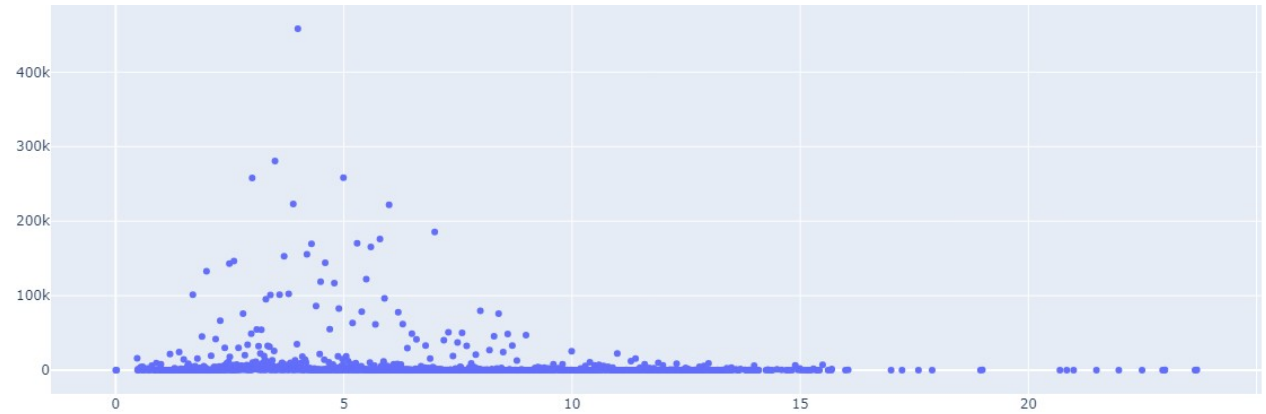
```
fig.add_trace(go.Scatter(x=df['price'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



На основании диаграммы рассеяния можно сделать вывод, что покупатели готовы приобретать сигареты за любую цену, но уровень с самым высоким спросом - около 2 у.е. При этом есть две ценовые категории, которые можно разделить на "обычные" и "элитные".

Анальгетики

```
df = df_ana_sales.groupby('price', as_index=False).agg({'move':
'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['price'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

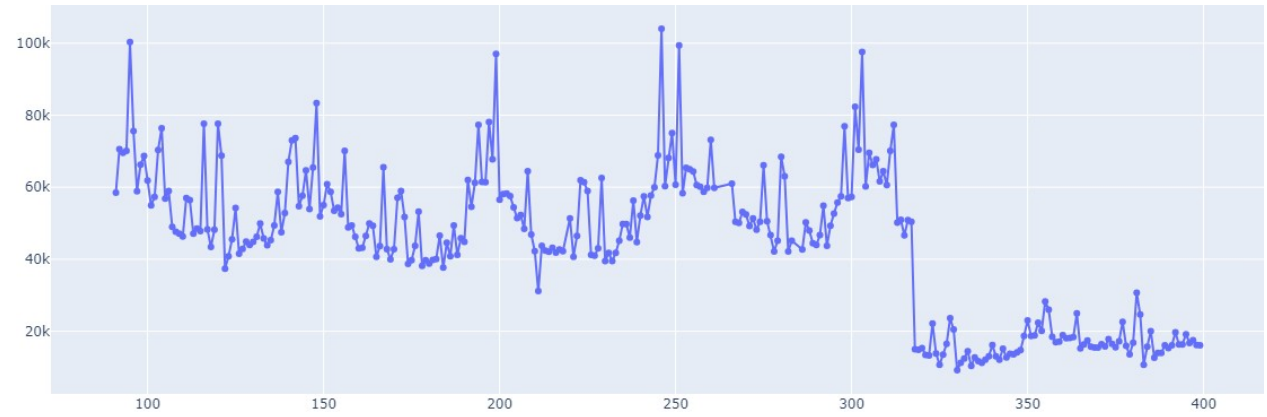


На основании диаграммы рассеяния можно сделать вывод, что комфортная цена на анальгетики для покупателя не превышает 13 у.е. Выше этой цены продажи значительно падают

Зависимость спроса от недели

Пиво

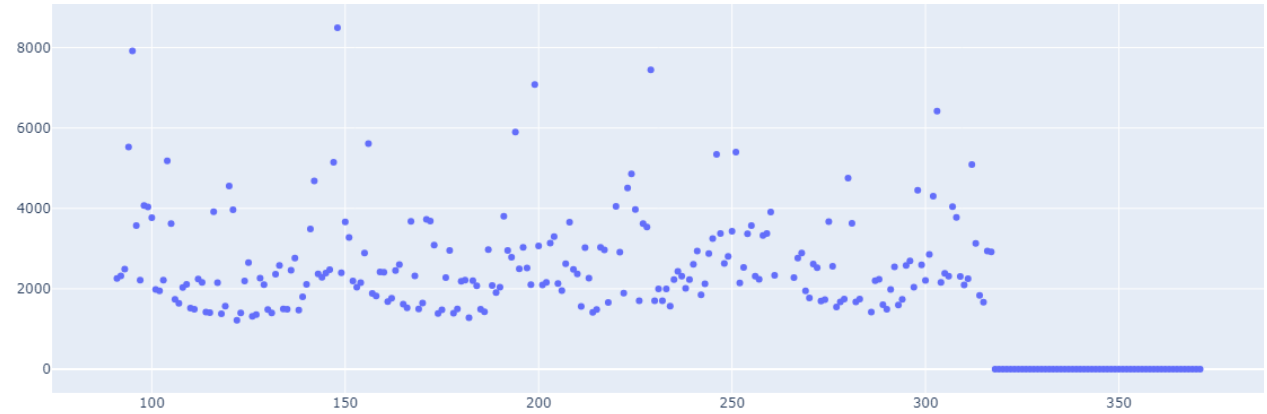
```
df = df_beer_sales.groupby('week', as_index=False).agg({'move':
'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['week'], y=df['move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



Видна сильная сезонная зависимость продажи пива. На праздничные дни спрос возрастает, летом в среднем продажи выше, чем зимой. Наблюдается очень резкий спад продаж пива в середине октября 1995 года. Примерно в это время в США был принят "закон о нулевой терпимости", запрещающий водителям моложе 21 года управлять автомобилем с содержанием алкоголя в крови не менее 0,02%, чтобы воспрепятствовать употреблению алкоголя несовершеннолетними. Также в ходе анализа обнаружилось, что многие популярные марки пива (см. анализ спроса от марки) в это время, судя по всему, закрылись, потому что продажи их стали равны нулю (см. пример ниже для самой популярной марки).

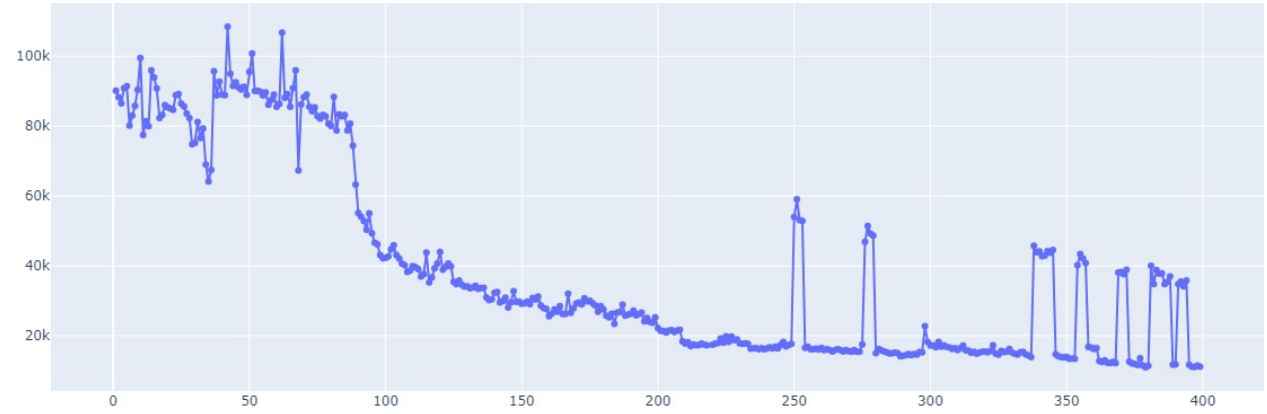
```
df = df_beer_sales[df_beer_sales['upc']==3410057306]
df_temp = df.groupby('week', as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
```

```
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



Сигареты

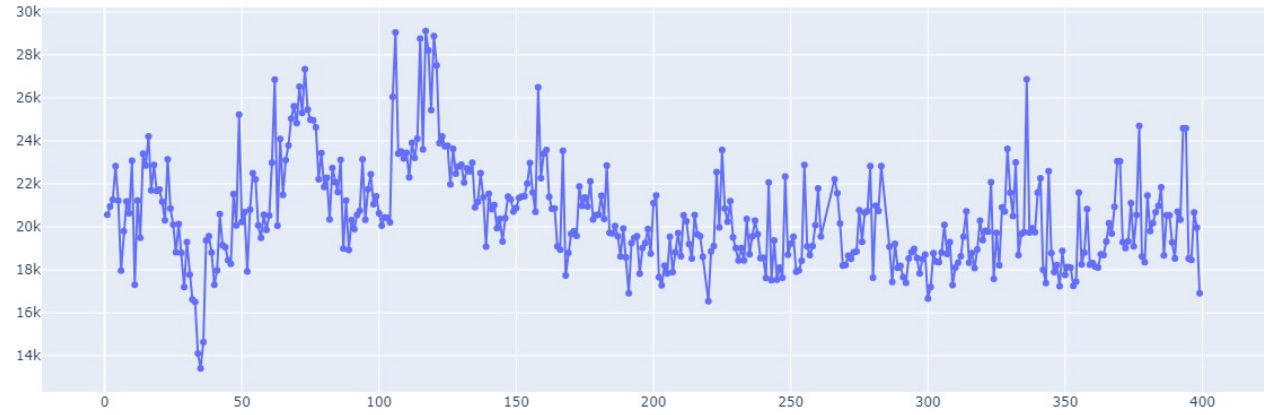
```
df = df_cig_sales.groupby('week', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df['week'], y=df['move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



Видна устойчивая тенденция к снижению покупки сигарет. До мая 1991 года наблюдается зависимость продаж от праздников, причем некоторые праздники уменьшают продажу, а некоторые увеличивают. В мае 1991 года отмечается резкое снижение продаж с дальнейшим сохранением тренда. Скорее всего это связано с тем, что в этом году расходы на рекламу табачных изделий в журналах и газетах достигли рекордно низкого уровня в сравнении с предыдущими периодами. Однако начиная с 1994 года продажи на сигареты в праздничные дни резко увеличились в сравнении с общим трендом, но даже в этих пиках есть снижение.

Анальгетики

```
df = df_ana_sales.groupby('week', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df['week'], y=df['move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

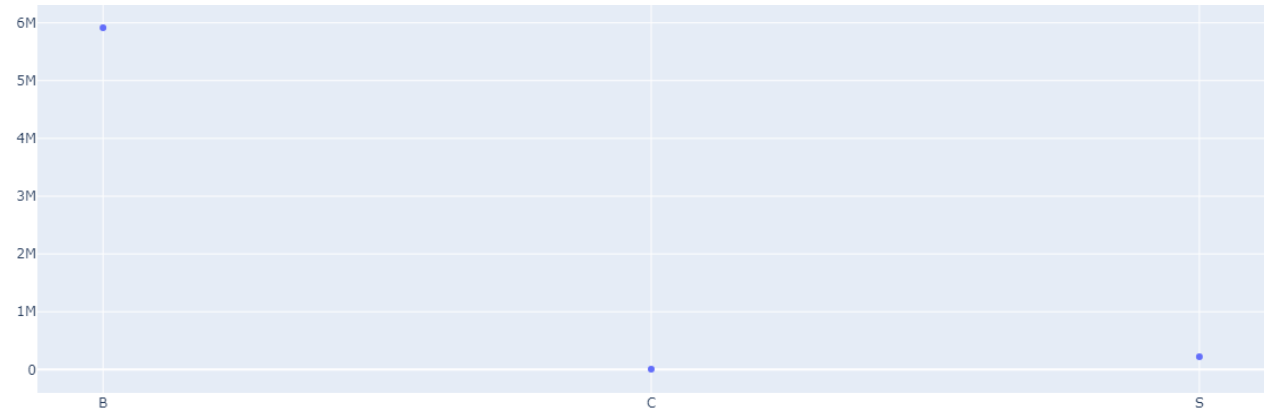


Наблюдается некоторая сезонность продаж анальгетиков. Летом продажи в среднем ниже, чем зимой. В праздничные недели продажи в среднем выше. В начале было сильное снижение продаж (май 1990), однако потом продажи вернулись на прежний уровень из-за распространения опиоидных препаратов для лечения хронической боли и активной рекламы фармацевтическими компаниями своих опиоидных препаратов для медицинских работников (начало первой волны эпидемии опиоидов).

Зависимость спроса от кода распродажи

Пиво

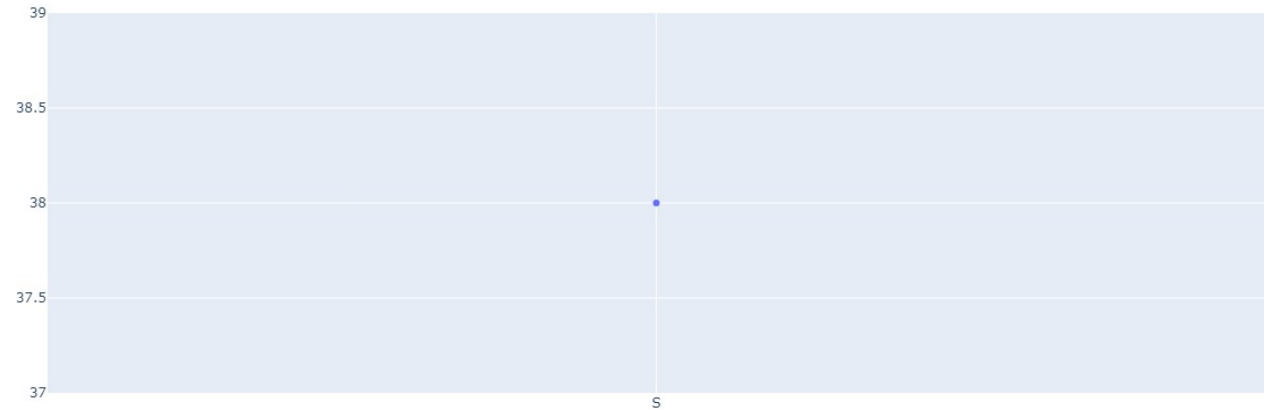
```
df = df_beer_sales.groupby('sale', as_index=False).agg({'move':
'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['sale'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



Заметно, что тип распродажи "бонусная покупка" значительно увеличивает спрос на пиво в сравнении с другими типами

Сигареты

```
df = df_cig_sales.groupby('sale', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df['sale'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

