**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский экономический университет имени Г. В. Плеханова»**

Кафедра информатики

Выпускная квалификационная работа

по программе профессиональной переподготовки «Процедурно-ориентированное программирование в прикладных задачах анализа данных в экономике»

на тему

«Анализ степени удовлетворенности авиапассажиров, выработка рекомендаций по ее увеличению»

Выполнил:

Иванов Илья Александрович

Преподаватель:

ст. преп. Савинова Виктория Михайловна

Москва

2022

# Понимание бизнес-целей

## Понимание бизнеса

В связи с нарастающим кризисом в сфере авиаперевозок как у российских, так и у зарубежных авиакомпаний, обусловленным многочисленными политическими (санкции на поставку авиатехники и обслуживание, закрытие воздушного пространства над многими странами) и экономическими факторами (рост цен на авиакеросин), требуется сохранение и повышение лояльности пассажиров. С этой целью необходим анализ объективных и субъективных факторов, которые влияют на степени удовлетворенности авиапассажиров, и построение модели, позволяющей прогнозировать итоговую удовлетворенность на основе этих факторов.

## Доступные ресурсы

Для успешной реализации проекта необходимы следующие категории специалистов: аналитик данных, бизнес-аналитик, специалист по базам данных, руководитель проекта.

Заказчик располагает всем необходимым оборудованием для проведения анализа данных.

## Риски

1. Несоблюдение сроков проекта

2. Риск неплатежеспособности заказчика

3. Риск нехватки и неполноты данных

4. Риск несоответствия полученных результатов требованиям заказчика.

## Ограничения

Анализ ограничен представленными данными, дополнительный сбор информации не проводится.

## Цели исследования данных

Задачи анализа данных, решаемые в рамках проекта:

- Построение визуализации по представленным датасетам.

- Решение задачи классификации (прогнозирования) степени удовлетворенности авиапассажиров на основании объективных (возраст, пол пассажира, дистанция полета и т.д.) и субъективных (удобство самолетных кресел, пространство для ног и т.д.) параметров.

## Критерии успешности изучения данных.

Построение расчетной модели, позволяющей с достаточной точностью выполнять прогнозирование степени удовлетворенности авиапассажиров.

## Начальное изучение данных.

## Сбор данных

Данные на анализ взяты из открытого доступа (https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction)

## Описание данных

Объем данных – на анализ представлено 2 файла (airline\_train.csv и airline\_test.csv), суммарным объемом ~15 МБ. Первый файл будет использоваться в качестве обучающей выборки, второй – в качестве тестовой.

Типы, виды данных и схемы кодирования:

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Тип данных | Вид данных | Схема кодирования |
| Unnamed | int64 | Непрерывный | - |
| id | int64 | Непрерывный | - |
| Gender | object | Дискретный | - |
| Customer Type | object | Дискретный | - |
| Age | int64 | Непрерывный | - |
| Type of Travel | object | Дискретный | - |
| Class | object | Дискретный | - |
| Flight Distance | int64 | Дискретный |  |
| Inflight wifi service | int64 | Дискретный | Целое число (от 0 до 5) |
| Departure/Arrival time convenient | int64 | Дискретный | Целое число (от 0 до 5) |
| Ease of Online booking | int64 | Дискретный | Целое число (от 0 до 5) |
| Gate location | int64 | Дискретный | Целое число (от 0 до 5) |
| Food and drink | int64 | Дискретный | Целое число (от 0 до 5) |
| Online boarding | int64 | Дискретный | Целое число (от 0 до 5) |
| Seat comfort | int64 | Дискретный | Целое число (от 0 до 5) |
| Inflight entertainment | int64 | Дискретный | Целое число (от 0 до 5) |
| On-board service | int64 | Дискретный | Целое число (от 0 до 5) |
| Leg room service | int64 | Дискретный | Целое число (от 0 до 5) |
| Baggage handling | int64 | Дискретный | Целое число (от 0 до 5) |
| Checkin service | int64 | Дискретный | Целое число (от 0 до 5) |
| Cleanliness | int64 | Дискретный | Целое число (от 0 до 5) |
| Departure Delay in Minutes | int64 | Непрерывный | - |
| Arrival Delay in Minutes | float64 | Непрерывный | - |
| satisfaction | object | Дискретный | - |

В качестве выходной переменной принимаем satisfaction.

Формат данных – файл csv, разделитель – “,”.

