КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра математичної інформатики

«До захисту допущено»
Завідувач кафедри
В. М. Терещенко _____
(підпис)

«___» _____20 21 р.

Дипломна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ПЕРЕДБАЧЕННЯ ЗАТРИМОК АВІАРЕЙСІВ

Виконала студентка 4 курсу
Лопуляк Жанна Ярославівна _____
(підпис)

Науковий керівник:
Доцент, кандидат фізико - математичних наук
Лівінська Ганна Володимирівна _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API –Прикладний програмний інтерфейс (англ. Application Programming Interface)

FAA – Федеральне авіаційне управління США (англ. Federal Aviation Administration)

HTML – мова розмітки гіпертексту (англ. HyperText Markup Language)

IATA –Міжнародна асоціація повітряного транспорту (англ. International Air Transport Association)

ICAO – Міжнародна організація цивільної авіації (англ. International Civil Aviation Organization)

KNN – Метод k-найближчих сусідів (англ. k-nearest neighbor method)

METAR – авіаційний метеорологічний код (англ. METeorological Aerodrome Report або англ. Meteorological Terminal Aviation Routine Weather Report)

SVM – метод опорних векторів (англ. support-vector machines)

URL – уніфікований локатор ресурсів або адреса ресурсу (англ. Uniform Resource Locator)

UTC – Всесвітній координований час (англ. Coordinated Universal Time, Time Zulu — «час зулу», Z)

ВСТУП

В епоху глобалізації авіаподорожі стали одним з основних способів пересування по всьому світу як для далеких міжнародних подорожей на тисячі кілометрів, так і внутрішніх рейсах в межах однієї країни. Водночас із зростанням кількості авіаліній та польотів, затримка рейсів стала серйозною проблемою як для авіакомпаній, так і для пасажирів.

У 2017 році Федеральне авіаційне управління США оцінило щорічну вартість затримок вильотів в Штатах у 26,6 млрд доларів. У 2018 році з-поміж скарг авіапасажирів скарги на затримки та скасування рейсів склали майже 33% для американських авіакомпаній [1]. Затримки та скасування рейсів не лише спричиняють дуже значні економічні втрати, а й несуть погані наслідки для пасажирів, погіршуючи їхнє враження про подорож.

До причин затримок належать погодні умови, перевантаженість аеропорту, технічні роботи на злітно-посадкових смугах та немало інших факторів. Проте, більшість суттєвих затримок пов’язані саме з погодою, і люди звикли вважати, що це може бути невід’ємною частиною авіаподорожей. Якби було можливо відносно точно, принаймні з точністю хоча б до 70 відсотків, передбачати затримки рейсів за кілька днів до польоту, це могло б заощадити як пасажирам так і авіалініям чимало часу та грошей.

Одним із підходів до такого передбачення є використання машинного навчання – підгалузі інформатики, яка займається дослідженням та побудовою алгоритмів, котрі здатні навчатися та надавати прогнози на основі даних. Воно було і залишається одним з основних методів побудови моделей для передбачення затримок авіарейсів. Методами машинного навчання, які зазвичай застосовуються для побудови таких моделей є Random forest, нейронні мережі, SVM, метод k-найближчих сусідів тощо.

У [2] було застосовано Random forest для передбачення затримки вильотів. Автори порівнювали цей підхід із регресійними моделями для

передбачення затримки вильотів у аеропортах Сполучених Штатів Америки. Вони розглядали часові проміжки на 2, 4, 6 та 24 години наперед. Із зростанням часового проміжку зростали помилки на тестових даних.

У [3] передбачалися прильоти до Міжнародного аеропорту імені Джона Ф. Кеннеді в Нью Йорку із використанням методів прийняття рішень в умовах невизначеності.

Автори дослідження [4] використовують алгоритми навчання з підкріпленням для передбачення затримок taxi-out (під taxi-out мається на увазі час від початку буксирування літака до зльоту). Автори моделювали Марківський процес та застосовували алгоритм машинного навчання. Коли ця модель запускалася за 15 хвилин до запланованого вильоту, вона показувала доволі хорошу точність для Міжнародного аеропорту імені Джона Ф. Кеннеді в Нью Йорку та Тампського міжнародного аеропорту у Флориді.

[5] описує побудову рекомендаційної системи передбачення затримок у деяких аеропортах. Передбачення базується на алгоритмі K-NN і використовує історичні дані щоб розпізнавати ситуації подібні до таких, що були в минулому.

Метою нашої роботи є побудувати моделі машинного навчання, які б вирішували задачу передбачення затримки рейсів на основі даних про польоти українських авіаліній. Для досягнення поставленої мети потрібно виконати такі завдання:

1. Знайти джерела даних про історію польотів та сформувати власний набір даних.
2. На основі сформованого набору даних побудувати моделі машинного навчання для передбачень затримок.
3. Порівняти якість моделей та визначити яка із них дає найкращі передбачення.

В якості інструменту розроблення було обрано мову програмування Python3, яка є безкоштовною, вільно поширюваною, з відкритим вихідним кодом та великим набором допоміжних бібліотек для обробки даних. Інтегрованими середовищами розробки на різних етапах роботи слугували PyCharm, Google Colaboratory та Jupyter Notebook.

Сфери застосування подібних моделей доволі широкі: вони можуть бути корисні як авіалініям, аеропортам, так і пасажирам. Авіалінії зацікавлені у розумінні ризиків затримок щоб зменшувати витрати, спричинені затримками. Якщо знати, що рейс потенційно затримується, можна оптимізовувати використання пального, коригувати плани для наступного рейсу авіалайнера, зрештою робити додаткові кроки для уникнення затримки. Для аеропортів, особливо завантажених, це потрібно щоб краще розпланувати послідовність посадок та вильотів, надавати інформацію диспетчерам, знижувати ризики безпеки, оптимізовувати роботу буксирів, іншої техніки та персоналу. Для пасажирів це може бути корисним задля кращого планування власного часу. До прикладу, при плануванні рейсу з пересадкою та обранні тривалості пересадки корисно було б заздалегідь знати, чи є ймовірність що перший рейс затримається. Тоді можна забронювати наступний рейс із достатнім запасом часу.

РОЗДІЛ 1. ЗБІР ТА ФОРМУВАННЯ НАБОРУ ДАНИХ

1.1 Збір даних про авіарейси та їх затримки

Спочатку було досліджено наявні у вільному доступі набори даних про затримки польотів. Найвідомішим подібним набором даних є [6], який містить інформацію про приблизно шість мільйонів польотів у США за 2015 рік. Також, у відомих нам дослідженнях [7,8] переважно використовуються набори даних про польоти в США. Цю інформацію неважко знайти, оскільки її періодично надає у вільний доступ Міністерство транспорту США.

Проте цікавість авторів полягала в тому, щоб зробити подібне дослідження для українських аеропортів, або принаймні авіаліній. Ми шукали, проте не знайшли жодних робіт, де була б зібрана така інформація. Було вирішено робити це самостійно. Протягом тривалих пошуків не було знайдено жодного подібного набору даних про цивільну авіацію у вільному доступі, проте був знайдений сервер [9]. Першочергово він використовується для відстежування польотів довкола усієї земної кулі у реальному часі (приклад інтерфейсу на Рисунку 1.1.1). Проте, за оформлення підписки можна отримати доступ до архіву польотів за останні три роки, чим ми і скористалися. На жаль, можливості побачити історію вильотів/прильотів/затримок для конкретного аеропорту немає, проте можна дивитися подібну історію для кожного конкретного авіалайнера. Окрім того, є можливість дивитися перелік лайнерів які належать конкретній авіалінії. Тому ми вирішили збирати інформацію про історію польотів лайнерів, які належать українським авіалініям.

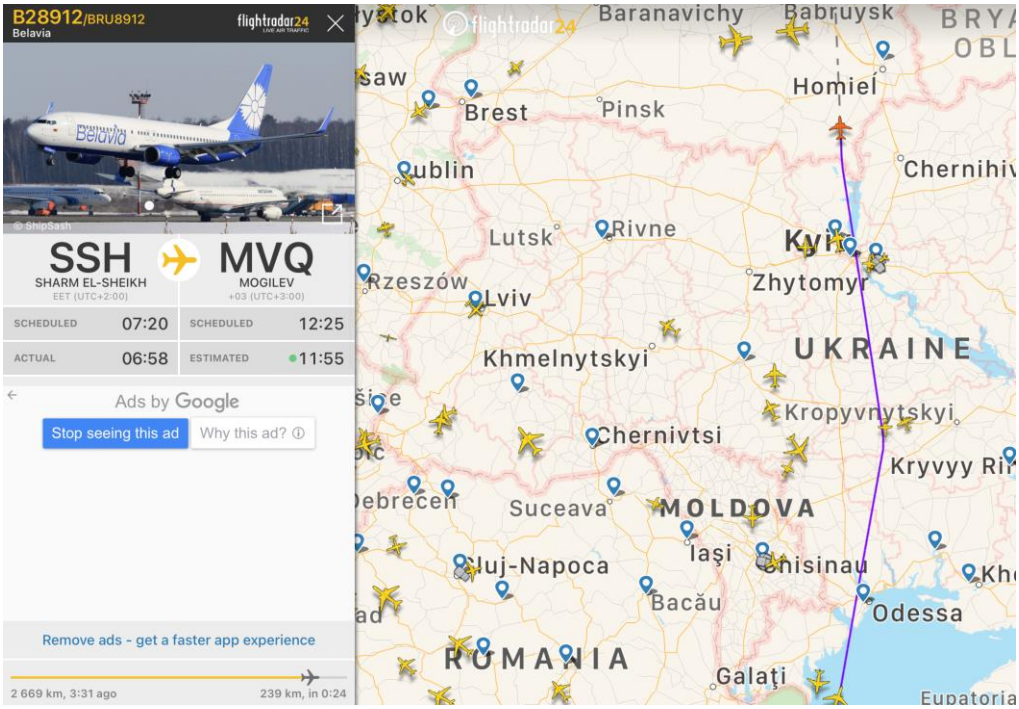


Рисунок 1.1.1 – вигляд застосунку flightradar24, зліва наведено інформацію про маршрут, час вильоту та прильоту тощо

По кожному конкретному літаку можна бачити таку інформацію, як наведено на Рисунку 1.1.2.

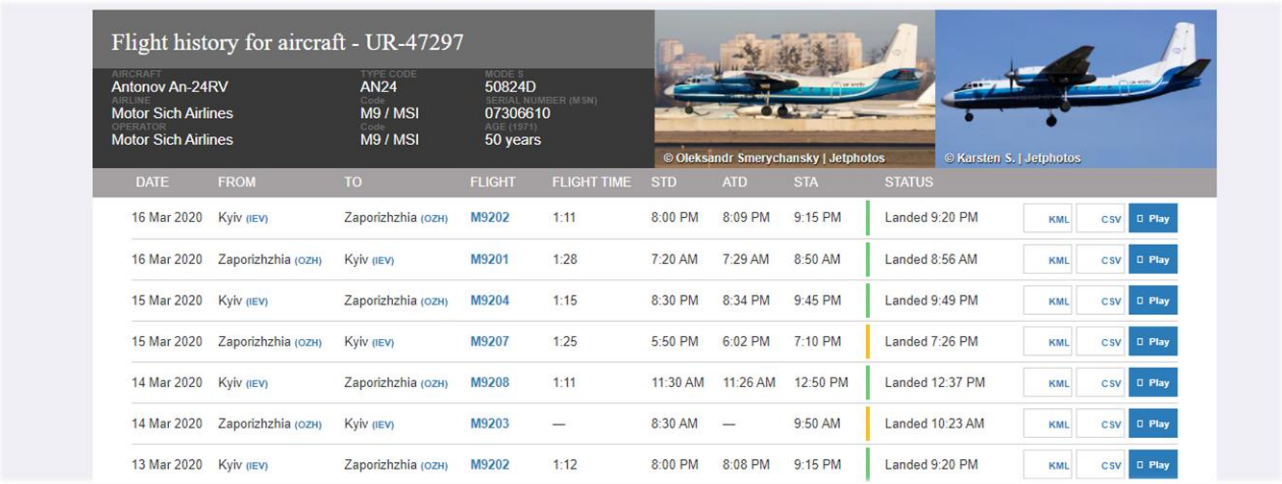


Рисунок 1.1.2 – початкова сторінка із переліками польотів для моделі UR-47297, джерело – веб-сайт flightradar24

Як видно з Рисунку 1.1.2, доступною є така інформація для авіалайнеру: унікальний ідентифікатор, авіалінія яка є нинішнім його оператором, код типу, серійний номер, вік, рік виробництва. По кожному із польотів, який він здійснював, маємо дату, аеропорти вильоту та прильоту та їхні трицифрові коди в дужках. Це – IATA коди аеропортів, один із унікальних ідентифікаторів для кожного аеропорту в світі, з якими ми ще стикатимемося надалі. Також бачимо номер рейсу, час польоту, час вильоту та прильоту за розкладом, справжній час прильоту та справжній час прильоту в останній колонці, щоправда в трохи іншому форматі: із словом “Landed” попереду. Далі є кнопки, що дають можливість завантажити інформацію про хід польоту: швидкість, зміна висоти, географічне положення тощо, які змінюються зі зміною часу. У своїй роботі ми цю інформацію не використовували.