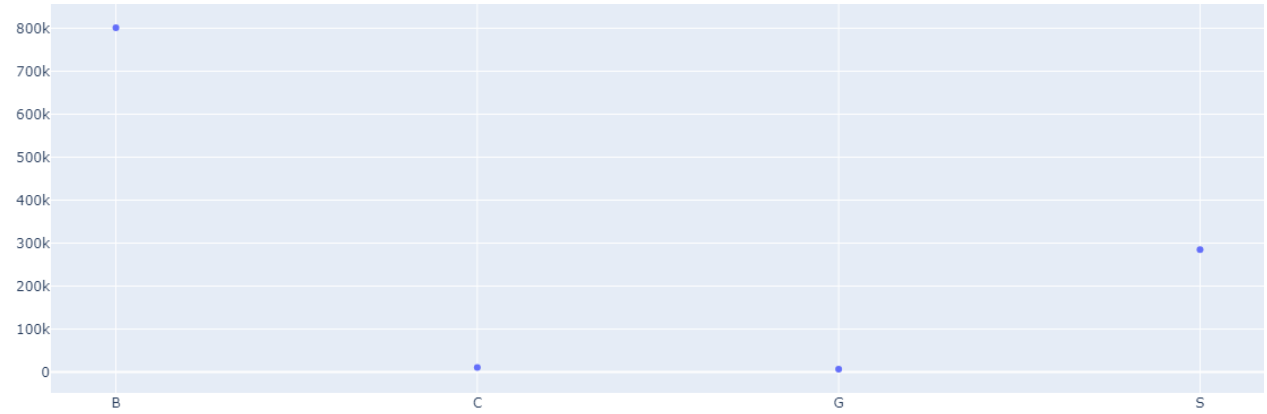


Видно, что единственный тип распродажи "снижение цены" для сигарет не привел к увеличению спроса (другие вообще не использовались)

Анальгетики

```
df = df_ana_sales.groupby('sale', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df['sale'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
```

```
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

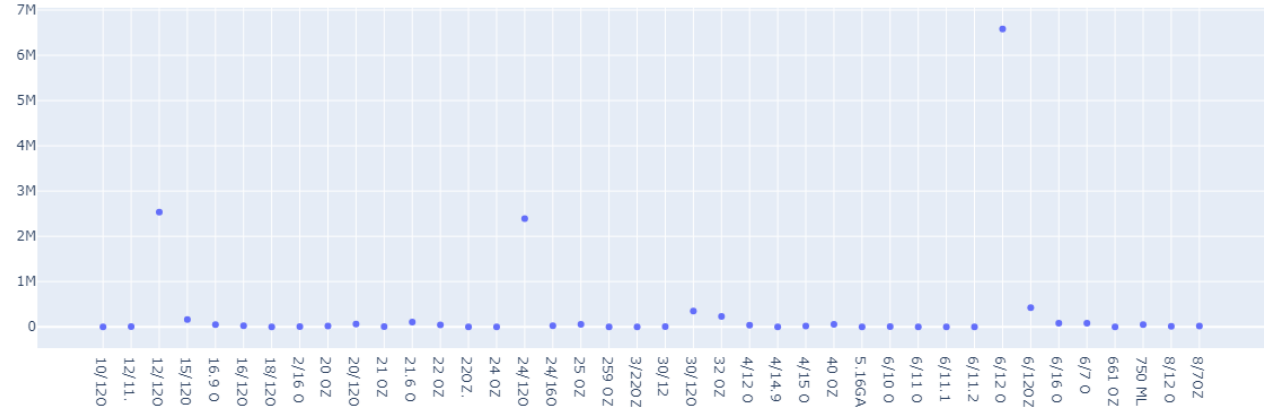


Заметно, что тип распродажи "бонусная покупка" значительно увеличивает спрос на пиво в сравнении с другими типами, затем идет "снижение цены". Непонятно, что такое "G" (???)

Зависимость спроса от размера товара

Пиво

```
df_1 = df_beer_sales.groupby('upc', as_index=False).agg({'move':
'sum'})
temp = df_beer_upc[['upc', 'size']]
df = pd.merge(df_1, temp, on="upc").groupby('size',
as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['size'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



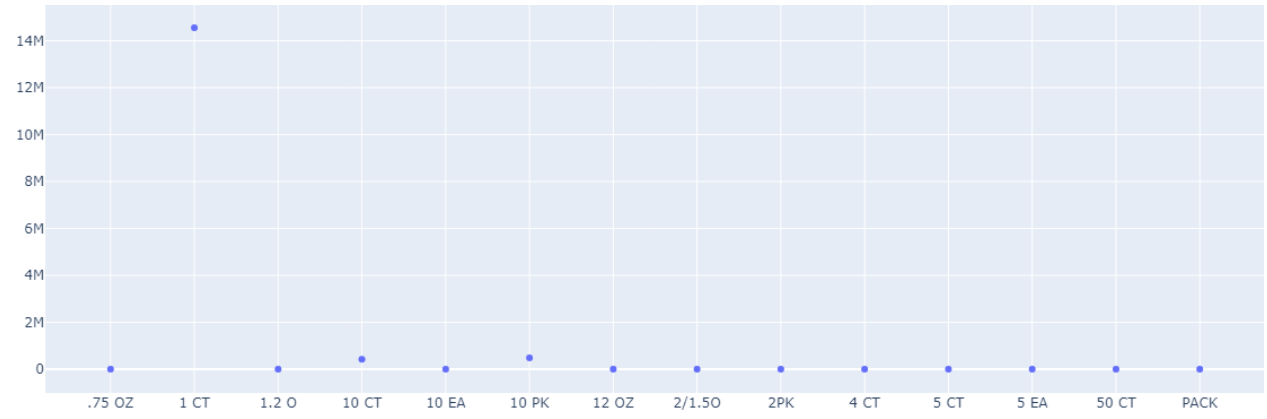
```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(10)
```

	size	move
32	6/12 O	6581540
2	12/12O	2536365
15	24/12O	2392989
33	6/12OZ	424972
21	30/12O	345766
22	32 OZ	230581
3	15/12O	161124
11	21.6 O	106270
35	6/7 O	81686
34	6/16 O	77307

Есть сильная зависимость спроса от размера бутылки пива

Сигареты

```
df_1 = df_cig_sales.groupby('upc', as_index=False).agg({'move':
'sum'})
temp = df_cig_upc[['upc', 'size']]
df = pd.merge(df_1, temp, on="upc").groupby('size',
as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['size'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(10)
```

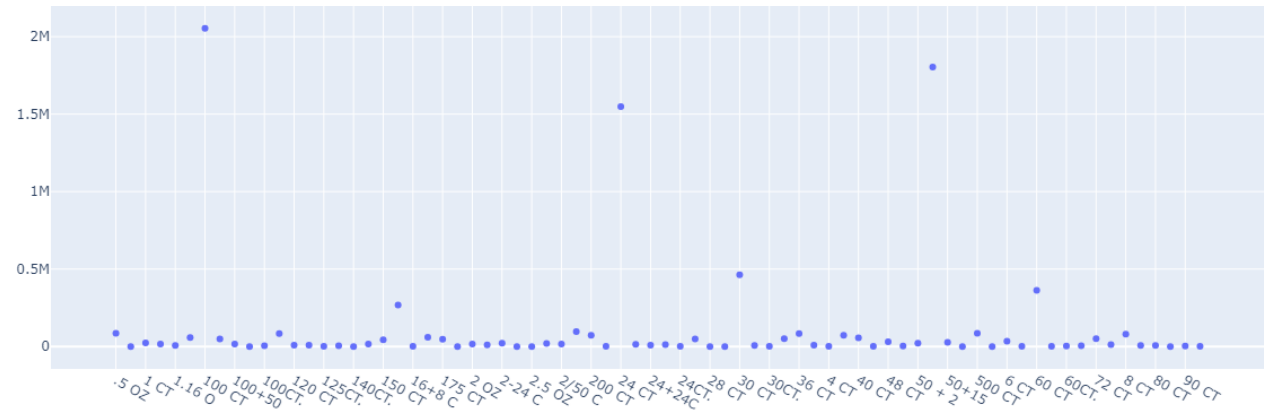
	size	move
1	1 CT	14557814
5	10 PK	479664

```
3    10 CT    421706
2    1.2 O      1811
11    5 EA       694
0   .75 OZ      119
10    5 CT        91
7   2/1.5O        42
9     4 CT        19
13    PACK        16
```

Есть сильная зависимость спроса от размера пачки сигарет

Анальгетики

```
df_1 = df_ana_sales.groupby('upc', as_index=False).agg({'move':
'sum'})
temp = df_ana_upc[['upc', 'size']]
df = pd.merge(df_1, temp, on="upc").groupby('size',
as_index=False).agg({'move': 'sum'})
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['size'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
sorted_df = df.sort_values(by='move', ascending=False)
sorted_df.head(10)
```

```
      size     move
6   100 CT  2053657
55   50 CT  1804317
34   24 CT  1548467
42   30 CT   463202
62   60 CT   363292
19   16 CT   268740
31   20 CT    96482
58  500 CT    85301
```

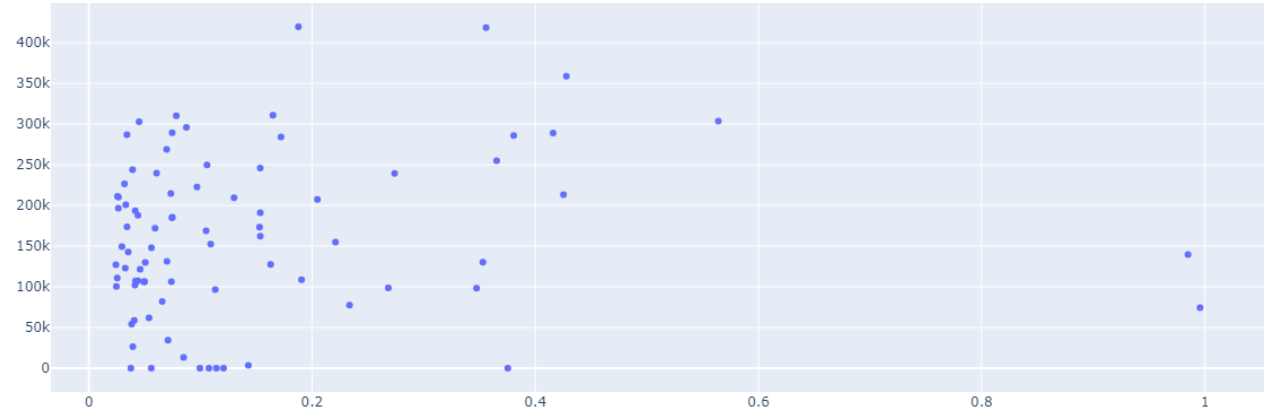
```
0    .5 OZ    84904
11   12 CT    84414
```

Есть сильная зависимость спроса от размера пачки анальгетиков

Зависимость спроса от % этнических групп

Пиво

```
df_1 = df_beer_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'ethnic']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['ethnic'], y=df['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df.corr()
```

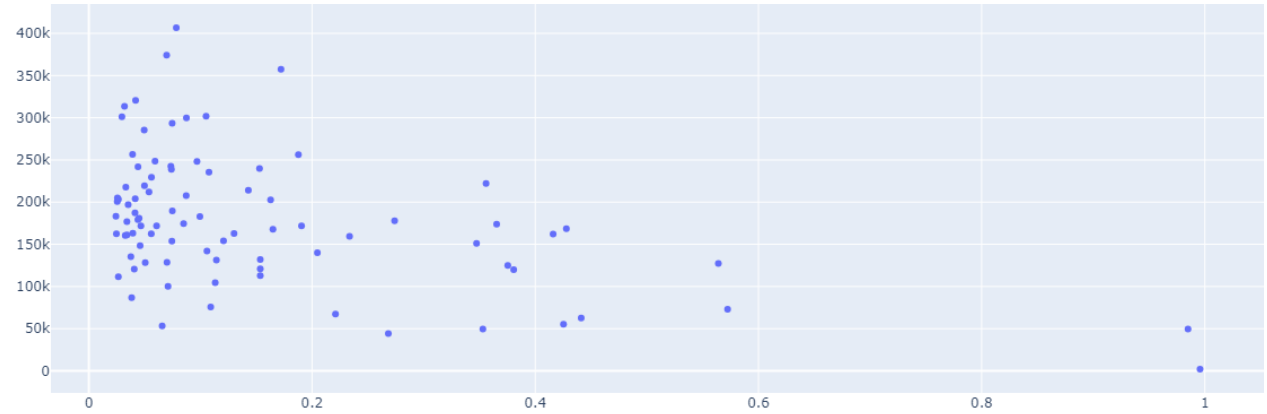
	store	move	ethnic
store	1.000000	0.383751	0.223823
move	0.383751	1.000000	0.136119
ethnic	0.223823	0.136119	1.000000

Нет явной зависимости между % этнических групп и спросом на пиво

Сигареты

```
df_1 = df_cig_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'ethnic']]
```

```
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['ethnic'], y=df['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



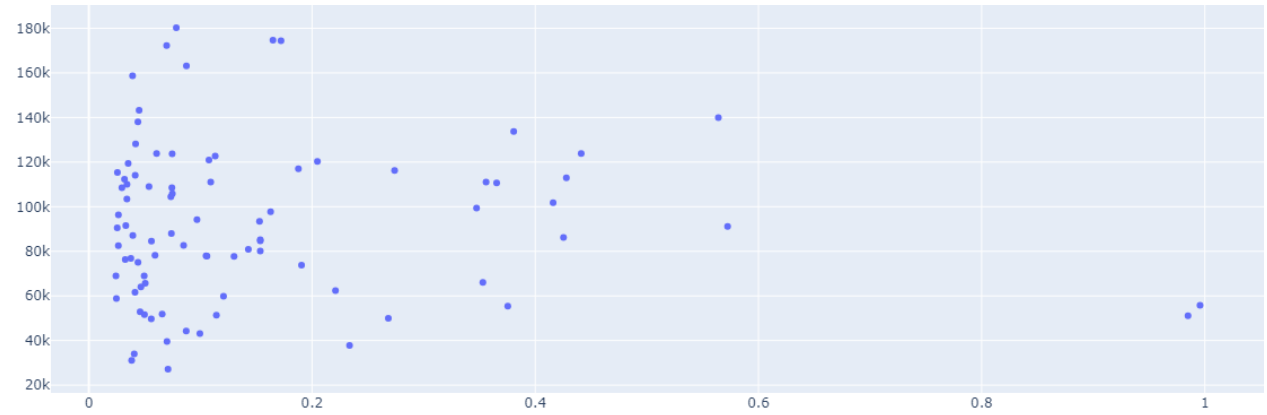

```
df.corr()
```

	store	move	ethnic
store	1.000000	0.020054	0.263485
move	0.020054	1.000000	-0.468082
ethnic	0.263485	-0.468082	1.000000

Можно заметить, что при увеличении доли этнических групп количество купленных сигарет в магазине в среднем уменьшается

Анальгетики

