In [ ]:
```
import numpy as np                # Массивы (матрицы, векторы, линейная ал
import matplotlib.pyplot as plt # Научная графика
%matplotlib inline

import pandas as pd               # Таблицы и временные ряды (dataframe, s
import seaborn as sns             # Еще больше красивой графики для визуал
import sklearn                    # Алгоритмы машинного обучения
```

Оценка уровня удовлетворенности перелета

Измерение удовлетворенности клиентов - ключевой элемент для современного бизнеса, поскольку он может внести значительный вклад в повышение качества обслуживания. Чтобы оправдать ожидания клиентов и достичь более высокого уровня, авиакомпаниям необходимо разработать специальный механизм измерения удовлетворенности пассажиров.

Необходимо проанализировать прикаких условиях достигается удовлетворенность пассажира приперелете. Таким обрахом требуется предсказать признак satisfaction по остальным признакам. Это задача восстановления регрессии.

https://www.kaggle.com/binaryjoker/airline-passenger-satisfaction

1. Загружаем данные

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving airline_passenger_satisfaction.csv to airline_passenger_satisfacti
on.csv
```

In [ ]:
```
import io
data_raw = pd.read_csv(io.BytesIO(uploaded['airline_passenger_satisfacti
```

Вначале посмотрим на размеры таблицы - количество строк (файл содержит информацио о 129879 различных пассажиров) и количество столбцов (каждый столбец соответствует конкретному признаку):

In [ ]:
```
data_raw.shape
```

Out[ ]:
```
(129880, 24)
```

In [ ]:
```
data_raw
```

Out[ ]:

	Unnamed:	Gender	customer_type	age	type_of_travel	customer_class	flight_dista

	0						
0	0	Male	Loyal Customer	13	Personal Travel	Eco Plus	
1	1	Male	disloyal Customer	25	Business travel	Business	
2	2	Female	Loyal Customer	26	Business travel	Business	1
3	3	Female	Loyal Customer	25	Business travel	Business	
4	4	Male	Loyal Customer	61	Business travel	Business	
...	...	...	...	...	...	...	
129875	129875	Male	disloyal Customer	34	Business travel	Business	
129876	129876	Male	Loyal Customer	23	Business travel	Business	
129877	129877	Female	Loyal Customer	17	Personal Travel	Eco	
129878	129878	Male	Loyal Customer	14	Business travel	Business	1
129879	129879	Female	Loyal Customer	42	Personal Travel	Eco	

129880 rows × 24 columns

In [ ]:
```
for x in data_raw.columns:
    print(x,data_raw[x].dtype)
```