Нашою ідеєю є якимось чином завантажити інформацію наведену на такій веб-сторінці та систематизувати її для різних літаків, сформувавши власний повноцінний набір даних. Єдиним відомим авторам варіантом для такої мети є завантаження вихідного коду сторінки у форматі .html. Цей код доступний до перегляду та завантаження будь-якому користувачеві. Щоб його побачити, варто по відкритій сторінці натиснути правою кнопкою миші і обрати «View page source».

Для завантаження документу в форматі .html, потрібно відкрити увесь перелік рейсів на сторінці, оскільки щоразу з підвантаженням нових рейсів вихідний код оновлюється. Щоб відкрити веб-сторінку із повним переліком рейсів, потрібно кожні сто польотів тиснути внизу на кнопку «Показати більше». На початку в автора були спроби робити це вручну, проте швидко стало зрозуміло, що таким чином витрачається надто багато часу і це суттєво сповільнить збір даних. Тоді було знайдено спосіб автоматизації натискання на кнопку на веб-сторінці. Для цього ми скористалися

розширенням для браузеру Google Chrome Tampermonkey. Було написано скрипт, котрий сам би автоматично натискав на кнопку.

Далі, після повного відкриття кожної із веб-сторінок із повним переліком доступних польотів та авіалайнера, було вручну завантажено вихідний код кожної із цих сторінок в форматі .html. В середньому розмір кожного із таких файлів становить 10Мб, варіюється залежно від кількості рейсів, здійснених літаком за останні три роки та записаних у базі даних джерела інформації. Всього маємо інформацію про польоти двох лайнерів авіалінії YanAir, дванадцяти – Windrose, двадцяти дев'яти – UIA, семи – Motorsich. На Рисунку 1.1.3 зображено розподіл між цими ж авіалініями за кількістю польотів (після подальшого очищення даних).

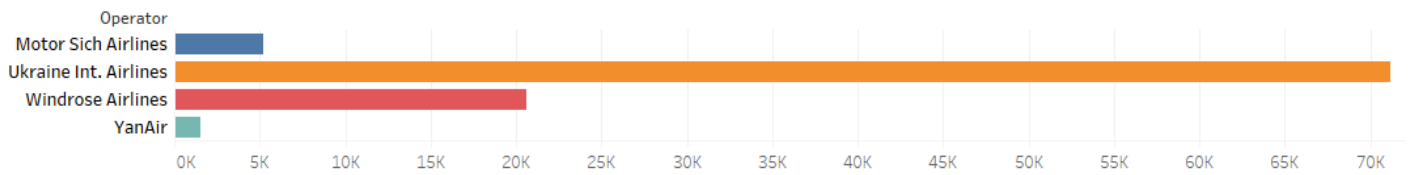


Рисунок 1.1.3 – кількість польотів для кожної із авіаліній в нашому наборі даних

Тепер, маючи файли із записами про польоти, будемо отримувати з них потрібну інформацію. Для перегляду самого HTML документу було використано середовище розробки Visual Studio Code, оскільки середовище розробки для Python не відображає коректно такі файли.

Спочатку знаходимо контейнер за класом, у якому знаходиться вся інформація про конкретний літак, така як вік, ідентифікаційний код тощо (інформація котру видно на верхній частині Рисунку 1.1.2). Записуємо інформацію про кожну із моделей в окремий файл models_info.csv. Окрім того, для інформації про вік літака ми користуємося не тим віком, що можна побачити на сайті. Ми записуємо рік виробництва, оскільки це зробить наш набір даних більш придатним для подальшого використання (до прикладу, якщо на сьогодні записати вік:18 років, то з плином часу це перестане

відображати коректну інформацію. А рік виробництва – величина абсолютна).

Далі зчитувалася інформація про кожен політ та записувалася у файл, окремий для кожного із літаків. Для цього, подібно до попередньо кроку, ми знаходили класи в яких міститься потрібна інформація та поелементно записували потрібну нам. Зауважимо, що чимало часу було втрачено на пошук проблеми, що спричиняла некоректне зчитування, якою зрештою виявилася наявність пробілу у назвах однакових змістовно класів із назвами «data-row» та «data -row», що показує необхідність давати повністю однакові імена ідентичним за змістом елементам. Після цього етапу зчитування для кожного із рейсів маємо такий перелік полів, як наведено на Рисунку1.1.4. Зазначимо, що всі з них зараз є текстовими, і поля що містять дату та час ще потребують додаткової обробки.

```
this_aircraft_flights_df = pd.DataFrame(
    {'Date of flight': dates,
     'Origin': origins,
     'Destination': destinations,
     'Flight number': flight_numbers,
     'Flight_times': flight_times,
     'Model': models,
     'STD': STDs,
     'ATD': ATDs,
     'STA': STAs,
     'ATA': ATAs
     })
```

Рисунок 1.1.4 – перелік наявних атрибутів для кожного польоту

Наступна задача – сформувати єдиний файл, де був би перелік усіх польотів, інформація про модель, що летіла рейс та час затримки вильоту і відповідно затримки прильоту. Об’єднування файлів та дописування інформації про модель не становить проблем, оскільки деталі про кожну модель ми вже мали в окремому файлі. Проте, на етапі обчислення

тривалості затримки, доводиться переводити кожен із часів вильоту та прильоту у формат часу. Для цього було використано бібліотеку datetime, яка дає можливість працювати з нашими форматами часу у текстовому вигляді, як-от 2:30 AM або 8:40 PM. Після зведення усього до одного формату часу, можемо обчислити абсолютні затримки для кожного із польотів. Окрім того, ми створюємо кілька нових стовпців в нашому наборі даних:

1. «month» із значеннями від 1 до 12, який показує номер місяця у році, коли відбувся рейс
2. «month_day» із значеннями від 1 до 31, який показує день місяця
3. «hour» із значеннями від 1 до 23, яка показує годину дня коли був запланований виліт. На основі цієї колонки також створюється «part_of_day», яка може набувати чотирьох значень: «Night», «Morning», «Day», «Evening»
4. «day_of_week», яка вказує день тижня
5. «is_delayed_departure», «is_delayed_arrival» які вказують чи був рейс затриманий, набувають значень True/False. В такому вигляді нам потрібна інформація для вирішення задачі класифікації
6. Стовпці «is_delayed_n_departure», «is_delayed_n_arrival», де значення n становлять 15, 30, 60, 120. Це зроблено для того, щоб мати на подальше можливість розрізняти довші та коротші затримки. Цілком імовірно, що затримки до 15-ти хвилин ми не вважатимемо повноцінними затримками і тоді зможемо скористатися стовпцем «is_delayed_15_departure»
7. «delayed_group_departure», «delayed_group_arrival» – стовпці, які набувають значень від 0 до 2, залежно від того, наскільки довгою була затримка вильоту та прильоту відповідно

На Рисунках 1.1.5 – 1.1.11 наведемо розподіл даних по новостворених стовпцях.

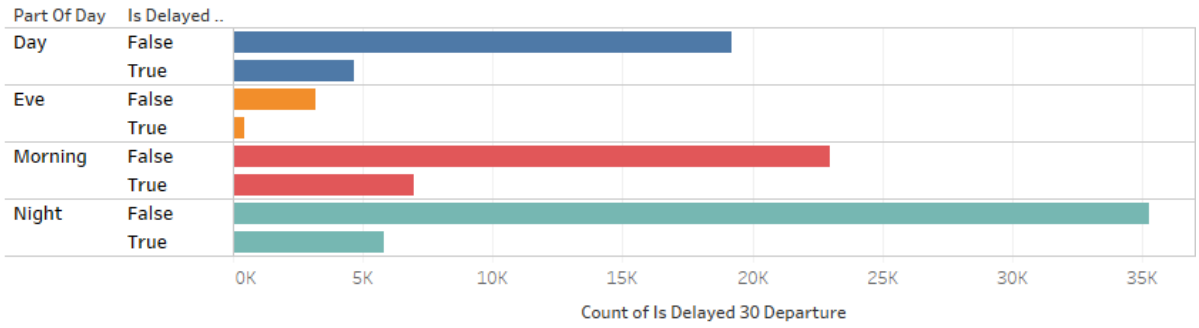


Рисунок 1.1.5 – Розподіл між вильотами, затримки яких становили мінімум 30 хвилин (True) та такими, що вилетіли із меншою затримкою або вчасно (False) залежно від частини дня.

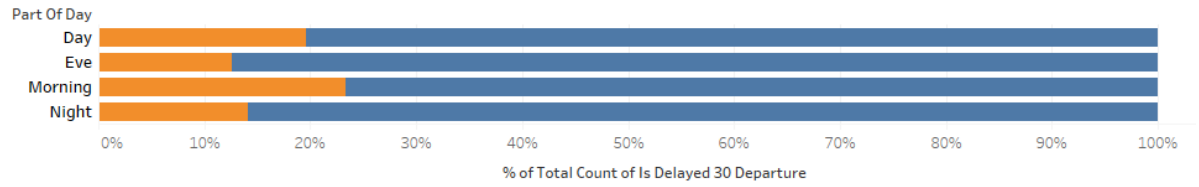


Рисунок 1.1.6 – Співвідношення між вильотами, затримка яких становила понад 30 хвилин (оранжевим) та до 30 хвилин (синім). Як видно, більший відсоток затримують вранці та вдень.

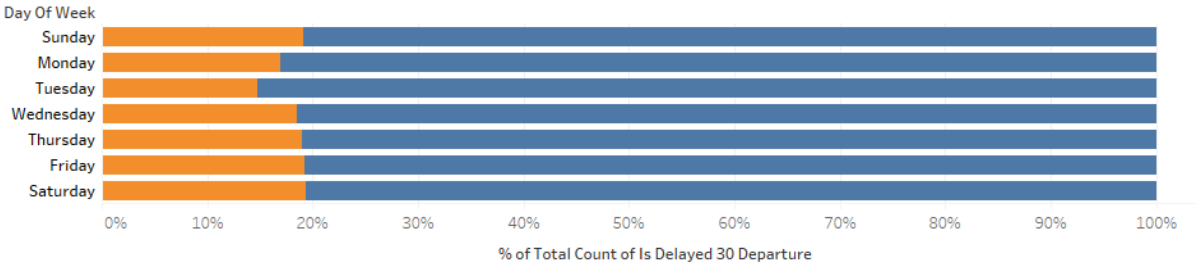


Рисунок 1.1.7 – Співвідношення між рейсами, що вилетіли із затримкою понад 30 хвилин та такими, затримка яких становила менше 30 хвилин. У понеділок та вівторок затримується менше рейсів, у інші дні тижня – більше.

