НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Класифікація рівня задоволеності пасажирів авіакомпанією»

Студента 2 курсу групи ІП-11
Спеціальності: 121
«Інженерія програмного забезпечення»
Сідака Кирила Ігоровича

«ПРИЙНЯВ» з оцінкою

_____

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____
Підпис Дата

Київ - 2023 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-11 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Сідака Кирила Ігоровича

1.Тема роботи Класифікація рівня задоволеності пасажирів авіакомпанією.
Методи K-Nearest Neighbors, Logistic Regression та Random Forest Classifier.

2.Строк здачі студентом закінченої роботи 29.05.2022

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
Вступ, постановка задачі, аналіз предметної області, робота з даними, інтелектуальний аналіз, висновки, перелік посилань, додаток А.

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6.Дата видачі завдання 16.04.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.04.2023	
2.	Визначення зовнішніх джерел даних	16.04.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	16.04.2023	
4.	Обробка та аналіз даних	17.05.2023	
5.	Обґрунтування методів інтелектуального аналізу даних	17.05.2023	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	17.05.2023	
7.	Підготовка пояснювальної записки	17.05.2023 – 29.05.2023	
8.	Здача курсової роботи на перевірку	20.05.2023	
9.	Захист курсової роботи	09.06.2023	

Студент _____ Сідак К. І.

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Ліхоузова Т.А

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім'я, по батькові)

"9" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 41 сторінка, 34 рисунки, 12 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: пошук, аналіз та обробка даних, реалізація ПЗ, що використовує отримані дані для подальшого аналізу та прогнозування результату.

Дана курсова робота включає в себе: постановку задачі, аналіз предметної області, роботу з даними, аналіз обраних методів для прогнозування та їх порівняння.

ДАТАСЕТ, МОДЕЛЬ КЛАСИФІКАЦІЇ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, K-NEAREST NEIGHBORS, LOGISTIC REGRESSION, RANDOM FOREST CLASSIFIER.

ЗМІСТ

ВСТУП

У сучасному світі туризм та авіаперельоти, зокрема, стали невід'ємною частиною життя багатьох людей та є основою економіки у деяких країнах. На даний момент існує безліч авіакомпаній по всьому світу, які пропонують різноманітні послуги авіаперельотів для різних категорій клієнтів. Їхня кількість постійно збільшується, тому важливо на основі даних про пасажира, рейс та оцінок пасажирка швидко та точно передбачити, чи буде він задоволений польотом саме через цю конкретну авіакомпанію. Таким чином, прогнозуючи на основі побідних даних про велику кількість пасажирів, чи вони задоволені авіакомпанією, можна швидко та досить точно отримати уявлення про якість цієї авіакомпанії.

У рамках даної курсової роботи було проаналізовано дані про пасажирів певної авіакомпанії (дані анонімні) та їхній рейс і шляхом інтелектуального аналізу обраних даних з використанням трьох різних методів класифікації спрогнозовано рівень задоволеності пасажирів авіакомпанією (задоволений або нейтрально/незадоволений).

Дана курсова робота буде розроблена з використанням технологій та бібліотек Python 3[1], Pandas[2], Seaborn[3], Matplotlib[4], Sklearn[5], NumPy[6], SciPy[7], os[8].

1 ПОСТАНОВКА ЗАДАЧІ

Під час виконання курсової роботи необхідно виконати наступні завдання:

Аналіз предметної області, завантаження датасету, дослідження його структури, форматування даних, виправлення помилок.

Створення застосунку, що розділяє набір даних на навчальні та тестові набори, проводить їх інтелектуальний аналіз для отримання передбачення за допомогою різних моделей класифікації, а саме наступними методами: K-Nearest Neighbors, Logistic Regression та Random Forest Classifier. Для кожного методу проаналізувати результати та, порівнявши їх, обрати отимальний метод для передбачення рівня задоволеності пасажира авіакомпанією.

Вхідними даними будуть стать пасажира, його тип (постійний клієнт чи ні), вік, тип подорожі (особиста подорож чи бізнес-подорож), клас (бізнес, еко, еко плюс), відстань перельоту, оцінки від 0 д 5 наступних аспектів: Wi-Fi на борту, зручність часу вильоту/прибуття, зручність онлайн бронювання, розташування виходу на посадку, їжа та напої, посадка на рейс онлайн, зручність крісел, розваги під час польоту, обслуговування на борту, обслуговування в багажному відділенні, обробка багажу, реєстрація на рейс, обслуговування під час польоту, бортове обслуговування, чистота – а також затримка вильоту в хвилинах, затримка прильоту в хвилинах та рівень задоволеності (задоволений або нейтральний/незадоволений).

Використати мову програмування Python 3 для реалізації застосунку.

Курсовий проект здати до дедлайну (початок сесії) та виконати у єдиному стилі написання коду (coding style).

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Авіакомпанії постійно звертають увагу на задоволеність своїх пасажирів, оскільки це є важливим фактором для їх успіху і конкурентоспроможності на ринку. У зв’язку з тим, що зараз їх кількість постійно збільшується, то цей фактор є особливо важливим. Таким чином, важливо виявити закономірності між оцінками пасажирів різних послуг авіакомпанії, їхнім віком й іншими аспектами рейсу та задоволеністю пасажирів.

Статистика задоволеності пасажирів може варіюватися в залежності від авіакомпанії та національності пасажирів, тому це завдання є досить складним та вимагає обробки та аналізу великої кількості даних, проте певну загальну тенденцію можна буде виявити й на даних однієї авіакомпанії, адже основні аспекти, які впливають рівень задоволеності пасажира в цих даних наявні.

Ця система дозволить авіакомпаніям отримати цінну інформацію про рівень задоволеності своїх пасажирів та виявити можливі напрямки для покращення якості обслуговування та задоволення потреб пасажирів.

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

– завантаження набору даних та дослідження його структури;
– інтелектуальний аналіз даних;
– використання декількох моделей прогнозування даних;
– прогнозування рівня задоволеності пасажиру авіакомпанією (задоволений або нейтрально/незадоволений);
– графічне відображення отриманих результатів та їх аналіз;
– порівняння використаних методів на основі різних метрик.

3 РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Для виконання курсової роботи був обраний датасет «Airline Passenger Satisfaction» на сайті https://www.kaggle.com//, що включають в себе. Даний набір даних скаладається з інформації по 129880 пасажирам. Даний датасет містить 2 csv-файли (які будуть об’єднані в один датафрейм), що складаються з 23 стовпців. Дані стовпці несуть в собі наступну інформацію:

- Gender – стать пасажирів (жінка, чоловік)
- Customer Type – тип клієнта (лояльний клієнт, нелояльний клієнт)
- Age – фактичний вік пасажирів
- Type of Travel – мета польоту пасажирів (особиста подорож, бізнес-подорож)
- Class – клас польоту пасажирів (бізнес, економ, економ плюс)
- Flight distance – відстань польоту цієї подорожі
- Inflight wifi service – рівень задоволеності сервісом Wi-Fi на борту (0: немає; 1-5)
- Departure/Arrival time convenient – рівень задоволеності зручністю часу відправлення/прибуття (0-5)
- Ease of Online booking – рівень задоволеності легкістю онлайн-бронювання (0-5)
- Gate location – рівень задоволеності розташуванням гейту (0-5)
- Food and drink – рівень задоволеності їжею та напоями (0-5)
- Online boarding – рівень задоволеності онлайн посадкою (0-5)
- Seat comfort – рівень задоволеності комфортом сидінь (0-5)
- Inflight entertainment – рівень задоволеності розвагами на борту (0-5)
- On-board service – рівень задоволеності сервісом на борту (0-5)
- Leg room service – рівень задоволеності сервісом з ніжним місцем (0-5)
- Baggage handling – рівень задоволеності обробкою багажу (1-5)
- Check-in service – рівень задоволеності сервісом реєстрації (0-5)
- Inflight service – рівень задоволеності сервісом на борту (0-5)

- Cleanliness – рівень задоволеності чистотою (0-5)
- Departure Delay in Minutes – затримка відправлення в хвилинах
- Arrival Delay in Minutes – затримка прибуття в хвилинах
- Satisfaction – рівень задоволеності авіалінією (задоволений, нейтральний/незадоволений)

3.2 Завантаження даних та заповнення пропущених значень

Для роботи з даними на мові Python буде використана бібліотека «pandas».

Для початку ми імпортуємо усі необхідні бібліотеки (зазначені вище) для подальшої роботи (рис. 3.1) та зчитуємо дані з двох csv-файлів у датафрейми й об’єднуємо їх. в один датафрейм (рис. 3.2).

```
In 1
import pandas as pd
import numpy as np
import os
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import chi2_contingency
from sklearn import ensemble, metrics
from sklearn.linear_model import LogisticRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
%matplotlib inline
Executed in 277ms, 28 May at 00:27:05
```

Рисунок 3.1 – Імпорт усіх необхідних бібліотек

```
In 2
directory = './data'
dataframes = []
for filename in os.listdir(directory):
    file_path = os.path.join(directory, filename)
    df = pd.read_csv(file_path, index_col=0)
    df = df.set_index('id')
    dataframes.append(df)

final_df = pd.concat(dataframes)
final_df = final_df.sort_index()
final_df.duplicated().sum()
Executed in 202ms, 20 May at 18:18:13

Out 2
0
```

Рисунок 3.2 – Завантаження даних та об’єднання їх в один датафрейм

Далі переглянемо перші 5 та останні 5 рядків отриманого датафрейму (рис. 3.3).