```
Unnamed: 0 int64
Gender object
customer_type object
age int64
type_of_travel object
customer_class object
flight_distance int64
inflight_wifi_service int64
departure_arrival_time_convenient int64
ease_of_online_booking int64
gate_location int64
food_and_drink int64
online_boarding int64
seat_comfort int64
inflight_entertainment int64
onboard_service int64
leg_room_service int64
baggage_handling int64
checkin_service int64
inflight_service int64
cleanliness int64
departure_delay_in_minutes int64
arrival_delay_in_minutes float64
satisfaction object
```

Мы видим, что столбцы (признаки) имеют имена. Рассмотрим следующие:

- `Gender` - пол.
- `customer_type` - тип покупатиеля.
- `age` - возраст.
- `type_of_travel` - тип поездки.

- `customer_class` - класс.
- `flight_distance` - расстояние перелета.
- `inflight_wifi_service`, `departure_arrival_time_convenient`, `ease_of_online_booking`, `gate_location`, `food_and_drink`, `online_boarding`, `seat_comfort`, `inflight_entertainment`, `onboard_service`, `leg_room_service`, `baggage_handling`, `checkin_service`, `inflight_service`, `cleanliness` - оценка качества wi-fi, времени вылета и прибытия, онлайн бронирования, местонахождение метса вылета, еды и напитков, онлайн посадки, сидений, развлечений во времня перелета, сервиса на ботру, места для ног, перевозки багажа, регистрации при вылете и по прилету, чистоты.
- `departure_delay_in_minutes` `arrival_delay_in_minutes` - задержка времени вылета и прибытия в минутах.
- `satisfaction` - удовлетворение.

Признаки Gender, customer_type, type_of_travel, customer_class, satisfaction - номинальный (категориальный), признаки departure_delay_in_minutes, arrival_delay_in_minutes - количественные (числовое в минутах), flight_distance - количественные (числовое в километрах), age - количественные (числовое в годах), остальные признаки - количественные выражают оценку тех или иных качеств полета от 1 до 5 (числовой).

In [ ]:
```
categorial_list = ['Gender', 'customer_type', 'type_of_travel','customer_
for x in categorial_list:
    data_raw[x] = data_raw[x].astype('category')
```

In [ ]:
```
data_raw['Gender'].dtype
```

Out[ ]:
```
CategoricalDtype(categories=['Female', 'Male'], ordered=False)
```

In [ ]:
```
data_raw['customer_type'].dtype
```

Out[ ]:
```
CategoricalDtype(categories=['Loyal Customer', 'disloyal Customer'], orde
red=False)
```

In [ ]:
```
data_raw['type_of_travel'].dtype
```

Out[ ]:
```
CategoricalDtype(categories=['Business travel', 'Personal Travel'], order
ed=False)
```

In [ ]:
```
data_raw['customer_class'].dtype
```

Out[ ]:
```
CategoricalDtype(categories=['Business', 'Eco', 'Eco Plus'], ordered=Fals
e)
```

In [ ]:
```
data_raw['satisfaction'].dtype
```

Out[ ]:
```
CategoricalDtype(categories=['neutral or dissatisfied', 'satisfied'], ord
ered=False)
```

Таким образом мы узнали каие значения принимают категориальные признаки.

2. Бинаризация номинальных признаков

Так как уровень удовлетворенности признак бинарный ,а нам необходимо найти зависимости между остальными призаками и удовлетворенностью, необходимо сделать его количественным, 'neutral or dissatisfied' - 0, 'satisfied' - 1.

Для останьных категориальных сделаем аналогично:

для Gender 0 - male 1 - female, для customer_type 0 - Loyal Customer 1 - disoyal Customer, для type_of_travel 0 - Personal Travel 1 - Business travel

In [ ]:
```
data_r = pd.read_csv(io.BytesIO(uploaded['airline_passenger_satisfaction
```

In [ ]:
```
data_r['satisfaction'] = pd.factorize(data_r['satisfaction'])[0]
data_r['customer_type'] = pd.factorize(data_r['customer_type'])[0]
data_r['Gender'] = pd.factorize(data_r['Gender'])[0]
data_r['type_of_travel'] = pd.factorize(data_r['type_of_travel'])[0]
```

К категориальному (небинарному) признаку `'customer_class'` применим метод *бинаризации* (*one-hot encoding*), который заключается в следующем.

Этот признак принимает 3 значения: 'Business', 'Eco', 'Eco Plus'.

Вместо признака `'customer_class'` будем использовать 3 новых признаков (dummy-признаков, dummy - фиктивный), которые так и назовем 'Business', 'Eco', 'Eco Plus'. При этом

- если признак `'customer_class'` принимает значение `'Business'`, то признак `'Business'` равен 1, а все остальные 0;
- и т.д.

In [ ]:
```
customer_class_dummies = pd.get_dummies(data_r['customer_class'])
```

Добавим эти dummy-столбцы к таблице и удалим столбец `Building`:

In [ ]:
```
data_r = pd.concat((data_r, customer_class_dummies), axis=1)
data_r = data_r.drop(['customer_class'], axis=1)
```

In [ ]:
```
data_r
```

Out[ ]:

	Unnamed: 0	Gender	customer_type	age	type_of_travel	flight_distance	inflight_wifi
0	0	0	0	13	0	460	
1	1	0	1	25	1	235	
2	2	1	0	26	1	1142	
3	3	1	0	25	1	562	
4	4	0	0	61	1	214	

...	...	...	...	...	...	...
129875	129875	0	1	34	1	526
129876	129876	0	0	23	1	646
129877	129877	1	0	17	0	828
129878	129878	0	0	14	1	1127
129879	129879	1	0	42	0	264

129880 rows × 26 columns

3. Визуализация и описательная статистика

Визуализация и описательная статистика - важные этапы анализа данных. Сводную информацию о признаках можем получить, вызвав метод `describe`:

In [ ]:
```
data_raw.head()
```

Out[ ]:

	Unnamed: 0	Gender	customer_type	age	type_of_travel	customer_class	flight_distance
0	0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460
1	1	Male	disloyal Customer	25	Business travel	Business	235
2	2	Female	Loyal Customer	26	Business travel	Business	1142
3	3	Female	Loyal Customer	25	Business travel	Business	562
4	4	Male	Loyal Customer	61	Business travel	Business	214

In [ ]:
```
data_raw.describe()
```

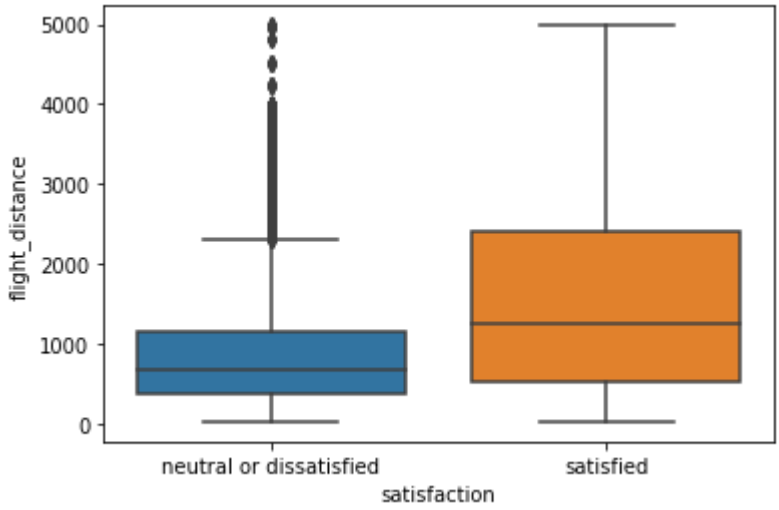
Out[ ]:

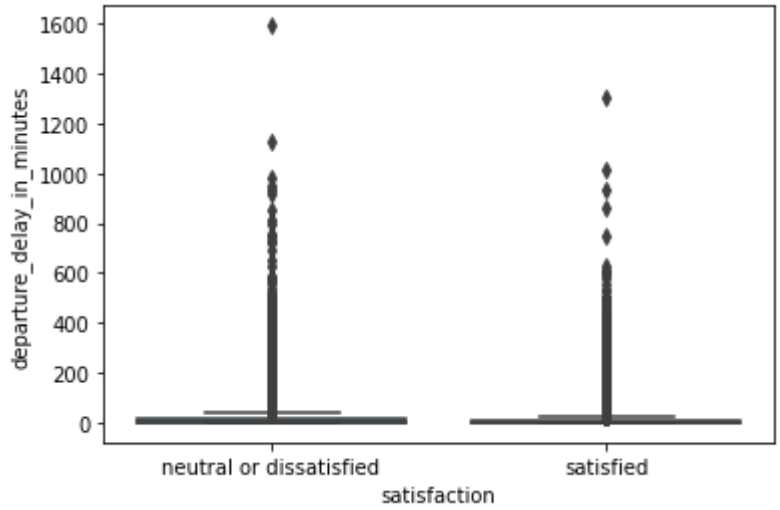
	Unnamed: 0	age	flight_distance	inflight_wifi_service	departure_arrival_t
count	129880.000000	129880.000000	129880.000000	129880.000000	
mean	64939.500000	39.427957	1190.316392	2.728696	
std	37493.270818	15.119360	997.452477	1.329340	
min	0.000000	7.000000	31.000000	0.000000	
25%	32469.750000	27.000000	414.000000	2.000000	
50%	64939.500000	40.000000	844.000000	3.000000	
75%	97409.250000	51.000000	1744.000000	4.000000	
max	129879.000000	85.000000	4983.000000	5.000000	

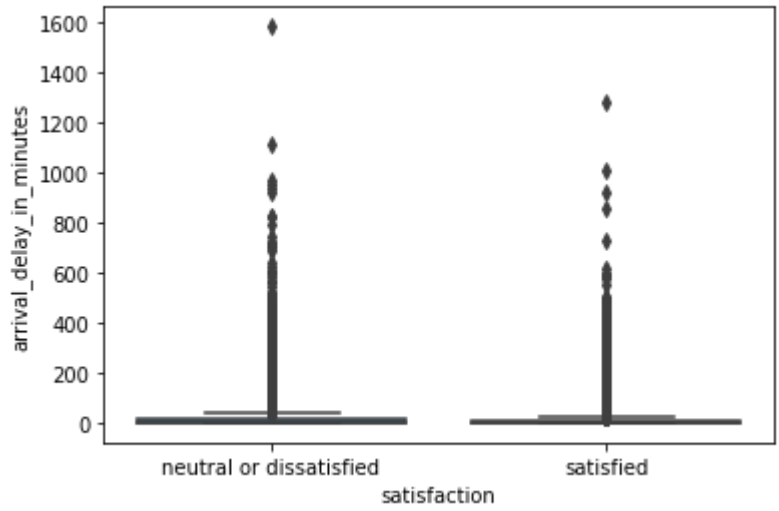
Для каждого количественного признака приведены средние значения, стандартное отклонение, минимальное и макимальное значения, медиана и значения квантилей.

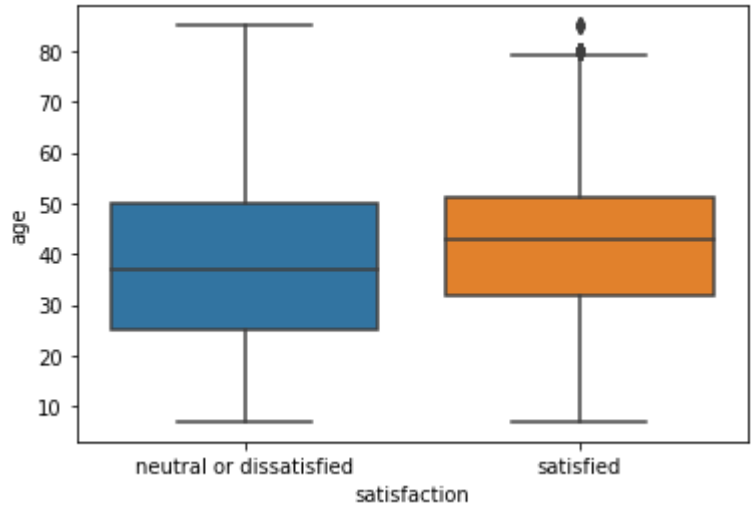
Сначала рассмотрим непрерывные величины, принимающие любые значения в допустимом для них диапозоне.

In [ ]:
```
int_features = ["flight_distance", "departure_delay_in_minutes", "arriva