```
df_1 = df_ana_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'ethnic']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['ethnic'], y=df['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df.corr()
```

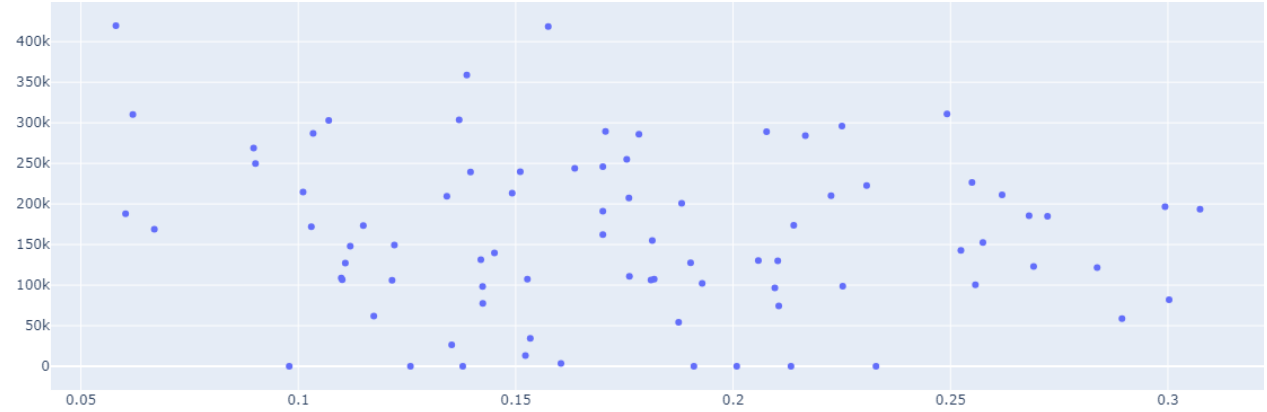
	store	move	ethnic
store	1.000000	0.264982	0.263485
move	0.264982	1.000000	-0.037524
ethnic	0.263485	-0.037524	1.000000

Нет явной зависимости между % этнических групп и спросом на анальгетики

Зависимость спроса от % пожилых людей

Пиво

```
df_1 = df_beer_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'age60']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['age60'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



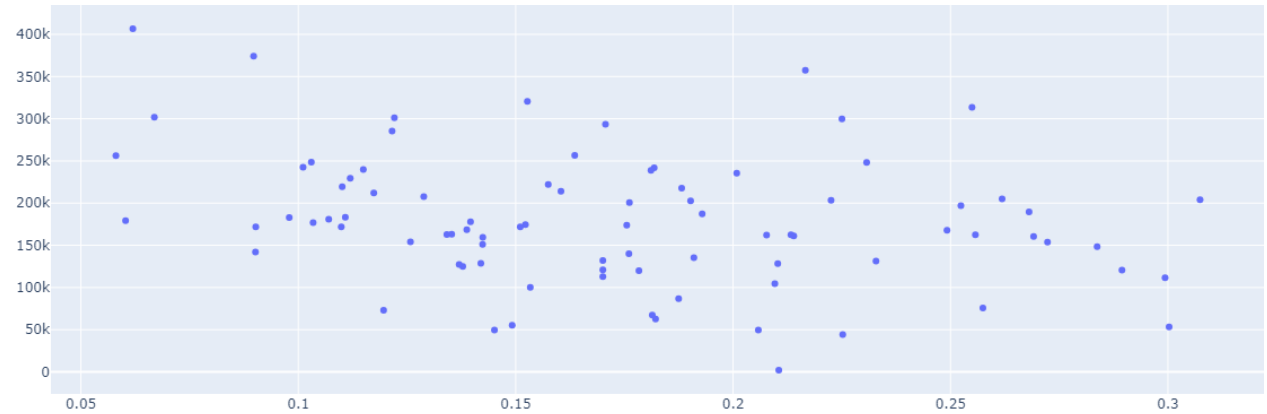
```
df.corr()
```

```
          store      move     age60
store  1.000000  0.383751 -0.246527
move   0.383751  1.000000 -0.136616
age60 -0.246527 -0.136616  1.000000
```

Нет явной зависимости между % пожилых клиентов и спросом на пиво

Сигареты

```
df_1 = df_cig_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'age60']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['age60'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df.corr()
```

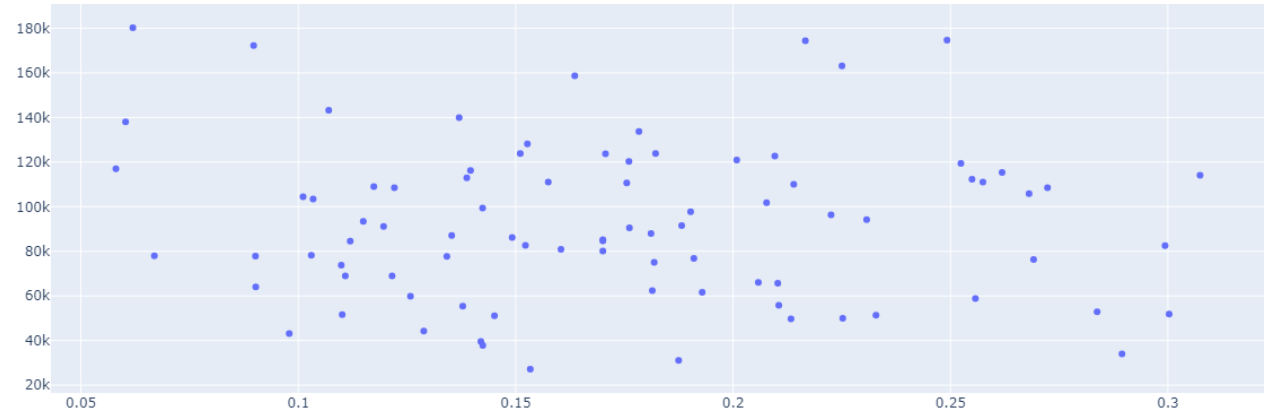
```
          store      move     age60
store  1.000000  0.020054 -0.225349
move   0.020054  1.000000 -0.282740
age60 -0.225349 -0.282740  1.000000
```

Есть очень слабо выраженный тренд к снижению спроса на сигареты при увеличении доли пожилых клиентов

Анальгетики

```
df_1 = df_ana_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'age60']]
```

```
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['age60'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df.corr()
```

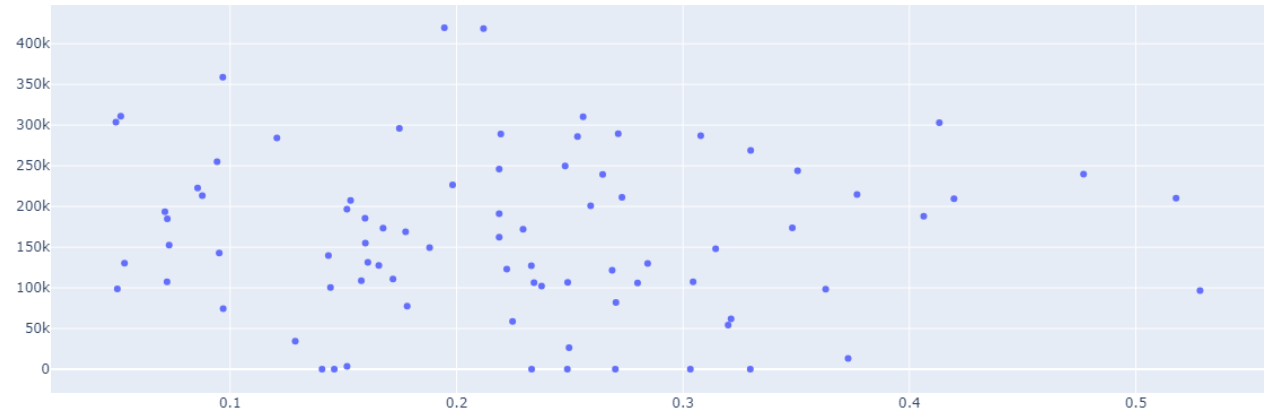
	store	move	age60
store	1.000000	0.264982	-0.225349
move	0.264982	1.000000	-0.061734
age60	-0.225349	-0.061734	1.000000

Нет явной зависимости между % пожилых клиентов и спросом на анальгетики

Зависимость спроса от % выпускников колледжей

Пиво

```
df_1 = df_beer_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'educ']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['educ'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



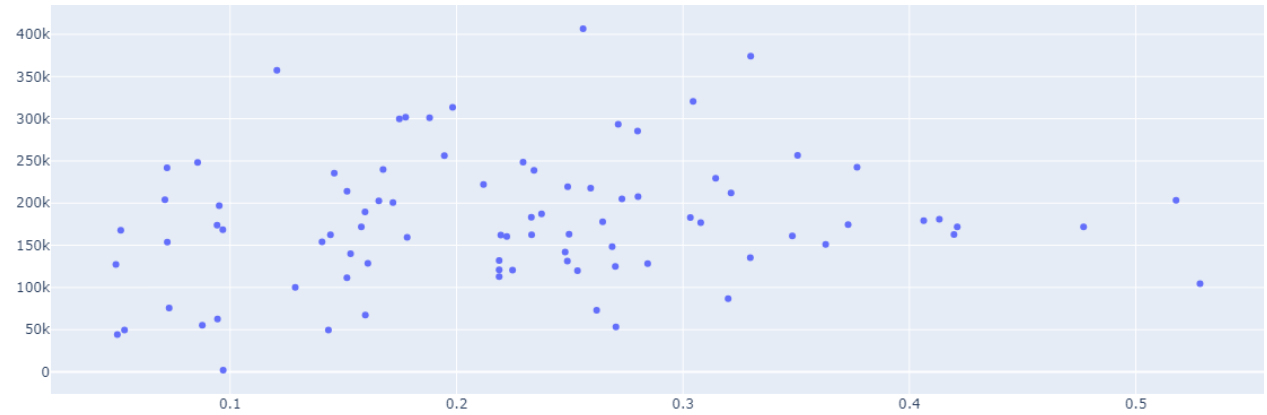

```
df.corr()
```

	store	move	educ
store	1.000000	0.383751	0.054774
move	0.383751	1.000000	-0.043535
educ	0.054774	-0.043535	1.000000

Нет явной зависимости между % выпускников колледжей и спросом на пиво

Сигареты

```
df_1 = df_cig_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'educ']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['educ'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



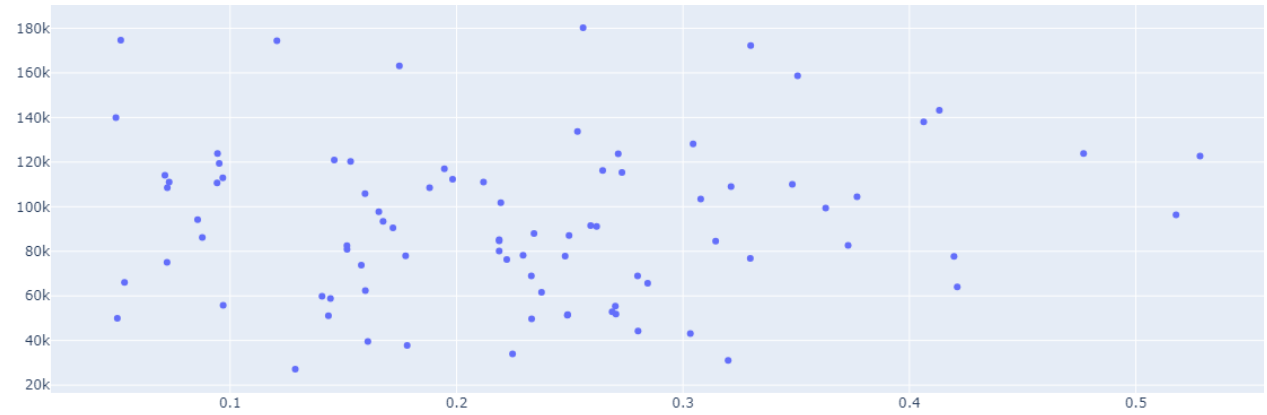

```
df.corr()
```

```
          store      move      educ
store  1.000000  0.020054  0.021790
move   0.020054  1.000000  0.161374
educ   0.021790  0.161374  1.000000
```

Нет явной зависимости между % выпускников колледжей и спросом на сигареты

Анальгетики

```
df_1 = df_ana_sales.groupby('store', as_index=False).agg({'move':
'sum'})
temp = df_demographic[['store', 'educ']]
df = pd.merge(df_1, temp, on="store")
fig = go.Figure()
fig.add_trace(go.Scatter(x=df['educ'], y=df['move'], mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df.corr()
```

```
          store      move      educ
store  1.000000  0.264982  0.021790
move   0.264982  1.000000  0.053149
educ   0.021790  0.053149  1.000000
```