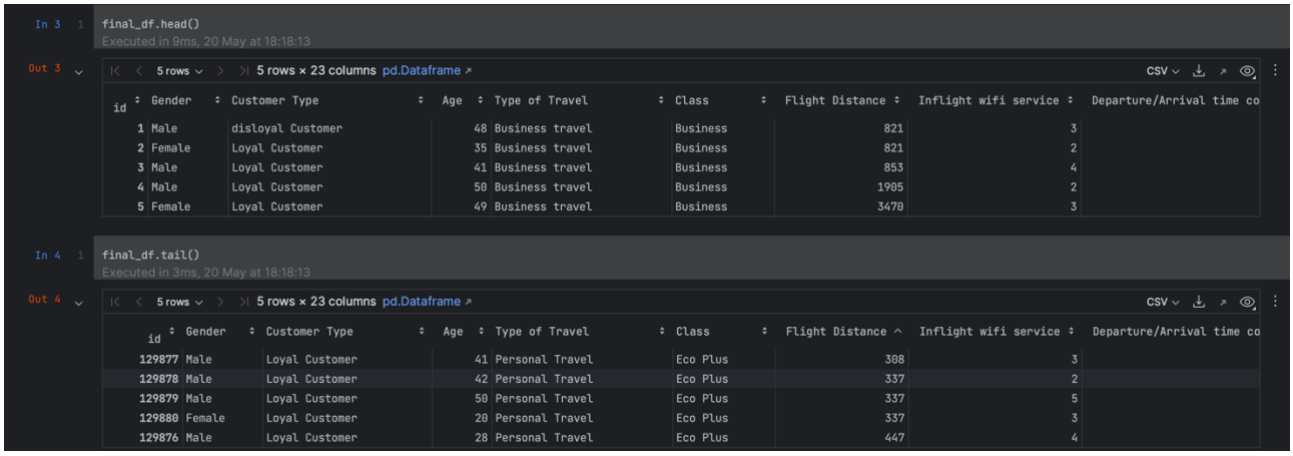


Рисунок 3.3 – Перші 5 та останні 5 рядків датафрейму

Наступним кроком виведемо стовпці, що містять пропущені значення, кількість цих значень та їхню частку у процентах (рис. 3.4).

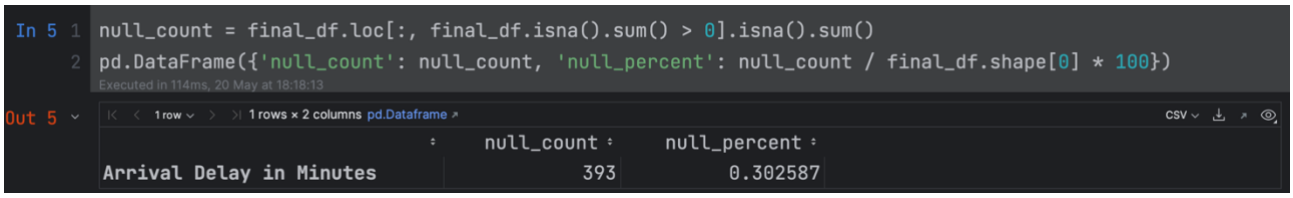


Рисунок 3.4 – Стовпці з пропущеними значеннями

Як бачимо, лише один стовпець містить пропущені значення. Оскільки частка цих значень є досить малою (трохи більша за 0.3%), а стовпець містить кількісні неперервні дані, то можна заповнити їх середнім значенням (рис. 3.5).

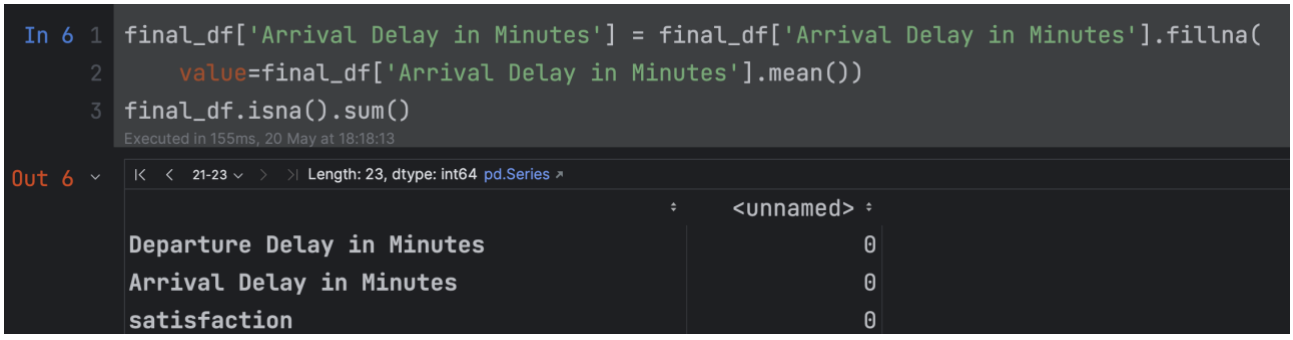


Рисунок 3.5 – Заповнення пропущених значень стовпця середнім значенням та перевірка їх наявності

3.3 Форматування датафрейму та дослідження його структури

Тепер відформатуємо назви стовпців датафрейму, привівши їх до нижнього регістру та замінивши пробіли підкреслюваннями (рис. 3.6).

```
In 7  1  final_df.columns = final_df.columns.str.lower().str.replace(' ', '_')
      2  final_df.columns.values
Executed in 103ms, 20 May at 18:18:13

Out 7   array(['gender', 'customer_type', 'age', 'type_of_travel', 'class',
               'flight_distance', 'inflight_wifi_service',
               'departure/arrival_time_convenient', 'ease_of_online_booking',
               'gate_location', 'food_and_drink', 'online_boarding',
               'seat_comfort', 'inflight_entertainment', 'on-board_service',
               'leg_room_service', 'baggage_handling', 'checkin_service',
               'inflight_service', 'cleanliness', 'departure_delay_in_minutes',
               'arrival_delay_in_minutes', 'satisfaction'], dtype=object)
```

Рисунок 3.6 – Форматування назв стовпців датафрейму

Наступним кроком виведемо інформацію про типи даних стовпців та кількість непорожніх значень стовпців (рис. 3.7).

```
In 8  1  final_df.info()
Executed in 159ms, 20 May at 18:18:13

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 23 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   gender                            129880 non-null  object
 1   customer_type                     129880 non-null  object
 2   age                               129880 non-null  int64
 3   type_of_travel                    129880 non-null  object
 4   class                             129880 non-null  object
 5   flight_distance                   129880 non-null  int64
 6   inflight_wifi_service             129880 non-null  int64
 7   departure/arrival_time_convenient 129880 non-null  int64
 8   ease_of_online_booking            129880 non-null  int64
 9   gate_location                     129880 non-null  int64
 10  food_and_drink                    129880 non-null  int64
 11  online_boarding                   129880 non-null  int64
 12  seat_comfort                      129880 non-null  int64
 13  inflight_entertainment            129880 non-null  int64
 14  on-board_service                  129880 non-null  int64
 15  leg_room_service                  129880 non-null  int64
 16  baggage_handling                  129880 non-null  int64
 17  checkin_service                   129880 non-null  int64
 18  inflight_service                  129880 non-null  int64
 19  cleanliness                       129880 non-null  int64
 20  departure_delay_in_minutes        129880 non-null  int64
 21  arrival_delay_in_minutes          129880 non-null  float64
 22  satisfaction                      129880 non-null  object
dtypes: float64(1), int64(17), object(5)
```

Рисунок 3.7 – Інформація про типи даних та кількість непорожніх значень стовпців

Як бачимо, стовпці з категоріальними змінними (у номінальній або порядковій шкалі) мають або цілочисельний, або ж рядковий тип.

Тепер перетворимо типи даних цих стовпців до категоріального (спеціальний тип даних у бібліотеці pandas) (рис. 3.8).

```
In 9
categorical_columns = final_df.columns[~final_df.columns.isin(
    ['age', 'flight_distance', 'departure_delay_in_minutes', 'arrival_delay_in_minutes', 'satisfaction'])]
final_df[categorical_columns] = final_df[categorical_columns].astype('category')
final_df.info()
Executed in 121ms, 20 May at 18:18:13
```

```
 ---  ------                                --------------  -----
  0   gender                                129880 non-null  category
  1   customer_type                         129880 non-null  category
  2   age                                   129880 non-null  int64
  3   type_of_travel                        129880 non-null  category
  4   class                                 129880 non-null  category
  5   flight_distance                       129880 non-null  int64
  6   inflight_wifi_service                 129880 non-null  category
  7   departure/arrival_time_convenient     129880 non-null  category
  8   ease_of_online_booking                129880 non-null  category
  9   gate_location                         129880 non-null  category
  10  food_and_drink                        129880 non-null  category
  11  online_boarding                       129880 non-null  category
  12  seat_comfort                          129880 non-null  category
  13  inflight_entertainment                129880 non-null  category
  14  on-board_service                      129880 non-null  category
  15  leg_room_service                      129880 non-null  category
  16  baggage_handling                      129880 non-null  category
  17  checkin_service                       129880 non-null  category
  18  inflight_service                      129880 non-null  category
  19  cleanliness                           129880 non-null  category
  20  departure_delay_in_minutes            129880 non-null  int64
  21  arrival_delay_in_minutes              129880 non-null  float64
  22  satisfaction                          129880 non-null  object
dtypes: category(18), float64(1), int64(3), object(1)
```

Рисунок 3.8 – Інформація про типи даних та кількість непорожніх значень стовпців

Потім виведемо основні описові статистики для стовпців, що містять кількісні неперервні змінні (рис. 3.9).

```
In 10
final_df.describe()
Executed in 51ms, 20 May at 18:18:13
```

Out 10 — 8 rows × 4 columns pd.Dataframe

	age	flight_distance	departure_delay_in_minutes	arrival_delay_in_minutes
count	129880.000000	129880.000000	129880.000000	129880.000000
mean	39.427957	1190.316392	14.713713	15.091129
std	15.119360	997.452477	38.071126	38.407410
min	7.000000	31.000000	0.000000	0.000000
25%	27.000000	414.000000	0.000000	0.000000
50%	40.000000	844.000000	0.000000	0.000000
75%	51.000000	1744.000000	12.000000	13.000000