for i, feature in enumerate(int_features):
    sns.boxplot(x = "satisfaction", y = feature, data=data_raw)
    plt.show()
```





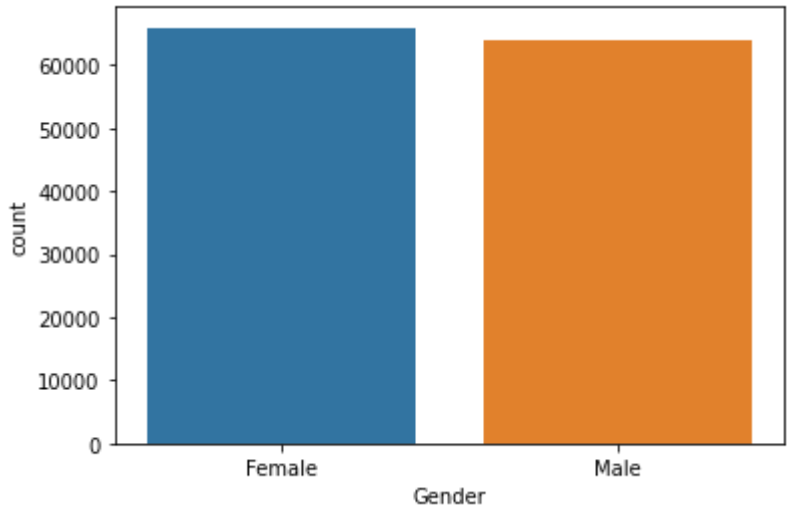




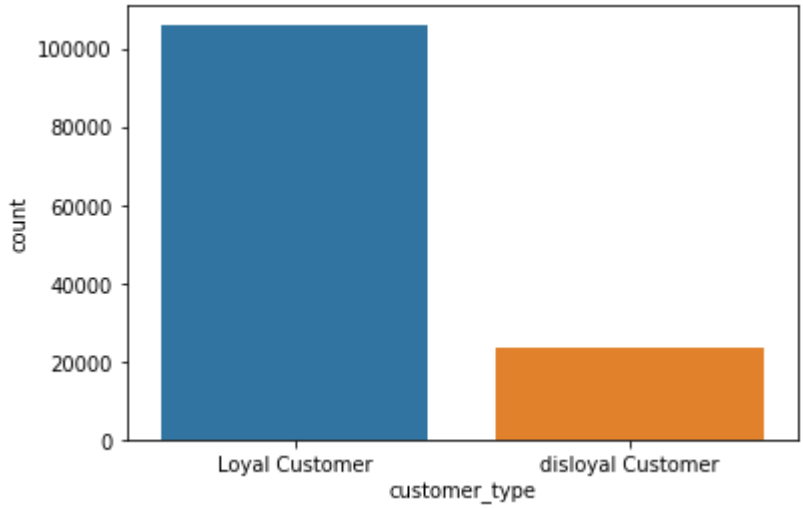
Только первй признак дал какое-то представление о разделении пассажиров. Во первых, медиана для удовлетворенных находится на большей дистанциии и значения примерно выше 1500 км относятся к удоблетворенным на много чаще.

Теперь рассмотрим категориальные признаки.

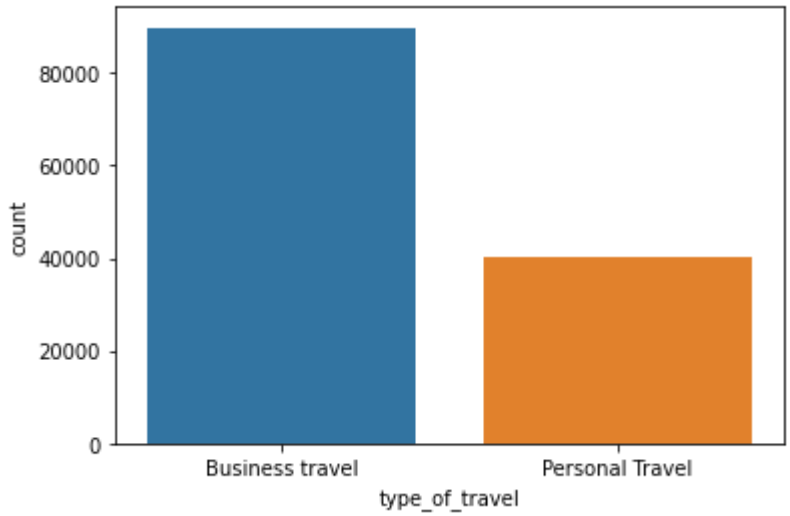
In [ ]:
```
sns.countplot(x='Gender',data=data_raw)
pass
```



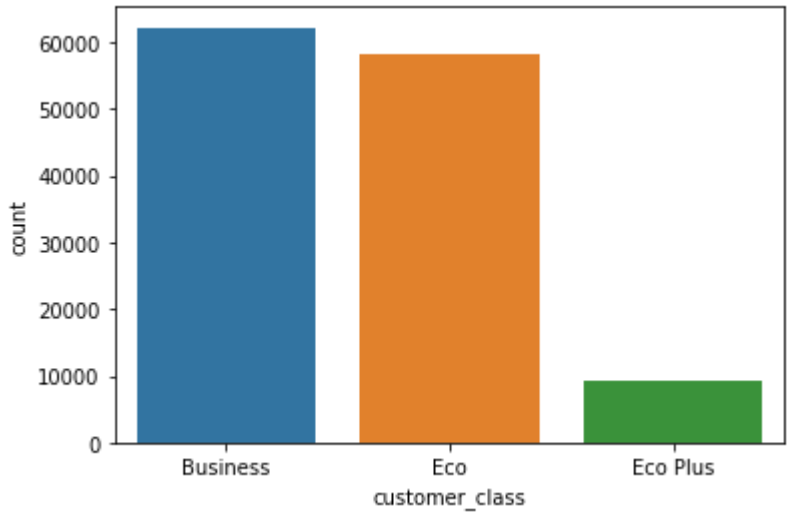
In [ ]:
```
sns.countplot(x='customer_type',data=data_raw)
pass
```



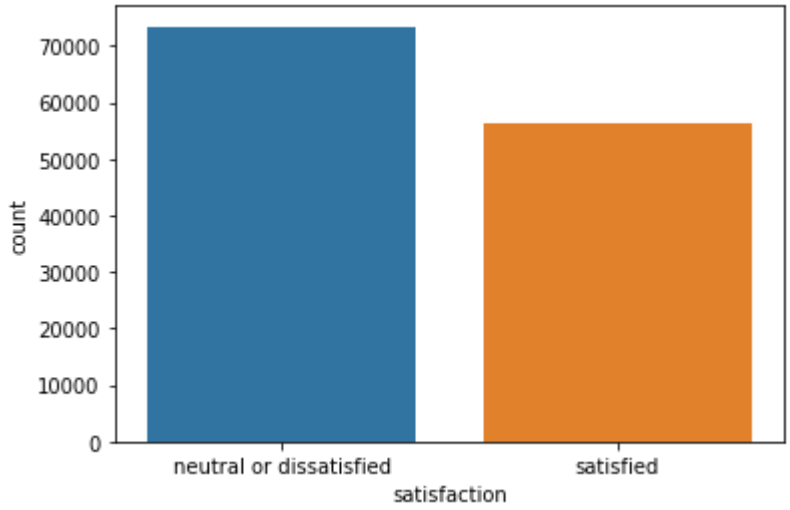
In [ ]:
```
sns.countplot(x='type_of_travel',data=data_raw)
pass
```



In [ ]:
```
sns.countplot(x='customer_class',data=data_raw)
pass
```

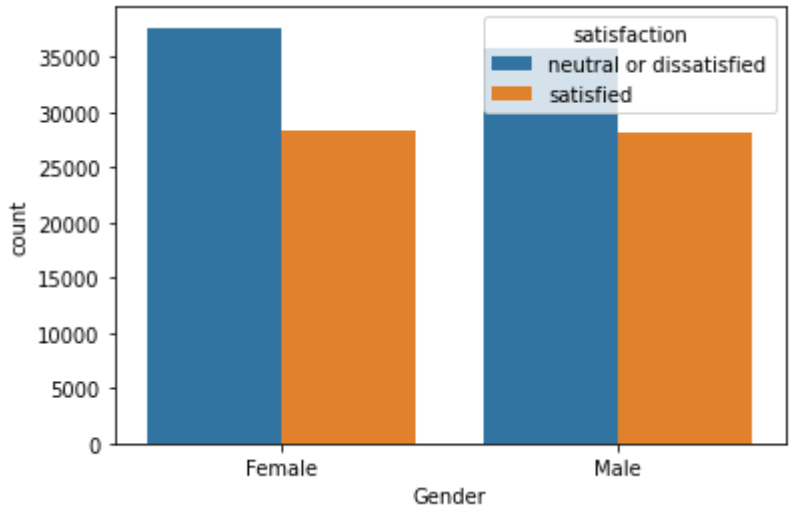


In [ ]:
```
sns.countplot(x='satisfaction',data=data_raw)
pass
```



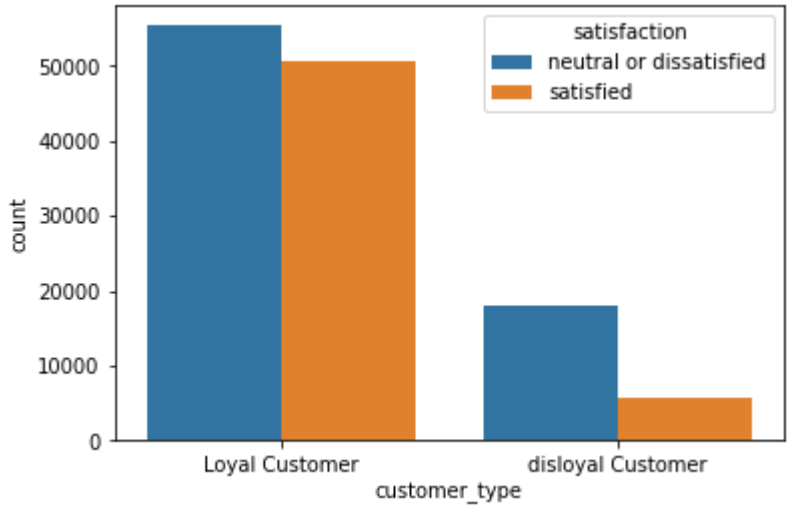
Из данных графиков для категориальных признаков можно заметить, что женщин и мужчин примерно одинаково, постоянных клиентов больше чем новых, рабочих поездок больше чем других, эконм и бизнес класс выбирают одинаково часто, чего не сказать о классе эконом плюс, удовлетворенных пассажиров немного меньше остальных.