## Исследование данных

Построение описательной статистики

Обучающая выборка:

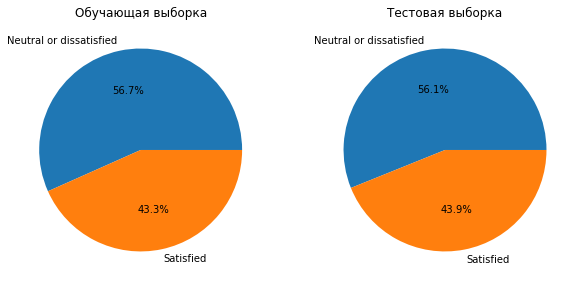
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Age** | **Flight Distance** | **Departure Delay in Minutes** | **Arrival Delay in Minutes** |
| **count** | 103904.000000 | 103904.000000 | 103904.000000 | 103594.000000 |
| **mean** | 39.379706 | 1189.448375 | 14.815618 | 15.178678 |
| **std** | 15.114964 | 997.147281 | 38.230901 | 38.698682 |
| **min** | 7.000000 | 31.000000 | 0.000000 | 0.000000 |
| **25%** | 27.000000 | 414.000000 | 0.000000 | 0.000000 |
| **50%** | 40.000000 | 843.000000 | 0.000000 | 0.000000 |
| **75%** | 51.000000 | 1743.000000 | 12.000000 | 13.000000 |
| **max** | 85.000000 | 4983.000000 | 1592.000000 | 1584.000000 |

Тестовая выборка:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Age | Flight Distance | Departure Delay in Minutes | Arrival Delay in Minutes |
| **count** | 25976.000000 | 25976.000000 | 25976.00000 | 25893.000000 |
| **mean** | 39.620958 | 1193.788459 | 14.30609 | 14.740857 |
| **std** | 15.135685 | 998.683999 | 37.42316 | 37.517539 |
| **min** | 7.000000 | 31.000000 | 0.00000 | 0.000000 |
| **25%** | 27.000000 | 414.000000 | 0.00000 | 0.000000 |
| **50%** | 40.000000 | 849.000000 | 0.00000 | 0.000000 |
| **75%** | 51.000000 | 1744.000000 | 12.00000 | 13.000000 |
| **max** | 85.000000 | 4983.000000 | 1128.00000 | 1115.000000 |

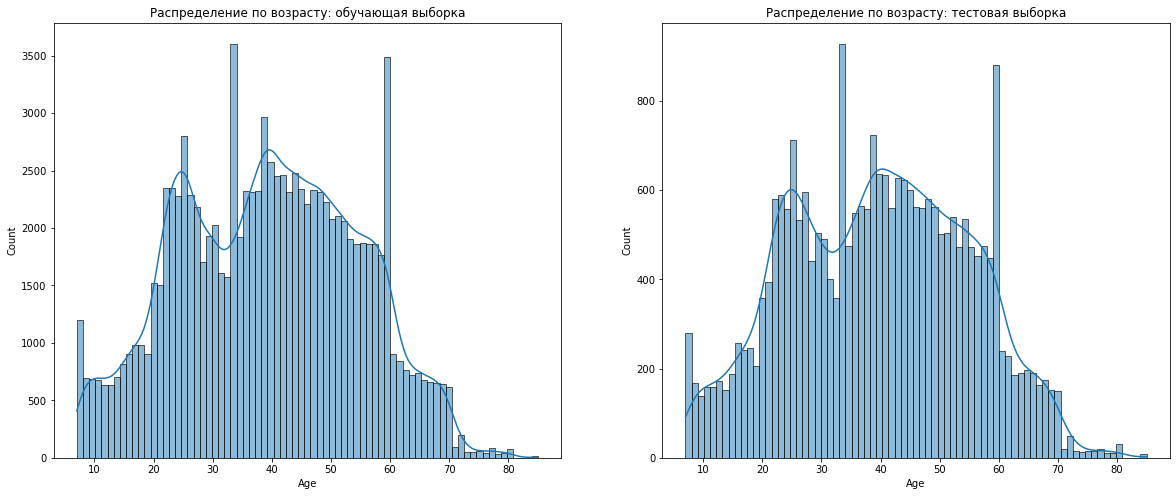
Для ознакомления со структурой данных и проведения предварительного анализа построим несколько графиков.

Распределение выходного параметра (satisfaction) по выборкам в целом:



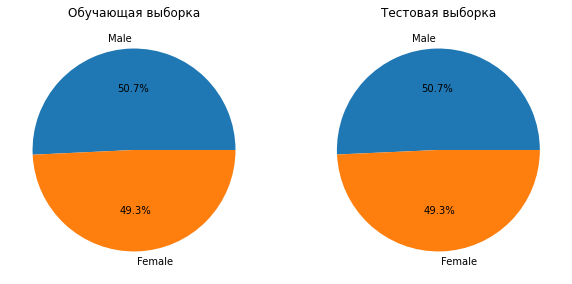
Таким образом, выборка достаточно сбалансирована.