Рисунок 3.9 – Описові статистики кількісних неперервних змінних

Після цього виведемо аналогічний опис для стовпців, що містить категоріальні змінні (рис. 3.10).

```
In 11  1  final_df.describe(include=['category'])
          Executed in 36ms, 20 May at 18:18:13
```

Out 11 — 4 rows × 18 columns pd.Dataframe

	1	2	3	4
	count	unique	top	freq
gender	129880	2	Female	65899
customer_type	129880	2	Loyal Customer	106100
type_of_travel	129880	2	Business trav…	89693
class	129880	3	Business	62160
inflight_wifi_service	129880	6	2	32320
departure/arrival_tim...	129880	6	4	31880
ease_of_online_booking	129880	6	3	30393
gate_location	129880	6	3	35717
food_and_drink	129880	6	4	30563
online_boarding	129880	6	4	38468
seat_comfort	129880	6	4	39756
inflight_entertainment	129880	6	4	36791
on-board_service	129880	6	4	38703
leg_room_service	129880	6	4	35886
baggage_handling	129880	5	4	46761
checkin_service	129880	6	4	36333
inflight_service	129880	6	4	47323
cleanliness	129880	6	4	33969

Рисунок 3.10 – Опис категоріальних змінних

3.4 Візуалізація даних та відбір ознак

Спочатку візуалізуємо проценте відношення значень цільової змінної за допомогою кругової діаграми (рис. 3.11).

```
In 12  1  plt.pie(final_df.satisfaction.value_counts(), labels=["Neutral or dissatisfied", "Satisfied"],
       2          colors=sns.color_palette("YlOrBr"), autopct='%1.1f%%');
          Executed in 76ms, 20 May at 18:18:13
```

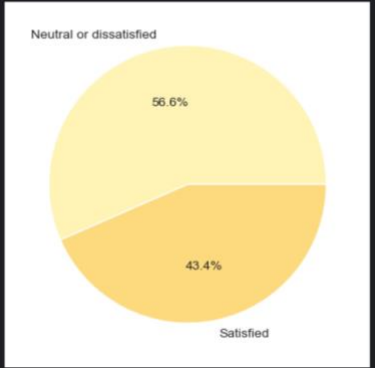


Рисунок 3.11 – Процентне відношення значень цільової змінної

Можна побачити, що більшість пасажирів є нейтральними/незадоволеними авіакомпанією, проте загалом відсоток задоволених пасажирів не є сильно меншим.

Наступним кроком візуалізуємо кореляційну матрицю для кількісних неперервних змінних, щоб дослідити ці ознаки на предмет колінеарності ознак (рис. 3.12).

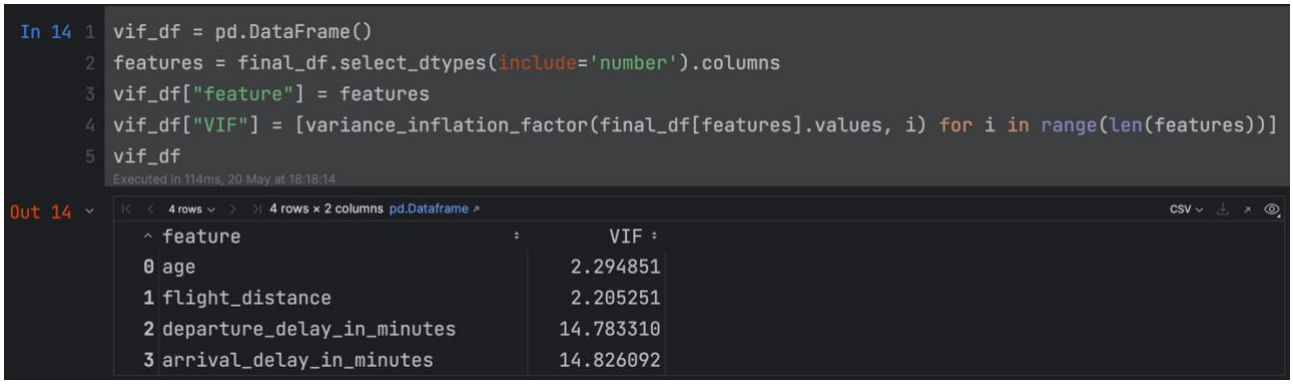
Рисунок 3.12 – Кореляційна матриця кількісних неперервних змінних

З цієї матриці відразу видно, що затримка вильоту та затримка прильоту сильно корелюють між собою, адже коефіцієнт кореляції (Пірсона) близький до 1. Отже, варто буде один з цих стовпців видалити.

На основі кореляційної матриці можна побачити кореляцію лише між двома ознаками, тому для дослідження на мультиколінераність ознак застосуємо фактор дисперсії інфляції (VIF):

$$VIF_i = \frac{1}{1-R_i^2},$$

де R_i^2 – це коефіцієнт детермінації лінійної регресійної для прогнозування і-ї змінної через інші (рис. 3.13).



```
In 14
vif_df = pd.DataFrame()
features = final_df.select_dtypes(include='number').columns
vif_df["feature"] = features
vif_df["VIF"] = [variance_inflation_factor(final_df[features].values, i) for i in range(len(features))]
vif_df
```

Executed in 114ms, 20 May at 18:18:14

Out 14

4 rows × 2 columns pd.Dataframe

	feature	VIF
0	age	2.294851
1	flight_distance	2.205251
2	departure_delay_in_minutes	14.783310
3	arrival_delay_in_minutes	14.826092

Рисунок 3.13 – Значення VIF для кількісних неперервних змінних

Маємо, що ситуація аналогічна кореляційній матриці. Ті ж 2 ознаки мають високі показники VIF (більше 5), що свідчить про сильну кореляцію. Оскільки затримка прильоту має трохи більше значення VIF, то цей стовпець буде видалено.

Далі за допомогою кругових діаграм візуалізуємо процентне відношення значень категоріальних змінних (рис. 3.14).

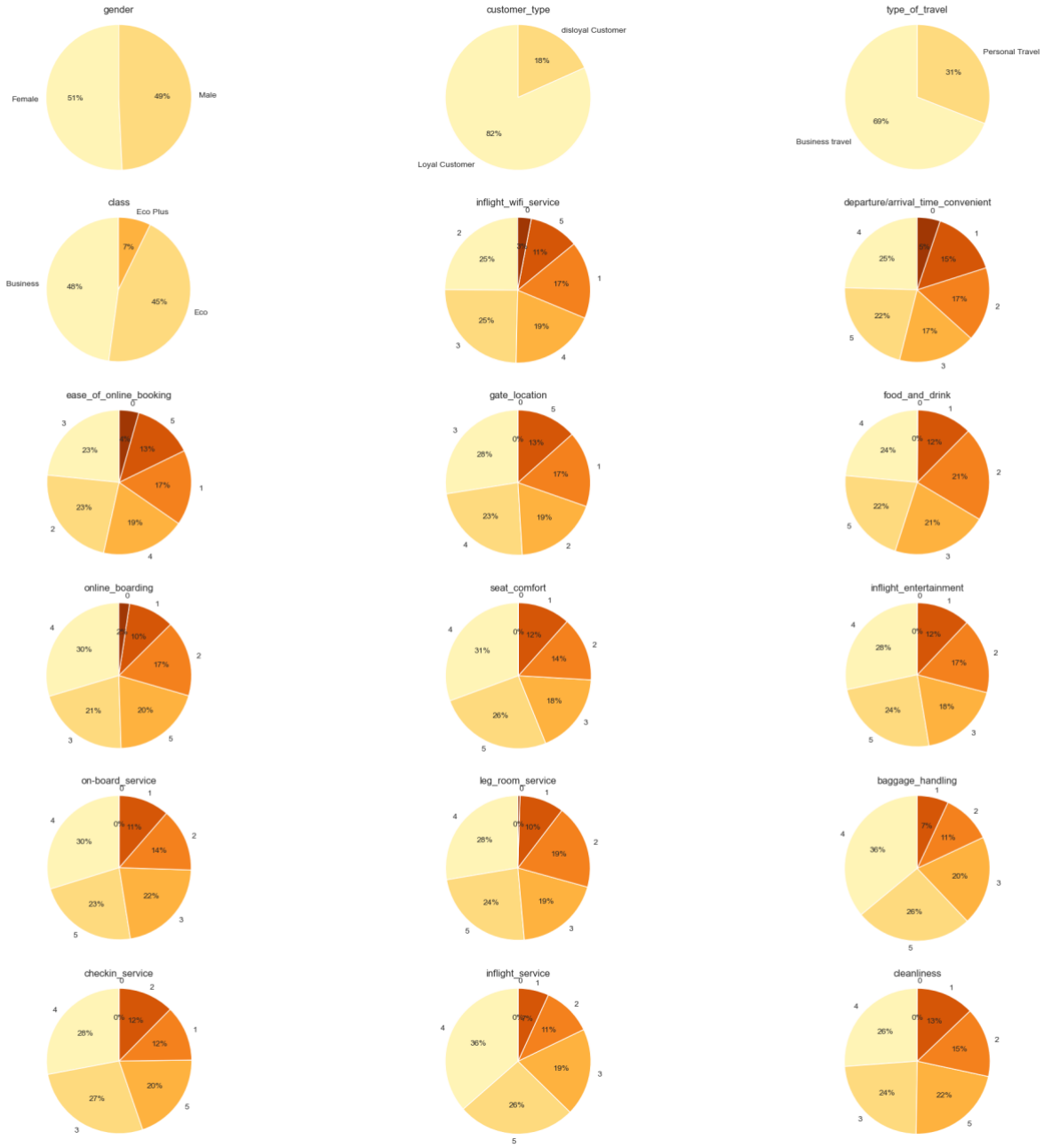


Рисунок 3.14 – Процентне відношення значень категоріальних змінних

Отже, можна побачити, що даласет містить приблизно однакову кількість жінок та чоловіків, більшість пасажирів – це вірні клієнти, трохи більше 2/3 подорожей становлять бізнес-подорожі, пасажири в основному літають економ- або бізнес-класом. Щодо оцінок пасажирів, то оцінок 0 досить незначна кількість, а від 1 до 5 розподілені досить рівномірно.

