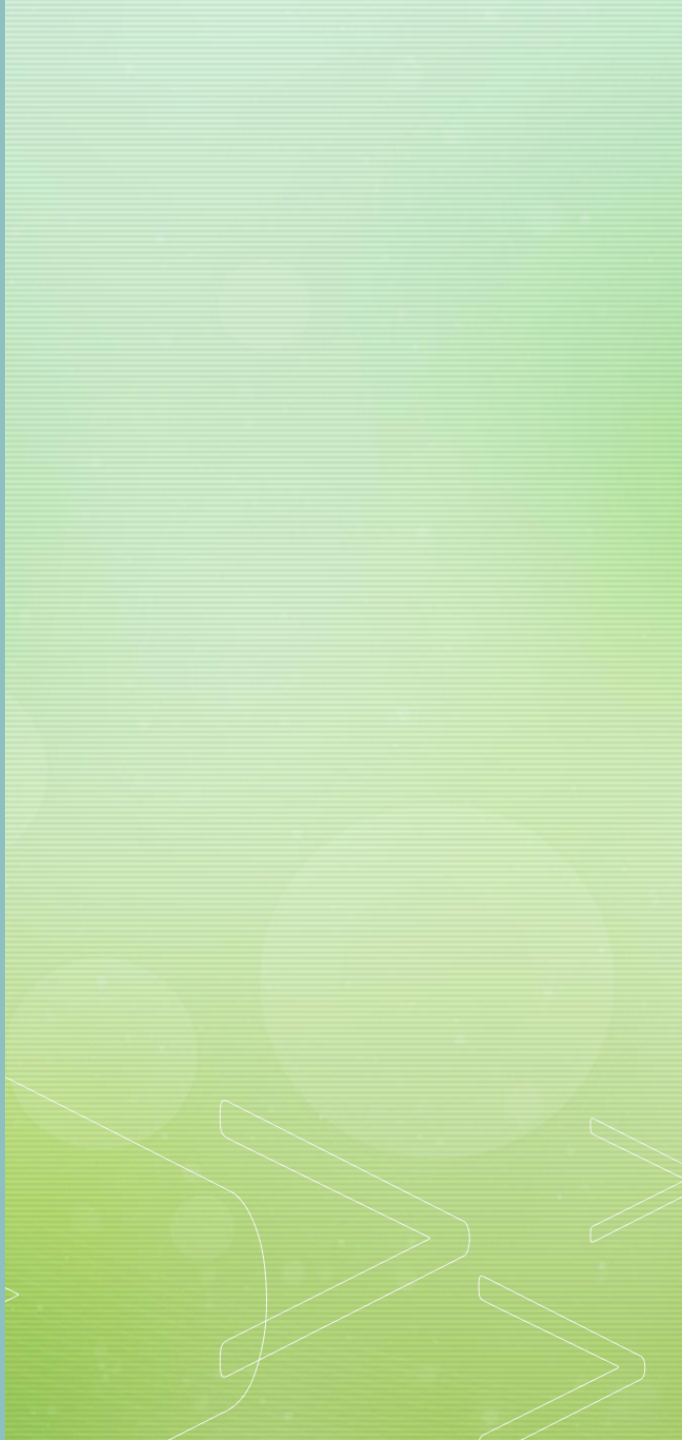




Стажировка компании GlowByte

# Причинно- следственный анализ

Выполнил: Белый Олег



# Постановка задачи

Имеется набор данных, отражающих состояние ключевых переменных сети кафе в течение 18 месяцев, необходимо:

- проверить наличие причинно-следственной связи между переменными **S\_sales** и **M\_sales**;
- восстановить максимальное количество аутентичных причинно-следственных связей между переменными

# Входные данные

Датасет описывает состояние продаж сети кафе в промежуток времени, соответствующий 5 месяцам до и 13 месяцам после введения в ассортимент нового **напитка М**. Данный напиток является улучшенным аналогом существовавшего ранее **напитка S**, который впоследствии предполагается вывести из ассортимента.