Распределение по возрасту:



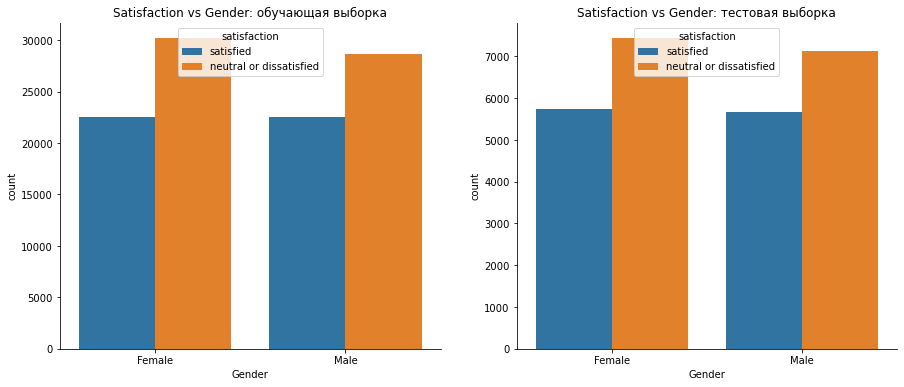
Видим, что в обоих выборках графики примерно одинаковые и соответствуют нормальному распределению (кроме левой части, где, вероятно, исключены дети младше 7 лет).

Распределение по полу в целом по выборкам:



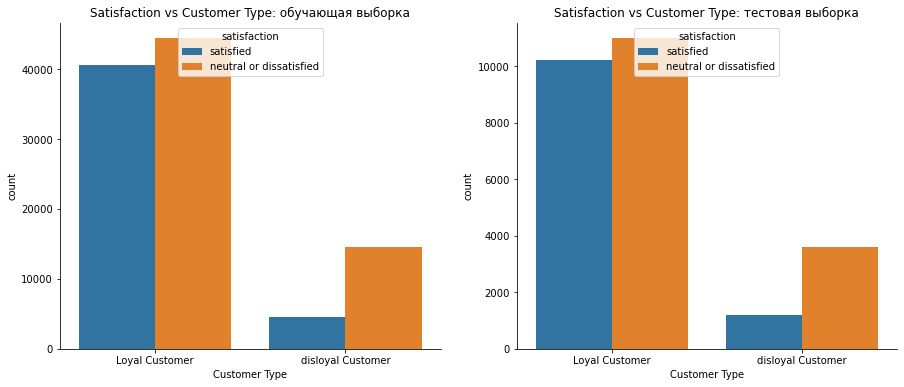
Количество мужчин и женщин в обоих выборках одинаково. Это, прежде всего, говорит о том, что выборка синтетическая.

Распределение по полу с учетом выходного параметра (satisfaction):



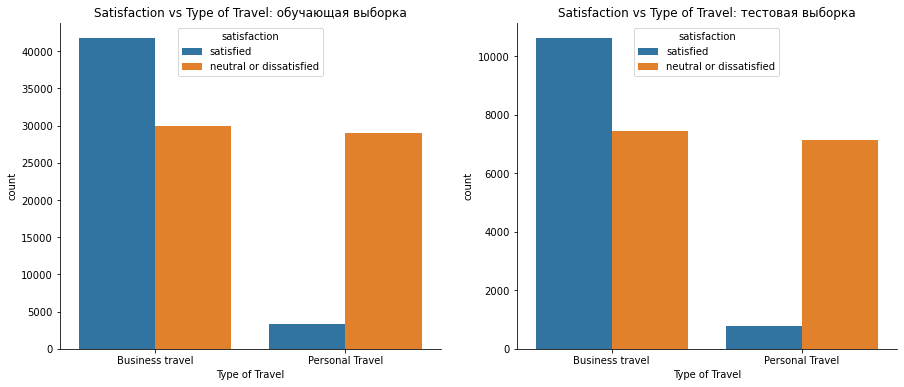
В данном случае видим, что процент довольных полетом и среди женщин, и среди мужчин примерно одинаковый.

Распределение по постоянным/непостоянным клиентам с учетом выходного параметра (satisfaction):



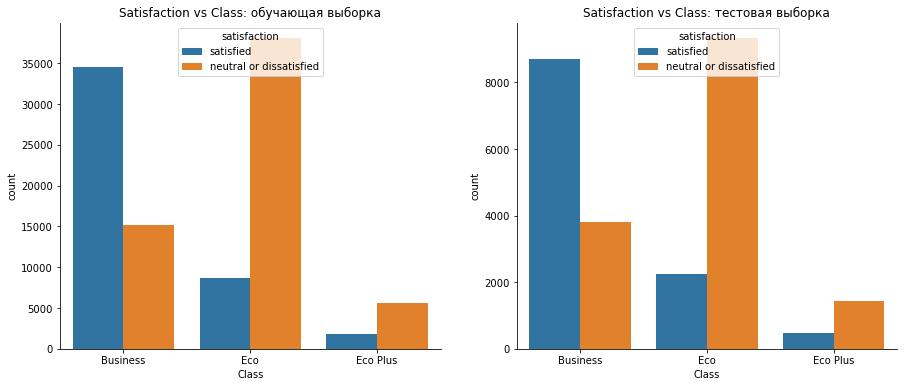
Видим, что процент людей, довольных полетом, среди постоянных клиентов гораздо выше, чем среди непостоянных.