Наступним кроком визначимо, чи корелюють між собою порядкові змінні та порядкові змінні з неперервними за допомогою кореляційної матриці, використавши коефіцієнт рангової кореляції Спірмана (рис. 3.15).

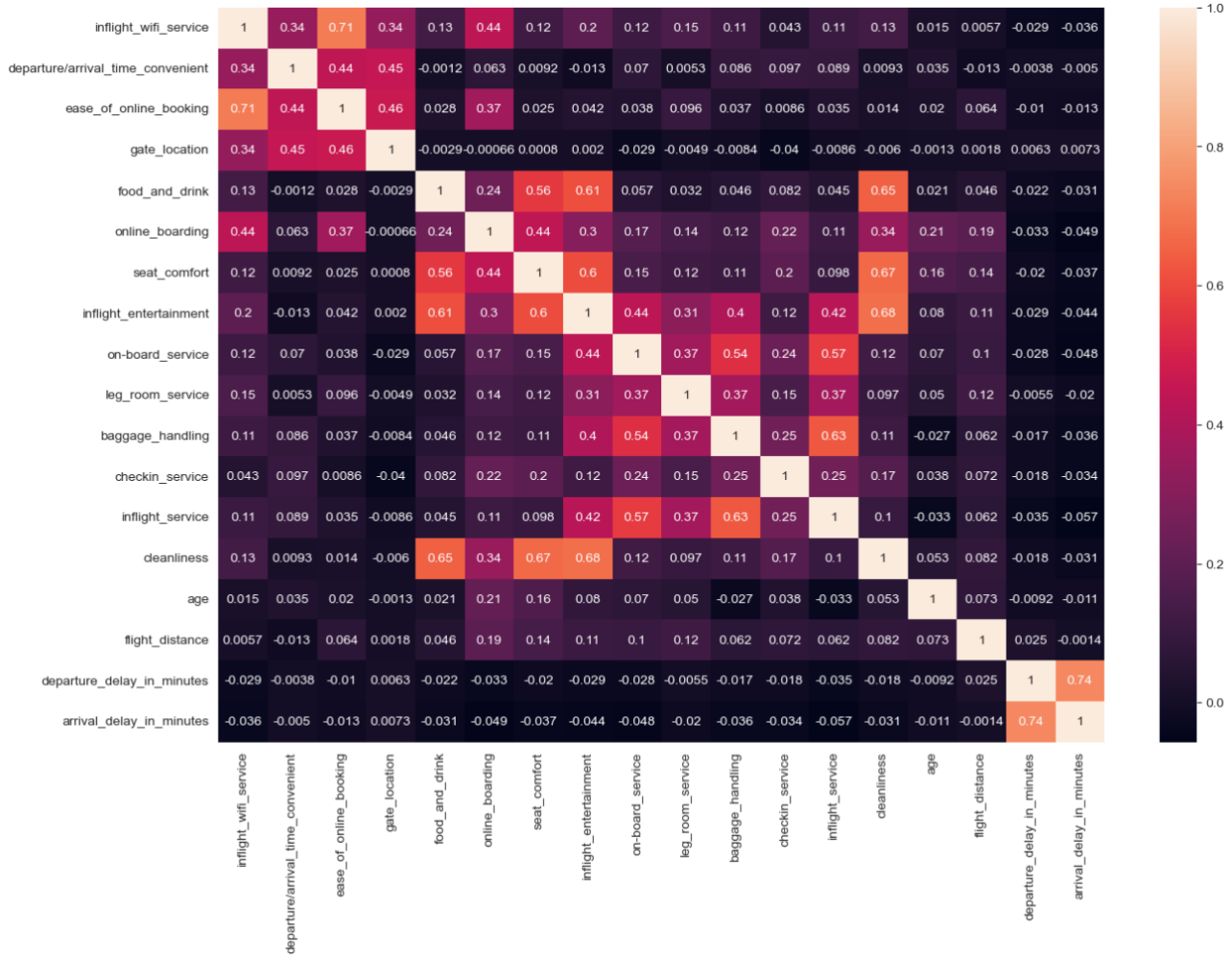


Рисунок 3.15 – Кореляційна матриця кількісних неперервних змінних та порядкових змінних

Бачимо, що стовпець inflight_entertainment має коефіцієнти кореляції більші за 0.6 з трьома стовпцями та більші за 0.4 з трьома іншими стовпцями, тому його буде доцільно видалити.

Тепер перетворимо значення цільової змінної на числові (0 – нейтральний/незадоволений, 1 – задоволений) та виведемо перші 5 значень (рис. 3.16).

```
In 17  satisfaction_mapping = {v: k for k, v in enumerate(final_df['satisfaction'].unique())}
       satisfaction_mapping
       Executed in 4ms, 26 May at 23:19:39

Out 17  {'neutral or dissatisfied': 0, 'satisfied': 1}

In 18  final_df['satisfaction'] = final_df['satisfaction'].map(satisfaction_mapping)
       final_df['satisfaction'].head()
       Executed in 7ms, 26 May at 23:19:39

Out 18  5 rows  Length: 5, dtype: int64  pd.Series  CSV
```

id	satisfaction
1	0
2	1
3	1
4	1
5	1

Рисунок 3.16 – Перетворення значень цільової змінної на числові

Після перетворення цільової змінної за допомогою стовпчастої діаграми візуалізуємо залежність цільової змінної (задоволеності) від якісних, зокрема: статі (рис. 3.17), типу клієнта (рис. 3.18), типу подорожі (рис. 3.19), класу (рис. 3.20).