In [ ]:
```
ax = sns.countplot(x="Gender", hue="satisfaction", data=data_raw)
```



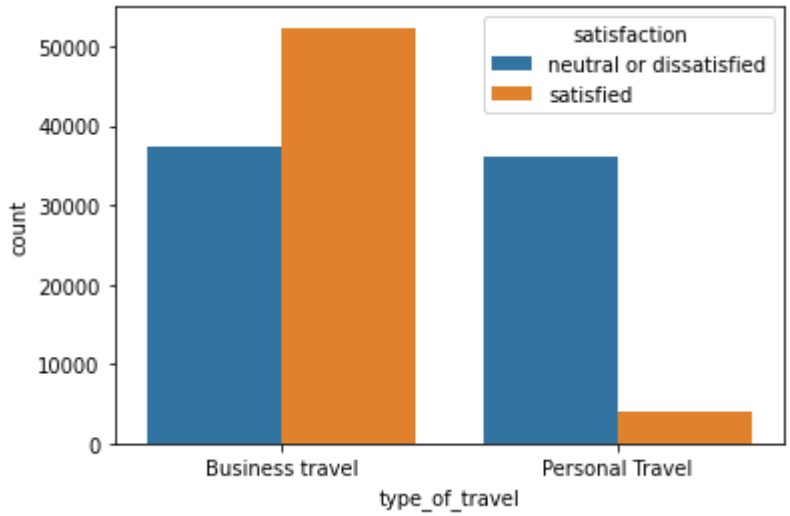
Из диаграммы видно, гендер не влияет на разделение мнений.

In [ ]:
```
ax = sns.countplot(x="customer_type", hue="satisfaction", data=data_raw)
```



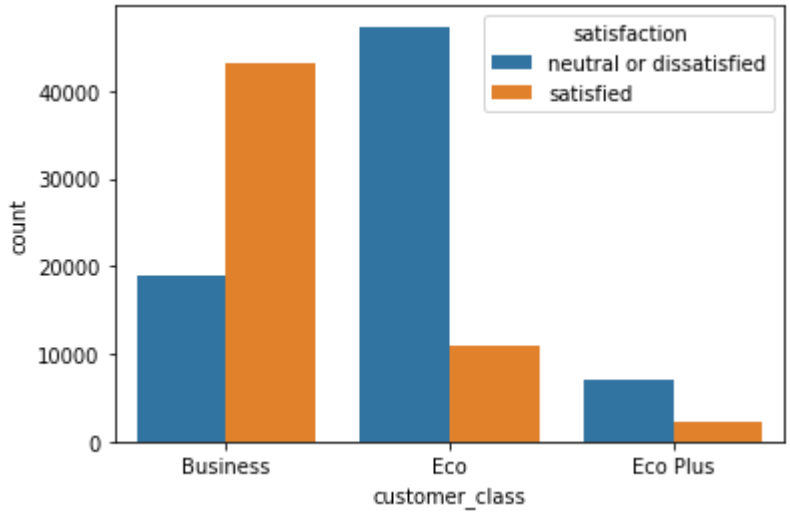
Можно заметить, что предположение о том, что постоянные клиенты будут чаще довольны не оправдалось. На диграмме приведено количество удовлетворенных и неудовлетворенных перелетом среди постоянных и новых клиентов.

In [ ]:
```
ax = sns.countplot(x="type_of_travel", hue="satisfaction", data=data_raw
```



Данная диаграмма дает чуть более лучшее представление о разделении мнения о качестве перелета. При не деловых поездах подавляющее большинство не удовлетворено или относится нейтрально к перелету.

In [ ]:
```
ax = sns.countplot(x="customer_class", hue="satisfaction", data=data_raw
```

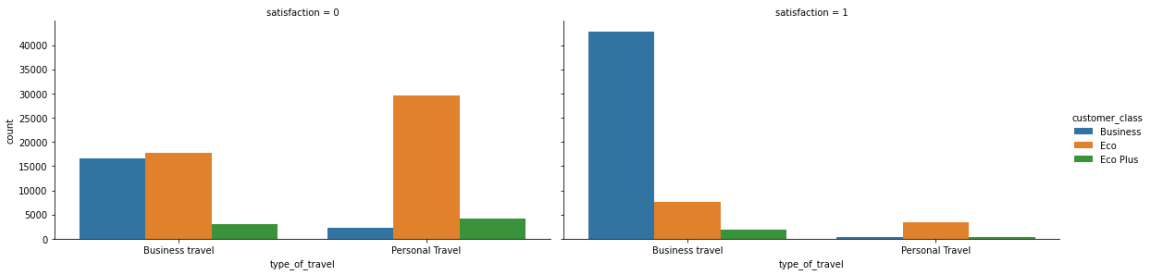


Как можно было предположить, люди летающие бизнес-классом чаще удовлетворены путешествием нежели те, кто летают эконом-классом. Это предположение подтвердилось графиком.