# Получение и обработка данных

```
df = pd.read_csv(r'ci_data_internship.csv', sep=',', encoding = 'cp1251', quoting=3)
df.index=[x[1] for x in df.index]
# извлечение численного значения (предположительно номера почтового индекса) из адреса
df.loc[df['Unnamed: 0'].notna(), 'Unnamed: 0']=df[df['Unnamed: 0'].notna()]['Unnamed: 0'].apply(lambda x: re.search('""(.+?) ', str(x)).group(1))
# избавление от кавычек
df.loc[df.S_sales.notna(), 'S_sales']=df[df.S_sales.notna()].S_sales.apply(lambda x: str(x)[:1])
# приведение данных типа object к float64
df.S_sales = df.S_sales.replace('', np.nan).astype('float64')
df.id = df.index
df.address = df['Unnamed: 0'].astype('float64')
df.drop(columns='Unnamed: 0', inplace=True)
df.reset_index(inplace=True, drop=True)
```

Извлечение численного значения из адреса, избавление от кавычек и т.д.

	id	address	is_holiday	day_type	season	comp_activity	M_promo_exp	M_promo_internet	M_promo_banners	S_price	M_price	menu_type	average_purchase_items	average
0	10111423.0	7435.0	0.0	1.0	3.0	0.0	NaN	0.0	0.0	279.0	NaN	1.0	1.397272	1
1	10111423.0	7435.0	0.0	1.0	3.0	0.0	0.00000	0.0	0.0	279.0	0.0	1.0	1.406686	
2	10111423.0	7435.0	0.0	0.0	3.0	0.0	0.00000	0.0	0.0	279.0	0.0	1.0	1.387417	1
3	10111423.0	7435.0	0.0	0.0	3.0	0.0	0.00000	0.0	0.0	279.0	0.0	1.0	1.503351	1
4	10111423.0	7435.0	0.0	0.0	3.0	0.0	0.00000	0.0	0.0	279.0	0.0	1.0	1.542436	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
539995	10112422.0	11522.0	0.0	0.0	4.0	0.0	26178.53131	28.0	4.0	349.0	389.0	7.0	2.425469	1
539996	10112422.0	11522.0	0.0	0.0	4.0	0.0	26178.53131	28.0	4.0	349.0	389.0	7.0	2.139832	1
539997	10112422.0	11522.0	NaN	0.0	4.0	0.0	26178.53131	28.0	4.0	349.0	389.0	7.0	2.156813	1
539998	10112422.0	11522.0	0.0	0.0	4.0	0.0	NaN	28.0	4.0	349.0	389.0	7.0	2.215092	1
539999	10112422.0	11522.0	0.0	1.0	4.0	0.0	26178.53131	28.0	4.0	349.0	389.0	7.0	2.309749	1

Датафрейм после первичной обработки

# Устранение пропусков и выбросов

```
# т.к. id кафе однозначно определяет его адрес и наоборот, то можно устранить часть пропусков медианой  
df.loc[:, 'address'] = df['address'].fillna(df.groupby('id')['address'].transform('median'))  
df.loc[:, 'id'] = df['id'].fillna(df.groupby('address')['id'].transform('median'))  
# те данные, где пропущены и id, и адрес восстановить не получится, поэтому избавимся от них  
df = df[~df.id.isna()]
```

Избавление от пропусков в переменных **address** и **id**

```
df.season.unique()  
  
array([3.00000000e+00, 0.00000000e+00, 4.00000000e+00, 1.00000000e+00,  
       6.83971682e+04, 4.10285962e+04, 2.00000000e+00, 2.61785313e+04,  
       3.62483616e+04, 1.01934330e+05])  
  
df = df[df.season <= 4]
```