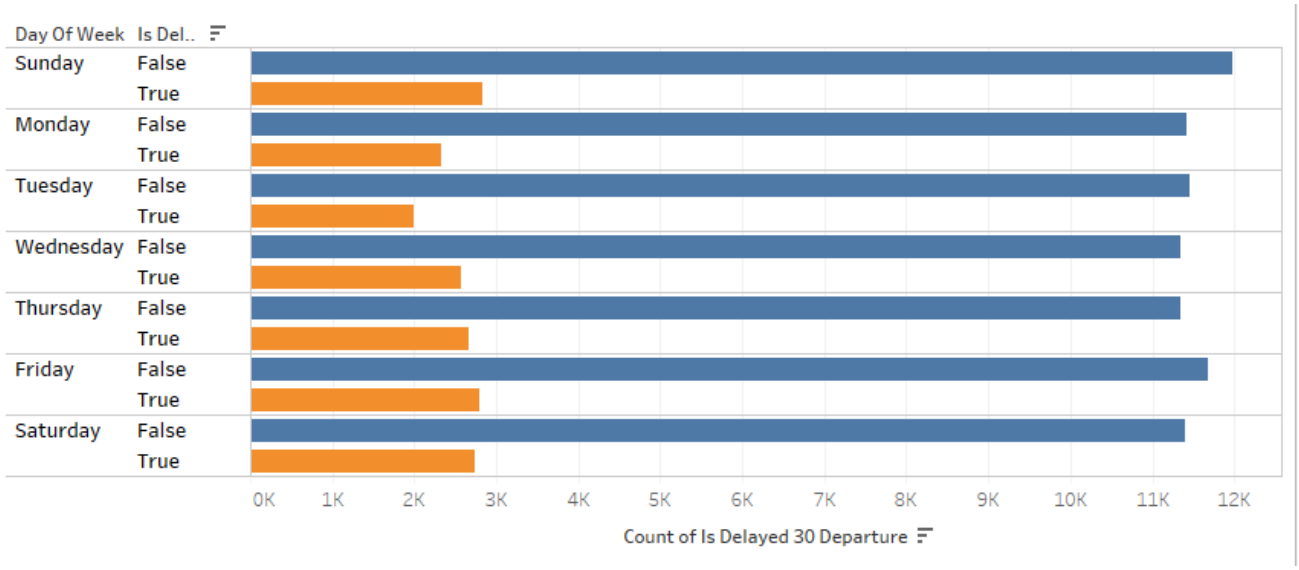


Рисунок 1.1.8 – Розподіл між вильотами, затримки яких становили мінімум 30 хвилин (оранжевим) та такими, що вилетіли із затримкою до 30 хвилин (синім) залежно від дня тижня.

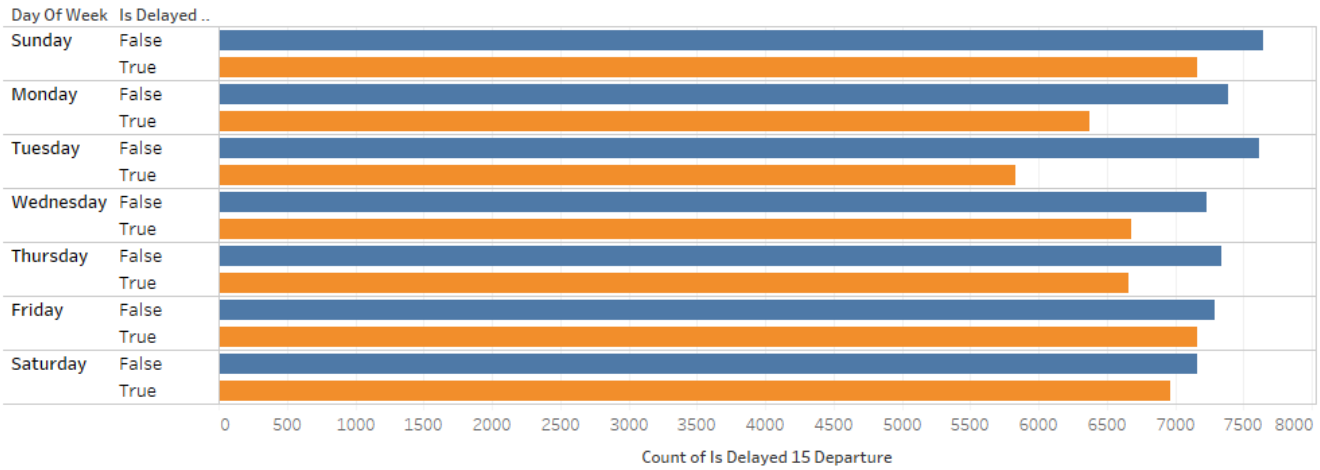


Рисунок 1.1.9 – Розподіл між вильотами, затримки яких становили мінімум 15 хвилин (оранжевим) та такими, що вилетіли із затримкою менше 15 хвилин (синім) залежно від дня тижня. Порівнюючи з попереднім рисунком, можемо бачити, що затримка значної частини рейсів становить 15-30 хвилин. Окрім того, зберігається паттерн із меншими затримками на початку тижня, зокрема у вівторок.

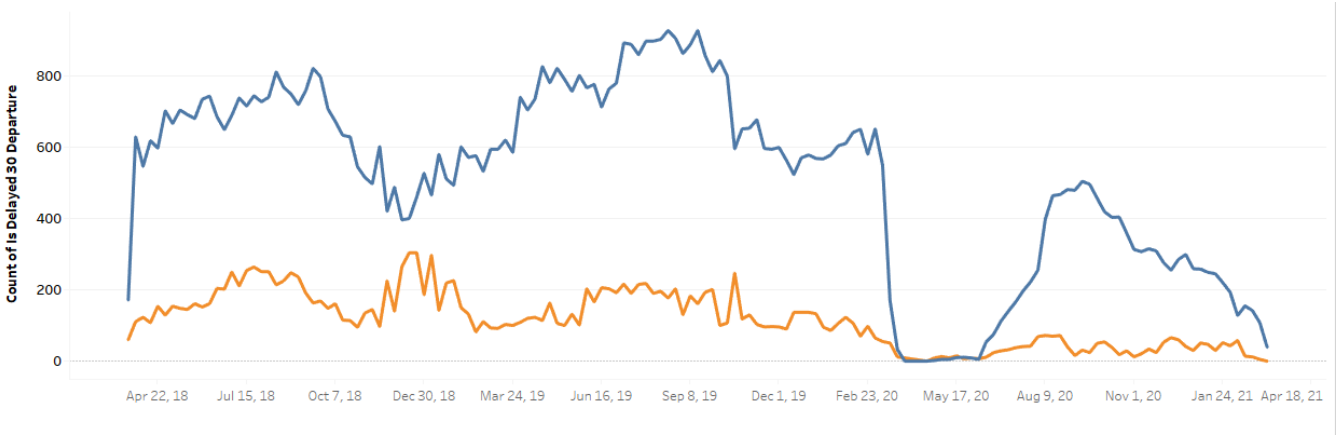


Рисунок 1.1.10 – Розподіл між вильотами, затримки яких становили мінімум 30 хвилин (оранжевим) та такими, що вилетіли із затримкою менше ніж 30 хвилин (синім) протягом усього проміжку часу для якого маємо набір даних.

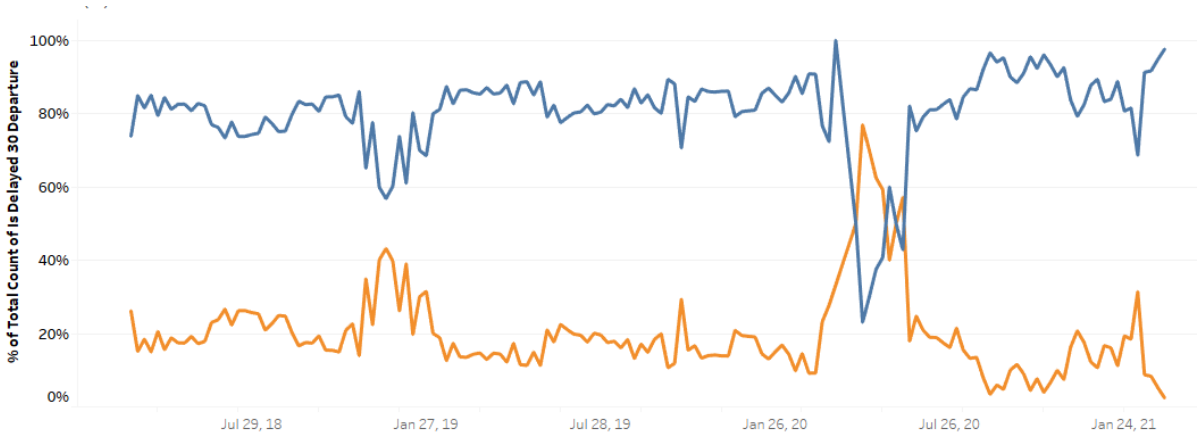


Рисунок 1.1.11 – Співвідношення між вильотами, затримки яких становили мінімум 30 хвилин (оранжевим) та такими, що вилетіли із затримкою менше ніж 30 хвилин (синім)

Як видно з Рисунку 1.1.11, на новорічні свята 2018/2019 зросла кількість затримок; у березні 2020 різко перестали літати через пандемію COVID-19; у квітні того ж року різко зросла кількість тривалих затримок; у серпні-вересні польоти частково відновилися, до того ж відбувалися без значної кількості затримок.

Отже, після цього етапу збору даних маємо у розпорядженні інформацію про 110228 польотів здійснених українськими авіалініями Мотор Січ, МАУ, Windrose та Yan Air протягом квітня 2018 – квітня 2021.

1.2. Збір даних про погоду

Оскільки попередньо зібраних даних лише про аеропорти, часи вильоту, час затримок тощо недостатньо для побудови якісної моделі визначення затримок рейсів, було вирішено розширювати наш набір даних додатковою інформацією. Даними, які підходять для цього, могли бути погода, завантаженість аеропорту тощо.

Погода є одним з найсуттєвіших факторів впливу на затримки у вильотах та прильотах [10]. Звичайні джерела даних про історичну погоду не підходять для нашої мети, бо

а) рідко надають достатньо критичної для авіації інформації, як-от видимість, обледеніння чи пікові пориви вітру;

б) мають недолік у тому що важче встановлювати відповідність між метеорологічною станцією та конкретним аеропортом.

З цих причин було обрано працювати із спеціальним метеорогічним кодом – METAR[11]. Цей формат подання про погоду зазвичай використовується в авіації: саме в такому форматі погода записується пілотами у брифінгу перед польотом. Сертифіковані пілоти дронів також послуговуються саме цим форматом погоди. За сприяння ICAO починаючи приблизно з 1990-х років цей формат став високо стандартизованим та використовується у більшості країн світу. Відповідно до [12], він є найпоширенішим у світі форматом передачі спостережень про погоду. Приклад такого METAR - коду наведено на Рисунку 1.2.1.

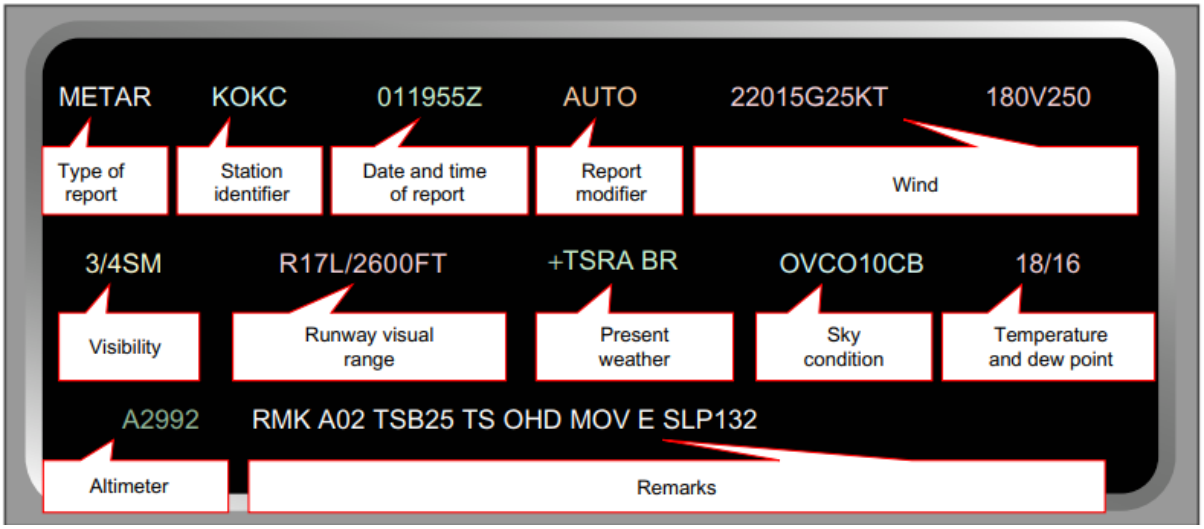


Рисунок 1.2.1– Приклад звіту METAR з аеропорту Оклахоми (код KOKC). Все, що передує “RMK” - це основна частина звіту, яка стандартна для всіх звітів, а далі йде розділ доповнень, які можуть варіюватися. Джерело – [13]

Інформація, яку кодує подібний звіт, є такою

1. **Type of report** – один із двох видів звіту. Це може бути або звичайний запис METAR, який записується кожні півгодини/годину, або SPECI, який ідентифікує різку зміну погоди і відповідно потребу в оновленому записі.
2. **Station identifier** – ICAO – код летовища.
3. **Date and time of report** – шестицифрова група, де перші дві цифри позначають число, а наступні чотири – конкретну дату. Z в кінці свідчить, що запис зроблений для часової зони UTC/Zulu.
4. **Modifier** – чи запис автоматичний чи скорегований вручну.
5. **Wind** – швидкість та напрям вітру.
6. **Visibility, Runway visual range** –видимість.
7. **Weather** – погодні явища.
8. **Sky condition**—хмарність.
9. **Temperature and dew point** – температура та точка роси.