Распределение по выходному параметру (satisfaction) с учетом типа поездки:



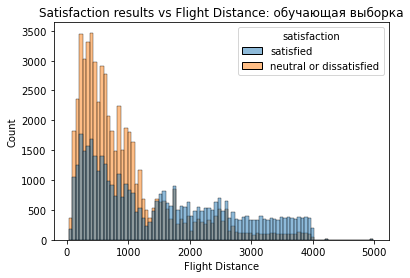
Среди тех, кто совершает бизнес-поездки, процент людей, довольных полетом, гораздо выше, чем среди тех, кто летает по личным делам.

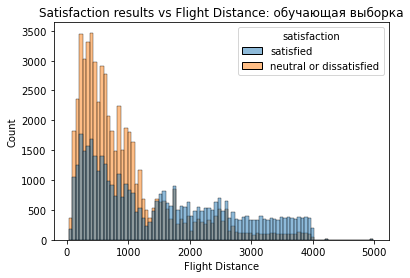
Распределение по выходному параметру (satisfaction) с учетом класса поездки:



Как видим, среди летающих бизнес-классом процент людей, довольных полетом, гораздо выше, чем среди тех, кто летает классами Eco и Eco Plus.

Рассмотрим зависимость параметра Satisfaction от расстояния полета:





Как видим, на коротких маршрутах процент тех, кто доволен полетом, гораздо ниже по сравнению с теми, кто летает на дальние расстояния.

Как представляется, на основе вышеприведенных графиков можно сделать вывод, что обе выборки являются сбалансированными по всем объективным параметрам и в достаточно полной мере отражают тенденции и распределения, существующие на данный момент в авиаотрасли.

Распределение субъективных параметров в графической форме в настоящем отчете не приводится в силу ограниченности места. Выводы по ним в целом соответствуют вышеприведенному (удовлетворенность путешествующих бизнес-классом выше, чем по остальным классам, например, по таким важным параметрам, как Leg room service и Seat comfort, и т.д.).

# Подготовка данных

Проверка на дубликаты: дубликаты отсутствуют.

Проверка на пропуски: имеются пропуски в столбце Arrival Delay in Minutes.

Проверка на противоречия: противоречия отсутствуют.

Разбиение на обучающую и тестовую выборки не требуется, так как на анализ представлено 2 файла.

Файлы сбалансированы (см. графики выше).

Дополнительная балансировка не требуется.

Данные требуют следующей очистки:

- Необходимо удалить столбцы Unnamed и id (обоснование: не имеют смысловой нагрузки при анализе)

- Необходимо удалить строки, соответствующие возрасту пассажира менее 14 лет (обоснование: ребенок не может адекватно оценить некоторые параметры, по которым проводится анализ, в частности, удобство онлайн-бронирования и удобство сдачи/получения багажа, т.к. не проводит эти действия самостоятельно). В пользу этого решения говорит тот факт, что выборка для возраста до 14 лет не является сбалансированной по выходному параметру satisfaction:

|  |
| --- |
|  |

- Необходимо удалить строки, соответствующие длине маршрута менее 250 км (обоснование: на коротких маршрутах должны применяться другие оценочные критерии. В частности, вряд ли в данном случае на оценку со стороны пассажира будет влиять наличие wi-fi на борту самолета или напитки/питание, а вот турбулентность, характерная для малоразмерных самолетов, напротив, будет иметь существенное значение).

- Имеется сильная прямая корреляция между столбцами Departure Delay in Minutes и Arrival Delay in Minutes (0,97 как на обучающей, так и на тестовой выборке).

|  |  |
| --- | --- |
| Обучающая выборка | Тестовая выборка |
|  |  |

В связи с тем, что (а) эти столбцы, очевидно, являются взаимозависимыми, и (b) в столбце Arrival Delay in Minutes имеются пропуски (310 в обучающей выборке, 83 в тестовой), принято следующее решение:

* заменить пропуски в Arrival Delay in Minutes на соответствующие значения из Departure Delay in Minutes;
* вычислить среднее значение между этими двумя столбцами и внести его в Departure Delay in Minutes;
* переименовать этот столбец в Delay in Minutes,
* удалить столбец Arrival Delay in Minutes;
* в расчетах далее использовать столбец Delay in Minutes.