Избавление от выбросов в переменной **season**

# Проверка причинно-следственной связи между переменными **S\_sales** и **M\_sales**

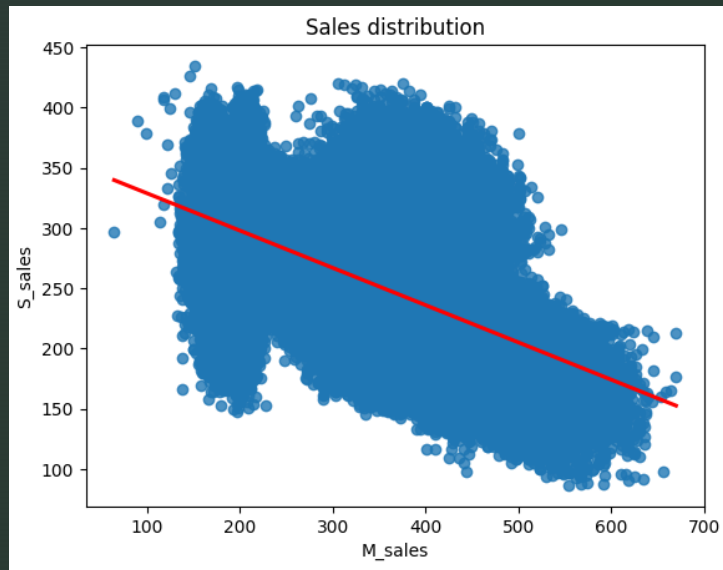


Диаграмма рассеяния  
**M\_sales** и **S\_sales**

По графику можем наблюдать наличие отрицательной корреляции.

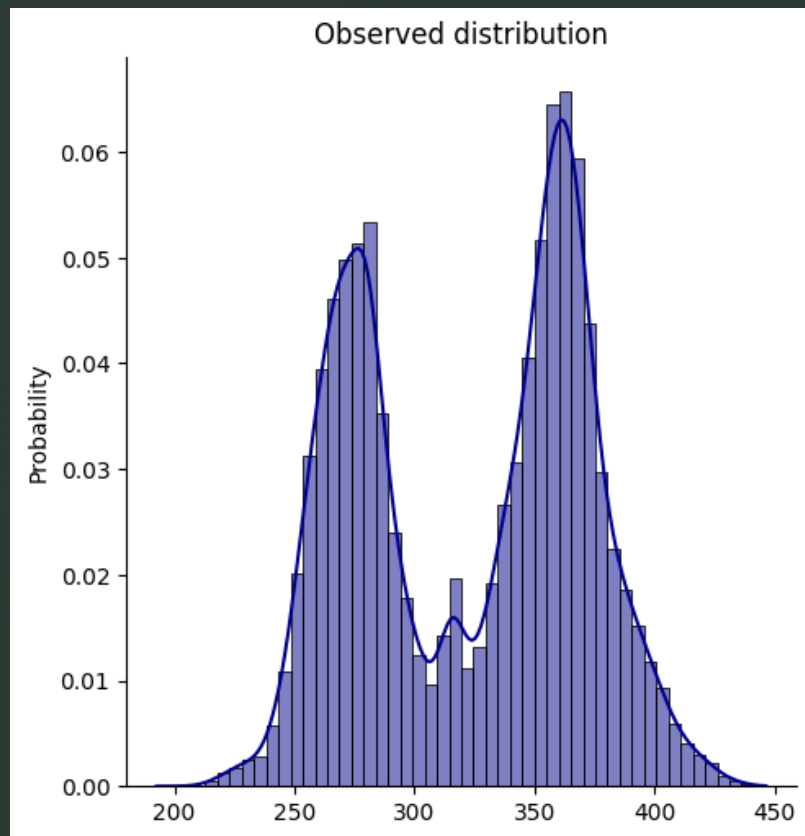
	n	r	CI95%	p-val
pearson	375756	-0.554591	[-0.56, -0.55]	0.0