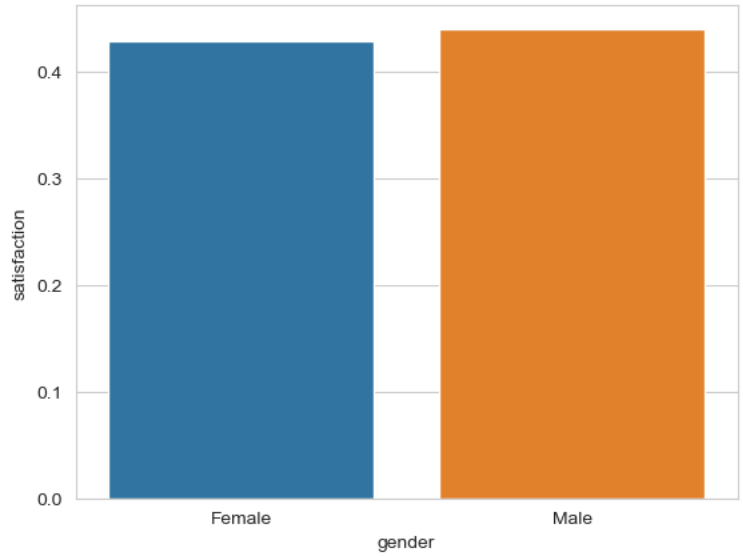


Рисунок 3.17 – Стовпчаста діаграма для статі пасажирів

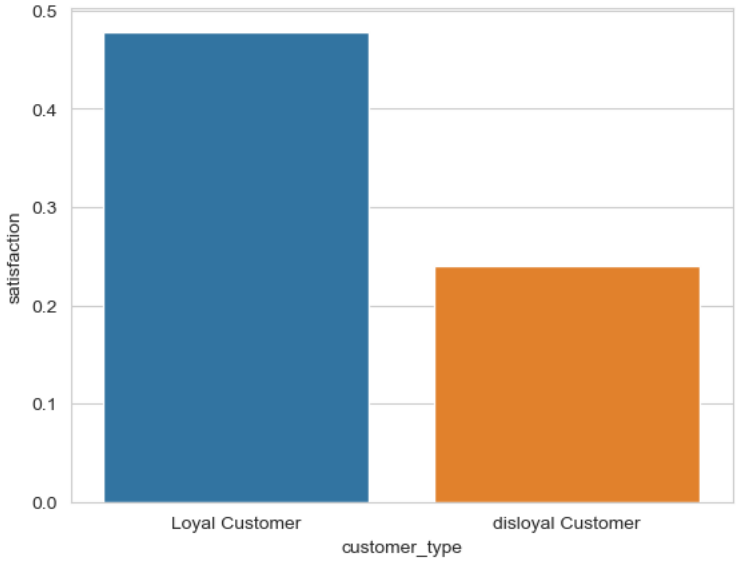


Рисунок 3.18 – Стовпчаста діаграма для типу пасажирів

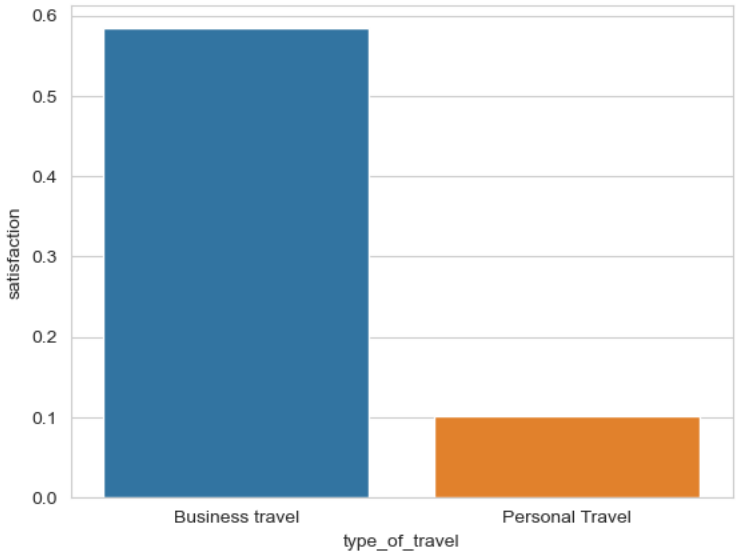


Рисунок 3.19 – Стовпчаста діаграма для типу подорожі

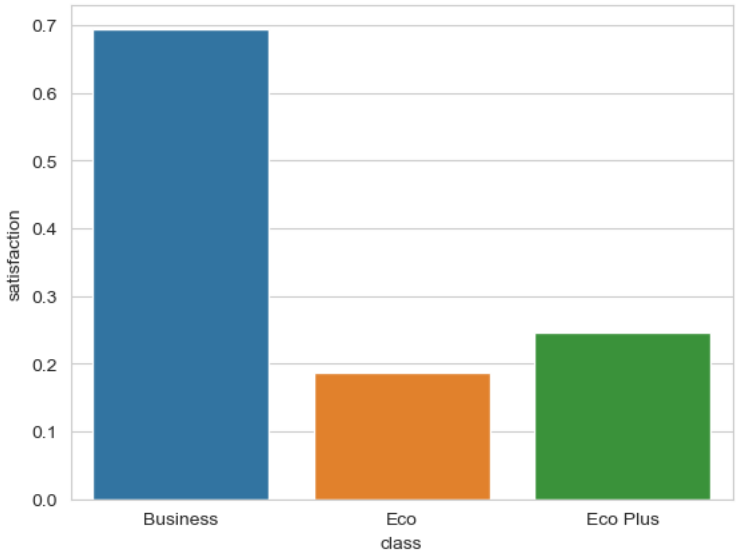


Рисунок 3.20 – Стовпчаста діаграма для класу

Отже, можна побачити, що в середньому рівень задоволеності жінок та чоловіків майже однаковий, вірні клієнти в середньому частіше задоволені авіакомпанією, що є очевидним фактом, пасажири, які виконують бізнес-подорож у середньому значно частіше задоволені, ніж ті, що виконують особисту подорож. Крім того, видно, що пасажири вищого класу частіше задоволені авіакомпанією, особливо це простежується для бізнес-класу.

Потім використаємо критерій Хі-квадрат для визначення наявності статистичного зв'язку між цільовою змінною (задоволеність) та якісними змінними (рис. 3.21).

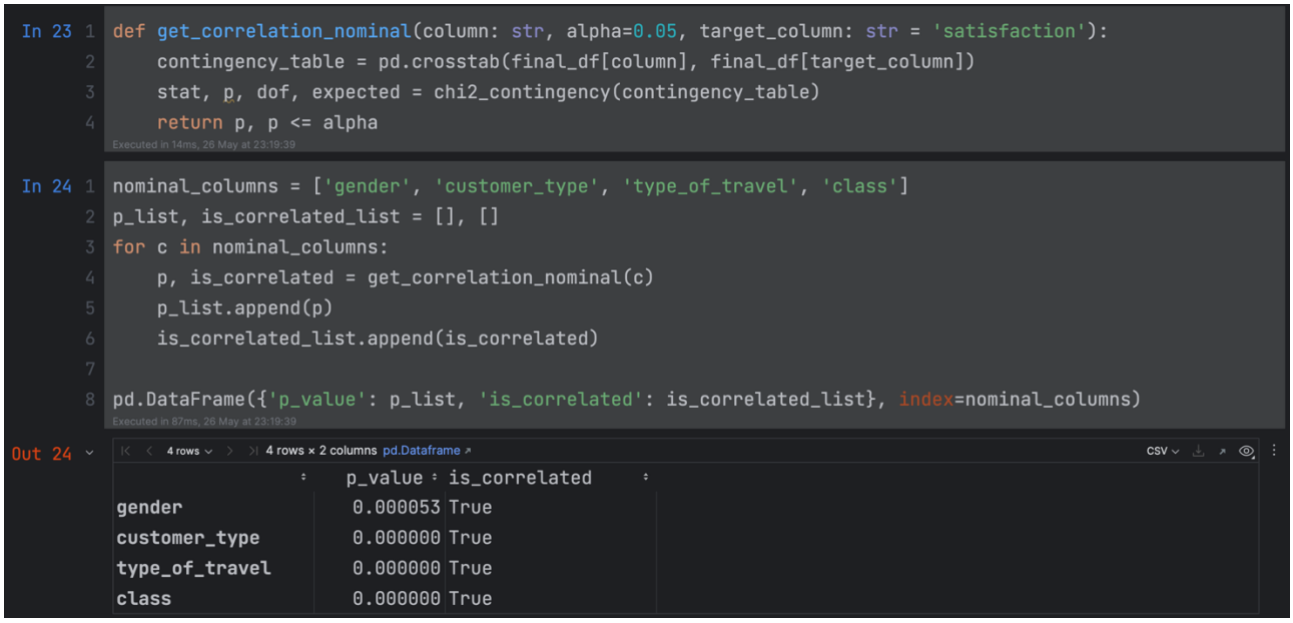


Рисунок 3.21 – Критерій Хі-квадрат для визначення статистичного зв'язку між якісними змінними та цільовою

Отримані результати вказують на те, що наявний статистичний зв'язок між кожною з цих незалежних якісних змінних та цільовою (залежною) змінною.

3.5 Підготовка датасету до інтелектуального аналізу

Спочатку згрупуємо стовпець віку у квартилі (4 квантилі), перетворивши таким чином шкалу цієї ознаки в інтервальну, та побудуємо стовпчасту діаграму для згрупованої ознаки відносно рівня задоволеності (рис. 3.22).

```
In 25  1  final_df['age_group'] = pd.qcut(final_df['age'], 4)
       2  sns.barplot(x='age_group', y='satisfaction', data=final_df, errorbar=None);
          Executed in 81ms, 26 May at 23:19:39
```

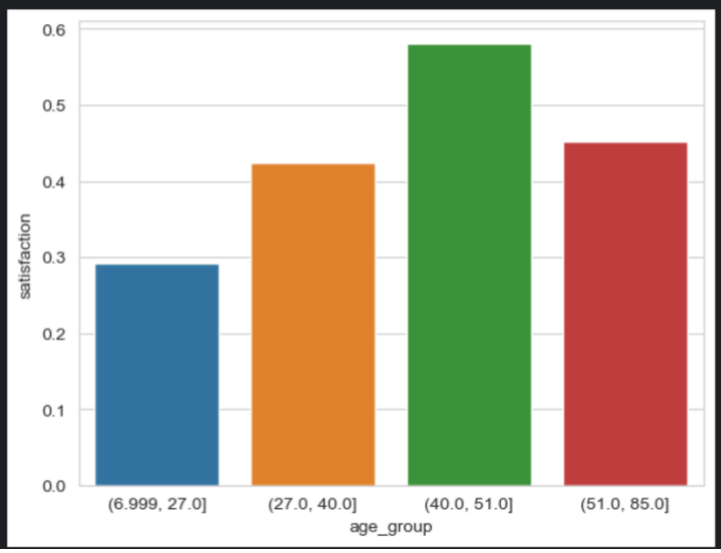


Рисунок 3.22 – Стовпчаста діаграма згрупованої ознаки віку у квартилі відносно рівня задоволеності

З отриманої діаграми видно, що найбільше задоволених пасажирів віком від 40 до 51 років включно, а найменше до 27 років.

Наступним кроком перетворимо інтервали на числа від 1 до 4 у порядку зростання віку (рис. 3.23).

```
In 26  1  final_df['age_group'] = final_df['age_group'].cat.codes + 1
       2  final_df['age_group'].head()
          Executed in 23ms, 26 May at 23:19:39

Out 26    id
          1    3
          2    2
          3    3
          4    3
          5    3
          Name: age_group, dtype: int8
```

Рисунок 3.23 – Перетворення інтервалів віку на числа від 1 до 4

Після цього видалимо стовпці, які були зазначено вище на основі показників кореляції, та стовпець age (вік), бо ми вже створили новий стовпець age_group (вікова група) (рис. 3.24).

```
In 27
final_df = final_df.drop(
    columns=['arrival_delay_in_minutes', 'inflight_entertainment', 'age'])
final_df.head()
Executed in 13ms, 26 May at 23:19:39
```

Out 27 — 5 rows × 21 columns pd.Dataframe

id	gender	customer_type	type_of_travel	class	flight_distance	inflight_wifi_service
1	Male	disloyal Customer	Business travel	Business	821	3
2	Female	Loyal Customer	Business travel	Business	821	2
3	Male	Loyal Customer	Business travel	Business	853	4
4	Male	Loyal Customer	Business travel	Business	1905	2
5	Female	Loyal Customer	Business travel	Business	3470	3

Рисунок 3.24 – Видалення стовпців датафрейму

Тепер створимо фіктивні змінні на основі якісних ознак для кращої їх інтерпретації статистичними моделями, причому для n унікальних значень ознаки буде створено n-1 фіктивних змінних, та приведемо назви стовпців до нижнього регістру, замінивши пробіли на нижнє підкреслювання (рис. 3.25).

```
In 28
final_df = pd.get_dummies(final_df, columns=['gender', 'customer_type', 'type_of_travel', 'class'], drop_first=True)
final_df.columns = final_df.columns.str.lower().str.replace(' ', '_')
final_df.head()
Executed in 17ms, 26 May at 23:19:39
```

Out 28 — 5 rows × 22 columns pd.Dataframe

n	age_group	gender_male	customer_type_disloyal_customer	type_of_travel_personal_travel	class_eco	class_eco_plus
0	3	1	1	0	0	0
1	2	0	0	0	0	0
1	3	1	0	0	0	0
1	3	1	0	0	0	0
1	3	0	0	0	0	0

Рисунок 3.25 – Створення фіктивних змінних на основі якісних ознак

В якості фінального кроку розділимо датасет на навчальні та тестові набори даних у відношенні 75% до 25% з приблизно рівним відношенням значень цільової змінної в обох наборах (рис. 3.26).

```
In 29
X, y = final_df.drop(columns=['satisfaction']), final_df['satisfaction']
Executed in 13ms, 26 May at 23:19:39

In 30
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y)
Executed in 67ms, 26 May at 23:19:39
```

Рисунок 3.26 – Розділення набору даних на навчальні та тестові набори

4 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Мною було обрано три методи для класифікації та подальшої їх оцінки: K-Nearest Neighbors, Logistic Regression та Random Tree Classifier.