- Необходимо удалить строки, соответствующие чрезмерно высоким задержкам (обоснование: это выбросы, согласно описательной статистике и вышеприведенным графикам). Отсечение проведем по правилу 5 сигм (~200 минут).

- В некоторых столбцах тестовой выборки диапазон значений, представляющих субъективную оценку пассажира, составляет от 1 до 5, в отличие от обучающей выборки, в которой диапазон значений составляет от 0 до 5 по всем указанным столбцам. В связи с этим необходимо при проведении анализа найти в обучающей выборке строки со значениями, отсутствующими в тестовой выборке, и добавить их в тестовую выборку.

После подготовки данных получаем 4 набора данных (X\_train, y\_train, X\_test, y\_test), которые и будем использовать при последующем анализе.

# Моделирование

Моделирование проведено при помощи следующих методов:

- Метод ближайших соседей

- Наивный Байесовский классификатор (Гаусс, Бернулли)

- Логистическая регрессия

- Дерево решений

- Искусственная нейронная сеть

Оценка результатов проводилась следующими метриками:

- Accuracy

- Recall

- Precision

- f1

Получены следующие результаты:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Расчетный метод | | | | | |
| Метрика оценки | Ближ. соседей | Наивный Байес | | Лог. регр. | Дерево реш. | Нейр. сеть |
| Гаусс | Бернулли |
| Accuracy | 0.9449 | 0.8015 | 0.8832 | 0.9355 | 0.9513 | 0.9612 |
| Recall | 0.9214 | 0.9559 | 0.8739 | 0.9201 | 0.9467 | 0.9405 |
| Precision | 0.9579 | 0.7130 | 0.8738 | 0.9393 | 0.9479 | 0.9747 |
| f1 | 0.9393 | 0.8168 | 0.8739 | 0.9296 | 0.9473 | 0.9573 |
| Время выполнения расчета | 51.43 | 0.5 (суммарно) | | 7.81 | 1.31 | 292.97 |

Примечание: время выполнения расчета указано приблизительно (с точностью до десятых долей секунд), с учетом верификации, и может использоваться только для относительной оценки, так как зависит от характеристик оборудования, на котором проводится расчет.

# Оценка результатов

Как видно из таблицы выше, все расчетные методы показывают хорошие результаты, однако наилучшие параметры по всем метрикам наблюдаются при расчете при помощи искусственной нейронной сети (однако этот метод и самый затратный с точки зрения времени).

# Внедрение

Исходя из того, что полученная модель показывает хорошие результаты, ее можно использовать для предсказания удовлетворенности пассажиров. Для дополнительного повышения точности представляется целесообразным провести следующие действия:

- Выполнить дополнительную балансировку выборок, получив соотношение 50/50 вместо текущих 56/44.

- Изменить текущие параметры субъективной оценки для пассажиров младшего возраста (удалить, например, удобство онлайн-бронирования).

- Выяснить параметры коротких рейсов, проанализировать применимость текущих критериев к ним.

- Ввести дополнительные параметры, такие, как тип самолета, способ посадки на самолет/выхода из него (трап + автобус либо напрямую из здания аэропорта), удобство стыковки (при наличии) и т.д.

- При необходимости выполнить подбор параметров расчетных моделей (учитывая затраты времени и имеющееся оборудование).

Выдача дополнительных рекомендаций по повышению степени удовлетворенности пассажиров по результатам проведенного анализа не представляется возможной (для этого необходим расчет влияния отдельных факторов на конечную оценку, что выходит за рамки данного проекта).

**Приложение 1.** Код программы Python для проведенного анализа

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

import sklearn

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import MinMaxScaler

from sklearn.naive\_bayes import BernoulliNB, GaussianNB

from sklearn.linear\_model import LogisticRegression

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score

# from sklearn.metrics import mean\_absolute\_percentage\_error, r2\_score

from sklearn.tree import DecisionTreeRegressor

from sklearn.neural\_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, f1\_score

import datetime

import warnings

warnings.filterwarnings("ignore")

# загрузка данных из файлов

df\_train = pd.read\_csv('airline\_train.csv')

df\_test = pd.read\_csv('airline\_test.csv')

#проверка на пропуски и дубликаты

# print(df\_train.head())

# print(df\_train.isnull().sum())

# print(df\_train.duplicated().sum())

# print(df\_test.head())

# print(df\_test.isnull().sum())

# print(df\_test.duplicated().sum())

# корреляция

# corr\_train = df\_train['Departure Delay in Minutes'].corr(df\_train['Arrival Delay in Minutes'])

# corr\_test = df\_test['Departure Delay in Minutes'].corr(df\_test['Arrival Delay in Minutes'])

############

##### Визуализация ##############

# plt.figure(figsize=(10, 10))

# plt.subplot(1, 2, 1)

# plt.pie(df\_train.satisfaction.value\_counts(), labels = ["Neutral or dissatisfied", "Satisfied"], autopct = '%1.1f%%')