10. **Altimeter** – налаштування альтиметру (пристороюю, призначеного для вимірювання висоти).
11. **Remarks** – додаткові коментарі які можуть містити дані вітру, зміну видимості, час початку та закінчення конкретного явища, інформацію про тиск тощо.

Повний опис кожного із пунктів, одиниці вимірів та інші деталі можна знайти в [13]. Ще більш точний опис полів, опис формування такого запису, розшифрування прикладів може бути знайдено в [11].

Початковим підходом було аналізувати METAR записи за допомогою регулярних виразів, що здається неважким враховуючи структурованість перших. Проте, через змінність від запису до запису як в довжині рядків, форматах символів та цілковитій варіативності поля з ремарками, довелося відмовитися від цього підходу. З’явилася потреба знайти дані METAR в уже розшифрованому вигляді, до того ж у часовому проміжку попередніх трьох років (відповідно до часового проміжку набору даних із польотами).

Проаналізувавши близько десятка існуючих API, ми помітили тенденцію, що зазвичай вони дають можливість надіслати лиш кілька тисяч безкоштовних запитів, чого в жодному випадку не вистачило б для заповнення нашої бази даних. Окрім того, таке джерело даних обмежувало б можливість розширення програми в подальшому. Тоді було знайдено джерело [14], яке надає можливість отримати історичні погодні дані METAR, із яких вже виділені всі основні показники, починаючи від 1928 року, до того ж, із можливістю автоматизувати збір цих самих даних (деталі наведемо нижче, у частині з описом практичної реалізації).

Цей архів збирається та підтримується Університетом штату Айова, США. Детальний опис самого архіву, його джерел даних та формування можна знайти в [15]. Недоліком цього джерела є те, що не всі поля доступні для більшості з нам потрібних аеропортів, як – от дані про рівень опадів,

хмарність та обледеніння. Проте, за браком кращих варіантів, було обрано працювати із цією базою даних.

Однією із переваг цієї бази даних є доступність записів про погоду для значного переліку світових аеропортів. Як видно з Рисунку 1.2.2, навіть для України перелік достатньо повний.

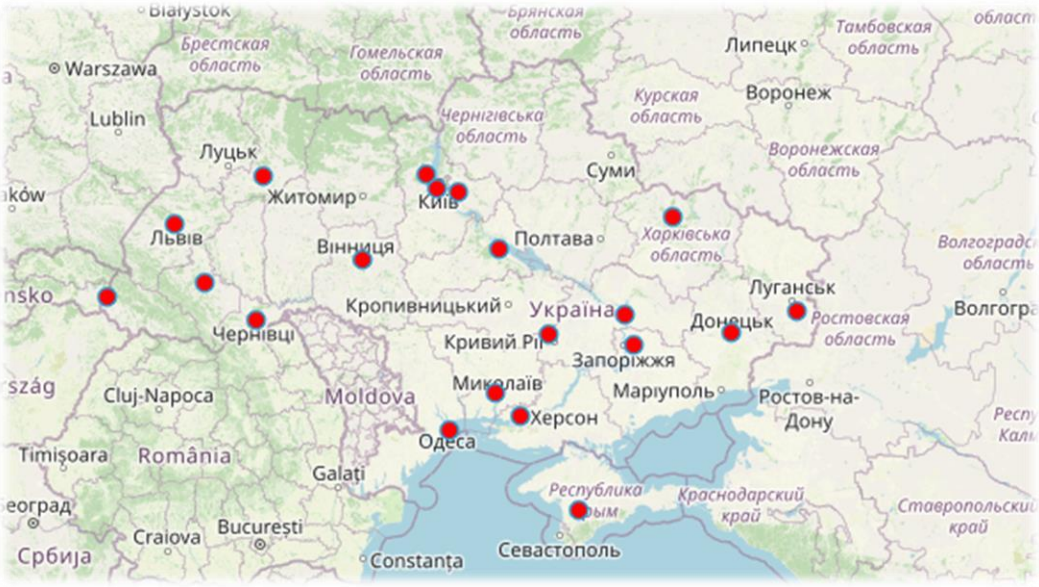


Рисунок 1.2.2 – Летовища України, для яких є доступ до історичної інформації про погоду. Самі аеропорти позначені на карті червоним кольором. Джерело [16]

Варто зазначити, що поточний стан погоди та відповідний METAR код для світових та зокрема українських аеродромів можна завжди бачити на сайті Національного управління океанічних і атмосферних досліджень [17].

Якщо перейти на основну сторінку архіву, звідки ми збирали дані про погоду [16], одразу можна побачити, що автори пропонують вже готові приклади для автоматизованого збору даних мовами програмування Python та R. У мові програмування R навіть створено повноцінний пакет для взаємодії з цим архівом. Проте, оскільки основною мовою програмування у нашій роботі є Python, для збору даних про погоду було обрано саме її.

Приклад, на який ми спиралися для написання коду нашої програми можна знайти в [18].

Як можна спостерігати, за надсилання запитів про погодні дані та демонстрації їх у браузері, а не завантаження на комп'ютер, відбувається формування файлу, всі дані про який можна зрозуміти з його URL. Як приклад, URL для доступних даних про погоду в Чернівецькому аеропорту в період 01.01.2021 – 25.04.2021 є таким *«https://mesonet.agron.iastate.edu/cgi-bin/request/asos.py?station=UKLN&data=all&year1=2021&month1=1&day1=1&year2=2021&month2=4&day2=25&tz=Etc%2FUTC&format=onlycomma&latlon=no&elev=no&missing=M&trace=T&direct=no&report_type=1&report_type=2»*. Уважно придивившись, можна помітити, що початкова і кінцева дати, значення для пропущених даних тощо уже записані в самій адресі. Ця ідея і використовується для формування та завантаження файлів з даними.

В самій програмі спочатку ми формуємо список з усіх унікальних аеропортів, які зустрічалися в пунктах вильотів та призначень. Далі, передаємо цей список у функцію, яка для кожного із аеропортів зі списку формує файл формату .csv із усіма доступними в архіві даними про погоду за останні три роки. Ми навмисне скачуємо усі доступні дані, щоб програма була легко розширюваною. Себто, за потреби заповнити додаткові дані про погоду із цього часового проміжку, нам не доведеться наново надсилати запити на сервер, а зможемо просто знайти інформацію серед уже завантажених даних.

Після запуску програми зі збору даних про погоду, маємо папку із файлами для кожного з аеропортів. Розмір їх варіюється від 68Мб до 2 Мб, в середньому – 11Мб; кількість записів у кожному – від 340 000 (нотування погоди щоп'ять хвилин) до 7000 (нотування погоди щогодини, проте не для всіх років), в середньому - 55000. Приклади того, як виглядають перші

рядки файлу із записами про погоду для Міжнародного аеропорту «Бориспіль» (код ICAO: UKBB) наведені на Рисунку 1.2.3.

A	B	C	D	E	F	G	H	I	J	K
station	valid	lon	lat	elevation	tmpf	dwpf	relh	drct	sknt	p01i
UKBB	3/1/2018 0:00	30.9667	50.3333	119	12.2	10.4	92.32	340	15.55	0
UKBB	3/1/2018 0:30	30.9667	50.3333	119	12.2	10.4	92.32	350	15.55	0
UKBB	3/1/2018 0:43	30.9667	50.3333	119	12.2	10.4	92.32	360	15.55	0

Рисунок 1.2.3 – Записані дані про погоду в «Борисполі» (частина 1)

L	M	N	O	P	Q	R	S	T	U	V
alti	mslp	vsby	gust	skyc1	skyc2	skyc3	skyc4	skyl1	skyl2	skyl3
29.91	M	1.06	M	BKN	OVC	M	M	700	1000	M
29.94	M	0.93	M	BKN	OVC	M	M	700	1000	M
29.94	M	0.81	M	BKN	OVC	M	M	600	900	M

Рисунок 1.2.3 – Записані дані про погоду в «Борисполі» (частина 2)

W	X	Y	Z	AA	AB	AC	AD	AE	AF
skyl4	wxcodes	ice_accret	ice_accret	ice_accret	peak_winc	peak_winc	peak_winc	feel	metar
M	SN BLSN	M	M	M	M	M	M	-5.17	UKBB 010000Z 34008MPS 1700 SN BLSN BKN007 OVC010 M1
M	SN BLSN	M	M	M	M	M	M	-5.17	UKBB 010030Z 35008MPS 1500 1300SW R36R/1800D R36L/1
M	SN BLSN	M	M	M	M	M	M	-5.17	UKBB 010043Z 36008MPS 1300 R36R/1700D R36L/1500D SN
M	SN BLSN	M	M	M	M	M	M	-7.51	UKBB 010100Z 36008MPS 1300 R36R/1600D R36L/1200D SN

Рисунок 1.2.3 – Записані дані про погоду в «Борисполі» (частина 3)

Основні колонки із даних, наведених вище, такі: «station» – ICAO код аеропорту; «valid» – час запису в форматі UTC; «lon, lat, elevation» – характеристики розміщення аеропорту (широта довгота та висота над рівнем моря); «vsby» – видимість. Значення та детальніший опис кожної із колонок можна знайти в [16].

1.3. Формування фінального набору даних

На цьому етапі у нас є файл із завантаженими польотами у форматі .csv та окремо файли з погодою для кожного із аеропортів у тому ж форматі. Наше завдання – для кожного із польотів додатково записати погоду у запланований час вильоту та прильоту. Здавалося б, все просто – потрібно взяти у файлі з погодою час вильоту/прильоту і перенести дані. Проте, тут з’являються такі дві проблеми:

- Для даних із погодою всі часові позначки зведені до UTC/Zulu часової зони, а у датасеті з польотами час записаний у часовій зоні країни, звідки рейс стартував/де приземлявся. Тобто, якщо до прикладу виліт був з Німеччини, а посадка в Україні, то навіть для одного рейсу часи вильоту та прильоту будуть записані у різних часових зонах.
- Ми не можемо просто брати час вильоту, а маємо у файлі шукати час, найближчий до нашого часу вильоту, що може мати немалу обчислювальну складність залежно від реалізації алгоритму пошуку.

Для того, щоб записати всі значення часових атрибутів в стандартній часовій зоні, нам потрібно знати у якій часовій зоні знаходиться кожен із аеропортів. Для цього ми скористалися базою даних Open Travel Data [19]. У ній наявні часові зони майже для всіх аеропортів із нашого набору даних. Часові зони кількох аеропортів яких не вистачало, було додано вручну. Тепер, маючи таблицю із часовою зоною кожного із аеропортів, у нашому наборі даних ми додаємо нові колонки, в яких зберігаємо години запланованого вильоту та прильоту у часовій зоні UTC/Zulu.

У нашому наборі даних унікальні ідентифікатори аеропортів це IATA коди, оскільки саме ці коди зазначалися у початковому джерелі [9]. Проте, у даних про погоду кожен аеропорт ідентифікується його ICAO кодом. Для додавання відповідного ICAO коду для кожного з аеропортів до нашого набору даних ми скористалися базою Open Flights [20].

Запис погоди для кожного з рейсів відбувався таким чином: відкривався файл із записами про погоду потрібного аеропорту; у файлі знаходився час, який найближчий до потрібного нам (зазвичай відхилення становить не більш ніж півгодини); показники погоди записувалися у файл із рейсами. Через велику кількість записів про погоду для кожного з аеропортів, щоб заповнити дані для всіх рейсів, програма, що виконувала перераховані кроки, працювала близько вісімдесяти годин.