Нет явной зависимости между % выпускников колледжей и спросом на анальгетики

Линейная регрессия на примере конкретного товара (пиво upc = 8248812345)

```
df = df_beer_sales[df_beer_sales['upc']==8248812345]
df = pd.merge(df, df_demographic, on="store")
```

```
df = df.drop(columns=['sale', 'upc', 'qty'])
X = df.drop(columns=['move'])
Y = df['move']
X.corr()
```

```
              store      week     price     age60      age9
educ  \
store      1.000000  0.008688  0.035365 -0.276633  0.390148  0.076376

week       0.008688  1.000000  0.112522 -0.008597 -0.000053  0.041580

price      0.035365  0.112522  1.000000  0.005449 -0.039586  0.068433

age60     -0.276633 -0.008597  0.005449  1.000000 -0.734259 -0.305879

age9       0.390148 -0.000053 -0.039586 -0.734259  1.000000  0.050401

educ       0.076376  0.041580  0.068433 -0.305879  0.050401  1.000000

ethnic     0.208755  0.000800 -0.040481 -0.082421  0.132283 -0.361520

income     0.019198  0.016499  0.004479 -0.152547  0.176661  0.664640

hhlarge    0.225209 -0.017693 -0.058597 -0.369565  0.769434 -0.382298

workwom    0.107533  0.022107  0.016898 -0.624314  0.188786  0.591283

hval150   -0.045022  0.035257  0.097358 -0.115618 -0.151605  0.900905

sstrdist   0.281345  0.016417 -0.047439  0.110173  0.038033 -0.163728

sstrvol    0.113356 -0.015663 -0.095884 -0.039692 -0.048506 -0.101658

cpdist5    0.066319 -0.024763 -0.006323  0.081205  0.161209 -0.149841

cpwvol5   -0.151670 -0.014301 -0.073238 -0.044344 -0.135985  0.288784


              ethnic    income   hhlarge   workwom   hval150
sstrdist  \
store      0.208755  0.019198  0.225209  0.107533 -0.045022  0.281345

week       0.000800  0.016499 -0.017693  0.022107  0.035257  0.016417

price     -0.040481  0.004479 -0.058597  0.016898  0.097358 -0.047439

age60     -0.082421 -0.152547 -0.369565 -0.624314 -0.115618  0.110173

age9       0.132283  0.176661  0.769434  0.188786 -0.151605  0.038033

educ      -0.361520  0.664640 -0.382298  0.591283  0.900905 -0.163728
```

```
ethnic    1.000000 -0.722668  0.239511 -0.275051 -0.424433  0.593730

income   -0.722668  1.000000 -0.056781  0.390841  0.636966 -0.444346

hhlarge   0.239511 -0.056781  1.000000 -0.254306 -0.501181  0.045185

workwom  -0.275051  0.390841 -0.254306  1.000000  0.490324 -0.258248

hval150  -0.424433  0.636966 -0.501181  0.490324  1.000000 -0.212989

sstrdist  0.593730 -0.444346  0.045185 -0.258248 -0.212989  1.000000

sstrvol   0.254793 -0.302700  0.007793 -0.101719 -0.198287  0.253588

cpdist5  -0.260968  0.284439  0.258270 -0.148635 -0.156733 -0.062091

cpwvol5  -0.354307  0.329812 -0.176962  0.264280  0.280842 -0.423518
```

```
            sstrvol   cpdist5   cpwvol5
store      0.113356  0.066319 -0.151670
week      -0.015663 -0.024763 -0.014301
price     -0.095884 -0.006323 -0.073238
age60     -0.039692  0.081205 -0.044344
age9      -0.048506  0.161209 -0.135985
educ      -0.101658 -0.149841  0.288784
ethnic     0.254793 -0.260968 -0.354307
income    -0.302700  0.284439  0.329812
hhlarge    0.007793  0.258270 -0.176962
workwom   -0.101719 -0.148635  0.264280
hval150   -0.198287 -0.156733  0.280842
sstrdist   0.253588 -0.062091 -0.423518
sstrvol    1.000000 -0.088759  0.367627
cpdist5   -0.088759  1.000000  0.054945
cpwvol5    0.367627  0.054945  1.000000
```

```
def correlation(dataset, threshold): # удалим колонки, сильно
коррелирующие между собой (чтобы избежать мультиколлинеарности)
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if (corr_matrix.iloc[i, j] >= threshold) and
(corr_matrix.columns[j] not in col_corr):
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
                if colname in dataset.columns:
                    del dataset[colname]

correlation(X, 0.7)
```

```
X.corr()

              store      week     price     age60      age9
educ  \
store     1.000000  0.008688  0.035365 -0.276633  0.390148  0.076376

week      0.008688  1.000000  0.112522 -0.008597 -0.000053  0.041580

price     0.035365  0.112522  1.000000  0.005449 -0.039586  0.068433

age60    -0.276633 -0.008597  0.005449  1.000000 -0.734259 -0.305879

age9      0.390148 -0.000053 -0.039586 -0.734259  1.000000  0.050401

educ      0.076376  0.041580  0.068433 -0.305879  0.050401  1.000000

ethnic    0.208755  0.000800 -0.040481 -0.082421  0.132283 -0.361520

income    0.019198  0.016499  0.004479 -0.152547  0.176661  0.664640

workwom   0.107533  0.022107  0.016898 -0.624314  0.188786  0.591283

sstrdist  0.281345  0.016417 -0.047439  0.110173  0.038033 -0.163728

sstrvol   0.113356 -0.015663 -0.095884 -0.039692 -0.048506 -0.101658

cpdist5   0.066319 -0.024763 -0.006323  0.081205  0.161209 -0.149841

cpwvol5  -0.151670 -0.014301 -0.073238 -0.044344 -0.135985  0.288784


              ethnic    income   workwom  sstrdist   sstrvol   cpdist5
cpwvol5
store     0.208755  0.019198  0.107533  0.281345  0.113356  0.066319 -
0.151670
week      0.000800  0.016499  0.022107  0.016417 -0.015663 -0.024763 -
0.014301
price    -0.040481  0.004479  0.016898 -0.047439 -0.095884 -0.006323 -
0.073238
age60    -0.082421 -0.152547 -0.624314  0.110173 -0.039692  0.081205 -
0.044344
age9      0.132283  0.176661  0.188786  0.038033 -0.048506  0.161209 -
0.135985
educ     -0.361520  0.664640  0.591283 -0.163728 -0.101658 -0.149841
0.288784
ethnic    1.000000 -0.722668 -0.275051  0.593730  0.254793 -0.260968 -
0.354307
income   -0.722668  1.000000  0.390841 -0.444346 -0.302700  0.284439
0.329812
workwom  -0.275051  0.390841  1.000000 -0.258248 -0.101719 -0.148635
0.264280
```

```
sstrdist  0.593730 -0.444346 -0.258248  1.000000  0.253588 -0.062091 -
0.423518
sstrvol   0.254793 -0.302700 -0.101719  0.253588  1.000000 -0.088759
0.367627
cpdist5  -0.260968  0.284439 -0.148635 -0.062091 -0.088759  1.000000
0.054945
cpwvol5  -0.354307  0.329812  0.264280 -0.423518  0.367627  0.054945
1.000000
```

```python
import statsmodels.api as sm
import numpy as np
y = Y.values
x = []
for i in X.columns:
    x.append(X[i])
def reg_m(y, x):
    ones = np.ones(len(x[0]))
    X = sm.add_constant(np.column_stack((x[0], ones)))
    for ele in x[1:]:
        X = sm.add_constant(np.column_stack((ele, X)))
    results = sm.OLS(y, X).fit()
    return results
print(reg_m(y, x).summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                          y   R-squared:
0.122
Model:                                OLS   Adj. R-squared:
0.121
Method:                     Least Squares   F-statistic:
227.0
Date:                   Sun, 15 Dec 2024   Prob (F-statistic):
0.00
Time:                           23:36:13   Log-Likelihood:
-85332.
No. Observations:                  21271   AIC:
1.707e+05
Df Residuals:                      21257   BIC:
1.708e+05
Df Model:                             13

Covariance Type:                nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
```

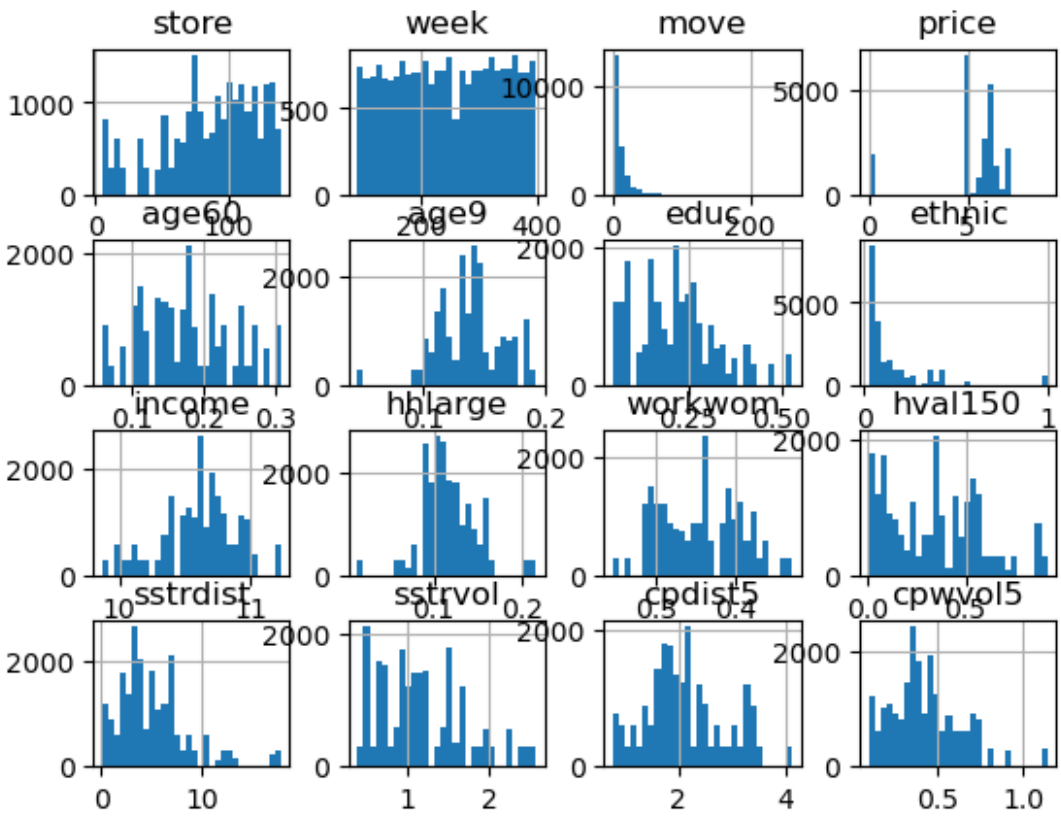
```
------------------------------------------------------------------------
--------
x1             -2.3513      0.670     -3.510      0.000      -3.664
-1.038
x2              0.1270      0.155      0.818      0.413      -0.177
0.431
x3             -2.6807      0.245    -10.943      0.000      -3.161
-2.201
x4              0.2500      0.040      6.206      0.000       0.171
0.329
x5            -51.7712      3.341    -15.494      0.000     -58.321
-45.222
x6            -23.7752      0.775    -30.689      0.000     -25.294
-22.257
x7            -20.9635      0.888    -23.595      0.000     -22.705
-19.222
x8             44.4787      1.608     27.665      0.000      41.327
47.630
x9           -121.8601      7.487    -16.276      0.000    -136.535
-107.185
x10           -44.4836      3.667    -12.132      0.000     -51.670
-37.297
x11             0.1238      0.052      2.374      0.018       0.022
0.226
x12            -0.0080      0.001     -7.734      0.000      -0.010
-0.006
x13             0.0323      0.003     10.753      0.000       0.026
0.038
const         300.6864      7.778     38.659      0.000     285.441
315.932
==============================================================================
========
Omnibus:                    17872.573   Durbin-Watson:
1.119
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
835690.542
Skew:                           3.786   Prob(JB):
0.00
Kurtosis:                      32.758   Cond. No.
2.58e+04
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.58e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Качество модели линейной регрессии очень плохое (R-squared = 0.122)

Исследование факторов на примере конкретного товара (пиво upc = 8248812345)

```
df.hist(bins = 30)
```

```
array([[<Axes: title={'center': 'store'}>,
        <Axes: title={'center': 'week'}>,
        <Axes: title={'center': 'move'}>,
        <Axes: title={'center': 'price'}>],
       [<Axes: title={'center': 'age60'}>,
        <Axes: title={'center': 'age9'}>,
        <Axes: title={'center': 'educ'}>,
        <Axes: title={'center': 'ethnic'}>],
       [<Axes: title={'center': 'income'}>,
        <Axes: title={'center': 'hhlarge'}>,
        <Axes: title={'center': 'workwom'}>,
        <Axes: title={'center': 'hval150'}>],
       [<Axes: title={'center': 'sstrdist'}>,
        <Axes: title={'center': 'sstrvol'}>,
        <Axes: title={'center': 'cpdist5'}>,
        <Axes: title={'center': 'cpwvol5'}>]], dtype=object)
```




```
from scipy import stats
for i in df.columns:
    alpha = 0.1
    stats_x, p_value_x = stats.shapiro(df[i][0:5000])
    if p_value_x < alpha:
        print(f"Выборка {i} не нормальна")
    else:
        print(f"Нельзя отклонить нулевую гипотезу о нормальности для
{i}")
```

```
Выборка store не нормальна
Выборка week не нормальна
Выборка move не нормальна
Выборка price не нормальна
Выборка age60 не нормальна
Выборка age9 не нормальна
Выборка educ не нормальна
Выборка ethnic не нормальна
Выборка income не нормальна
Выборка hhlarge не нормальна
Выборка workwom не нормальна
Выборка hval150 не нормальна
Выборка sstrdist не нормальна
Выборка sstrvol не нормальна
```

```
Выборка cpdist5 не нормальна
Выборка cpwvol5 не нормальна
```

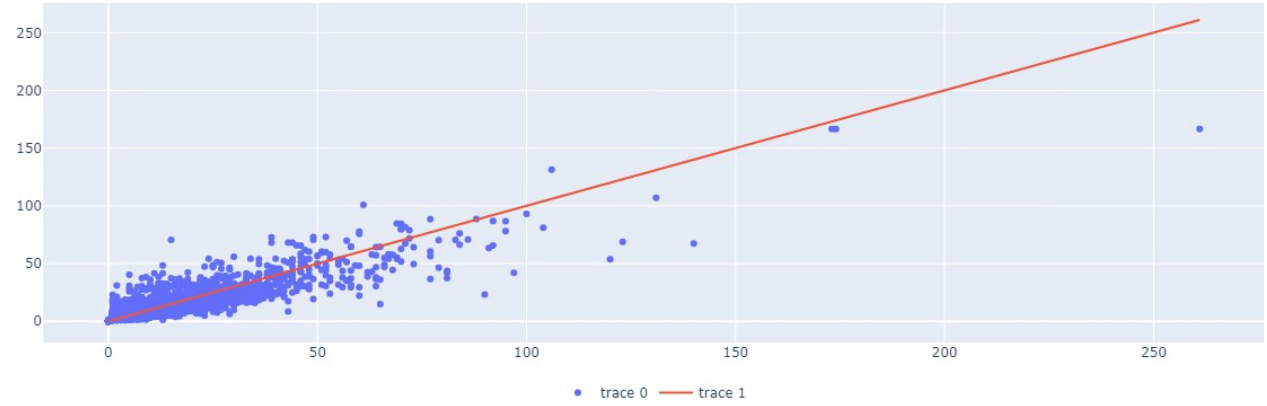
Видно, что все факторы распределены ненормально (согласно тесту Шапиро-Уилка), поэтому модель линейной регрессии не даст хорошего прогноза на таких данных, в чем мы убедились ранее на построенной модели и гистограммах распределения

Метод k ближайших соседей на примере конкретного товара (пиво upc = 8248812345)

```
df = df_beer_sales[df_beer_sales['upc']==8248812345]
df = pd.merge(df, df_demographic, on="store")

df = df.drop(columns=['sale', 'upc', 'qty'])
X = df.drop(columns=['move'])
Y = df['move']

from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsRegressor # Метод ближайших
соседей
from sklearn.preprocessing import StandardScaler # Стандартизация
from sklearn.model_selection import train_test_split # Кросс-валидация
from sklearn.metrics import r2_score, mean_squared_error

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    random_state=0,
                                                    test_size=0.2)

def kernel(distances, h=1): # перевзвешиваем таргеты соседей таким
образом, чтобы ближайшие давали больший вклад, чем самые далекие
    const = 1 / (np.sqrt(2 * np.pi))
    power = (-1/2) * ((distances)**2) / h**2
    return const * np.exp(power)


pipe = Pipeline([('scaler', StandardScaler()),
                 ('KNN', KNeighborsRegressor(n_neighbors = 4,
weights=kernel))])

pipe.fit(X_train, Y_train)
Y_pred = pipe.predict(X_test)

r2 = r2_score(Y_test, Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
print(f'R-squared: {r2}, MSE = {MSE}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=Y_test, y=Y_pred, mode='markers'))
```

```
fig.add_trace(go.Scatter(x=Y_test, y=Y_test, mode='lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

```
R-squared: 0.785470809217751, MSE = 45.90591752380458
```



Видно, что метод k ближайших соседей работает неплохо

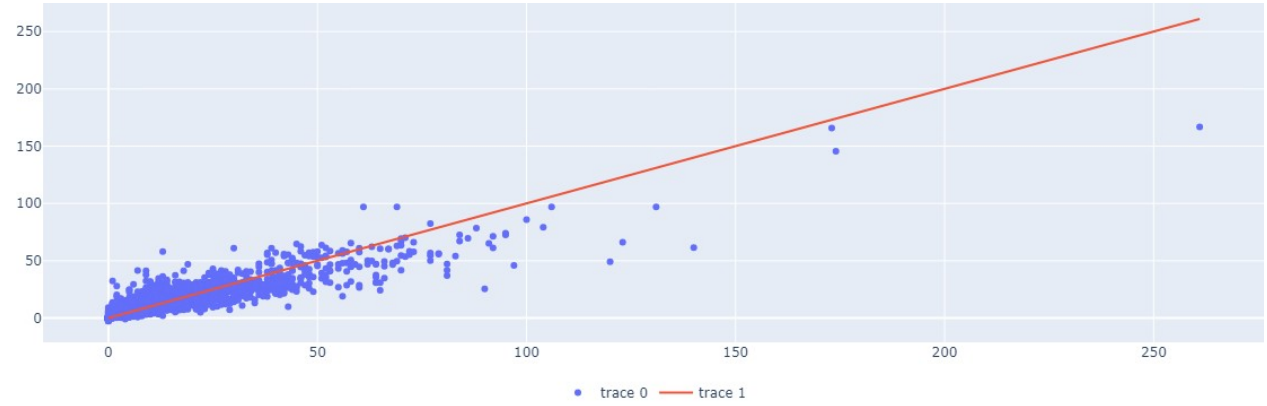
```
# проверка статистической значимости прогнозов
alpha = 0.001
t_score, p_value = stats.mannwhitneyu(Y_test, Y_pred)
print(t_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, распределения выборок
различаются значительно ")
else:
    print("Нельзя отклонить нулевую гипотезу, распределения выборок
различаются незначительно")

l_score, p_value = stats.levene(Y_test, Y_pred, center = 'median')
print(l_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, дисперсии выборок различаются
значительно")
else:
    print("Нельзя отклонить нулевую гипотезу, дисперсии выборок
различаются незначительно")
```

```
8731160.0 0.004550759614773671
Нельзя отклонить нулевую гипотезу, распределения выборок различаются
незначительно
3.2450548168267 0.07167468544012792
Нельзя отклонить нулевую гипотезу, дисперсии выборок различаются
незначительно
```