# plt.title('Обучающая выборка')

# plt.subplot(1, 2, 2)

# plt.pie(df\_test.satisfaction.value\_counts(), labels = ["Neutral or dissatisfied", "Satisfied"], autopct = '%1.1f%%')

# plt.title('Тестовая выборка')

# plt.show()

####################################

# df\_train\_young = df\_train[df\_train['Age'] < 14]

# df\_test\_young = df\_test[df\_test['Age'] < 14]

# plt.figure(figsize=(15, 15))

# plt.subplot(1, 2, 1)

# plt.pie(df\_train\_young.satisfaction.value\_counts(), labels = ["Neutral or dissatisfied", "Satisfied"], autopct = '%1.1f%%')

# plt.title('Возраст < 14 лет: обучающая выборка')

# plt.subplot(1, 2, 2)

# plt.pie(df\_test\_young.satisfaction.value\_counts(), labels = ["Neutral or dissatisfied", "Satisfied"], autopct = '%1.1f%%')

# plt.title('Возраст < 14 лет: тестовая выборка')

# plt.show()

###########################################

# plt.figure(figsize=(10, 10))

# plt.subplot(1, 2, 1)

# plt.pie(df\_train.Gender.value\_counts(), labels = ["Male", "Female"], autopct = '%1.1f%%')

# plt.title('Обучающая выборка')

# plt.subplot(1, 2, 2)

# plt.pie(df\_test.Gender.value\_counts(), labels = ["Male", "Female"], autopct = '%1.1f%%')

# plt.title('Тестовая выборка')

# plt.show()

####################################################################

# plt.figure(figsize=(15, 6))

# plt.subplot(1, 2, 1)

# sns.countplot(x ="Gender", data = df\_train, hue ="satisfaction", order = df\_train['Gender'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Gender: обучающая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.subplot(1, 2, 2)

# sns.countplot(x ="Gender", data = df\_test, hue ="satisfaction", order = df\_train['Gender'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Gender: тестовая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.show()

###################################################################

# plt.figure(figsize=(15, 6))

# plt.subplot(1, 2, 1)

# sns.countplot(x ="Customer Type", data = df\_train, hue ="satisfaction", order = df\_train['Customer Type'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Customer Type: обучающая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.subplot(1, 2, 2)

# sns.countplot(x ="Customer Type", data = df\_test, hue ="satisfaction", order = df\_train['Customer Type'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Customer Type: тестовая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.show()

# ###################################

# plt.figure(figsize=(20, 8))

# plt.subplot(1, 2, 1)

# sns.histplot( x= "Age", data = df\_train, kde= True, bins = 75)

# plt.title('Распределение по возрасту: обучающая выборка')

# plt.subplot(1, 2, 2)

# sns.histplot( x= "Age", data = df\_test, kde= True, bins = 75)

# plt.title('Распределение по возрасту: тестовая выборка')

# plt.show()

# ###################################

# ###################################

# plt.figure(figsize=(15, 6))

# plt.subplot(1, 2, 1)

# sns.countplot(x ="Type of Travel", data = df\_train, hue ="satisfaction", order = df\_train['Type of Travel'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Type of Travel: обучающая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.subplot(1, 2, 2)

# sns.countplot(x ="Type of Travel", data = df\_test, hue ="satisfaction", order = df\_train['Type of Travel'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Type of Travel: тестовая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.show()

###################################

# ###################################

# plt.figure(figsize=(15, 6))

# plt.subplot(1, 2, 1)

# sns.countplot(x ="Class", data = df\_train, hue ="satisfaction", order = df\_train['Class'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Class: обучающая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.subplot(1, 2, 2)

# sns.countplot(x ="Class", data = df\_test, hue ="satisfaction", order = df\_train['Class'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

# })

# plt.title("Satisfaction vs Class: тестовая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='upper center', title = "satisfaction")

# plt.show()

# def classes (df):

# plt.figure(figsize=(5, 5))

# sns.countplot(x ="Class", data = df, hue ="satisfaction", order = df\_train['Class'].value\_counts().index, hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'

#

# plt.title("Satisfaction results vs Class: обучающая выборка")

# sns.despine(top = True, right = True, left = False, bottom = False)

# plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), title = "satisfaction")

# plt.show()

# classes(df\_train)

# classes(df\_test)

# ###################################

# sns.histplot(x = "Flight Distance", data = df\_train, hue ="satisfaction", hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'})

# plt.title("Satisfaction results vs Flight Distance: обучающая выборка")

# plt.show()

# sns.histplot(x = "Flight Distance", data = df\_test, hue ="satisfaction", hue\_order = ['satisfied', 'neutral or dissatisfied'], palette = {