Після цього кроку маємо повністю сформований початковий набір даних. На Рисунку 1.3.1а) візуалізовано ці дані таким чином: кружечками позначені аеропорти, розмір круга відповідає кількості рейсів, що вилітали з цього аеропорту, колір відповідає відсотку затриманих із них – що ближче колір до зеленого, то більший відсоток рейсів вилетіли із затримкою.

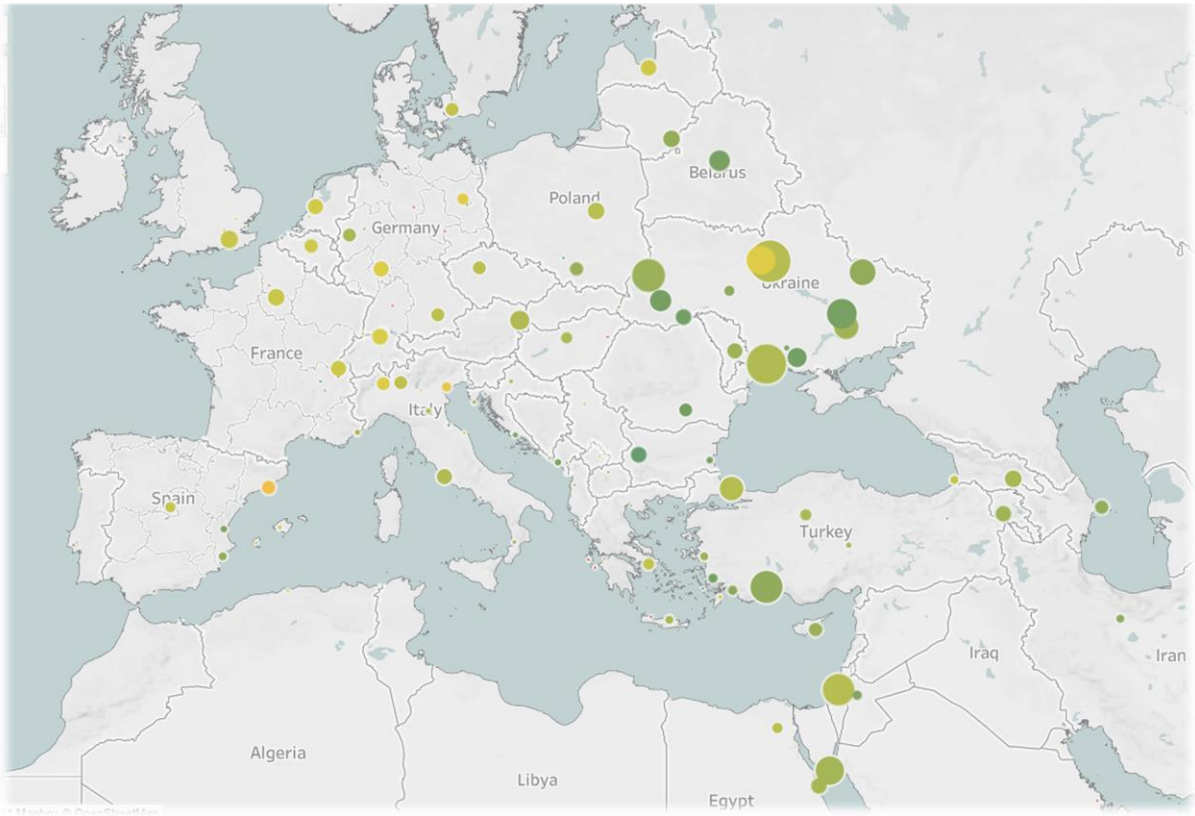


Рисунок 1.3.1а) – Аеропорти із нашого набору даних, розмір показує кількість вильотів, колір показує відсоток затриманих рейсів

Водночас, на Рисунку 1.3.1б) також позначено аеропорти та кількості вильотів з них, проте тут колір кодує середню затримку рейсу. Як видно з Рисунку 1.3.1, в наборі даних маємо багато рейсів які були затримані, проте затримка не становила довгий час. Щоб чіткіше побачити це, можна звернути увагу на аеропорти «Бориспіль» та «Київ Жуляни» на першій та другій частинах Рисунку 1.3.1. Як видно з першого рисунку, в обох із них, а особливо в Жулянах, відсоток затриманих рейсів високий порівняно з іншими аеропортами України. Проте з другої частини малюнку видно, що

середня тривалість затримки не перевищує середньої по Україні (інші аеропорти держави мають приблизно такі ж кольори). Тобто, для цих двох аеропортів затримок багато, проте в середньому вони доволі короткі.

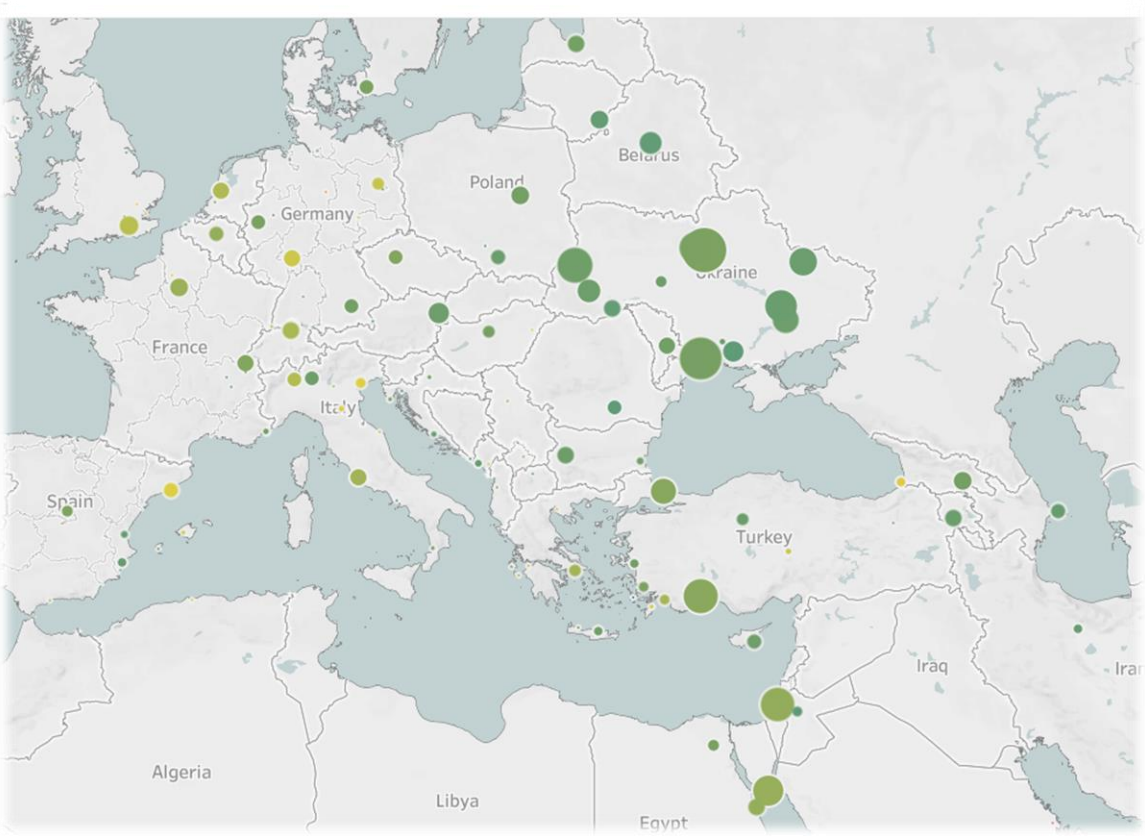


Рисунок 1.3.1б) – Аеропорти із нашого набору даних, розмір показує кількість вильотів, колір – середню затримку рейсів

РОЗДІЛ 2. РОЗВІДУВАЛЬНИЙ АНАЛІЗ ТА ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

2.1. Етапи попередньої обробки даних

Етап що слідує після збору даних – розвідувальний аналіз та попередня обробка даних. Цей підхід включає в себе початковий аналіз даних із поєднанням як візуальних так і невізуальних методів. Метою є поглиблення розуміння наявних даних та підготовка їх для подальшого використання в моделях.

Самі кроки розвідувального аналізу даних можуть виділятися різним чином, ми зупинимося на такому переліку:

1. Описовий аналіз
    - Розуміння наявних змінних
    - Перевірка якості та релевантності даних
    - Перевірка центральних тенденцій для кожної із змінних
2. Попередня обробка даних

    Цей крок потрібен, оскільки на практиці набори даних як правило містять багато так званого «шуму», себто небажаної інформації, яка може впливати на результат. Він включає в себе такі кроки:

    - Визначення та корекція так званих «bad values», до яких належать пропущені дані; дані, тип яких зазначено неправильно тощо.
    - Визначення та корекція аномальних даних, викидів, які лежать поза очікуваним діапазоном. Це можуть бути як неправильно введені дані, так і дані про виняткові ситуації які зустрічаються надзвичайно рідко та потребують окремої обробки. До основних способів їх виявлення належать візуалізація розподілу даних використовуючи зокрема графік boxplot, обчислення

відповідного zscore. До основних способів усунення викидів належать їх видалення; зміна їх значень так, щоб вони перестали бути викидами.

- Виявлення та видалення непотрібних даних. До них належать повторювані записи, які рекомендується видаляти; атрибути, у яких значення або однакові для кожного із записів, або різні в усіх (ідентифікатори), оскільки вони не несуть корисної інформації; атрибути, значення яких відсутні для більш ніж 25-30% усіх записів, оскільки вони, навіть якщо заповнити пропущені записи середніми або медіанними значеннями, не відображатимуть коректну інформацію.

3. Візуалізація даних

Це ще один важливий крок в розумінні датасету. Він зручний, оскільки людина значно краще сприймає та виявляє закономірності у візуальних зображеннях, а не цифрових таблицях. Окрім того, хороша візуалізація дає можливість швидко оцінити структуру та зв’язки навіть у великій кількості даних. Основні пункти що включає аналіз даних за допомогою візуалізації такі:

- Одновимірний аналіз (гістограми, коробкові діаграми, скрипкові діаграми)
- Двовимірний аналіз (діаграми розсіювання, графіки кореляцій, теплові карти)
- Багатовимірний аналіз

Окрім того, зауважимо, що інколи візуалізація може виявити і приховані паттерни в даних, які неможливо помітити просто аналізуючи стандартні описові статистики. Прикладом цього є

як відомий Квартет Анскомбе, так і менш відомий «The Datasaurus Dozen», зображений на Рисунку 2.1.1 [21].

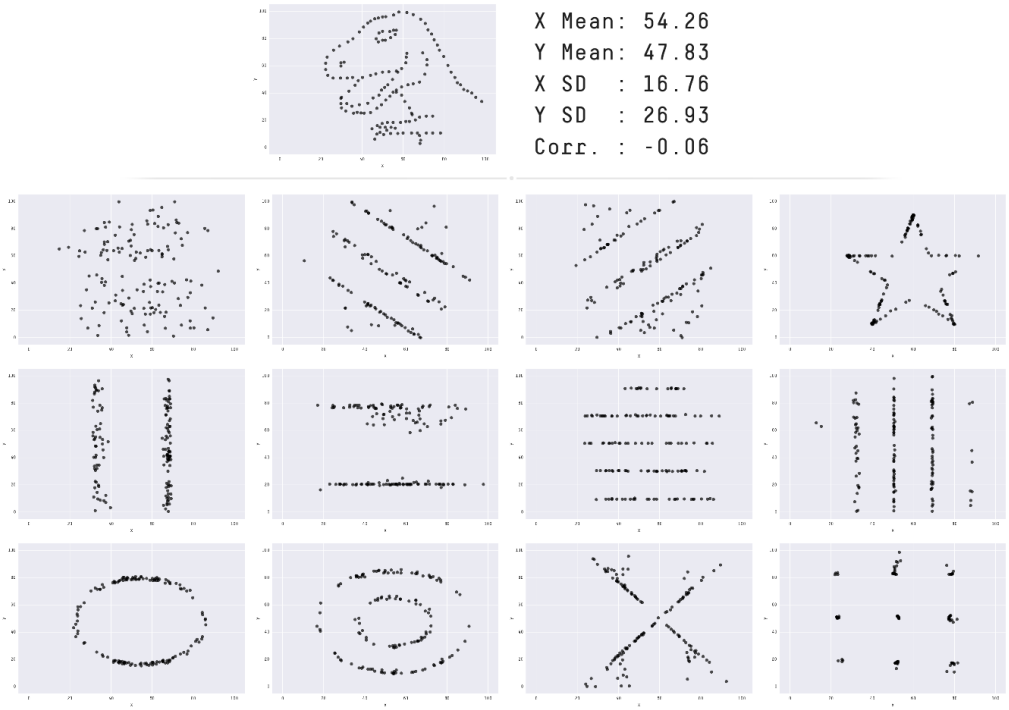


Рисунок 2.1.1 – Візуальне зображення тринадцяти наборів даних, кожен із яких має однакові до двох знаків після коми середні значення, стандартні відхилення по обом координатам та кореляцію Пірсона, при цьому графіки різко різняться.