Результаты теста Пирсона

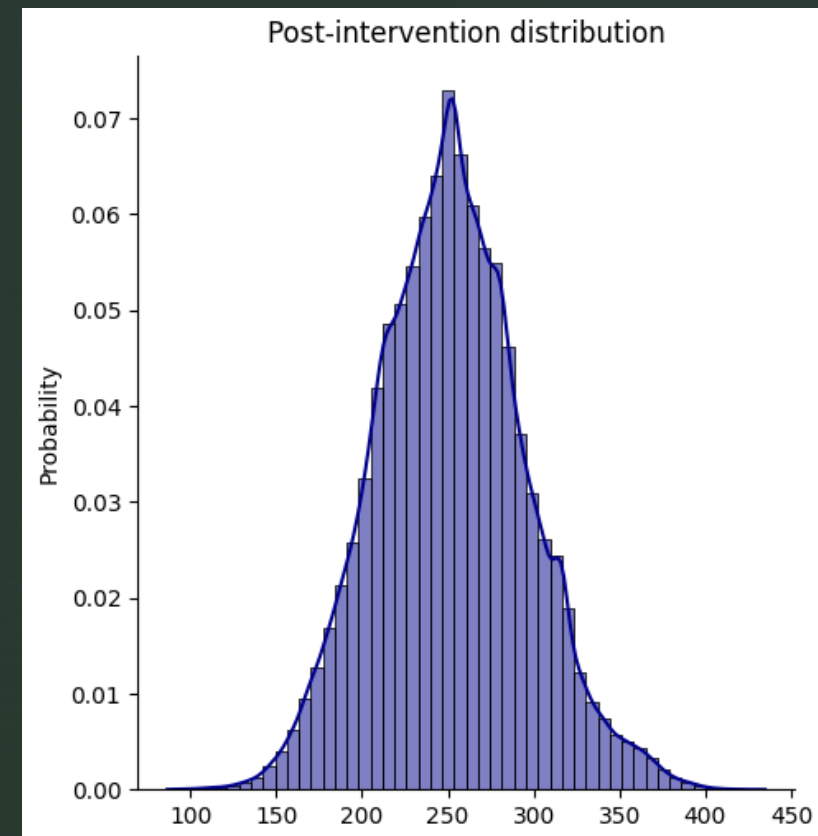
Тест вернул значение p-value, меньшее 0.05, поэтому мы отвергаем нулевую гипотезу о независимости **M\_sales** и **S\_sales**, т.е. **M\_sales**  $\nparallel$  **S\_sales**



Проверим наше утверждение, сравнив наблюдаемое и пост-интервенционное распределения переменной  $S\_sales$ . В качестве наблюдаемого будем считать распределение  $S\_sales$  до введения напитка  $M$  в ассортимент, а в качестве пост-интервенционного - распределение  $S\_sales$  после введения напитка  $M$  в ассортимент



Наблюдаемое распределение



Пост-интервенционное распределение

Видим, что наблюдаемое бимодальное распределение заметно отличается от унимодального пост-интервенционного. Удостоверимся в этом, сравнив распределения, используя тест Колмогорова-Смирнова:

```
stats.ks_2samp(S_sales_before, S_sales_after)
```

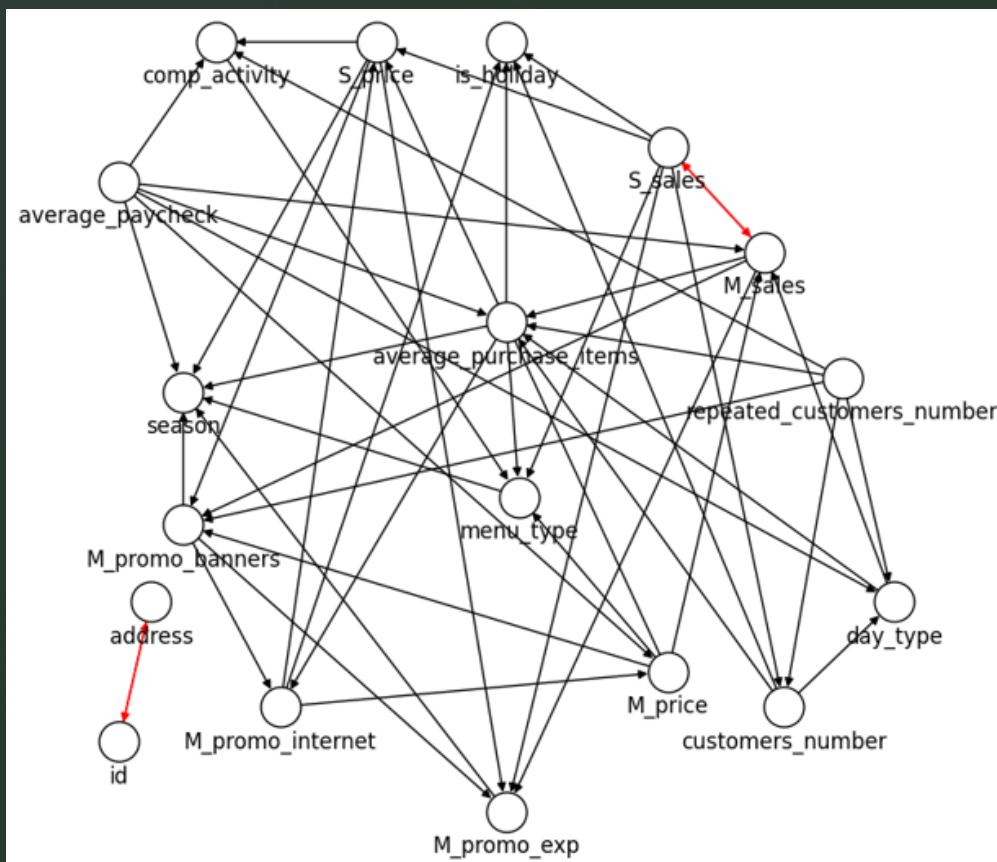
```
KstestResult(statistic=0.5041838296640158, pvalue=0.0, statistic_location=265.0453853, statistic_sign=-1)
```