# 'satisfied': 'tab:blue',

# 'neutral or dissatisfied': 'tab:orange'})

# plt.title("Satisfaction results vs Flight Distance: тестовая выборка")

# plt.show()

########################################

# оценка субъективных параметров

# df\_not\_business = df\_train[df\_train['Class'] != 'Business'].loc[:, 'Inflight wifi service':'Cleanliness']

# df\_business = df\_train[df\_train['Class'] == 'Business'].loc[:, 'Inflight wifi service':'Cleanliness']

# df\_male = df\_train[df\_train['Gender'] == 'Male'].loc[:, 'Inflight wifi service':'Cleanliness']

# df\_female = df\_train[df\_train['Gender'] == 'Female'].loc[:, 'Inflight wifi service':'Cleanliness']

# df\_train\_all = df\_train.loc[:, 'Inflight wifi service':'Cleanliness']

# df\_test\_all = df\_test.loc[:, 'Inflight wifi service':'Cleanliness']

# def figures(df):

# plt.figure(figsize=(20, 20))

# for i in range(df.shape[1]):

# plt.subplot(4, 4, i+1)

# labels = sorted(list(df.iloc[:, i].unique()))

# plt.pie(df.iloc[:, i].value\_counts(), labels = labels, counterclock = False, startangle = 90, autopct = '%1.1f%%')

# plt.title(df.columns[i])

# plt.show()

# print('По выборке в целом:')

# figures(df\_train\_all)

# print('По бизнес-классу:')

# figures(df\_business)

# print('По остальным классам:')

# figures(df\_not\_business)

# print('По мужчинам:')

# figures(df\_male)

# print('По женщинам:')

# figures(df\_female)

# ######################

########################################### Расчеты ###########################

# проверка на полноту значений в столбцах с субъективной оценкой со стороны пассажира (должно быть от 0 до 5 во всех столбцах)

df\_check\_train = df\_train.loc[:, 'Inflight wifi service':'Cleanliness'] # отбираем нужные столбцы

df\_check\_test = df\_test.loc[:, 'Inflight wifi service':'Cleanliness']

for i in range(df\_check\_test.shape[1]): # перебор по столбцам

if df\_check\_train.iloc[:,i].nunique() != df\_check\_test.iloc[:,i].nunique(): # если число уникальных значений в столбцах

#обучающей и тестовой выборки отличается, то:

for j in range(len(df\_check\_train.iloc[:,i].unique())): # перебор по каждому значению в столбце

if j not in df\_check\_test.iloc[:,i].unique(): # если не находим:

nr\_add = (df\_check\_train[df\_check\_train.iloc[:, i] == j].index.values)[0] # добавляем отсутствующую строку в тестовую выборку

df\_add = pd.DataFrame(df\_train.iloc[nr\_add:nr\_add+1])

df\_test = df\_test.append(df\_add)

df\_test.drop\_duplicates(inplace = True)

df\_test = df\_test.reset\_index(inplace=False, drop = True)

# очистка данных

def cleaning(df):

df = df[df['Age'] >= 14] # фильтрация по возрасту, расстоянию и задержке

df = df[df['Flight Distance'] >= 250]

df = df[df['Departure Delay in Minutes'] < (df['Departure Delay in Minutes'].mean() + 5\*(df['Departure Delay in Minutes'].std()))]

df['satisfaction']= pd.get\_dummies(df['satisfaction'])[list(df['satisfaction'].unique())[0]] # переводим satisfaction в 0/1

target = df['satisfaction'] # назначаем целевую переменную

df = df.drop(df.iloc[:,[0, 1]], axis = 1) # удаляем первые 2 столбца

categorical\_indexes = [0, 1, 3, 4] + list(range(6, 20)) # определяем, какие переменные будут категориальными

df.iloc[:,categorical\_indexes] = df.iloc[:,categorical\_indexes].astype('category') # переводим их в категориальные

df['Arrival Delay in Minutes'].fillna(df['Departure Delay in Minutes'], inplace = True) # ищем пропуски в задержках, заполняем их и оставляем один столбец

df['Departure Delay in Minutes'] = (df['Departure Delay in Minutes'] + df['Arrival Delay in Minutes'])/2

df = df.drop(df.iloc[:,[-2]], axis = 1)

df.rename(columns = {'Departure Delay in Minutes':'Delay in Minutes'}, inplace = True)

numerical\_columns = [c for c in df.columns if df[c].dtype.name != 'category'] # разбиваем столбцы на количественные и категориальные

numerical\_columns.remove('satisfaction')

categorical\_columns = [c for c in df.columns if df[c].dtype.name == 'category']

df\_describe = df.describe(include = ['category'])