Метод K-Nearest Neighbors я обрав через його простоту , інтерпретабельність та відносно непогану швидкість навчання. Крім того, даний метод не потребує особливої підготовки даних перед навчанням, окрім бажано стандартизації, та має лише 2 гіперпараметри: кількість сусідів k та порядок p метрики Мінковського[9] (хоча зазвичай використовується Евклідова відстань p=2, і цей гіперпараметр окремо не підбирають). Відстань Мінковського між двома точкам P та $Q \in R^n$ обчислюється за наступною формулою:

$$d(P,Q) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}},$$

де $P = (x_1, x_2, \dots, x_n)$, $Q = (y_1, y_2, \dots, y_n)$.

Принцип роботи цього методу полягає у «запам’ятовуванні» навчальної вибірки та передбачення класу шляхом призначення точок даних до класу, до якого належить k найближчих за обраною метрикою відстані точок (Евклідова відстань). На мою думку, такий підхід є досить ефективним для задачі курсової роботи, адже загалом пасажири зі схожими оцінками певних послуг, віковою групо і т.д. будуть досить близько між собою знаходитись.

Одним з недоліків цього методу є так зване прокляття розмірності[10]. Дане явище характеризується тим, що при збільшенні розмірності даних вони можуть ставати більш розрідженими, тому це може призвести до перенавчання моделі, адже алгоритм сильно адаптується до навчальних даних. Крім того, при збільшенні розмірності також збільшується швидкість обчислень.

В якості наступного методу я обрав Logistic Regression (логістичну регресію) через швидкість її навчання, адже датасет у мене досить об’ємний, інтерпретабельність та простоту, що зменшує ризик перенавчання.

Даний метод припускає, що дані можна розділити гіперплощиною. Принцип роботи цього методу полягає в обчисленні лінійної комбінації

незалежних змінних та використання функції активації сигмоїди[11], яка має наступний вигляд:

$$f(z) = \frac{1}{1+e^{-z}},$$

де z – лінійна комбінація незалежних змінних.

Область значень сигмоїди [0;1], тобто на виході отримаємо ймовірність приналежності значення до класу 1. Якщо отримана ймовірність менше 0.5, то призначаємо приналежність до класу 0, інакше – 1.

Недоліком даного методу є насамперед те, що він часто показує погані результати для даних, які явно є нелінійними, тобто їх не можна розділити гіперплощиною.

Третім методом я обрав Random Forest Classifier (випадковий ліс), який є ансамблем дерев рішень. На відміну від методів вище, його важче інтерпретувати та потребує більших обчислень, особливо зі збільшенням кількості дерев, тому час навчання відповідно збільшується. Тим не менш, цей метод зазвичай показує значно кращі результати, ніж ті ж самі K-Nearest Neighbors та Logistic Regression. Ще однією причиною, чому я обрав даний метод, є його стійкість до шуму в даних, адже він є ансамблем. Крім того, для випадкового лісу часто достатньо налаштувати лише один гіперпараметр – кількість дерев, а дані не треба стандартизувати перед навчанням.

4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors

Перед навчанням моделі KNN бажано стандартизувати дані, тому буде використано клас Pipeline, який складається із самої моделі та стандартизатора, завдяки якому перед навчанням моделі та прогнозуванням дані будуть стандартизовуватись. Для підбору гіперпараметрів, а саме кількості сусідів k, буде використано алгоритм GridSearchCV[12], який для кожного набору значень гіперпараметрів з заданого списку (я обрав від 1 до 29 включно для k) навчає модель та використовує перехресну перевірку (я обрав 5-кратну) для її оцінки на основі заданої метрики (я обрав точність). Модель з найвищим показником метрики буде обрана (рис. 4.1).

```
In 32
knn_pipe = Pipeline([('sc', StandardScaler()),
                     ('knn', KNeighborsClassifier())])
param_grid = {'knn__n_neighbors': range(1, 30)}
knn_gs = GridSearchCV(knn_pipe, param_grid=param_grid, scoring='accuracy', n_jobs=-1)
knn_gs.fit(X_train, y_train)
print(f'Best accuracy: {knn_gs.best_score_:.2%}, best params: {knn_gs.best_params_}')
print(f'Test accuracy: {knn_gs.score(X_test, y_test):.2%}')
Executed in 1m, 28 May at 00:28:26

Best accuracy: 92.89%, best params: {'knn__n_neighbors': 7}
Test accuracy: 92.84%
```

Рисунок 4.1 – Підбір оптимального гіперпараметра для методу KNN та навчання оптимальної моделі

Як бачимо, для даного розбиття на навчальні та тестові набори даних оптимальним значенням k є 7, при якому найкраща точність моделі буде становити 92.89%, а її точність на тестовому наборі – 92.84%. Отриманий показник метрики на тестовому наборі є досить високим.

Тепер побудуємо матрицю невідповідностей для даної моделі, а також виведемо значення різних метрик: точність, влучність, повнота та міра F1 (рис. 4.2).

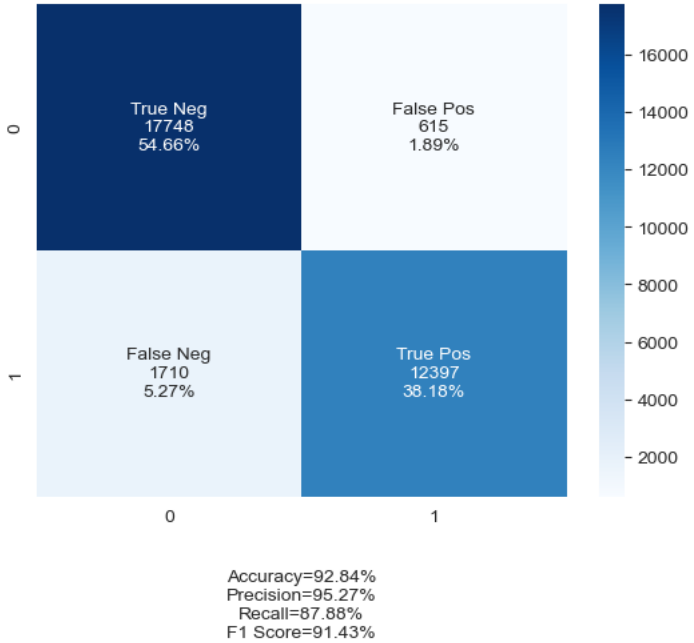


Рисунок 4.2 – Матриця невідповідностей зі значенням метрик для KNN

Отримана матриця показує, що кількість хибних класифікацій пасажирів як нейтральних/незадоволених, коли вони насправді були задоволені, майже втричі більша за кількість хибно класифікованих незадоволених/нейтральних пасажирів. Цим і пояснюється менше значення метрики повноти (recall), ніж

метрики влучності (precision). Міра F1 є більшою за. 0.9, що є високим показником.

4.3 Аналіз отриманих результатів для методу Logistic Regression

Аналогічно з використанням алгоритму GridSearchCV з 5-кратною перехресною перевіркою підберемо гіперпараметр C, що є оберненим значенням «сили» L2 регуляризації (гребнева регресія), серед заданих значень (0.1, 0.05, 0.1, 0.5, 1, 1.5, 10, 100). Також буде застосовуватись Pipeline зі стандартизацією (рис. 4.3).

```
In 34
logistic_regression_pipe = Pipeline([('sc', StandardScaler()), ('lr', LogisticRegression())])
param_grid = {'lr__C': [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100]}
logistic_regression_gs = GridSearchCV(logistic_regression_pipe, param_grid=param_grid,
 scoring='accuracy', n_jobs=-1)
logistic_regression_gs.fit(X_train, y_train)
print(f'Best accuracy: {logistic_regression_gs.best_score_:.2%}, best params: {logistic_regression_gs
 .best_params_}')
print(f'Test accuracy: {logistic_regression_gs.score(X_test, y_test):.2%}')
Executed in 982ms, 28 May at 00:28:28

Best accuracy: 87.45%, best params: {'lr__C': 0.05}
Test accuracy: 87.23%
```

Рисунок 4.3 – Підбір оптимального гіперпараметра для методу Logistic Regression та навчання оптимальної моделі

Оптимальним значенням гіперпараметра C серед заданих для даного розбиття на навчальні та тестові набори даних є 0.05. Отриманий показник метрики точності на тестовому наборі є досить непоганим, зважаючи також на те, що навчання моделі є дуже швидким.

Аналогічно побудуємо матрицю невідповідностей для даної моделі, а також виведемо значення різних метрик: точність, влучність, повнота та міра F1 (рис. 4.4).

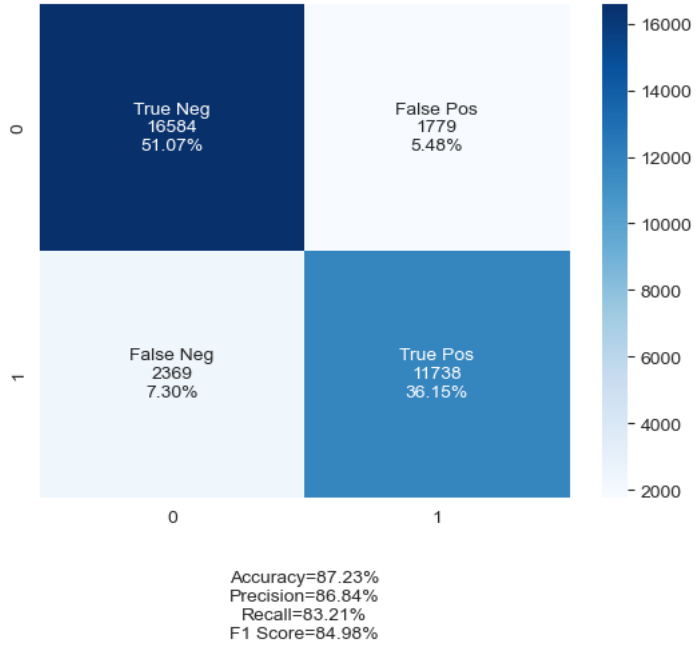


Рисунок 4.4 – Матриця невідповідностей зі значенням метрик для Logistic Regression

Отримана матриця показує, що кількість хибних класифікацій пасажирів як нейтральних/незадоволених, коли вони насправді були задоволені, дещо більша за кількість хибно класифікованих незадоволених/нейтральних пасажирів. Оскільки в даному випадку різниця між ними невелика, то й різниця значення метрики повноти (recall) та метрики влучності (precision) також невелика. Міра F1 лежить у межах від 0.8 до 0.9, що є непоганим показником.