4. Зміна діапазону значень

   Діапазони атрибутів часто суттєво відрізняються. Це може впливати на результат усіх алгоритмів, які враховують відстань між точками у певних обчисленнях (PCA, нейронні мережі тощо). За такої ситуації атрибути, які мають більші діапазони, матимуть відповідно більші ваги і суттєвіше впливатимуть на результат, що не є бажаним. Для уникнення цього змінюється масштаб атрибутів. Найбільш часто використовуваними техніками є такі:

- Z Score – зміна даних таким чином, що середнє значення стає нулем, а стандартне відхилення одиницею; більш застосовні до методів із врахуванням ваг.
- Min-max scaling – зміна даних таким чином, що їхній діпазон стає від нуля до одиниці.
- Log, Exponential, Cube and Square Root transformations – зміни масштабу даних, котрі зменшують міру асиметрії розподілу.

5. Перекодування даних

   Дані, що мають нечисловий формат (такі як стать, країни, категорії тощо) потрібно трансформовувати у числовий. Цей крок необхідний, оскільки всі математичні моделі на вхід приймають саме числові значення. Для моделей машинного навчання найбільш широко використовуваними є два підходи до подібної трансформації:

   - One Hot Encoding – кожен рівень категорії перетворюється на окремий стовпець у датасеті, котрий може набувати лише булевих значень, залежно від належності до цього рівня категорії. Рекомендується для номінальних даних та у випадку, якщо рівнів категорії не дуже багато (приблизно до 25ти).
   - Label Encoding – кожен рівень категорії кодується конкретним числом і стовпець з даними переписується відповідно до закодованих значень. Підходить для порядкових даних, оскільки числа зберігають впорядкованість рівнів категорії.

За обробки нашого набору даних було першочергово здійснено такі кроки:

- Видалення записів, для яких були відсутні справжній час вильоту або прильоту.
- Для даних про погоду заміна значення «M» на нульові значення. Саме так позначалася відсутність даних у архіві із даними погоди.
- Після попереднього кроку було виявлено, що деякі стовпці із даними про погоду мають понад 30% нульових значень, а деякі і понад 90%. Такі стовпці не несуть корисної інформації, тому їх було видалено.

```
27  alti                98590 non-null  object
28  mslp                115 non-null    object
29  vsby                98621 non-null  object
30  gust                1456 non-null   object
31  skyc1               50645 non-null  object
32  skyc2               13771 non-null  object
33  skyc3               1611 non-null   object
34  skyc4               50 non-null     object
35  skyl1               45592 non-null  object
36  skyl2               13722 non-null  object
37  skyl3               1607 non-null   object
38  skyl4               48 non-null     object
39  wxcodes             15253 non-null  object
40  ice_accretion_1hr   0 non-null      float64
41  ice_accretion_3hr   0 non-null      float64
42  ice_accretion_6hr   0 non-null      float64
```

Рисунок 2.1.2 – кількість ненульових значень у деяких стовпцях із даними про погоду

- Видалення стовпців які вже не потрібні, як-от із часами вильоту/прильоту в стандартній часовій зоні, ICAO коди аеропортів.
- Видалення стовпця із номером рейсу, оскільки наявність номеру рейсу у вхідних даних для моделей зробила б їх непридатними для рейсів, котрих немає в нашому наборі даних
- Під час перевірки значень стовпця із роком виробництва, було виявлено, що значна частина значень некоректні (Рисунок 2.1.3). Неправильні значення з’явилися через нестандартизованість записів у початковому джерелі даних. Окрім того, було виявлено, що

зазначеним роком виробництва багатьох літаків був 2021, що не відповідає дійсності. Некоректні роки виробництва виправлені вручну, використовуючи ресурс planespotters.net як джерело даних. Після цього у наборі даних створено стовпець «age», в якому записано вік для кожного з літаків.

```
all.production.unique()
```

```
array(['2003', '1971', '1992', '1980', '1973', 'v-89', 'r-90', 'r-01',
       '2002', 'r-03', 'r-00', 'v-11', '2012', 'n-13', 'l-13', '2007',
       'r-93', 'r-97', 'p-91', 'n-98', 'y-05', '1975', 'b-12', 'r-12',
       'y-99', 't-13', 'y-09', 'c-09', 'n-00', 'b-17', 't-06', 'v-06',
       'p-14', 'l-15', 'l-17', 't-17', 'v-17', 'r-05', 'n-06', 'v-02',
       'g-02'], dtype=object)
```

Рисунок 2.1.3 – початкові значення років виробництва літаків

- Частково попередня обробка відбувалася ще на етапі формування самого набору даних із завантажених файлів. До таких кроків належали: видалення зайвих пробілів у рядках, видалення символів переносів рядків, заміна дати виробництва із вигляду «AGE (1971)» на «1971», видалення непотрібних дужок, ком, слова «Landed» із запису про реальний час посадки та подібна дрібна робота із очищення тексту.

На Рисунку 2.1.4 наведено описові статистики змінних які залишилися після очищення даних. Дописування «_a» означає, що це значення погоди для аеропорту призначення (a від arrival). Як видно, тривалості затримок вильотів та прильотів мають розподіл сильно перекошений вправо. Окрім того, середнє значення затримки прильотів значно нижче, аніж затримки вильотів. Це спричинено значною кількістю таких ситуацій, коли рейс вилітав із невеликою затримкою і встигав прилетіти вчасно, компенсувавши незначну затримку протягом польоту. Можна помітити, що мінімальне значення змінної elevation, яка вказує на висоту летовища над рівнем моря, є від’ємною. Це помилкове значення

лише для одного аеропорту в Баку, яке було виправлено вручну. Окрім того, можемо сказати що у більшості рейсів видимість була добра (значення 6.21 відповідає хорошій видимості); розподіл віку доволі симетричний (середнє та медіанне значення близькі між собою).

У деяких змінних, що характеризують погоду, була незначна кількість нульових значень, які ми заповнили медіанними. Після цього не залишилося жодних нульових значень в наборі даних.

	count	mean	std	min	25%	50%	75%	max
departure_delay	98515.0	23.123920	47.836432	0.0000	6.0000	14.0000	24.0000	1427.0000
arrival_delay	98515.0	14.199168	80.622276	0.0000	0.0000	0.0000	6.0000	1439.0000
lon	98515.0	28.476697	12.384842	-150.0261	28.0333	30.9667	30.9667	121.7667
lat	98515.0	46.838017	7.509627	-8.7482	46.4333	50.3333	50.3333	67.7010
elevation	98515.0	144.371294	144.340295	-5.0000	62.0000	119.0000	120.4700	1758.0000
tmpf	98515.0	56.854471	18.868558	-13.0000	42.8000	59.0000	71.6000	109.4000
dwpf	98515.0	45.573721	14.430801	-20.2000	35.6000	46.4000	55.4000	86.0000
relh	98515.0	70.743453	21.654773	6.7700	54.6000	74.4400	88.0000	100.0000
drct	98515.0	185.123342	105.958585	0.0000	100.0000	190.0000	270.0000	360.0000
sknt	98515.0	7.112119	4.159686	0.0000	3.8900	6.0000	9.7200	40.0000
alti	98515.0	29.984901	0.243248	28.5000	29.8300	29.9700	30.1200	50.0000
vsby	98515.0	5.710123	1.377295	0.0000	6.2100	6.2100	6.2100	15.0000
lon_a	98515.0	28.660834	12.488209	-150.0261	28.0333	30.9667	30.9667	121.7667
lat_a	98515.0	46.642918	7.680025	-8.7482	46.4333	50.3333	50.3333	67.7010
elevation_a	98515.0	145.570119	145.773100	-5.0000	62.0000	119.0000	122.0000	1758.0000
tmpf_a	98515.0	57.782723	18.985253	-13.0000	42.8000	59.0000	73.4000	113.0000
dwpf_a	98515.0	45.675842	14.293793	-50.8000	35.6000	46.4000	55.4000	87.8000
relh_a	98515.0	69.096432	22.221471	6.1500	51.7200	72.3800	87.6500	100.0000
drct_a	98515.0	186.440045	106.984006	0.0000	100.0000	190.0000	280.0000	360.0000
sknt_a	98515.0	7.239982	4.205285	0.0000	3.8900	6.0000	9.7200	45.0000
alti_a	98515.0	29.982654	0.240506	28.4700	29.8300	29.9700	30.1200	50.0000
vsby_a	98515.0	5.754805	1.314349	0.0000	6.2100	6.2100	6.2100	15.0000
Age	98515.0	15.429792	10.148619	4.0000	8.0000	14.0000	20.0000	59.0000

Рисунок 2.1.4 – Описові статистики змінних

На Рисунку 2.1.5 зображено розподіли та графіки «ящик з вусами» для тривалостей затримок вильотів та прильотів. Оскільки в обох величин багато викидів в правій частині (великих значень), то на графіку «ящик з вусами» ми не показували викиди, інакше «ящик» стиснувся б майже в одну лінію. На Рисунку 2.1.6 наведено аналогічні графіки для віку літаків, вік літаків, що здійснили 75% з польотів не перевищує 20 років. З Рисунку 2.1.7 видно, що більшість польотів здійснено між аеропортами, що знаходяться на висоті до 150м над рівнем моря. Інколи зустрічаються аеропорти, що розташовані відчутно вище, зокрема крайній викид справа відповідає аеропорту Erzurum в Туреччині, висота якого 1757м над рівнем моря.

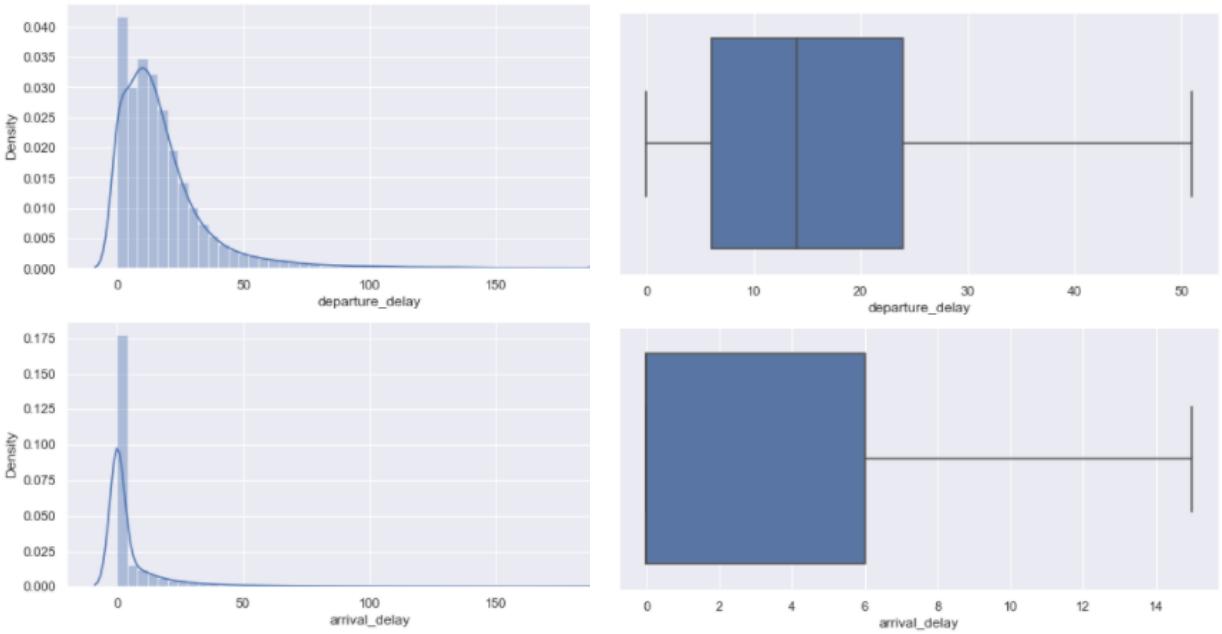


Рисунок 2.1.5 – розподіли та графіки «ящик з вусами» для тривалостей затримок вильотів та прильотів