Градиентный бустинг на примере конкретного товара (пиво upc = 8248812345)

```
df = df_beer_sales[df_beer_sales['upc']==8248812345]
df = pd.merge(df, df_demographic, on="store")

df = df.drop(columns=['sale', 'upc', 'qty'])
X = df.drop(columns=['move'])
Y = df['move']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    random_state=0,
                                                    test_size=0.2)

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt


# Step 3: Defining parameter grid
param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5]
}

gbr = GradientBoostingRegressor()
#Initializing Grid Search
grid_search = GridSearchCV(estimator=gbr, param_grid=param_grid, cv=5,
scoring='neg_mean_squared_error')
grid_search.fit(X_train, Y_train)
print(f'Best parameters found: {grid_search.best_params_}')

best_model = grid_search.best_estimator_
Y_pred = best_model.predict(X_test)

r2 = r2_score(Y_test, Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
print(f'R-squared: {r2}, MSE = {MSE}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=Y_test, y=Y_pred, mode='markers'))
fig.add_trace(go.Scatter(x=Y_test, y=Y_test, mode='lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

```
Best parameters found: {'learning_rate': 0.1, 'max_depth': 5,
'n_estimators': 200}
R-squared: 0.8031113942235528, MSE = 42.131106098864414
```



Видно, что градиентный бустинг работает неплохо

```
# проверка статистической значимости прогнозов
alpha = 0.001
t_score, p_value = stats.mannwhitneyu(Y_test, Y_pred)
print(t_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, распределения выборок
различаются значительно ")
else:
    print("Нельзя отклонить нулевую гипотезу, распределения выборок
различаются незначительно")

l_score, p_value = stats.levene(Y_test, Y_pred, center = 'median')
print(l_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, дисперсии выборок различаются
значительно")
else:
    print("Нельзя отклонить нулевую гипотезу, дисперсии выборок
различаются незначительно")
```

```
8517355.0 2.3176428695760536e-06
Отклонить нулевую гипотезу, распределения выборок различаются
значительно
8.021818900124142 0.0046325727222666395
Нельзя отклонить нулевую гипотезу, дисперсии выборок различаются
незначительно
```

Случайный лес на примере конкретного товара (пиво upc = 8248812345)

```
df = df_beer_sales[df_beer_sales['upc']==8248812345]
df = pd.merge(df, df_demographic, on="store")

df = df.drop(columns=['sale', 'upc', 'qty'])
X = df.drop(columns=['move'])
Y = df['move']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    random_state=0,
                                                    test_size=0.2)

np.random.seed = 1

param_grid = {
    "random_forest__max_depth": [10, 15, 20],
    "random_forest__min_samples_split": [2, 5, 10],
    "random_forest__min_samples_leaf": [1, 3, 5]
}

custom_cv = [(X_train.index.to_list(), X_test.index.to_list())]

pipe = Pipeline([('scaler', StandardScaler()),
                 ("random_forest",
RandomForestRegressor(random_state=1))])

search = GridSearchCV(pipe,
                      param_grid,
                      cv=custom_cv,
                      scoring='neg_mean_squared_error',
                      verbose=10)

search.fit(X, Y)

print(f"Best parameter (CV score={search.best_score_:.5f}):")
print(search.best_params_)
```

```
Fitting 1 folds for each of 27 candidates, totalling 27 fits
[CV 1/1; 1/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=2
[CV 1/1; 1/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=2;, score=-46.175 total time=   2.8s
[CV 1/1; 2/27] START random_forest__max_depth=10,
```

```
random_forest__min_samples_leaf=1, random_forest__min_samples_split=5
[CV 1/1; 2/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=5;, score=-46.223 total time=   2.6s
[CV 1/1; 3/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=10
[CV 1/1; 3/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=10;, score=-46.434 total time=   2.6s
[CV 1/1; 4/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=2
[CV 1/1; 4/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=2;, score=-47.292 total time=   2.6s
[CV 1/1; 5/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=5
[CV 1/1; 5/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=5;, score=-47.292 total time=   2.6s
[CV 1/1; 6/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=10
[CV 1/1; 6/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=10;, score=-47.624 total time=   2.6s
[CV 1/1; 7/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=2
[CV 1/1; 7/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=2;, score=-49.445 total time=   2.6s
[CV 1/1; 8/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=5
[CV 1/1; 8/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=5;, score=-49.445 total time=   2.6s
[CV 1/1; 9/27] START random_forest__max_depth=10,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=10
[CV 1/1; 9/27] END random_forest__max_depth=10,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=10;, score=-49.445 total time=   2.6s
[CV 1/1; 10/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=2
[CV 1/1; 10/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=2;, score=-38.771 total time=   3.9s
[CV 1/1; 11/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=5
[CV 1/1; 11/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=5;, score=-39.389 total time=   3.7s
```

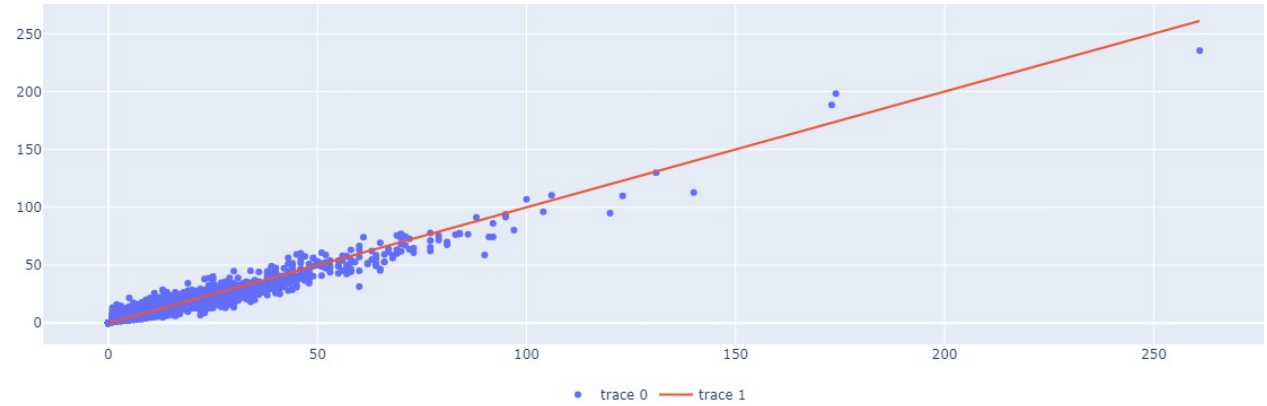
```
[CV 1/1; 12/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=10
[CV 1/1; 12/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=10;, score=-40.367 total time=   3.5s
[CV 1/1; 13/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=2
[CV 1/1; 13/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=2;, score=-41.392 total time=   3.6s
[CV 1/1; 14/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=5
[CV 1/1; 14/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=5;, score=-41.392 total time=   3.7s
[CV 1/1; 15/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=10
[CV 1/1; 15/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=10;, score=-42.206 total time=   3.4s
[CV 1/1; 16/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=2
[CV 1/1; 16/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=2;, score=-44.541 total time=   3.4s
[CV 1/1; 17/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=5
[CV 1/1; 17/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=5;, score=-44.541 total time=   3.3s
[CV 1/1; 18/27] START random_forest__max_depth=15,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=10
[CV 1/1; 18/27] END random_forest__max_depth=15,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=10;, score=-44.541 total time=   3.4s
[CV 1/1; 19/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=2
[CV 1/1; 19/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=2;, score=-39.029 total time=   4.8s
[CV 1/1; 20/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=5
[CV 1/1; 20/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=1,
random_forest__min_samples_split=5;, score=-39.110 total time=   4.5s
[CV 1/1; 21/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=1, random_forest__min_samples_split=10
[CV 1/1; 21/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=1,
```

```
random_forest__min_samples_split=10;, score=-39.949 total time=   4.0s
[CV 1/1; 22/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=2
[CV 1/1; 22/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=2;, score=-41.241 total time=   4.2s
[CV 1/1; 23/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=5
[CV 1/1; 23/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=5;, score=-41.241 total time=   4.0s
[CV 1/1; 24/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=3, random_forest__min_samples_split=10
[CV 1/1; 24/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=3,
random_forest__min_samples_split=10;, score=-41.965 total time=   3.7s
[CV 1/1; 25/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=2
[CV 1/1; 25/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=2;, score=-44.395 total time=   3.6s
[CV 1/1; 26/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=5
[CV 1/1; 26/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=5;, score=-44.395 total time=   3.7s
[CV 1/1; 27/27] START random_forest__max_depth=20,
random_forest__min_samples_leaf=5, random_forest__min_samples_split=10
[CV 1/1; 27/27] END random_forest__max_depth=20,
random_forest__min_samples_leaf=5,
random_forest__min_samples_split=10;, score=-44.395 total time=   3.7s
Best parameter (CV score=-38.77098):
{'random_forest__max_depth': 15, 'random_forest__min_samples_leaf': 1,
'random_forest__min_samples_split': 2}
```

```
pipe = Pipeline([('scaler', StandardScaler()),
                 ("random_forest", RandomForestRegressor(max_depth=15,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         random_state=1))])
pipe.fit(X_train, Y_train)
Y_pred = search.best_estimator_.predict(X_test)
r2 = r2_score(Y_test, Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
print(f'R-squared: {r2}, MSE = {MSE}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=Y_test, y=Y_pred, mode='markers'))
fig.add_trace(go.Scatter(x=Y_test, y=Y_test, mode='lines'))
fig.update_layout(legend_orientation="h",
```

```
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

```
R-squared: 0.9369884984800164, MSE = 13.483483442416437
```



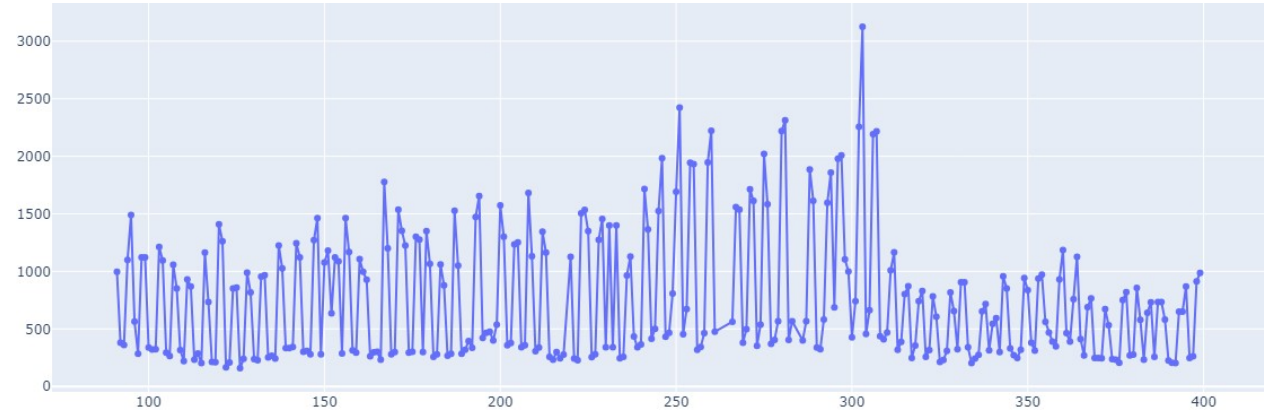
Видно, что случайный лес показал наилучший результат

```
# проверка статистической значимости прогнозов
alpha = 0.001
t_score, p_value = stats.mannwhitneyu(Y_test, Y_pred)
print(t_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, распределения выборок
различаются значительно ")
else:
    print("Нельзя отклонить нулевую гипотезу, распределения выборок
различаются незначительно")

l_score, p_value = stats.levene(Y_test, Y_pred, center = 'median')
print(l_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, дисперсии выборок различаются
значительно")
else:
    print("Нельзя отклонить нулевую гипотезу, дисперсии выборок
различаются незначительно")
```

```
8685455.5 0.0011922474034526409
Нельзя отклонить нулевую гипотезу, распределения выборок различаются
незначительно
4.680842288318482 0.03052853869319544
Нельзя отклонить нулевую гипотезу, дисперсии выборок различаются
незначительно
```

ARIMA на примере конкретного товара (пиво upc = 8248812345)

```
import warnings
import itertools
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm

df = df_beer_sales[df_beer_sales['upc']==8248812345]
df = pd.merge(df, df_demographic, on="store")
df_temp = df.groupby('week', as_index=False).agg({'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

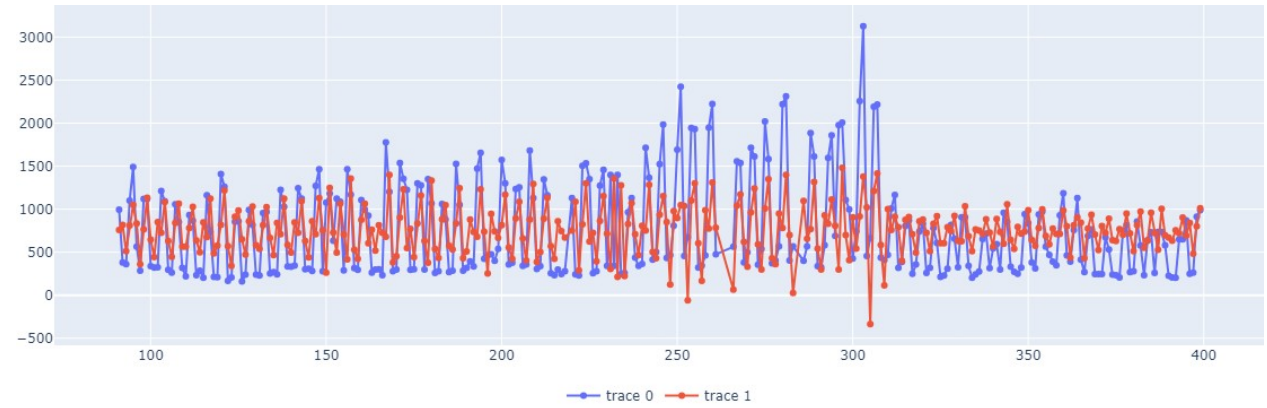



```
from statsmodels.tsa.stattools import adfuller

print('Результат теста:')
df_result = adfuller(df_temp['move'])
df_labels = ['ADF Test Statistic', 'p-value', '#Lags Used', 'Number of
Observations Used']
for result_value, label in zip(df_result, df_labels):
    print(label + ' : ' + str(result_value))

if df_result[1] <= 0.05:
    print("Сильные доказательства против нулевой гипотезы, ряд
является стационарным.")
else:
```