По результатам теста значение  $p$ -value меньше 0.05, значит нулевая гипотеза о том, что два распределения идентичны, отвергается, а значит, увеличение продаж напитка **M** отрицательно сказывается на продажах напитка **S**, иными словами, наблюдается эффект каннибализации, когда объём продаж одного продукта компании сокращается за счет вывода на рынок другого продукта



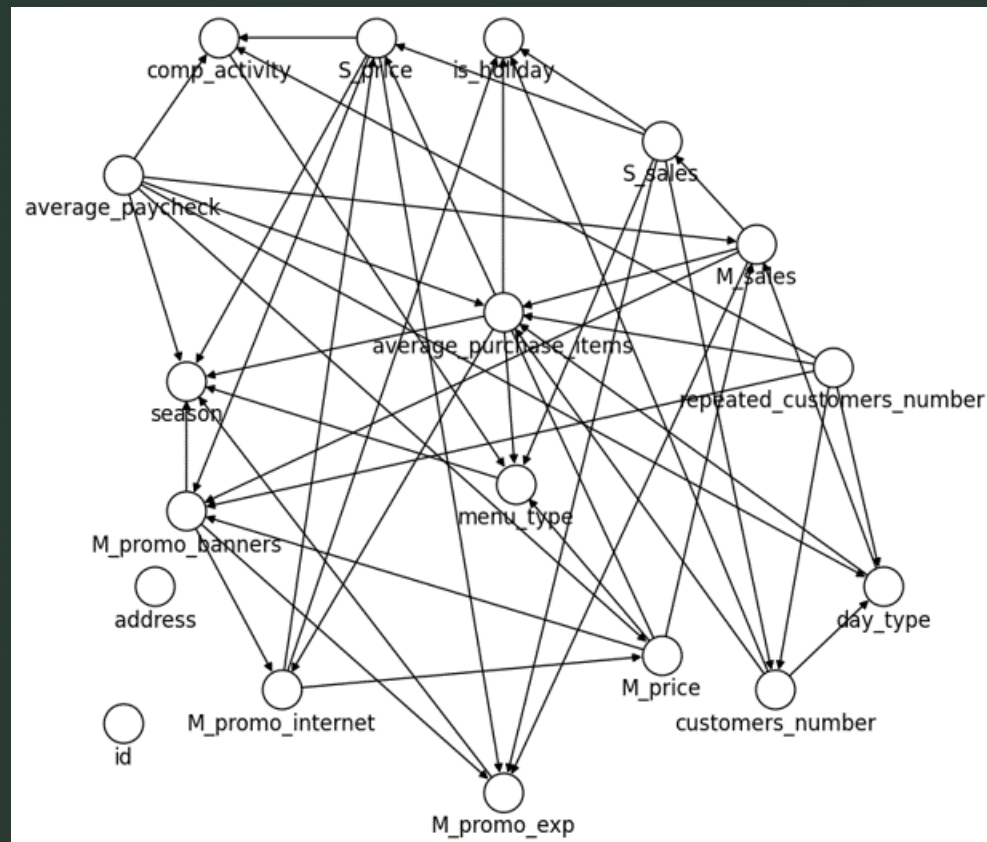
# Восстановление максимального количества аутентичных причинно-следственных связей между переменными

В результате применения **constraint-based** алгоритма **Петера-Кларка (PC)** был получен следующий граф:




Неориентированными остались рёбра ('S\_sales', 'M\_sales') и ('id', 'address'). В первом случае можем выбрать только стрелку 'M\_sales' -> 'S\_sales', т.к. она не порождает новых циклов и v-структур, в то время как ориентация в обратном направлении приводит к возникновению новых v-структур, например: 'S\_sales' -> 'M\_sales' <- 'average\_paycheck'.