Мы обнаружили зависимость удовлетворенности летающих от типа поездки и класса пассажиров. Можем поробовать построить диаграмму, иллюстрирующую зависимость этих трех характеристик.

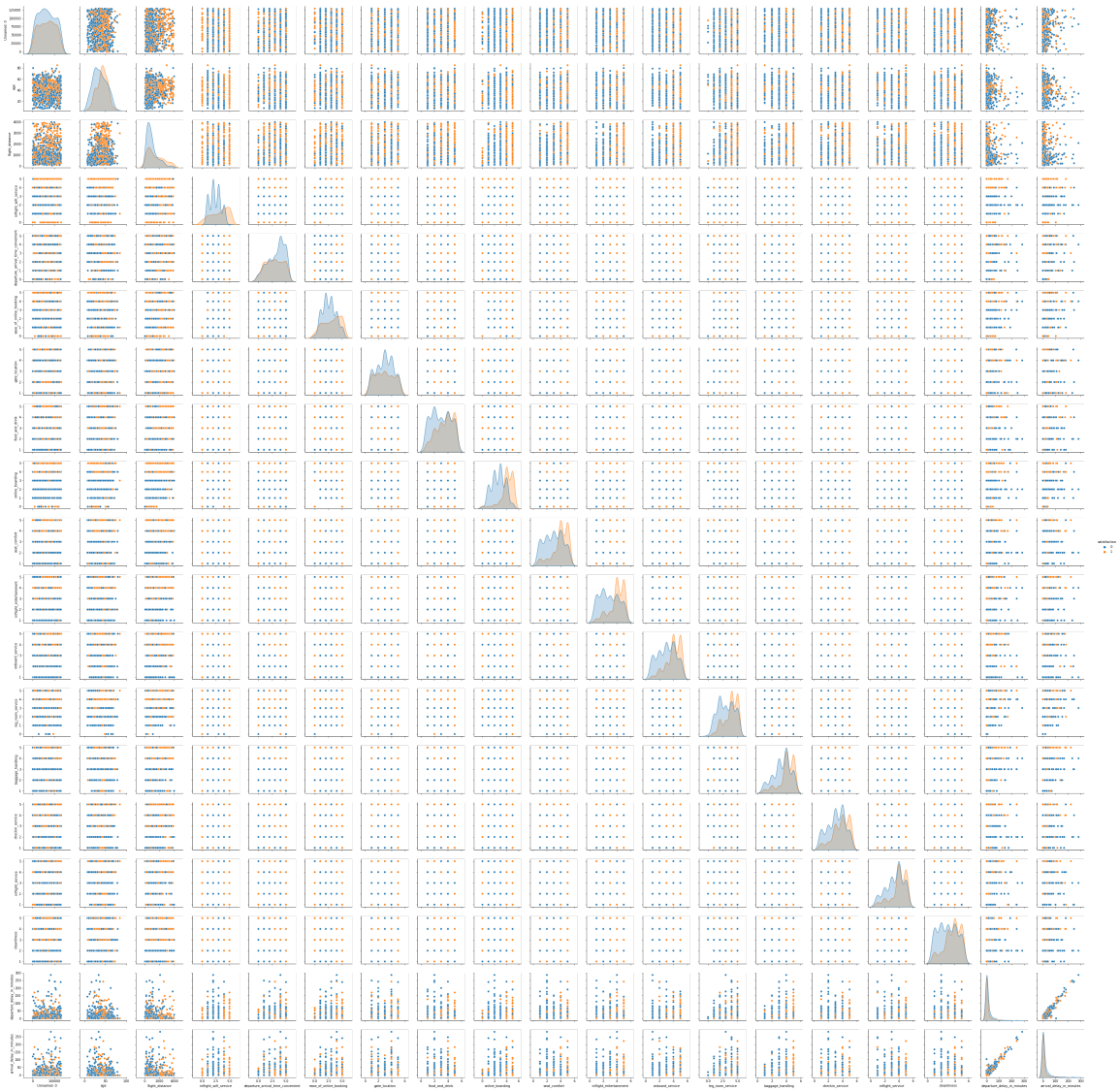
In [ ]:
```
data_raw['satisfaction'] = pd.factorize(data_raw['satisfaction'])[0]
```

In [ ]:
```
g = sns.catplot(x="type_of_travel", hue="customer_class", col="satisfact
                data=data_raw, kind="count",
                height=4, aspect=2);
```



Построим все возможные диаграммы рассеивания для каждой пары переменных:

In [ ]:
```
sns.pairplot(data_raw.sample(1000), hue='satisfaction')
pass
```

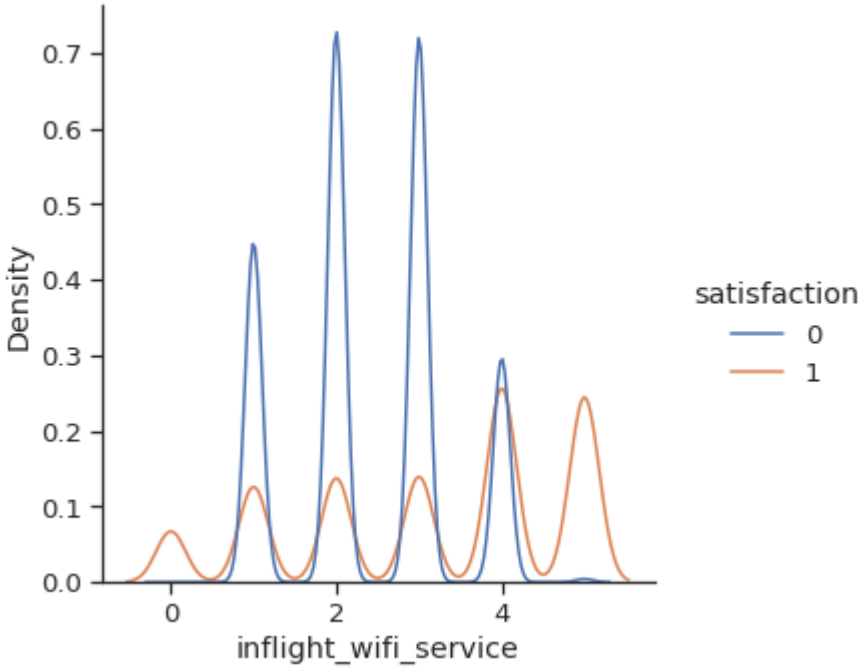


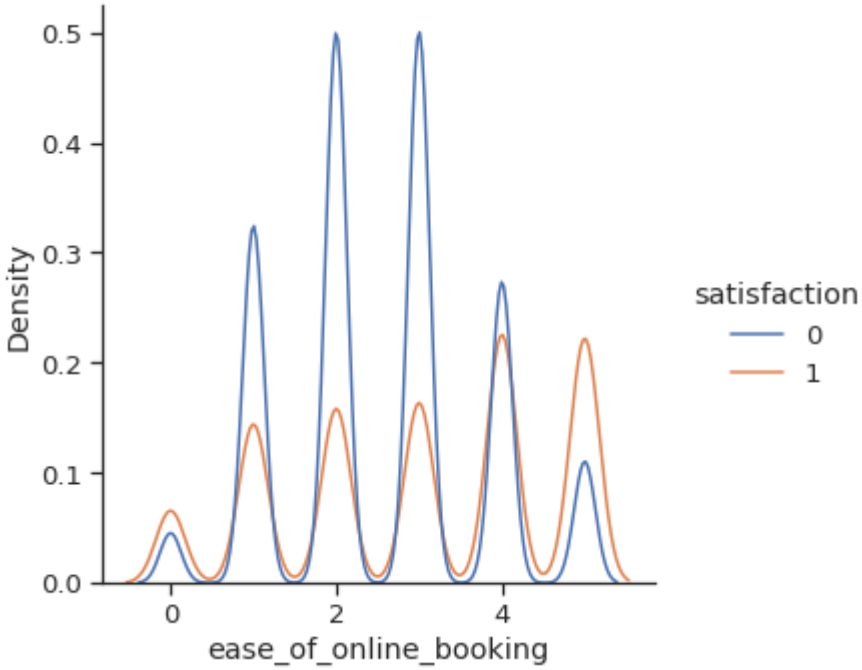
На диаграммах, в частности, наблюдается попарная корреляция между `inflight_wifi_service`, `ease_of_online_booking`, `online_boarding`, `seat_comfort`, `inflight_entertainment` и `satisfaction`

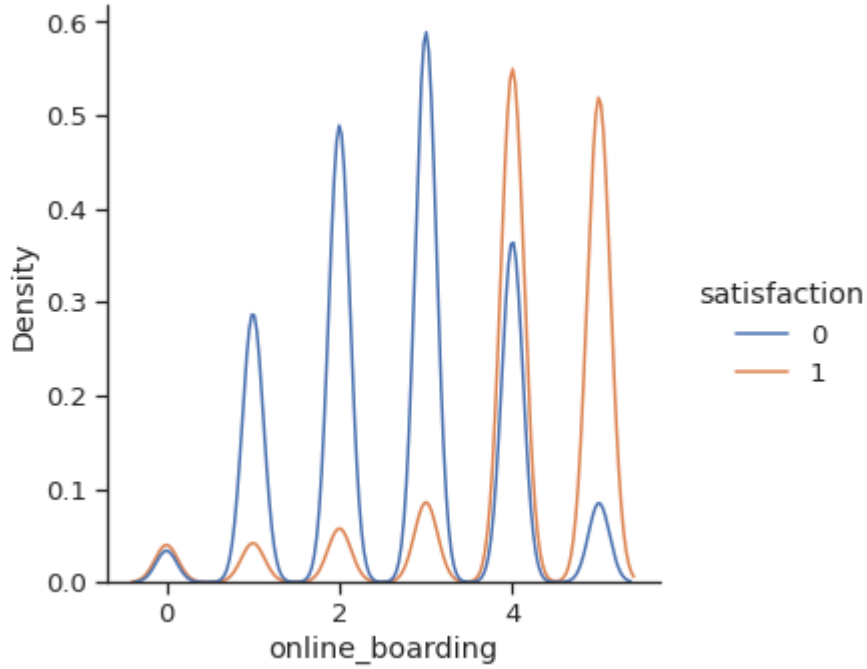
На диагонали расположены гистограммы распределения признаков. Построим их еще раз для каждой пары признаков - отдельно.

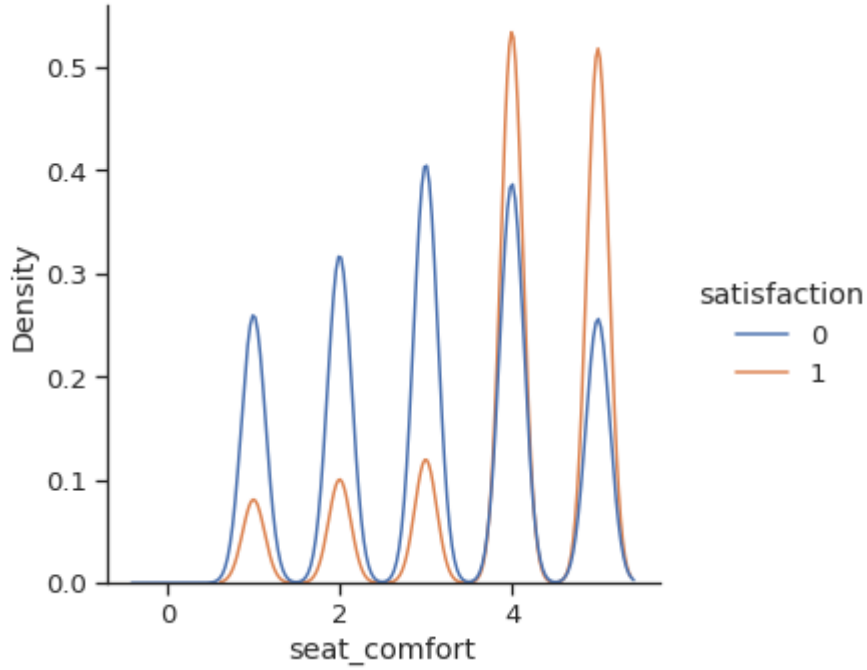
In [ ]:

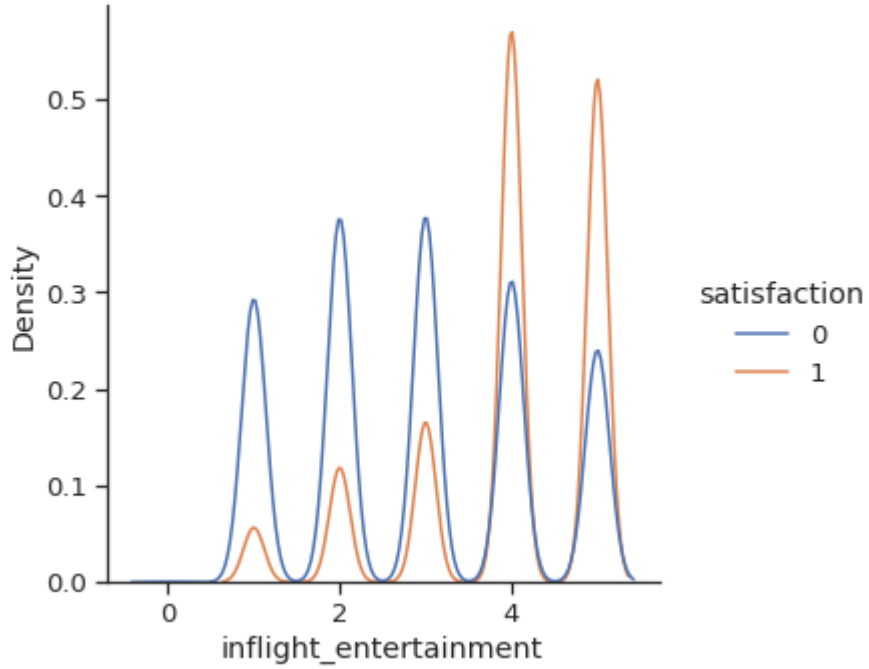
```
main_features = ['inflight_wifi_service', 'ease_of_online_booking', 'onl