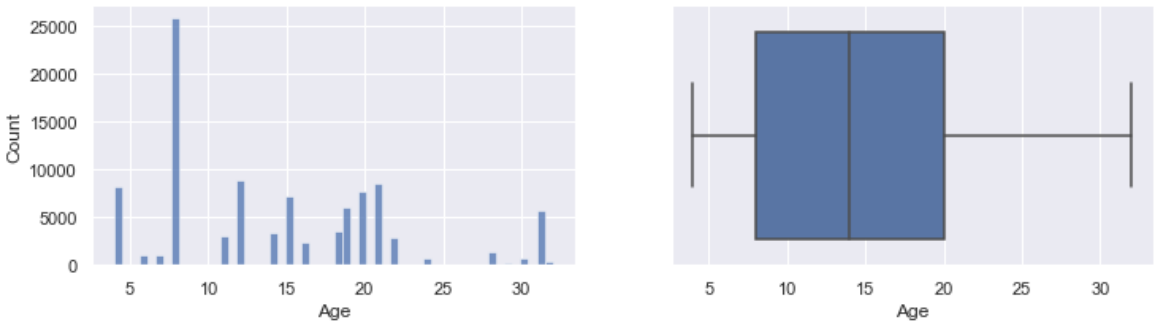


Рисунок 2.1.6 – Розподіл віку літаків

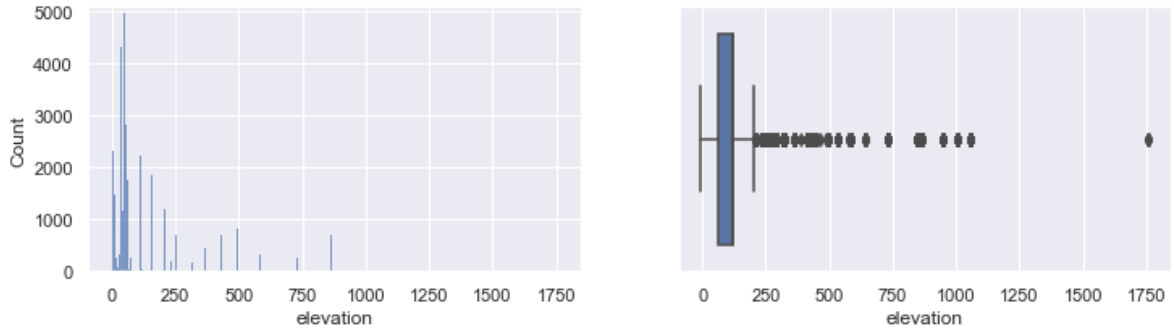


Рисунок 2.1.7 – Розподіл польотів за висотою аеропортів над рівнем моря

На Рисунках 2.1.8 – 2.1.11 зображено залежність затримок вильотів від показників погоди в аеропортах вильоту та затримок прильоту в аеропортах призначення. На кожному із Рисунків значення 0 позначає затримки до 60 хвилин, 1 – інтервал затримок 60-120 хвилин, 2 – затримки понад 120 хвилин. Інтервали 1 та 2 мають однаковий розподіл тиску, інтервал 0 відрізняється від них. Щодо вітру, його швидкість цілком може впливати на затримки, проте напрям не видається суттєвою ознакою: всі медіанні значення збігаються. Рисунок 2.1.11 зображує затримки залежно від температури (за шкалою Фаренгейта). Суттєвих відмінностей у температурі для різних груп затримок немає.

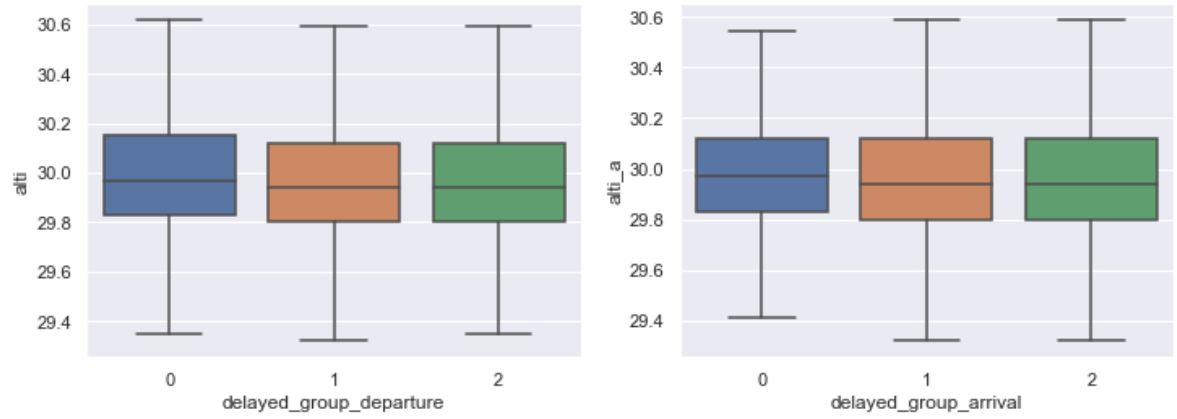


Рисунок 2.1.8 – Затримки в залежності від тиску

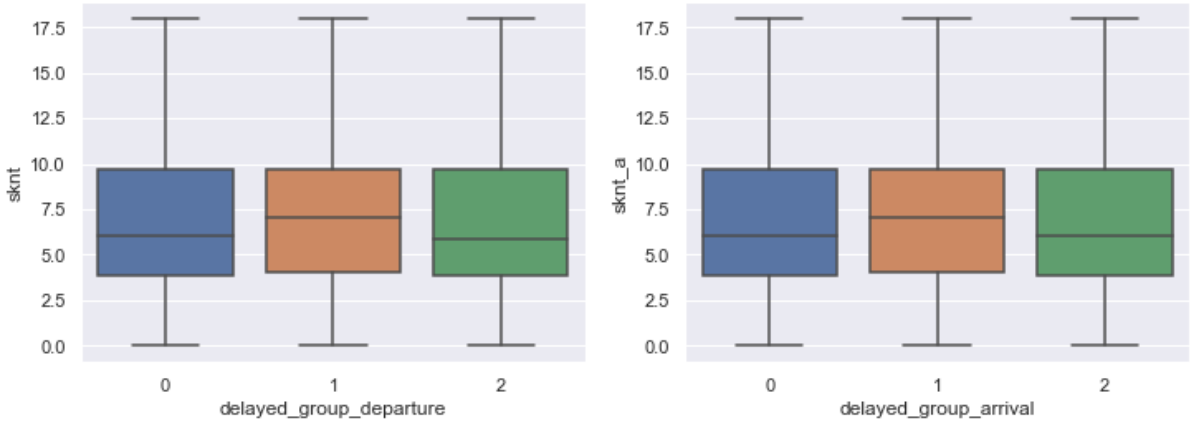


Рисунок 2.1.9 – Затримки в залежності від швидкості вітру

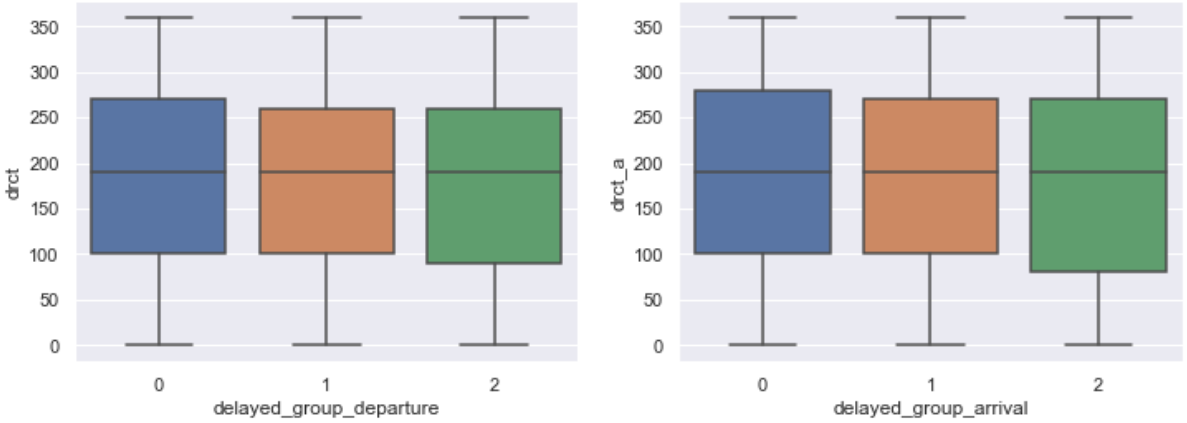


Рисунок 2.1.10 – Затримки в залежності від напряму вітру

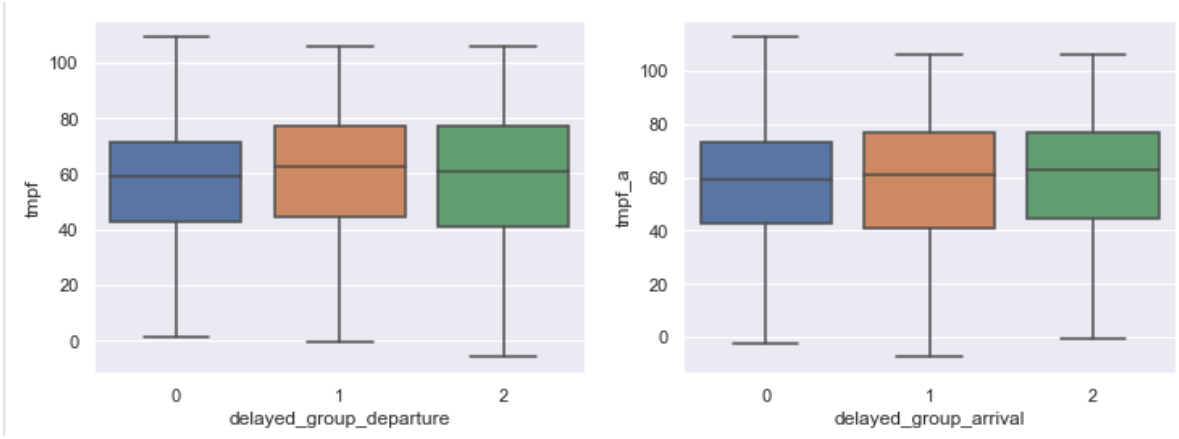


Рисунок 2.1.11 – Затримки в залежності від температури

З Рисунків 2.1.12 та 2.1.13 можемо побачити кількість затримок вильотів та прильотів для кожної з авіаліній. Знову бачимо тенденцію, що затримок прильотів менше у кожної з авіаліній. Окрім того, видно, що

найменше затримок у відсотковому співвідношенні спостерігається у авіалінії Windrose.