Находить направление ребра ('id', 'address') не имеет смысла, т.к. обе эти вершины больше не имеют связей, кроме как друг с другом и выполняют, впрочем, одну и ту же функцию - однозначно идентифицируют кафе, однако, следует сказать, что на самом деле, если бы переменную **address** удалось преобразовать из категориальной в количественную, связав, например, адрес кафе с его удалённостью от жилых комплексов, то это, вероятно, повлияло бы на вид графа



Граф, полученный в результате применения **background knowledge**



# Способы улучшения полученных результатов

- Устранение мультиколлинеарности переменных
- Устранение дисбаланса классов
- Агрегирование
- Замена номинальных переменных на количественные
- Нормализация

# Устранение дисбаланса классов и агрегирование

```
df = df.sort_values(by=['address'])
min=df.groupby('address')['address'].count().min()
for i in df.address.unique():
    while df[df['address']==i].address.count() > min:
        df=df.drop(index=df[df['address']==i].sample(n=1).index)
```

Андерсэмплинг по переменной **address**

Преобразуем переменную **is\_holiday** в бинарную, где 1 - это праздники, а 0 - будни

```
ds.loc[:, 'is_holiday'] = np.where(ds['is_holiday'] >= 1, 1, 0)
```

Прореживаем данные, у которых **menu\_type** > 3 и избавляемся от остальных

```
menu_type
1.0      817
2.0      840
3.0     8968
4.0    33991
5.0    17037
6.0    25526
7.0    25747
Name: id, dtype: int64
```

```
ds_3 = ds[ds['menu_type']==3]
ds_4 = resample(ds[ds['menu_type']==4], replace = False, n_samples = len(ds_3), random_state = 0)
ds_5 = resample(ds[ds['menu_type']==5], replace = False, n_samples = len(ds_3), random_state = 0)
ds_6 = resample(ds[ds['menu_type']==6], replace = False, n_samples = len(ds_3), random_state = 0)
ds_7 = resample(ds[ds['menu_type']==7], replace = False, n_samples = len(ds_3), random_state = 0)
ds = pd.concat([ds_3, ds_4, ds_5, ds_6, ds_7])
```

# Замена номинальных переменных на количественные

Идея: связать `address` и `season` с `customers_number`, а `menu_type` - с `average_paycheck` через частоты повторения

Адрес:

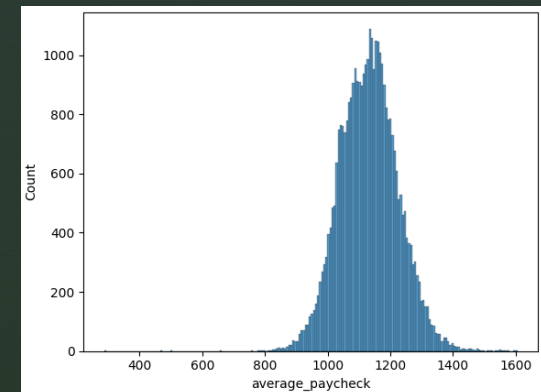
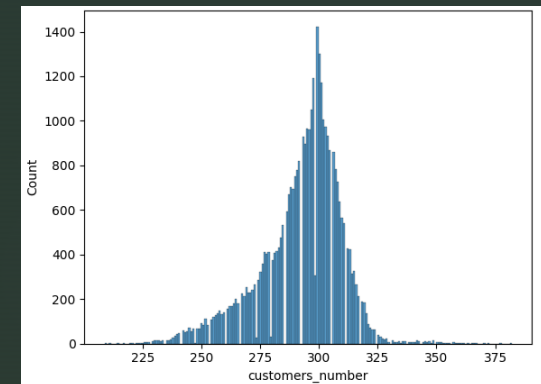
```
ds.loc[:, 'address'] = ds.address.map(ds[ds.customers_number >= 297].address.value_counts())
```

Сезон:

```
ds.loc[:, 'season'] = ds.season.map(ds[ds.customers_number >= 297].season.value_counts())
```