sns.set(font_scale= 1.2)
sns.set_style('ticks')

for i, feature in enumerate(main_features):
    sns.displot(data=data_raw, x=feature, kind='kde', hue='satisfaction'

sns.despine()
```


Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
inflight_wifi_service
satisfaction
0
1


Density
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
ease_of_online_booking
satisfaction
0
1


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
0
2
4
online_boarding
satisfaction
0
1

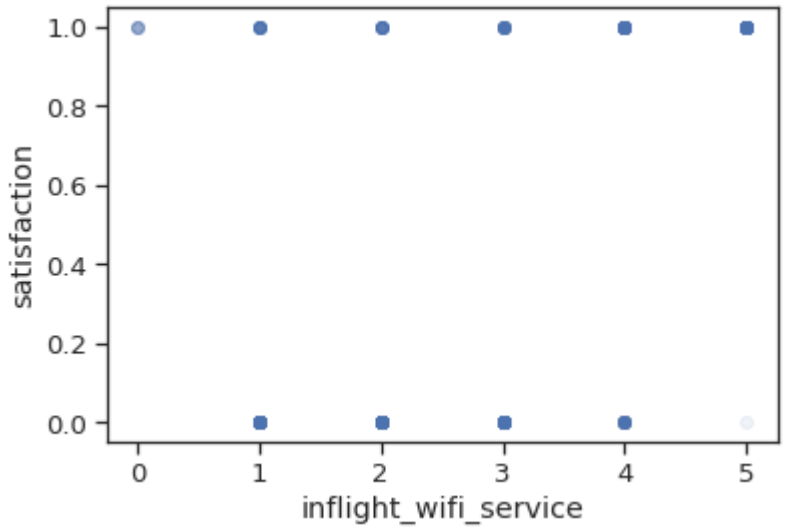

0.5
0.4
0.3
0.2
0.1
0.0
Density
0
2
4
seat_comfort
satisfaction
0
1

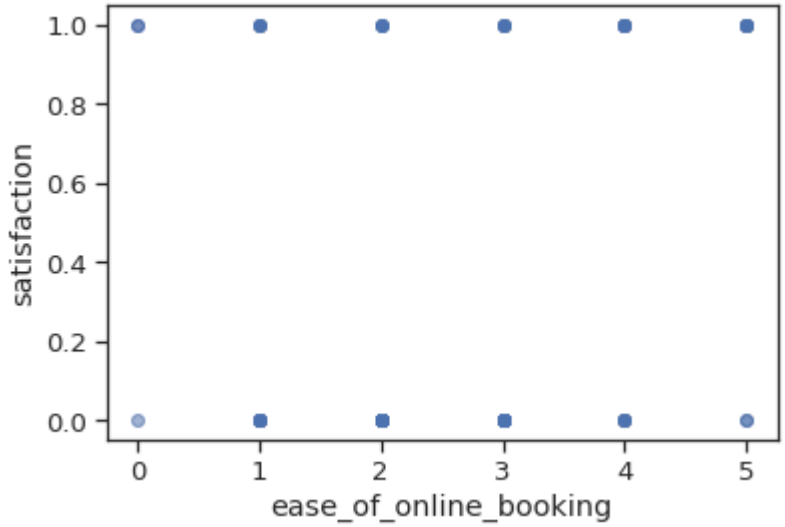


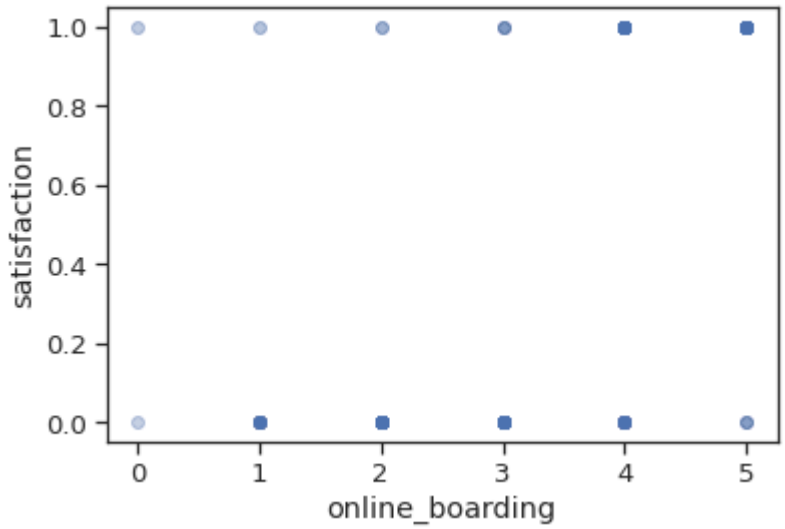
Попробуем построить диагрммы рассеивания для выделенных признаков и признака 'satisfaction'.

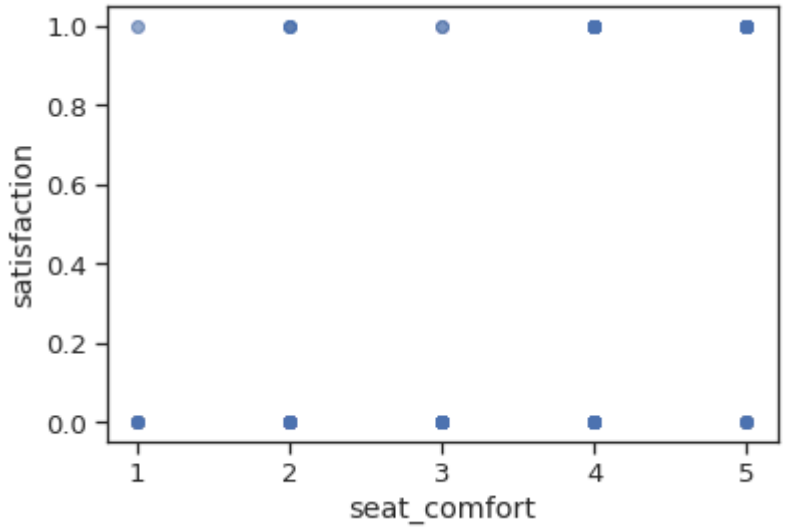
In [ ]:

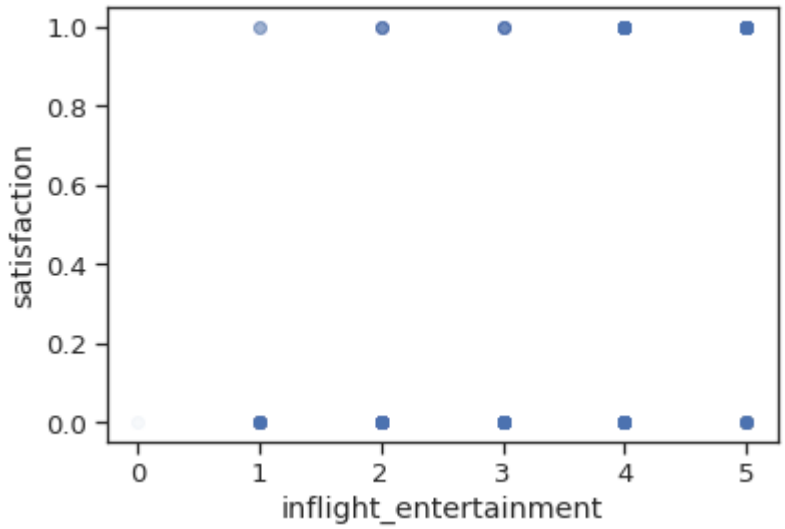
```
np.random.seed(42)