```
    print("Слабые доказательства против нулевой гипотезы, ряд не
является стационарным.")
```

```
Результат теста:
ADF Test Statistic : -2.8988347078059706
p-value : 0.04548367300272037
#Lags Used : 7
Number of Observations Used : 294
Сильные доказательства против нулевой гипотезы, ряд является
стационарным.
```

```
# Оценка модели ARIMA
p = 2  # Порядок авторегрессии (AR)
d = 0  # Порядок разностей (I)
q = 2  # Порядок скользящего среднего (MA)
model_arima = sm.tsa.ARIMA(df_temp['move'], order=(p, d, q)).fit()


forecast_arima = model_arima.predict(start=0,
end=len(df_temp['move']))

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers+lines'))
fig.add_trace(go.Scatter(x=df_temp['week'], y=forecast_arima,
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
Y_test = df_temp['move'][200:]
Y_pred = forecast_arima[201:]
r2 = r2_score(Y_test, Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
print(f'R-squared: {r2}, MSE = {MSE}')
```

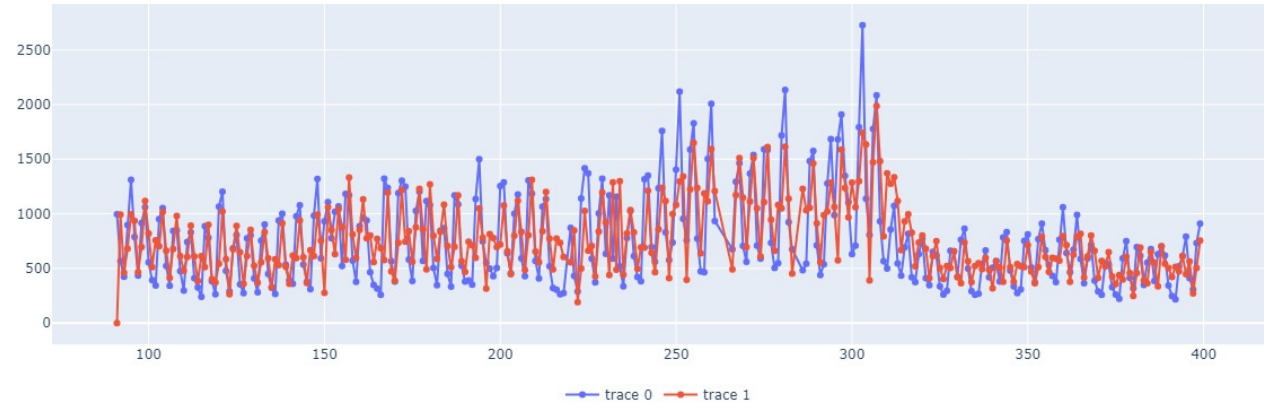
```
R-squared: 0.12782609599417194, MSE = 188407.30853361718
```

```
# проверка статистической значимости прогнозов
Y_test = df_temp['move']
Y_pred = forecast_arima

alpha = 0.001
t_score, p_value = stats.mannwhitneyu(Y_test, Y_pred)
print(t_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, распределения выборок
различаются значительно ")
else:
    print("Нельзя отклонить нулевую гипотезу, распределения выборок
различаются незначительно")

l_score, p_value = stats.levene(Y_test, Y_pred, center = 'median')
print(l_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, дисперсии выборок различаются
значительно")
else:
    print("Нельзя отклонить нулевую гипотезу, дисперсии выборок
различаются незначительно")
```

```
38928.0 0.00149998350263957
Нельзя отклонить нулевую гипотезу, распределения выборок различаются
незначительно
76.4478264135181 2.2444456270345027e-17
Отклонить нулевую гипотезу, дисперсии выборок различаются значительно
```

```
# Коэффициент сглаживания
alpha = 0.7

# Рассчитываем экспоненциальное скользящее среднее
df_temp['ema_move'] = df_temp['move'].ewm(alpha=alpha,
adjust=False).mean()

# Оценка модели ARIMA
p = 2  # Порядок авторегрессии (AR)
d = 1  # Порядок разностей (I)
q = 2  # Порядок скользящего среднего (MA)
model_arima = sm.tsa.ARIMA(df_temp['ema_move'], order=(p, d, q)).fit()


forecast_arima = model_arima.predict(start=0,
end=len(df_temp['ema_move']))

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_move'],
mode='markers+lines'))
```

```
fig.add_trace(go.Scatter(x=df_temp['week'], y=forecast_arima,
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



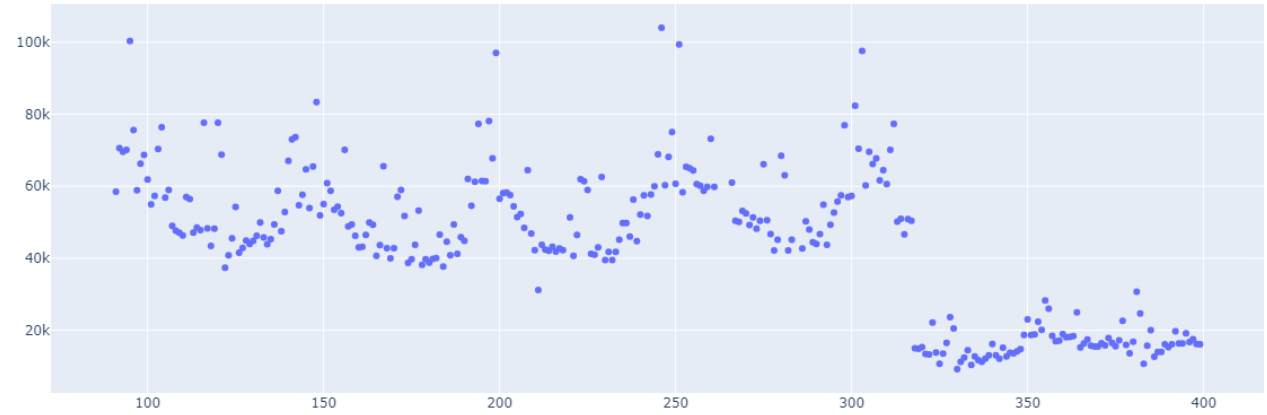

```
Y_test = df_temp['move'][200:]
Y_pred = forecast_arima[201:]
r2 = r2_score(Y_test, Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
print(f'R-squared: {r2}, MSE = {MSE}')

R-squared: 0.5232930871927368, MSE = 102978.39225511382

# проверка статистической значимости прогнозов
Y_test = df_temp['move']
Y_pred = forecast_arima

alpha = 0.001
t_score, p_value = stats.mannwhitneyu(Y_test, Y_pred)
print(t_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, распределения выборок
различаются значительно ")
else:
    print("Нельзя отклонить нулевую гипотезу, распределения выборок
различаются незначительно")

l_score, p_value = stats.levene(Y_test, Y_pred, center = 'median')
print(l_score, p_value)
if p_value < alpha:
    print("Отклонить нулевую гипотезу, дисперсии выборок различаются
значительно")
else:
    print("Нельзя отклонить нулевую гипотезу, дисперсии выборок
различаются незначительно")
```

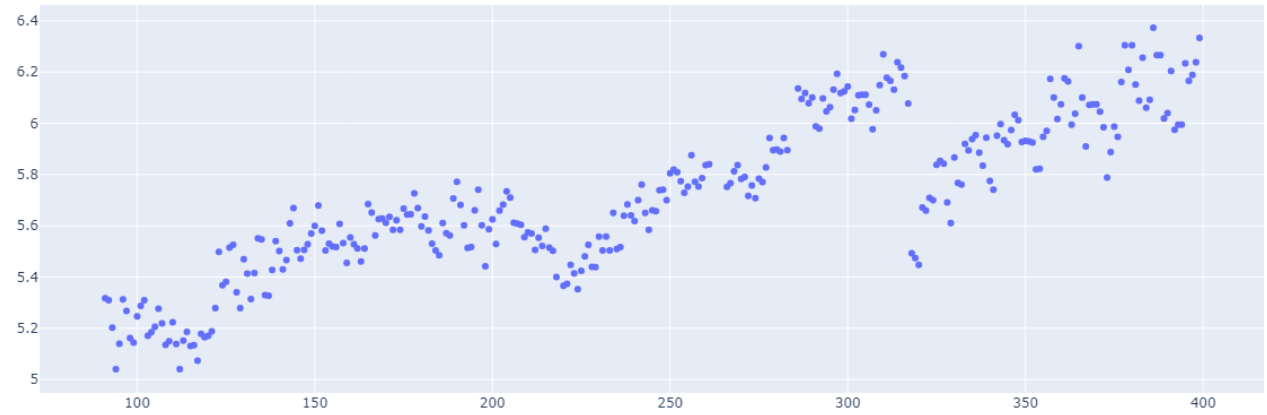
```
38696.0 0.0010285058825652118
Нельзя отклонить нулевую гипотезу, распределения выборок различаются незначительно
49.46695203738556 5.490976612654734e-12
Отклонить нулевую гипотезу, дисперсии выборок различаются значительно
```

Анализ эластичности

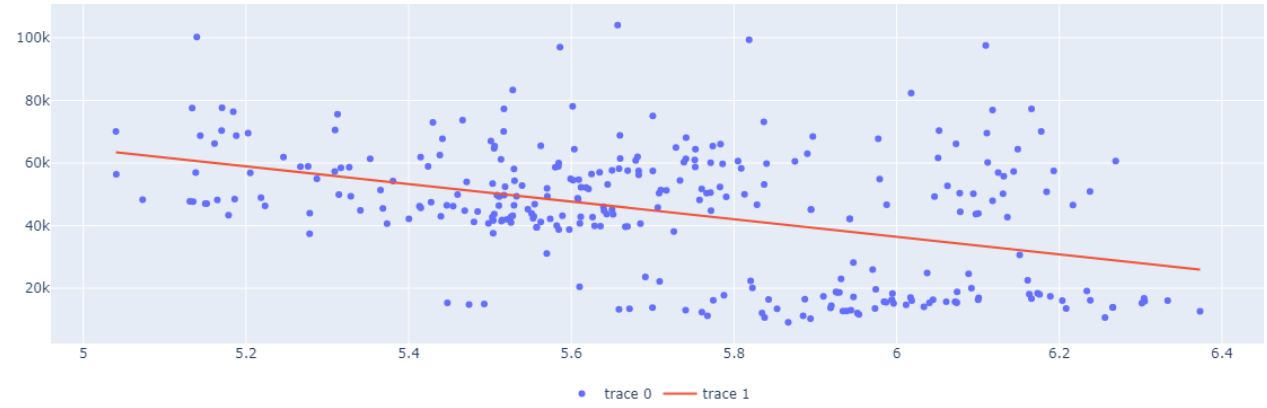
Пиво

```
df = df_beer_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)

df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['price'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```





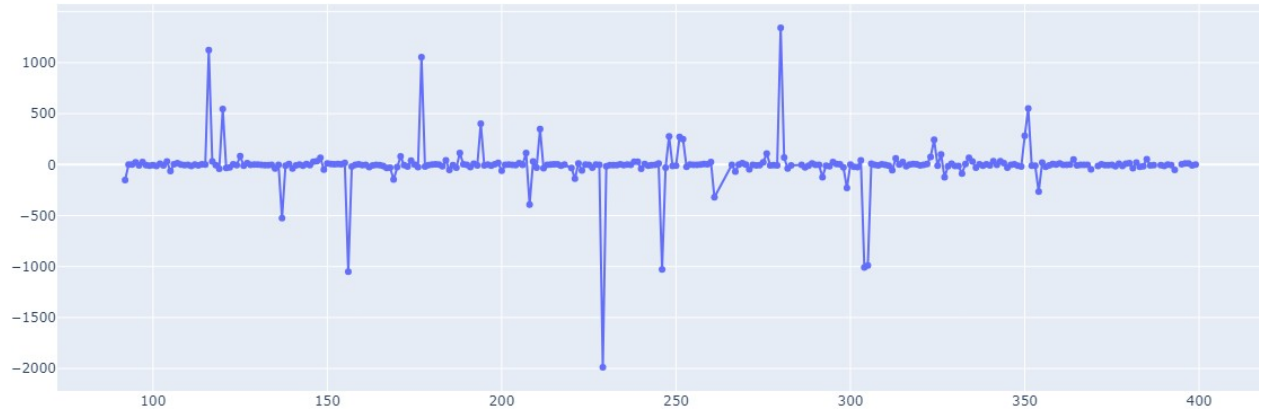

```
x = np.array(df_temp['price'])
y = np.array(df_temp['move'])
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

markers = go.Scatter(x=df_temp['price'], y=df_temp['move'],
mode='markers')
trendline = go.Scatter(x=df_temp['price'], y=p(x), mode='lines')

fig = go.Figure(data=[markers, trendline])
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df_temp["% Change in move"] = df_temp["move"].pct_change()
df_temp["% Change in price"] = df_temp["price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]
print(f'Среднее значение эластичности =
{df_temp["elasticity"].mean()}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
```

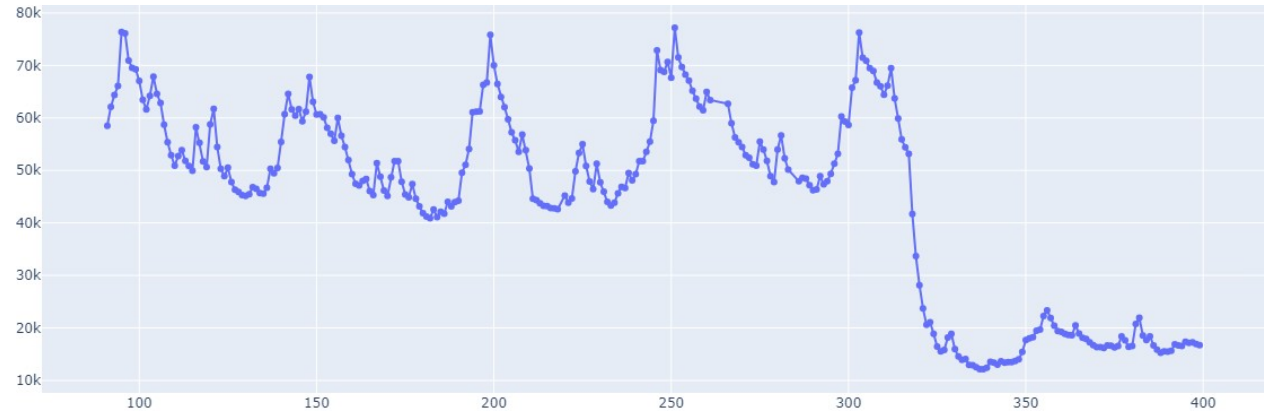
```
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