Тип меню:

```
ds.loc[:, 'menu_type'] = ds.menu_type.map(ds[ds.average_paycheck >= 1135].menu_type.value_counts())
```





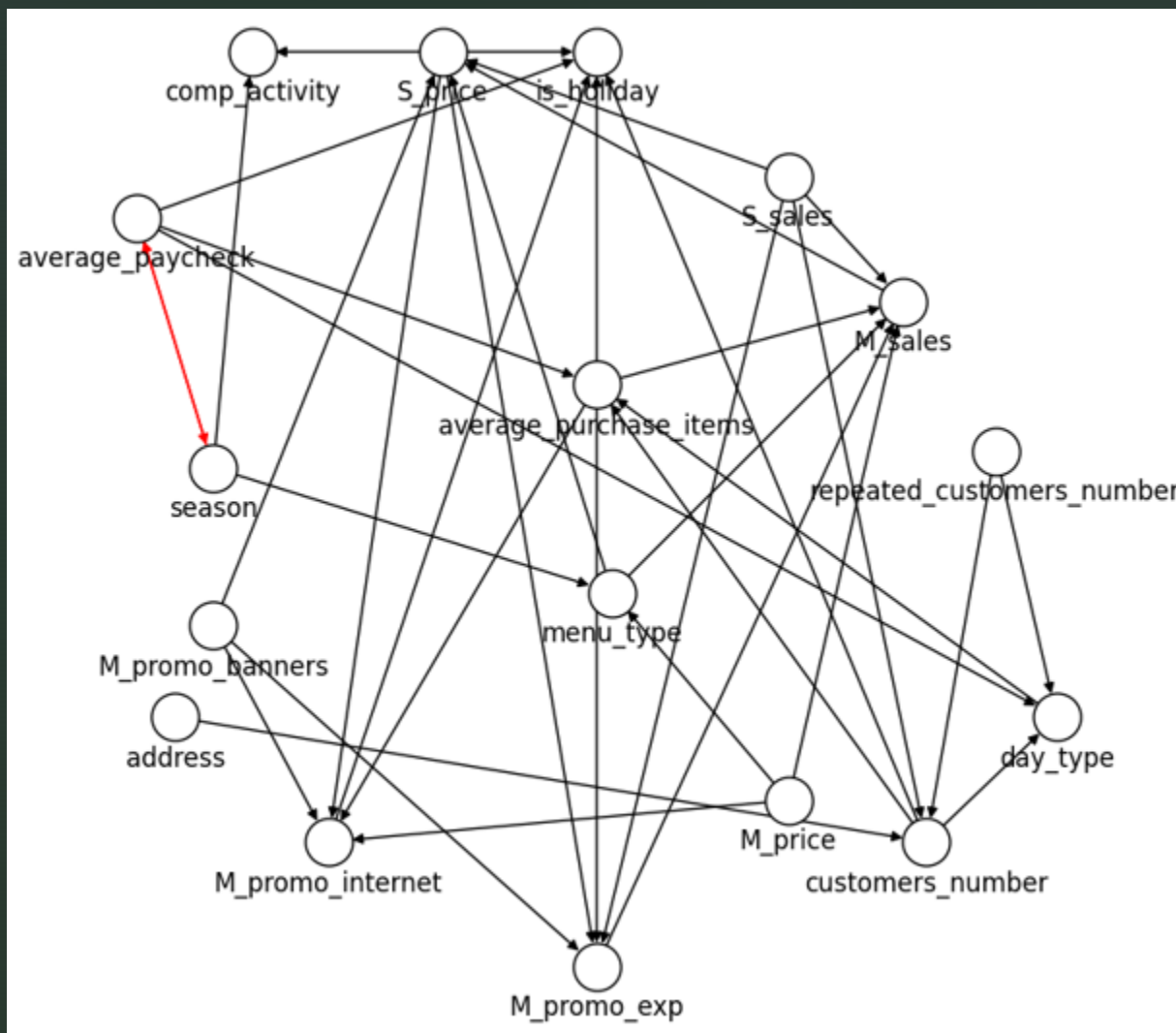
# Нормализация

Нормализовал все признаки, кроме бинарных и принимающих небольшие дискретные значения