for i, feature in enumerate(main_features):
    random_subset = np.random.choice(np.arange(data_raw.shape[0]), size=
    plt.scatter(data_raw.iloc[random_subset][feature], data_raw.iloc[ran
    plt.xlabel(feature)
    plt.ylabel('satisfaction')
    plt.show()
```




satisfaction
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
ease_of_online_booking


satisfaction
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
online_boarding


satisfaction
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
seat_comfort

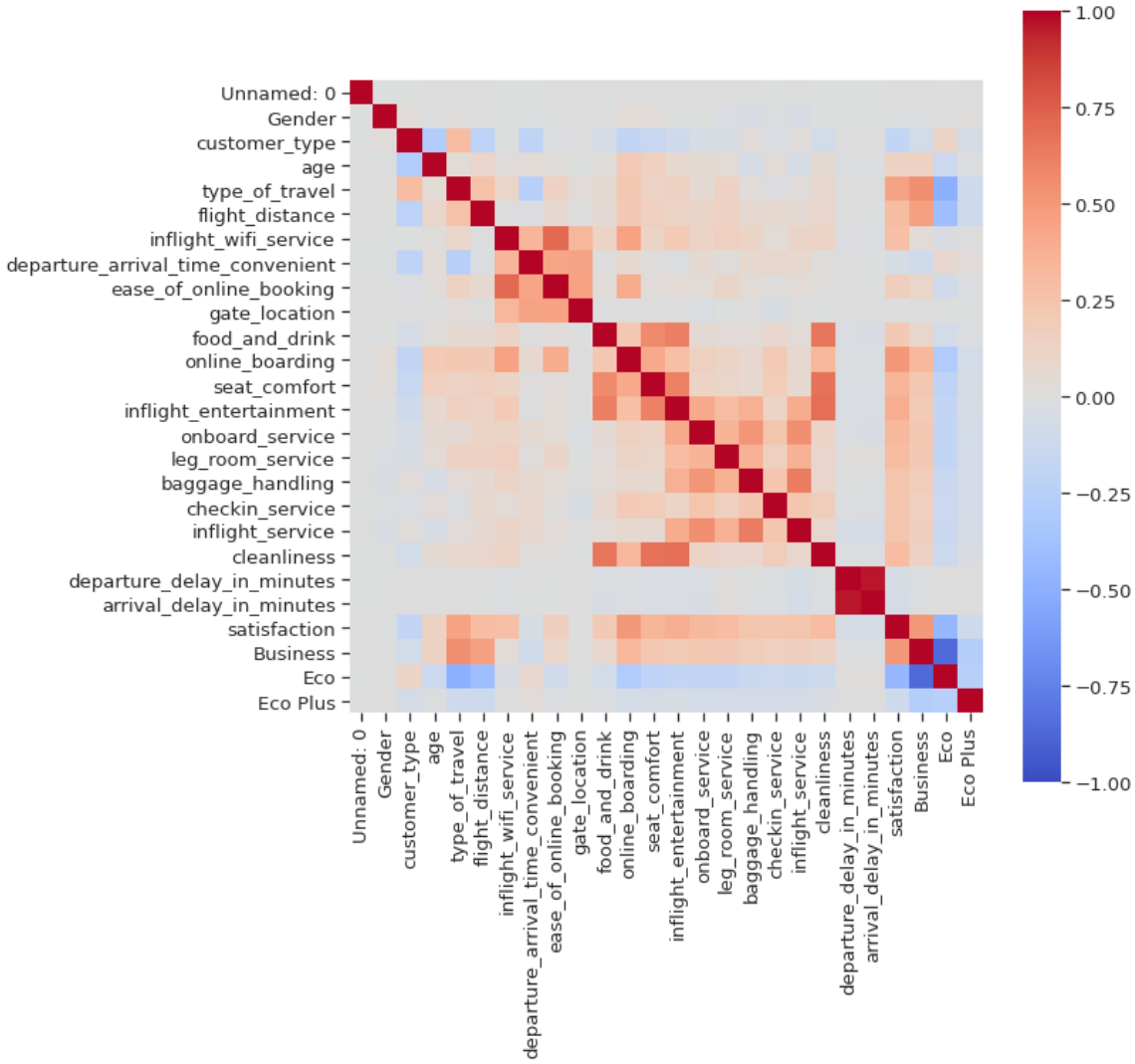


Так как изнвчально 'satisfaction' был категориальным, из данных диаграмм не просто вявить зависимость. интенсивность цвета точек показывает частоту встречаемости , например , по первой диаграмме можно определить что в большинстве случаев чем лучше wi-fi (5) тем люди более удовлетворены.

Посмотрим еще раз как сильно связаны между собой признаки. Чем светлее ячейка, тем меньше по абсолютной величине коэффициент корреляции:

In [ ]:

```
plt.figure(figsize = (10, 10))
corr_mat = data_r.corr()
sns.heatmap(corr_mat, square=True, vmin=-1, vmax=1, cmap='coolwarm')
pass
```



Выведем коэффициенты корреляции, большие заданного значения:

In [ ]:
```
corr_mat.where(corr_mat > 0.4).stack().tail(10)
```

Out[ ]:
```
satisfaction  type_of_travel     0.449861
              online_boarding    0.501749
              satisfaction       1.000000
              Business           0.502476
Business      type_of_travel     0.552173
              flight_distance    0.466594
              satisfaction       0.502476
              Business           1.000000
Eco           Eco                1.000000
Eco Plus      Eco Plus           1.000000
dtype: float64
```

Можно заметить, что для параметра satisfaction высокая связь наблюдается с online_boarding и с Business

In [ ]:
```
corr_mat.where(np.triu(corr_mat > 0.5, k=1)).stack().sort_values(ascendi
```

Out[ ]:
```
departure_delay_in_minutes  arrival_delay_in_minutes    0.965291
inflight_wifi_service       ease_of_online_booking      0.714807
inflight_entertainment      cleanliness                 0.692511
seat_comfort                cleanliness                 0.679613
food_and_drink              cleanliness                 0.658054
baggage_handling            inflight_service            0.629237
food_and_drink              inflight_entertainment      0.623461
```

```
seat_comfort                 inflight_entertainment      0.611837
food_and_drink               seat_comfort                0.575846
type_of_travel               Business                    0.552173
onboard_service              inflight_service            0.551569
                             baggage_handling            0.520296
satisfaction                 Business                    0.502476
online_boarding              satisfaction                0.501749
dtype: float64
```

В итоге, попробуем расположить в порядке убывания самые важные для нас признаки : online_boarding, customer_class, type_of_travel, inflight_entertainment, (flight_distance, inflight_wifi_service, ease_of_online_booking, seat_comfort - более слабо связанные признаки)

4. Заполнение пропущенных значений

Посмотрим, сколько пропущенных значений в каждом столбце матрицы:

In [ ]:
```
data_r.isna().sum()
```

Out[ ]:
```
Unnamed: 0                             0
Gender                                 0
customer_type                          0
age                                    0
type_of_travel                         0
flight_distance                        0
inflight_wifi_service                  0
departure_arrival_time_convenient      0
ease_of_online_booking                 0
gate_location                          0
food_and_drink                         0
online_boarding                        0
seat_comfort                           0
inflight_entertainment                 0
onboard_service                        0
leg_room_service                       0
baggage_handling                       0
checkin_service                        0
inflight_service                       0
cleanliness                            0
departure_delay_in_minutes             0
arrival_delay_in_minutes             393
satisfaction                           0
Business                               0
Eco                                    0
Eco Plus                               0
dtype: int64
```

Так как задержка прилета зависит от задержки вылета (это видно из предыдущего пункта анализа - между этими параметрами очень высокий уровень корреляции 0,94), логично заполнить пропущенные ячейки в столбце задержка прилета значениями из столбца задерка вылета. Можно было и удалить (пропущенных значений не так много по сравнению со всем набором данных), в данном случае заменять медианой или средним значением кажется не целесообразным.

Заполним пропущенные значения в столбцах, соответствующих числовым признакам:

In [ ]:
```
data_r['arrival_delay_in_minutes'].fillna(data_r['departure_delay_in_min
print (data_r.isnull().sum())
```

```
Unnamed: 0                            0
Gender                                0
customer_type                         0
age                                   0
type_of_travel                        0
flight_distance                       0
inflight_wifi_service                 0
departure_arrival_time_convenient     0
ease_of_online_booking                0
gate_location                         0
food_and_drink                        0
online_boarding                       0
seat_comfort                          0
inflight_entertainment                0
onboard_service                       0
leg_room_service                      0
baggage_handling                      0
checkin_service                       0
inflight_service                      0
cleanliness                           0
departure_delay_in_minutes            0
arrival_delay_in_minutes              0
satisfaction                          0
Business                              0
Eco                                   0
Eco Plus                              0
dtype: int64
```

In [ ]:
```
data_r.isna().sum()
```

Out[ ]:
```
Unnamed: 0                            0
Gender                                0
customer_type                         0
age                                   0
type_of_travel                        0
flight_distance                       0
inflight_wifi_service                 0
departure_arrival_time_convenient     0
ease_of_online_booking                0
gate_location                         0
food_and_drink                        0
online_boarding                       0
seat_comfort                          0
inflight_entertainment                0
onboard_service                       0
leg_room_service                      0
baggage_handling                      0
checkin_service                       0
inflight_service                      0
cleanliness                           0
departure_delay_in_minutes            0
arrival_delay_in_minutes              0
satisfaction                          0
Business                              0
Eco                                   0
Eco Plus                              0
dtype: int64
```

5. Нормализация количественных признаков

Выполним стандартизацию - линейное преобразование, приводящее все значения к нулевому среднему и единичному стандартному отклонению,всех признаков:

In [ ]:
```
data_stand = (data_r - data_r.mean(axis = 0))/data_r.std(axis = 0)
```

In [ ]:
```
data_stand.describe()
```

Out[ ]:

	Unnamed: 0	Gender	customer_type	age	type_of_travel	flight_dis
count	1.298800e+05	1.298800e+05	1.298800e+05	1.298800e+05	1.298800e+05	1.29880
mean	3.164495e-18	6.856576e-16	9.267046e-17	-8.443226e-17	8.642626e-16	1.72180
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.00000
min	-1.732031e+00	-1.014874e+00	-4.734200e-01	-2.144797e+00	-1.493946e+00	-1.16227
25%	-8.660154e-01	-1.014874e+00	-4.734200e-01	-8.219896e-01	-1.493946e+00	-7.78299
50%	0.000000e+00	9.853362e-01	-4.734200e-01	3.783516e-02	6.693632e-01	-3.47200
75%	8.660154e-01	9.853362e-01	-4.734200e-01	7.653792e-01	6.693632e-01	5.55097
max	1.732031e+00	9.853362e-01	2.112273e+00	3.014152e+00	6.693632e-01	3.80237

6. Боремся с выбросами (outliers)

Для обнаружения выбросов найдем, квантили для признаков `flight_distance`, `departure_delay_in_minutes` `arrival_delay_in_minutes` - задержка времени вылета и прибытия в минутах.

In [ ]:
```
data_stand['flight_distance'].quantile([0.005,.01,.05,.1,.5,.9,.95,.99,.
```

Out[ ]:
```
0.005   -1.107137
0.010   -1.092099
0.050   -1.015955
0.100   -0.956754
0.500   -0.347201
0.900    1.564670
0.950    2.195276
0.990    2.700563
0.995    2.761323
Name: flight_distance, dtype: float64
```

In [ ]:
```
data_stand['departure_delay_in_minutes'].quantile([0.005,.01,.05,.1,.5,.
```

Out[ ]:
```
0.005   -0.386480
0.010   -0.386480
0.050   -0.386480
0.100   -0.386480
0.500   -0.386480
0.900    0.769252
0.950    1.636051
0.990    4.341513
0.995    5.707377
Name: departure_delay_in_minutes, dtype: float64
```

In [ ]:
```
data_stand['arrival_delay_in_minutes'].quantile([0.005,.01,.05,.1,.5,.9,
```

Out[ ]:
```
0.005   -0.392753
0.010   -0.392753
0.050   -0.392753
0.100   -0.392753
0.500   -0.392753
0.900    0.747155
0.950    1.653900
0.990    4.348229
0.995    5.747207
Name: arrival_delay_in_minutes, dtype: float64
```

Удалим все строки таблицы, в которых `flight_distance`, `departure_delay_in_minutes` или `arrival_delay_in_minutes` выходят за пределы квантилей 0.005, 0.995.

In [ ]:
```
rows_to_drop = data_stand[
    (data_stand['flight_distance'] < data_stand['flight_distance'].quant
    (data_stand['arrival_delay_in_minutes']  < data_stand['arrival_delay
    (data_stand['departure_delay_in_minutes']  < data_stand['departure_d
data_outliers = data_stand.drop(rows_to_drop)
data_outliers.shape
```

Out[ ]:
```
(128034, 26)
```