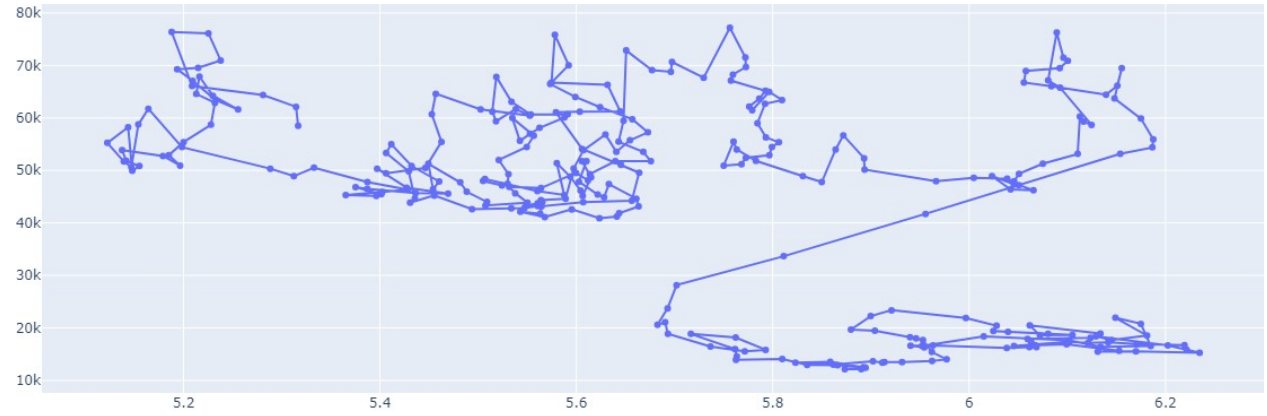
```
Среднее значение эластичности = -6.830950041712979
```



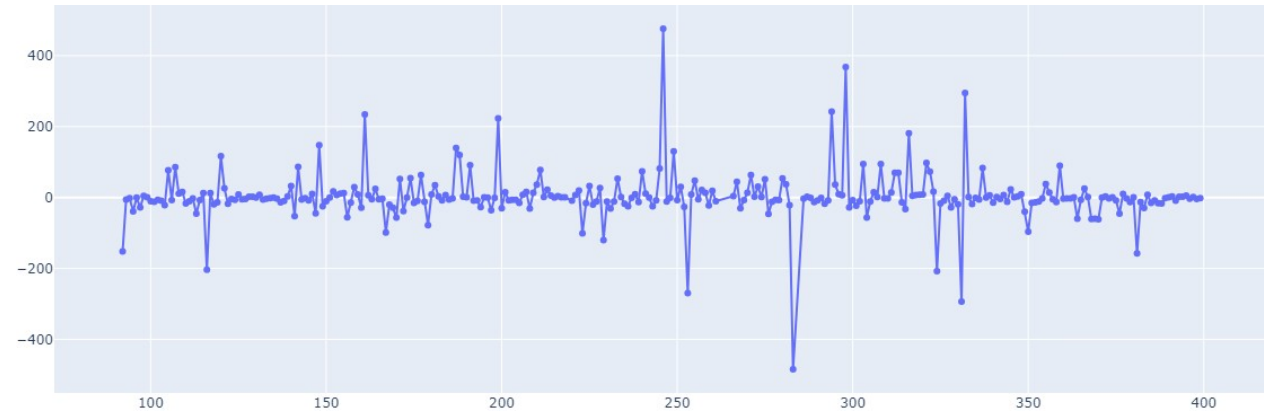

```
# через сглаживание
df = df_beer_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)
df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})


# Коэффициент сглаживания
alpha = 0.3

# Рассчитываем экспоненциальное скользящее среднее
df_temp['ema_move'] = df_temp['move'].ewm(alpha=alpha,
adjust=False).mean()
df_temp['ema_price'] = df_temp['price'].ewm(alpha=alpha,
adjust=False).mean()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_price'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
```

```
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['ema_price'],
y=df_temp['ema_move'], mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



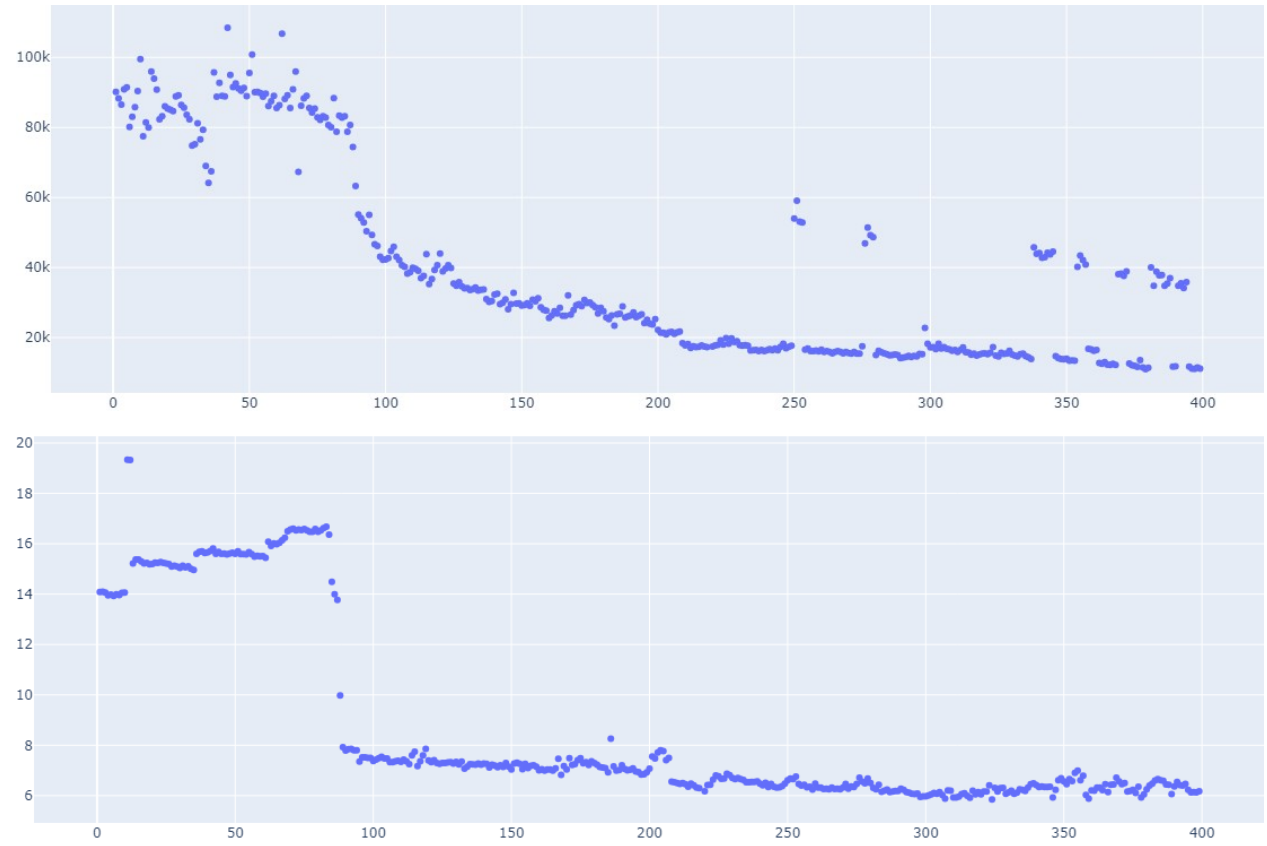


```
df_temp["% Change in move"] = df_temp["ema_move"].pct_change()
df_temp["% Change in price"] = df_temp["ema_price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

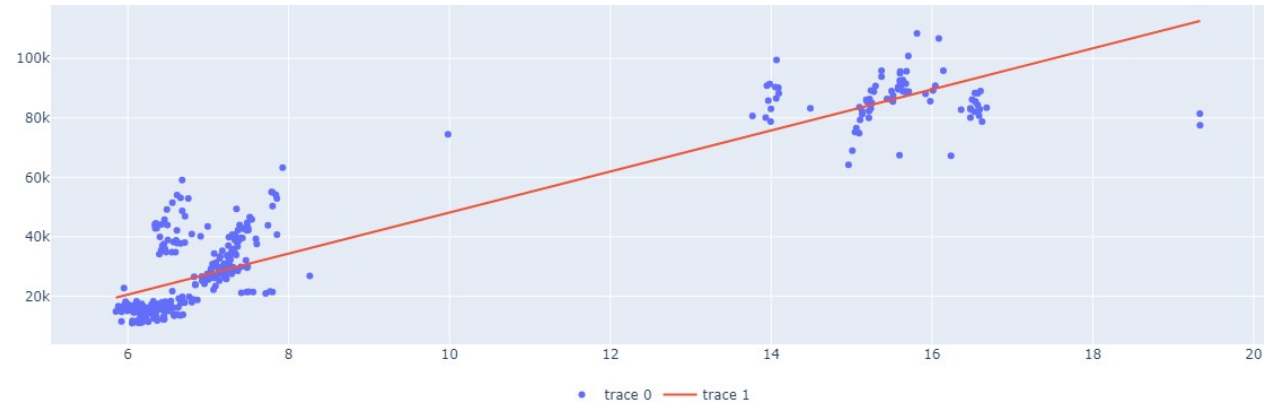


Сигареты

```
df = df_cig_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)

df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['price'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

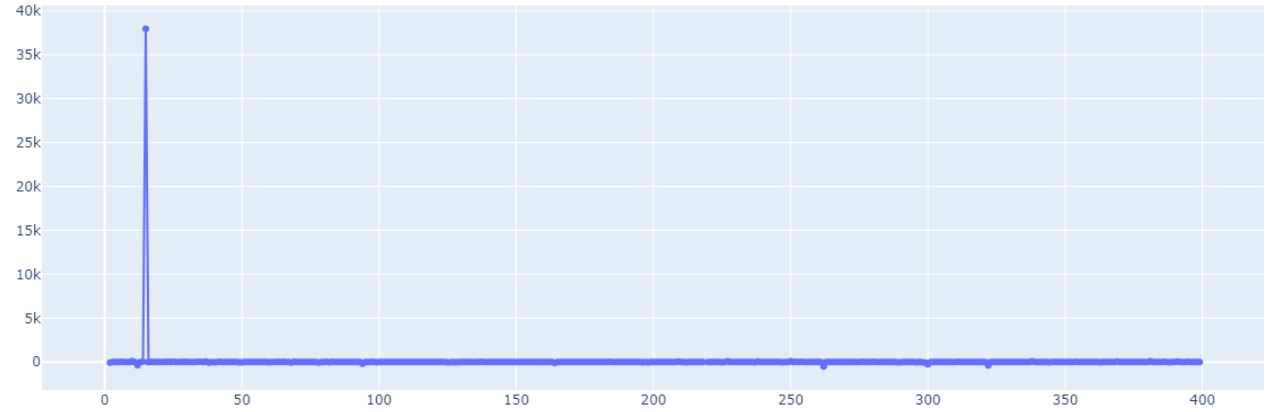



```
x = np.array(df_temp['price'])
y = np.array(df_temp['move'])
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

markers = go.Scatter(x=df_temp['price'], y=df_temp['move'],
mode='markers')
trendline = go.Scatter(x=df_temp['price'], y=p(x), mode='lines')

fig = go.Figure(data=[markers, trendline])
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



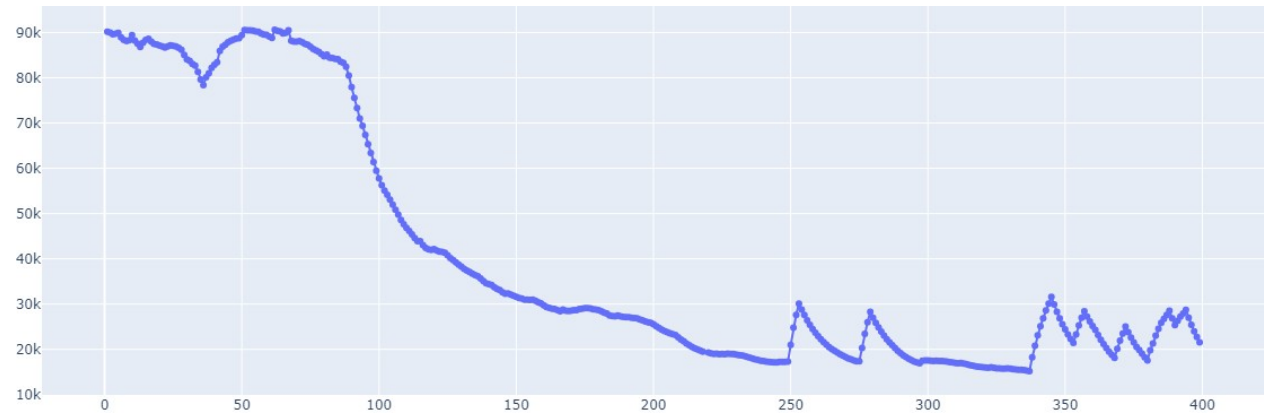

```
df_temp["% Change in move"] = df_temp["move"].pct_change()
df_temp["% Change in price"] = df_temp["price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]
print(f'Среднее значение эластичности =
{df_temp["elasticity"].mean()}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

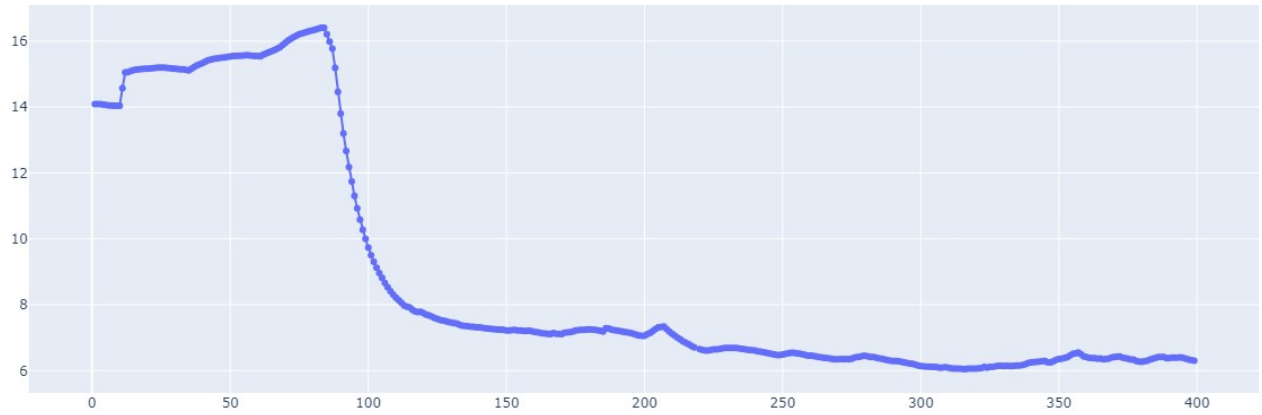
```
Среднее значение эластичности = 92.91670988985095
```

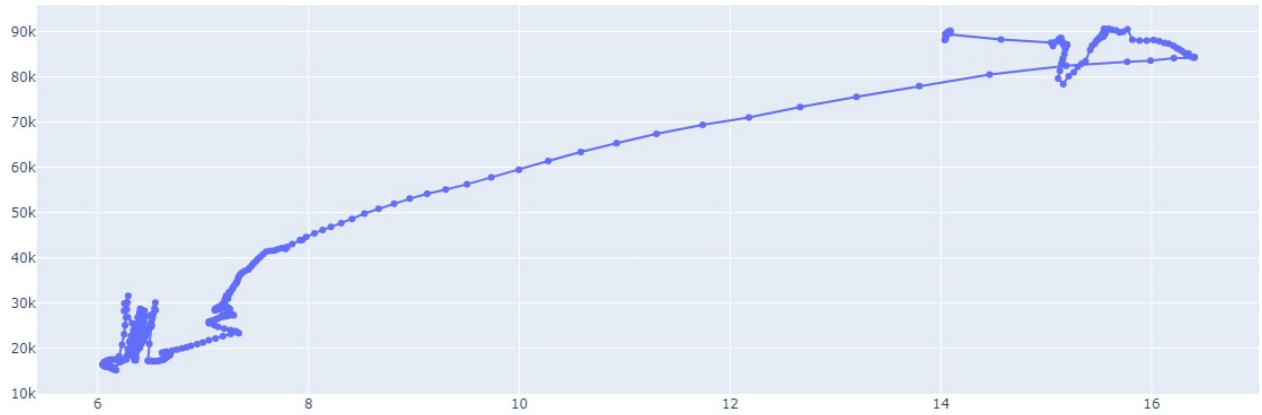