4.4 Аналіз отриманих результатів для методу Random Forest Classifier

На відміну від двох попередніх методів, Random Forest Classifier (випадковий ліс) – це ансамблевий метод, оскільки він використовує певну кількість дерев рішень. З цієї причини та через досить велику кількість ознак і самих екземплярів використання алгоритму GridSearchCV не є оптимальним, адже це буде вимагати важких обчислень та займати занадто багато часу. Алгоритм RandomizedSearchCV, який не перебирає всі значення гіперпараметрів серед заданих, також не є прийнятним рішенням з аналогічної причини. Отже, я вирішив підібрати найвпливовіший гіперпараметр для випадкового лісу,

кількість дерев рішень, використовуючи ітеративний підхід навчання моделей з різною кількістю дерев серед заданого списку, обираючи модель з тією кількістю дерев, при якій вона показує найвищу точність на тестовому наборі даних (рис. 4.5).

```
In 36  1  estimators = range(10, 201, 10)
       2  best_accuracy = 0
       3  best_forest = None
       4  best_n = 0
       5  accuracies = []
       6  for n_estimators in estimators:
       7      random_forest = ensemble.RandomForestClassifier(n_estimators=n_estimators)
       8      random_forest.fit(X_train, y_train)
       9      accuracy = random_forest.score(X_test, y_test)
      10      accuracies.append(accuracy)
      11      if accuracy > best_accuracy:
      12          best_accuracy = accuracy
      13          best_forest = random_forest
      14          best_n = n_estimators
      15
      16  print(f'Test accuracy: {best_accuracy:.2%}, best number of trees: {best_n}')
          Executed in 2m, 28 May at 00:30:34

Test accuracy: 96.13%, best number of trees: 160
```

Рисунок 4.5 – Підбір оптимальної кількості дерев для методу Random Forest Classifier та навчання оптимальної моделі

У результаті такого підбору серед значень кількості дерев від 10 до 200 включно з кроком у 10 отримали для даного розбиття на навчальні та тестові набори даних оптимальну кількість дерев рівну 160.

Для більшої наочності побудуємо графік залежності точності моделі на тестовому наборі від кількості дерев (рис. 4.6).

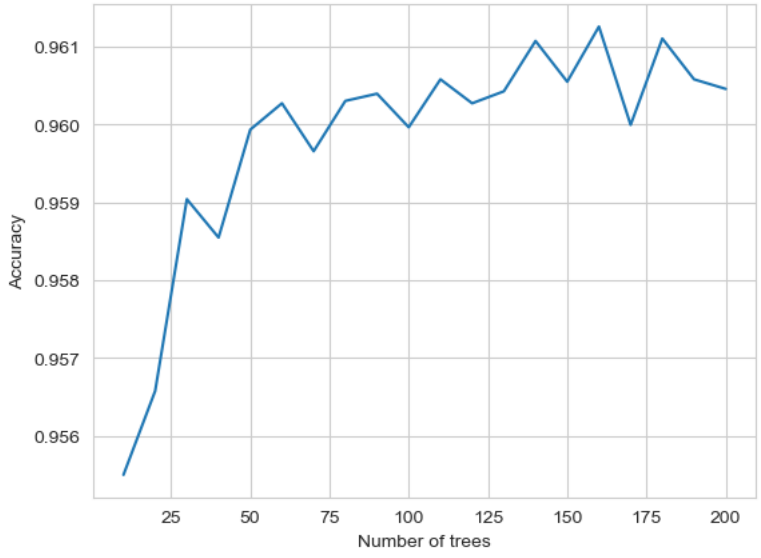


Рисунок 4.6 – Графік залежності точності моделі від кількості дерев

Можна побачити, що спочатку точність різко зростає, а потім дещо коливається і набуває локального максимуму для 160 дерев.

Аналогічно побудуємо матрицю невідповідностей для даної моделі, а також виведемо значення різних метрик: точність, влучність, повнота та міра F1 (рис. 4.7).

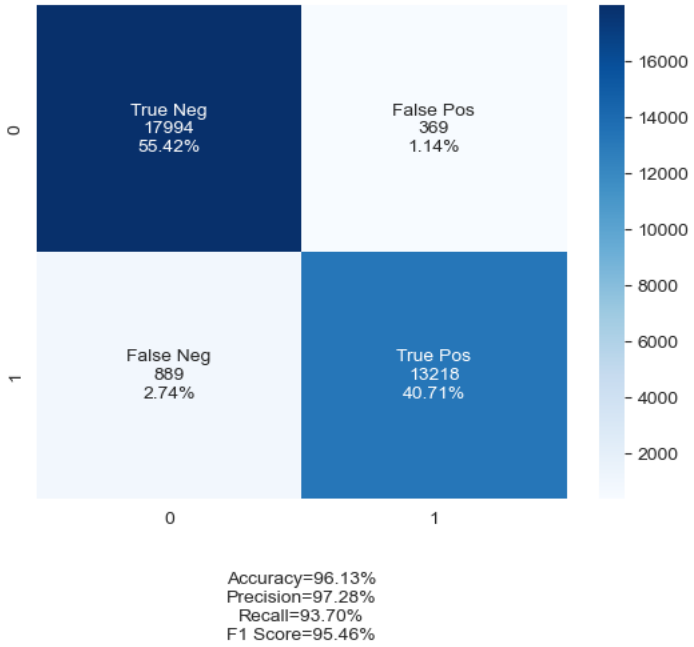


Рисунок 4.7 – Матриця невідповідностей зі значенням метрик для Random Forest Classifier

Отримана матриця показує, що кількість хибних класифікацій пасажирів як нейтральних/незадоволених, коли вони насправді були задоволені, дещо більша за кількість хибно класифікованих незадоволених/нейтральних пасажирів. Оскільки в даному випадку різниця між ними невелика, то й різниця значення метрики повноти (recall) та метрики влучності (precision) також невелика. Міра F1 більша за 0.95, що є дуже високим показником.

4.5 Порівняння отриманих результатів методів

Проаналізувавши окремо кожен із методів, варто провести порівняння даних методів. Для цього створимо датафрейм з різними метриками та їх значеннями для кожного з методів, відсортувавши його за спаданням метрики точності (рис. 4.8).

In 39

```
def get_score_list(classifiers, X_data, y_data):
    predicted = {c: c.predict(X_data) for c in classifiers}
    metrics_list = [metrics.accuracy_score, metrics.precision_score, metrics.recall_score, metrics
     .f1_score]
    result = []
    for c in classifiers:
        result.append([metric(y_data, predicted[c]) for metric in metrics_list])
    return result


classifiers_list = [knn_gs, logistic_regression_gs, best_forest]
pd.DataFrame(get_score_list(classifiers_list, X_test, y_test), columns=['accuracy', 'precision',
 'recall', 'f1_score'],
             index=['knn', 'logistic_regression', 'random_forest']).sort_values(by='accuracy',
              ascending=False)
```

Executed in 1s, 28 May at 00:30:36

Out 39

3 rows × 4 columns pd.Dataframe

CSV

	accuracy	precision	recall	f1_score
random_forest	0.961257	0.972842	0.936982	0.954575
knn	0.928395	0.952736	0.878784	0.914267
logistic_regression	0.872251	0.868388	0.832069	0.849841

Рисунок 4.8 – Датафрейм значень різних метрик для трьох обраних методів

Отже, бачимо, що по всім метрикам оптимальним Random Forest Classifier, який до того ж є прийнятним з точки зору часу та складності обчислень. KNN також має високі показники та є більш інтерпретабельним і швидшим при цьому. Логістична регресія демонструє найгірші результати, які є непоганими, проте для даної задачі краще використовувати перші два методи. Загалом використання Random Forest Classifier для даної задачі є, як на мене, найбільш обґрунтованим.

ВИСНОВКИ

Аналіз рівня задоволеності пасажирів є дуже важливим для кожної авіакомпанії, щоб покращувати якість послуг, які вона пропонує та в такий спосіб залишатись конкурентоспроможною.

Для прогнозування рівня задоволеності пасажирів, а саме бінарної класифікації на задоволений або нейтральний/незадоволений, було використано моделі трьох методів: K-Nearest Neighbor, Logistic Regression та Random Forest Classifier, проаналізовано їх за допомогою матриці невідповідностей і різних метрик та порівняно ці методи між собою.

Внаслідок інтелектуального аналізу для даного випадкового розбиття на навчальні та тестові набори даних отримали наступні результати: модель KNN. показала точність 92.84%, що є досить високим показником, модель Logistic Regression – 87.23%, що все ще є непоганим показником, проте не таким високим для нашої задачі, модель Random Forest Classifier – 96.13%, що є дуже високим показником, тому ця модель є гарним та обґрунтованим вибором, адже все ще є прийнятною з точки зору складності та часу обчислень. Усі моделі мають більше хибних негативних класифікацій, ніж хибних позитивних, тому значення метрики влучності для них більша, ніж значення метрики повноти.