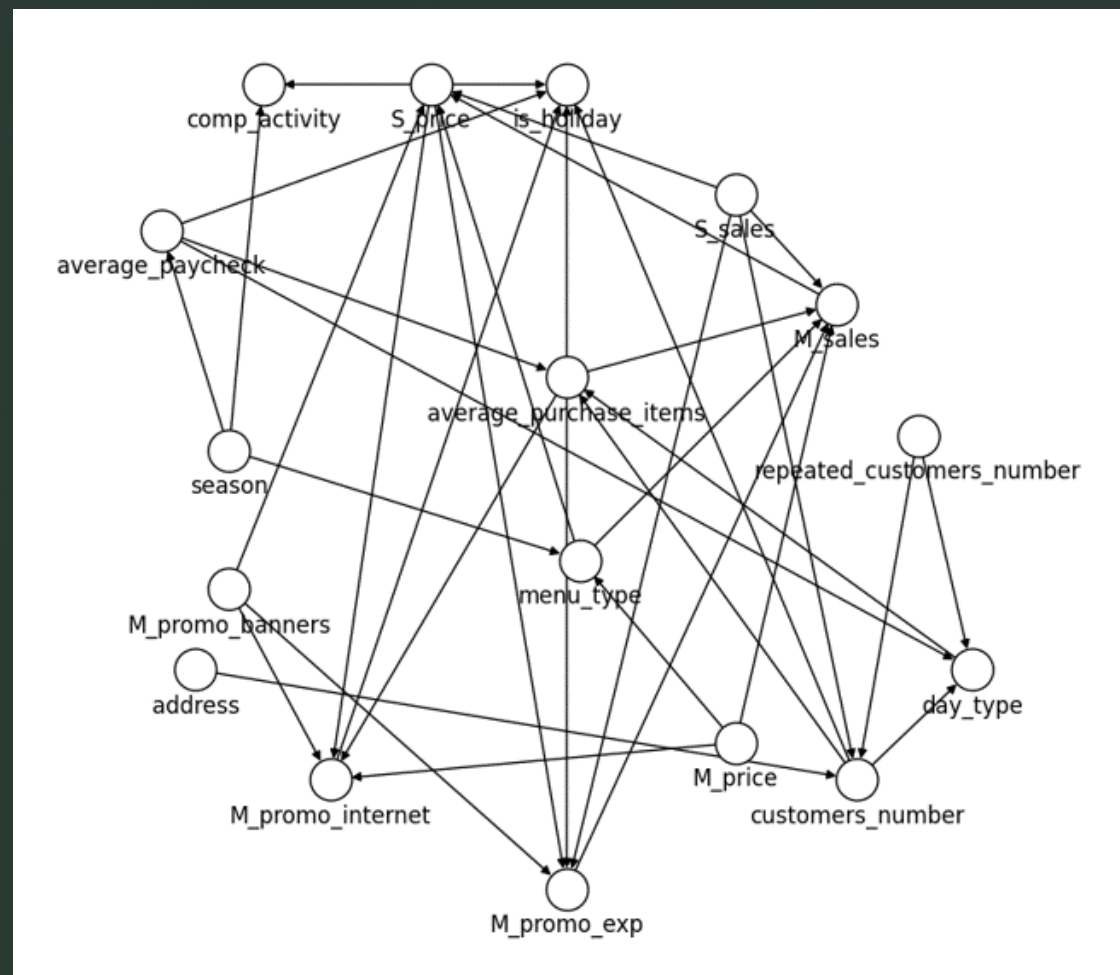
```
#не нормализую: id, address, is_holiday, day_type, comp_activity, т.к. это либо бинарные, либо небольшие дискретные значения  
ds.reset_index(inplace=True, drop=True)  
ds_1 = ds.loc[:,['id','address','is_holiday','day_type','comp_activity']]  
ds_2 = ds.drop(columns=['id','address','is_holiday','day_type','comp_activity'],axis=1)  
mms=MinMaxScaler()  
mms.fit(ds_2)  
ds_2_norm = pd.DataFrame(mms.transform(ds_2), columns=ds_2.columns)  
  
ds_norm = pd.merge(ds_1, ds_2_norm, left_index=True, right_index=True)
```



# Получившийся граф



# Граф после Background knowledge





Спасибо за внимание!