```
# через сглаживание
df = df_cig_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)
df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})
```

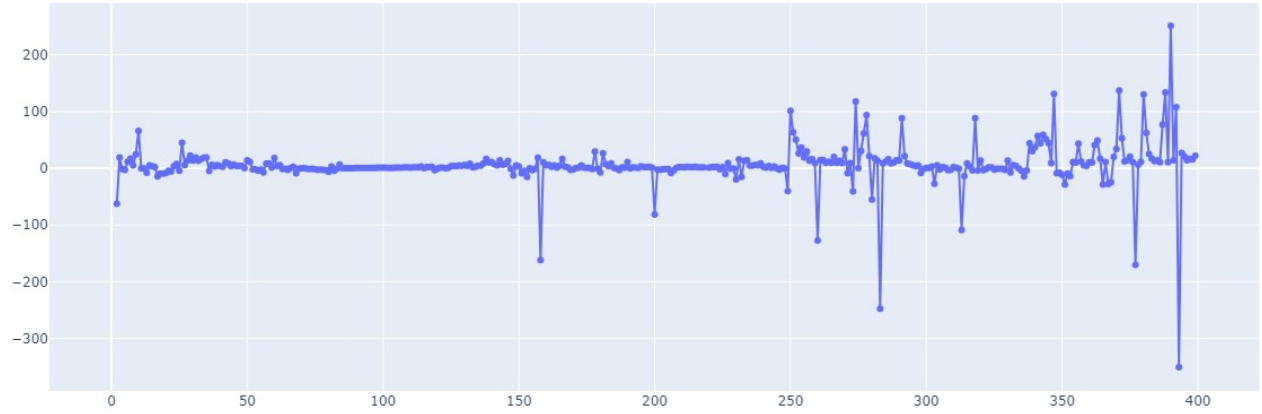
```
# Коэффициент сглаживания
alpha = 0.1

# Рассчитываем экспоненциальное скользящее среднее
df_temp['ema_move'] = df_temp['move'].ewm(alpha=alpha,
adjust=False).mean()
df_temp['ema_price'] = df_temp['price'].ewm(alpha=alpha,
adjust=False).mean()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_price'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['ema_price'],
y=df_temp['ema_move'], mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```





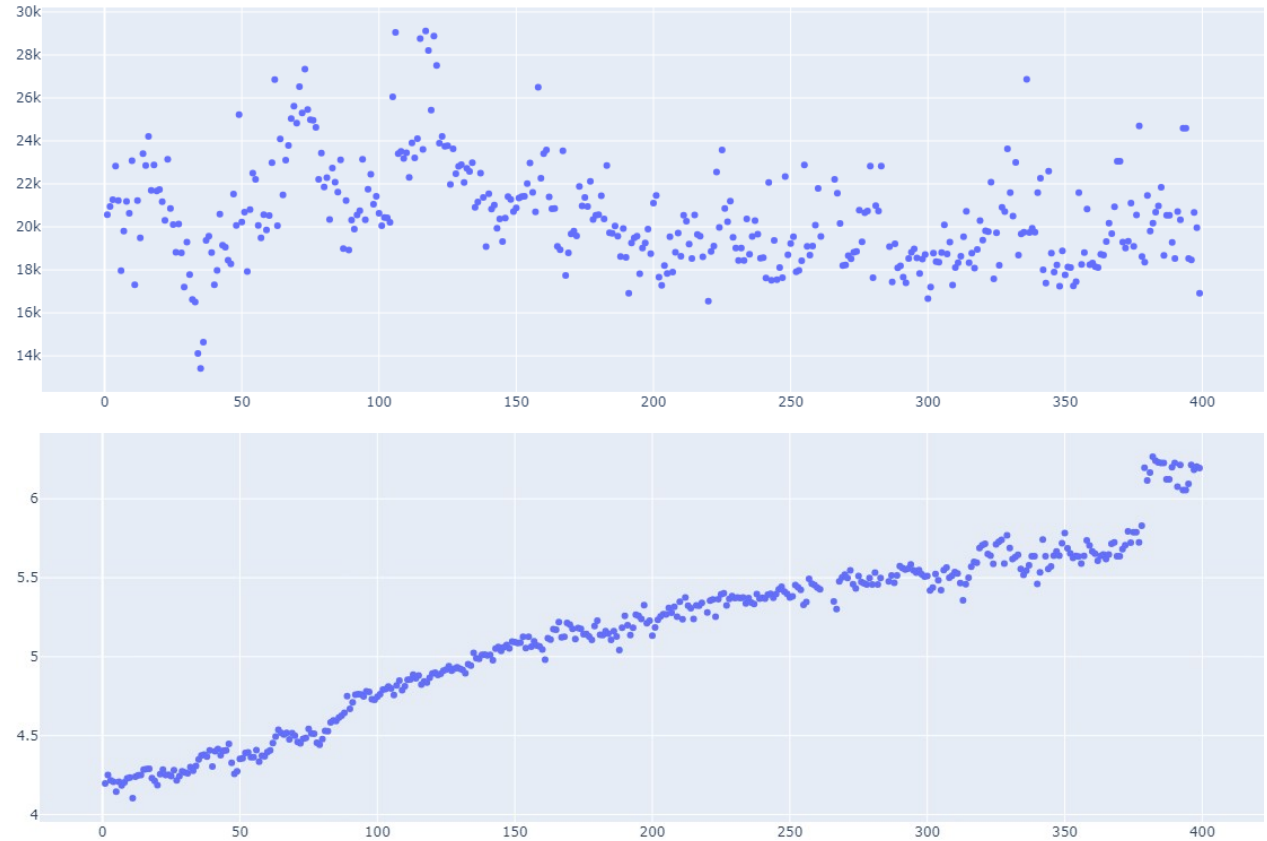



```
df_temp["% Change in move"] = df_temp["ema_move"].pct_change()
df_temp["% Change in price"] = df_temp["ema_price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

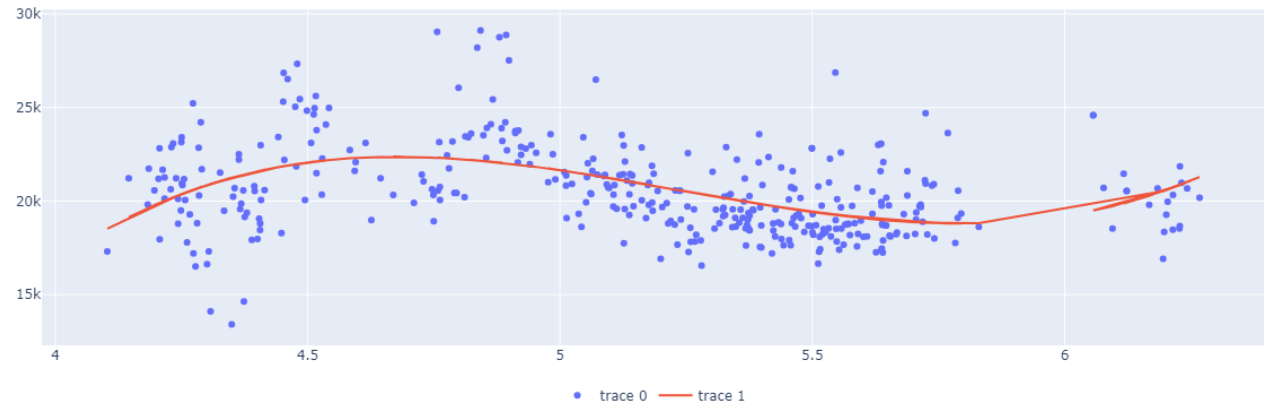


Анальгетики

```
df = df_ana_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)

df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['move'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['price'],
mode='markers'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

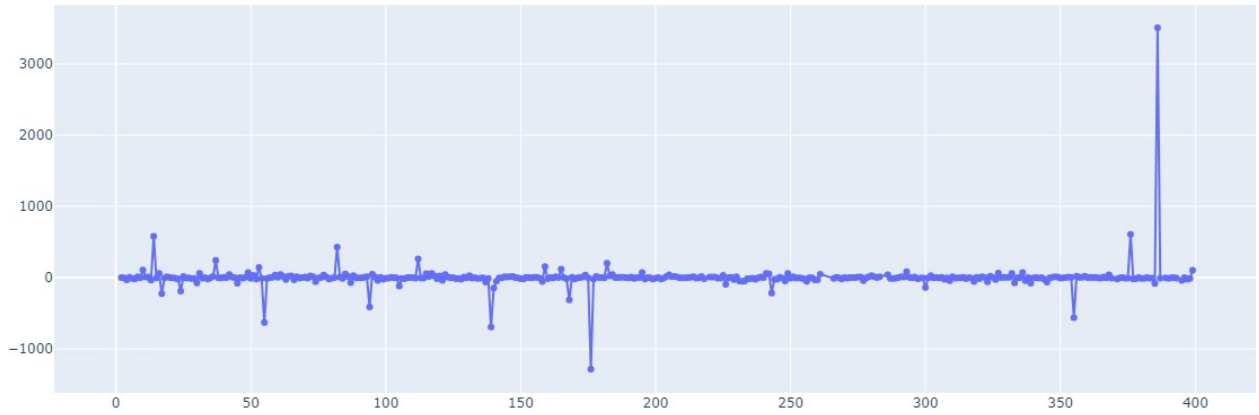


```
x = np.array(df_temp['price'])
y = np.array(df_temp['move'])
z = np.polyfit(x, y, 3)
p = np.poly1d(z)

markers = go.Scatter(x=df_temp['price'], y=df_temp['move'],
mode='markers')
trendline = go.Scatter(x=df_temp['price'], y=p(x), mode='lines')

fig = go.Figure(data=[markers, trendline])
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



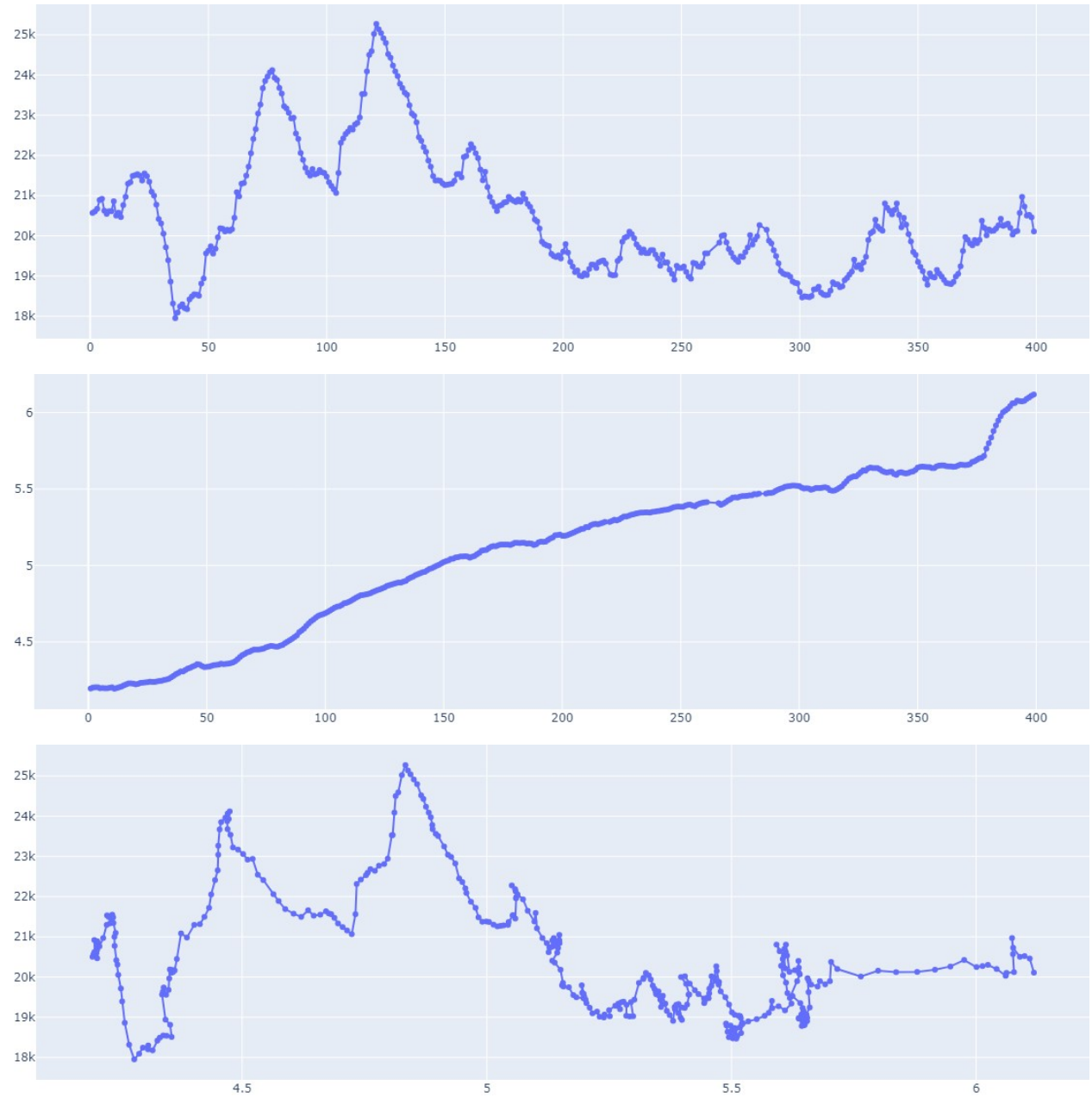

```
df_temp["% Change in move"] = df_temp["move"].pct_change()
df_temp["% Change in price"] = df_temp["price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]
print(f'Среднее значение эластичности =
{df_temp["elasticity"].mean()}')

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```

```
Среднее значение эластичности = 2.9376159955186236
```




```
# через сглаживание
df = df_ana_sales
df = df.drop(columns = ['upc', 'sale', 'qty', 'store'])
df = df.drop(df[df['move'] == 0].index)
df_temp = df.groupby('week', as_index=False).agg({'price': 'mean',
'move': 'sum'})


# Коэффициент сглаживания
alpha = 0.1

# Рассчитываем экспоненциальное скользящее среднее
df_temp['ema_move'] = df_temp['move'].ewm(alpha=alpha,
adjust=False).mean()
df_temp['ema_price'] = df_temp['price'].ewm(alpha=alpha,
adjust=False).mean()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_move'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['ema_price'],
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['ema_price'],
y=df_temp['ema_move'], mode='markers+lines'))
fig.update_layout(legend_orientation="h",
```

```
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```




```
df_temp["% Change in move"] = df_temp["ema_move"].pct_change()
df_temp["% Change in price"] = df_temp["ema_price"].pct_change()
df_temp["elasticity"] = df_temp["% Change in move"] / df_temp["%
Change in price"]

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_temp['week'], y=df_temp['elasticity'],
```

```
mode='markers+lines'))
fig.update_layout(legend_orientation="h",
                  legend=dict(x=.5, xanchor="center"),
                  margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```