Отже, для подальшого використання з метою класифікації пасажирів авіакомпанії обрано модель Random Forest Classifier через її високі показники усіх метрик, зокрема, метрики точності, що є особливо важливим для авіакомпанії, щоб робити правильні висновки на основі аналізу, та прийнятий час навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація мови програмування Python. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/
2. Бібліотека Pandas. [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/
3. Бібліотека Seaborn. [Електронний ресурс] – Режим доступу до ресурсу: https://seaborn.pydata.org/introduction.html
4. Бібліотека Matplotlib. [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/stable/
5. Бібліотека Sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/user_guide.html
6. Бібліотека NumPy. [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org
7. Бібліотека SciPy. [Електронний ресурс] – Режим доступу до ресурсу: https://scipy.org
8. Бібліотека os. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/os.html
9. Метрика Мінковського. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Метрика_Мінковського
10. Прокляття розмірності. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Прокляття_розмірності
11. Сигмоїда. [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Сигмоїда
12. GridSearchCV for Beginners. [Електронний ресурс] — Режим доступу до ресурсу: https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду Класифікація рівня задоволеності пасажирів авіакомпанією.

(Найменування програми (документа))

SSD

(Вид носія даних)

8 арк, 64 Кб

(Обсяг програми (документа), арк.,

студента групи ІП-11 ІІ курсу

Сідака К. І.

```
import pandas as pd
import numpy as np
import os
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import chi2_contingency
from sklearn import ensemble, metrics
from sklearn.linear_model import LogisticRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
%matplotlib inline
#%%
directory = './data'
dataframes = []
for filename in os.listdir(directory):
    file_path = os.path.join(directory, filename)
    df = pd.read_csv(file_path, index_col=0)
    df = df.set_index('id')
    dataframes.append(df)

final_df = pd.concat(dataframes)
final_df = final_df.sort_index()
final_df.duplicated().sum()
#%%
final_df.head()
#%%
```

```
final_df.tail()
#%%
null_count = final_df.loc[:, final_df.isna().sum() > 0].isna().sum()
pd.DataFrame({'null_count': null_count, 'null_percent': null_count / final_df.shape[0]
* 100})
#%%
final_df['Arrival Delay in Minutes'] = final_df['Arrival Delay in Minutes'].fillna(
    value=final_df['Arrival Delay in Minutes'].mean())
final_df.isna().sum()
#%%
final_df.columns = final_df.columns.str.lower().str.replace(' ', '_')
final_df.columns.values
#%%
final_df.info()
#%%
categorical_columns = final_df.columns[~final_df.columns.isin(
    ['age', 'flight_distance', 'departure_delay_in_minutes', 'arrival_delay_in_minutes',
'satisfaction'])]
final_df[categorical_columns] = final_df[categorical_columns].astype('category')
final_df.info()
#%%
final_df.describe()
#%%
final_df.describe(include=['category'])
#%%
plt.pie(final_df.satisfaction.value_counts(), labels=["Neutral or dissatisfied",
"Satisfied"],
        colors=sns.color_palette("YlOrBr"), autopct='%1.1f%%');
#%%
heatmap = sns.heatmap(final_df.corr(), vmin=-1, vmax=1, annot=True)
```

```
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize': 12}, pad=12)
plt.xticks(rotation=45);
#%%
vif_df = pd.DataFrame()
features = final_df.select_dtypes(include='number').columns
vif_df["feature"] = features
vif_df["VIF"] = [variance_inflation_factor(final_df[features].values, i) for i in
range(len(features))]
vif_df
#%%
fig, axes = plt.subplots(6, 3, figsize=(25, 25))
for i, col in enumerate(final_df[categorical_columns]):
    column_values = final_df[col].value_counts()
    labels = column_values.index
    sizes = column_values.values
    axes[i // 3, i % 3].pie(sizes, labels=labels, colors=sns.color_palette("YlOrBr"),
autopct='%1.0f%%', startangle=90)
    axes[i // 3, i % 3].axis('equal')
    axes[i // 3, i % 3].set_title(col)
#%%
ordinal_df = final_df.select_dtypes('category').loc[:,
'inflight_wifi_service':].astype(int)
ordinal_continuous = ordinal_df.merge(final_df.select_dtypes(np.number),
left_index=True, right_index=True)
plt.figure(figsize=(15, 10))
sns.heatmap(ordinal_continuous.corr(method='spearman'), annot=True);
#%%
satisfaction_mapping = {v: k for k, v in enumerate(final_df['satisfaction'].unique())}
satisfaction_mapping
#%%
```

```
final_df['satisfaction'] = final_df['satisfaction'].map(satisfaction_mapping)
final_df['satisfaction'].head()
#%%
sns.barplot(x='gender', y='satisfaction', data=final_df, errorbar=None);
#%%
sns.barplot(x='customer_type', y='satisfaction', data=final_df, errorbar=None);
#%%
sns.barplot(x='type_of_travel', y='satisfaction', data=final_df, errorbar=None);
#%%
sns.barplot(x='class', y='satisfaction', data=final_df, errorbar=None);
#%%
def get_correlation_nominal(column: str, alpha=0.05, target_column: str =
'satisfaction'):
    contingency_table = pd.crosstab(final_df[column], final_df[target_column])
    stat, p, dof, expected = chi2_contingency(contingency_table)
    return p, p <= alpha
#%%
nominal_columns = ['gender', 'customer_type', 'type_of_travel', 'class']
p_list, is_correlated_list = [], []
for c in nominal_columns:
    p, is_correlated = get_correlation_nominal(c)
    p_list.append(p)
    is_correlated_list.append(is_correlated)

pd.DataFrame({'p_value': p_list, 'is_correlated': is_correlated_list},
index=nominal_columns)
#%%
final_df['age_group'] = pd.qcut(final_df['age'], 4)
sns.barplot(x='age_group', y='satisfaction', data=final_df, errorbar=None);
#%%
```

```
final_df['age_group'] = final_df['age_group'].cat.codes + 1
final_df['age_group'].head()
#%%
final_df = final_df.drop(
    columns=['arrival_delay_in_minutes', 'inflight_entertainment', 'age'])
final_df.head()
#%%
final_df = pd.get_dummies(final_df, columns=['gender', 'customer_type',
'type_of_travel', 'class'], drop_first=True)
final_df.columns = final_df.columns.str.lower().str.replace(' ', '_')
final_df.head()
#%%
X, y = final_df.drop(columns=['satisfaction']), final_df['satisfaction']
#%%
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y)
#%%
def plot_confusion_matrix(classifier):
    predicted = classifier.predict(X_test)
    confusion_matrix = metrics.confusion_matrix(y_test, predicted)
    group_names = ['True Neg', 'False Pos', 'False Neg', 'True Pos']
    group_counts = ['{0:0.0f}'.format(value) for value in
                    confusion_matrix.flatten()]
    group_percentages = ['{0:.2%}'.format(value) for value in
                         confusion_matrix.flatten() / np.sum(confusion_matrix)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
              zip(group_names, group_counts, group_percentages)]
    labels = np.asarray(labels).reshape(2, 2)
    accuracy = np.trace(confusion_matrix) / float(np.sum(confusion_matrix))
    precision = confusion_matrix[1, 1] / sum(confusion_matrix[:, 1])
    recall = confusion_matrix[1, 1] / sum(confusion_matrix[1, :])
```

```
    f1_score = 2 * precision * recall / (precision + recall)
    stats_text =
f"\n\nAccuracy={accuracy:.2%}\nPrecision={precision:.2%}\nRecall={recall:.2%}\n
F1 Score={f1_score:.2%}"
    sns.heatmap(confusion_matrix, annot=labels, fmt='', cmap='Blues')
    plt.xlabel(stats_text)
#%%
knn_pipe = Pipeline([('sc', StandardScaler()),
                     ('knn', KNeighborsClassifier())])
param_grid = {'knn__n_neighbors': range(1, 30)}
knn_gs = GridSearchCV(knn_pipe, param_grid=param_grid, scoring='accuracy',
n_jobs=-1)
knn_gs.fit(X_train, y_train)
print(f'Best accuracy: {knn_gs.best_score_:.2%}, best params:
{knn_gs.best_params_}')
print(f'Test accuracy: {knn_gs.score(X_test, y_test):.2%}')
#%%
plot_confusion_matrix(knn_gs)
#%%
logistic_regression_pipe = Pipeline([('sc', StandardScaler()), ('lr',
LogisticRegression())])
param_grid = {'lr__C': [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100]}
logistic_regression_gs = GridSearchCV(logistic_regression_pipe,
param_grid=param_grid, scoring='accuracy', n_jobs=-1)
logistic_regression_gs.fit(X_train, y_train)
print(f'Best accuracy: {logistic_regression_gs.best_score_:.2%}, best params:
{logistic_regression_gs.best_params_}')
print(f'Test accuracy: {logistic_regression_gs.score(X_test, y_test):.2%}')
#%%
plot_confusion_matrix(logistic_regression_gs)
```

```
#%%
estimators = range(10, 201, 10)
best_accuracy = 0
best_forest = None
best_n = 0
accuracies = []
for n_estimators in estimators:
    random_forest = ensemble.RandomForestClassifier(n_estimators=n_estimators)
    random_forest.fit(X_train, y_train)
    accuracy = random_forest.score(X_test, y_test)
    accuracies.append(accuracy)
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_forest = random_forest
        best_n = n_estimators

print(f'Test accuracy: {best_accuracy:.2%}, best number of trees: {best_n}')
#%%
plt.plot(estimators, accuracies)
plt.xlabel('Number of trees')
plt.ylabel('Accuracy');
#%%
plot_confusion_matrix(best_forest)
#%%
def get_score_list(classifiers, X_data, y_data):
    predicted = {c: c.predict(X_data) for c in classifiers}
    metrics_list = [metrics.accuracy_score, metrics.precision_score,
metrics.recall_score, metrics.f1_score]
    result = []
    for c in classifiers:
```

```
        result.append([metric(y_data, predicted[c]) for metric in metrics_list])
    return result


classifiers_list = [knn_gs, logistic_regression_gs, best_forest]
pd.DataFrame(get_score_list(classifiers_list, X_test, y_test), columns=['accuracy',
'precision', 'recall', 'f1_score'],
             index=['knn', 'logistic_regression',
'random_forest']).sort_values(by='accuracy', ascending=False)
```