binary\_columns = [c for c in categorical\_columns if df\_describe[c]['unique'] == 2] # разбиваем категориальные столбцы на бинарные и небинарные

nonbinary\_columns = [c for c in categorical\_columns if df\_describe[c]['unique'] > 2]

df\_binary = df[binary\_columns].reset\_index(inplace=False, drop = True)

for col in binary\_columns:

df\_binary[col]= pd.get\_dummies(df\_binary[col])[list(df\_binary[col].unique())[0]] # обрабатываем бинарные

df\_nonbinary = pd.get\_dummies(df[nonbinary\_columns]).reset\_index(inplace=False, drop = True)

df\_numerical = df[numerical\_columns]

scaler = MinMaxScaler() # нормализуем данные в количественных столбцах

df\_numerical = scaler.fit\_transform(df\_numerical)

df\_numerical = pd.DataFrame(df\_numerical, columns = numerical\_columns)

df = pd.concat((df\_numerical, df\_nonbinary, df\_binary), axis = 1) # итоговый датасет

return df, target

# получение обучающей и тестовой выборок

X\_train, y\_train = cleaning(df\_train)

X\_test, y\_test = cleaning(df\_test)

# расчетные функции

def neighbor (x\_train, x\_test, y\_train, n):

classifier = KNeighborsClassifier(n\_neighbors = n)

classifier.fit(x\_train, y\_train.values.ravel())

y\_pred = classifier.predict(x\_test)

return y\_pred

def Bayes (x\_train, x\_test, y\_train):

classif = BernoulliNB()

#Обучение модели Наивного Байеса

classif.fit(x\_train, y\_train.values.ravel())

#Проверка модели на тестовой выборке

y\_pred = classif.predict(x\_test)

classif = GaussianNB()

classif.fit(x\_train, y\_train.values.ravel())

y\_pred3 = classif.predict(x\_test)

return y\_pred, y\_pred3

def Tree (x\_train, x\_test, y\_train):

clf = DecisionTreeClassifier(criterion = 'entropy', splitter = 'best', min\_samples\_split = 2,

min\_samples\_leaf = 1)

clf.fit(x\_train, y\_train)

y\_pred = clf.predict(x\_test)

return y\_pred

def neural (x\_train, x\_test, y\_train, n):

clf = MLPClassifier(hidden\_layer\_sizes= (n), activation = 'logistic', solver = 'sgd',

learning\_rate\_init = 0.01, max\_iter = 2000, tol = 0.000001)

clf.fit(x\_train, y\_train.values.ravel())

y\_pred = clf.predict(X\_test)

return y\_pred

def LogRegr (x\_train, x\_test, y\_train):

classifier = LogisticRegression(solver = 'sag', penalty = 'l2')

classifier.fit(x\_train, y\_train.values.ravel())

y\_pred = classifier.predict(x\_test)

return y\_pred

# метрики проверки

def verif (y\_test, y\_pred):

print('Accuracy = ', accuracy\_score(y\_test, y\_pred))

print('Recall = ', recall\_score(y\_test, y\_pred))

print('Precision = ', precision\_score(y\_test, y\_pred))

print('f1 = ', f1\_score(y\_test, y\_pred))

# вызов расчетных функций

dt\_start = datetime.datetime.now()

print('Нейронная сеть:')

verif (y\_test, neural(X\_train, X\_test, y\_train, 5))

dt\_end = datetime.datetime.now()

print('Продолжительность вычислений, с: ', (dt\_end - dt\_start).total\_seconds())

dt\_start = datetime.datetime.now()

print('Ближайший сосед:')

verif (y\_test, neighbor(X\_train, X\_test, y\_train, 10))

dt\_end = datetime.datetime.now()

print('Продолжительность вычислений, с: ', (dt\_end - dt\_start).total\_seconds())

dt\_start = datetime.datetime.now()

print('Наивный Байес:')

y1, y2 = Bayes(X\_train, X\_test, y\_train)

print('Бернулли:')

verif(y\_test, y1)

print('Гаусс:')

verif(y\_test, y2)

dt\_end = datetime.datetime.now()

print('Продолжительность вычислений, с: ', (dt\_end - dt\_start).total\_seconds())

dt\_start = datetime.datetime.now()

print('Дерево:')

verif(y\_test, Tree (X\_train, X\_test, y\_train))

dt\_end = datetime.datetime.now()

print('Продолжительность вычислений, с: ', (dt\_end - dt\_start).total\_seconds())

dt\_start = datetime.datetime.now()

print('Лог. регрессия:')

verif(y\_test, LogRegr (X\_train, X\_test, y\_train))

dt\_end = datetime.datetime.now()

print('Продолжительность вычислений, с: ', (dt\_end - dt\_start).total\_seconds())