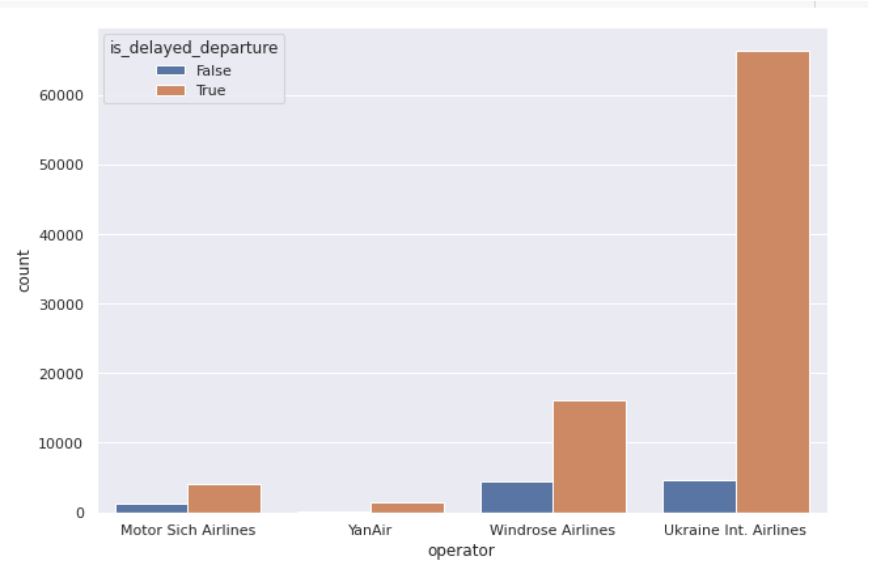


Рисунок 2.1.12 – Кількість затримок вильотів залежно від авіалінії

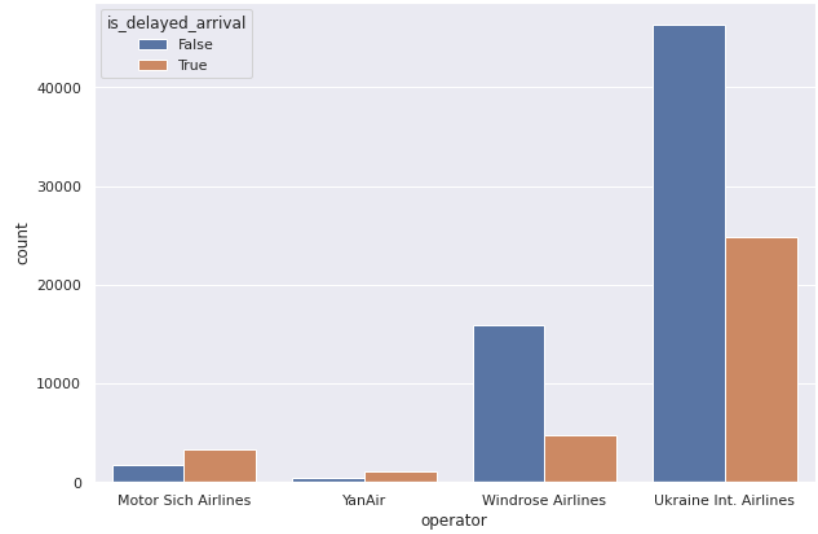


Рисунок 2.1.13 – Кількість затримок прильотів залежно від авіалінії

На Рисунку 2.1.14 наведені кореляції між змінними у нашому наборі даних. Більшість змінних не корельовані між собою сильно. Найбільш суттєвою кореляцією виділяються змінні температури та точки роси, що є цілком очікувано. Окрім того, можна побачити що температури в точках відправлення та призначення також достатньо корельовані між собою.

Можливо, це спричинено тим, що багато із рейсів виконувалося в межах України, де часто температура між містами відрізняється не надто сильно. Окрім того, перельоти досліджено в північній півкулі, в країнах якої пора року однакова і немає радикальної різниці в температурі. Загалом, сильної мультиколінеарності не спостерігається.

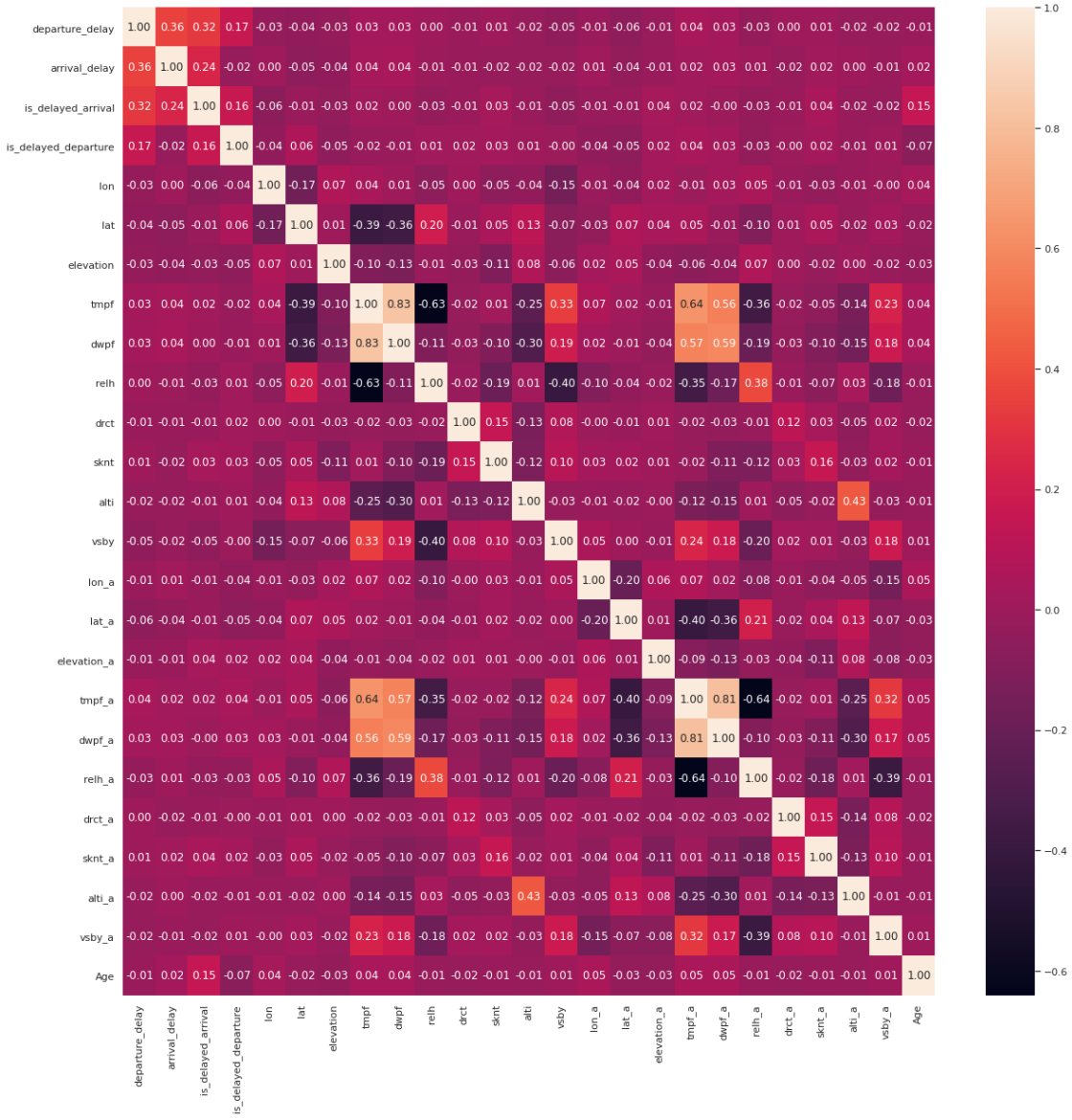


Рисунок 2.1.14 – Кореляції між змінними

2.2. Балансування даних

У нашому наборі даних ми маємо справу із незбалансованими даними. Дисбаланс зустрічається може бути таких типів:

- Недостатня представленість класу в одній або більше із важливих незалежних змінних
- Недостатня представленість одного із класів у залежній змінній, яку і намагаємося передбачити

Незбалансовані дані зустрічаються доволі часто у наборах даних із реального світу, і вони становлять проблему для більшості алгоритмів машинного навчання, зокрема тих, що використовуються для класифікації. Переважно збалансованість даних є одним з припущень цих алгоритмів. Прикладами таких моделей є нейронні мережі, Random forest та деякі інші. (Існують і моделі які менш чутливі до незбалансованих даних, як-от логістична регресія або SVM. За побудови кожної окремої моделі потрібно перевіряти її припущення та коригувати свої дані відповідно до них).

Дисбаланс даних може призводити до побудови моделей які мають низькі показники прогнозування, особливо для класу, представленому в меншості, який часто є більш важливим (зазвичай як приклад наводяться класи здоровий/хворий, де не пропустити хворобу є важливішим завданням, але таких даних менше). Це відбувається тому, що під час тренування модель з’ясовує, що переважно зустрічається клас більшості і стає упередженою щодо розподілу класів. Такі моделі можуть показувати високу метрику точності, проте інші показники якості мати низькими.

До прикладу, на Рисунку 2.2.1 показано метрики якості моделі Random forest на тренувальних та тестувальних даних датасету із затримками польотів. Як бачимо, незважаючи на доволі високу точність на обох наборах даних (83%), інша суттєва метрика якості – f1-score є низькою. Це стається через недостатнє розпізнавання моделлю випадків, коли рейс було затримано, у нашому випадку саме належність до цього класу представлена у меншій кількості.

	precision	recall	f1-score	support
False	0.83	0.99	0.90	60446
True	0.68	0.12	0.20	13440
accuracy			0.83	73886
macro avg	0.76	0.55	0.55	73886
weighted avg	0.81	0.83	0.78	73886

	precision	recall	f1-score	support
False	0.83	0.99	0.90	20167
True	0.65	0.11	0.19	4462
accuracy			0.83	24629
macro avg	0.74	0.55	0.55	24629
weighted avg	0.80	0.83	0.78	24629

Рисунок 2.2.1 – Метрики якості моделі Random forest на тренувальних та тестувальних даних відповідно

Отже, той набір даних що маємо зараз не задовольняє припущення моделі і для побудови більш точної моделі потрібно мати більш збалансовані по залежній змінній дані.

Для компенсації присутнього у наборі дисбалансу даних застосовуються over-sampling або under-sampling – методи, коли добираються додаткові спостереження класу меншості або ігноруються частина спостережень класу більшості відповідно.

До найвідоміших технік under-sampling належать такі:

- Випадковий вибір

  Із спостережень які належать до класу більшості випадковим чином вибираються ті, які потрапляють в новий набір даних. Обирається така кількість спостережень, яка рівна кількості спостережень у класі меншості. Таким чином, отримується новий збалансований набір даних.

- Центроїди кластерів

На незалежних даних створюються кластери. Далі, кожен кластер заміняється його центроїдом. Таким чином зменшується кількість спостережень що належать до класу більшості.

Більше деталей та варіацій методів можна знайти у [22].

Загалом, under-sampling використовується лише в тих випадках, коли дослідник має справді величезний набір даних, який в будь-якому випадку мав би бути зменшений через обчислювальну складність його опрацювання. У інших випадках, для того, щоб не втрачати корисної інформації, використовується over-sampling, до основних технік якого належать:

- Випадковий вибір

  Ця техніка полягає у дублюванні випадкових спостережень, які належать до класу меншості. Випадкові спостереження дублюються допоки не зрівняється кількість спостережень в обох (або більше) класах. Ця техніка добре підходить для компенсації незначного дисбалансу, і гірше – для дуже суттєвого, як-от 99.5% належностей одного класу і 0,5% до іншого.

- SMOTE: Synthetic Minority Over-sampling Technique

  Техніка полягає у створенні нових спостережень, які належали б до класу меншості і мали відповідні значення незалежних величин. Це досягається такими кроками:

  1) Випадковим чином обирається спостереження, яке належить до класу меншості та кілька його найближчих сусідів (кількість сусідів – параметр методу).
  2) Із обраних сусідів обирається випадковий та проводиться лінія між ними.

3) На проведеній лінії обирається випадкова точка, яка і буде новим спостереженням

Таким чином створюються нові спостереження що належать до класу меншості аж допоки набір даних не стане збалансованим. Стаття із описом цієї техніки вперше була опублікована в 2002 році [23].

На Рисунку 2.2.2 можемо побачити візуалізацію роботи цього алгоритму.

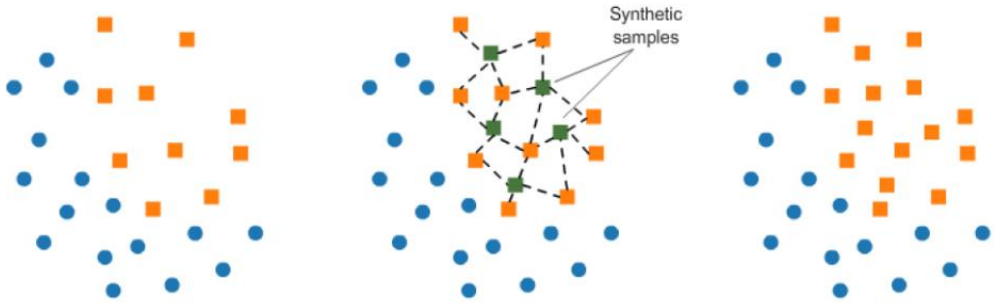


Рисунок 2.2.2 – Приклад роботи SMOTE, створення штучних спостережень для збалансування набору даних

Оскільки за використання under-sampling втрачається частина набору даних і відповідно інформації, цей метод є небажаним для нас, бо сформований набір даних і так не є дуже великим. За ігнорування значної його частини може виявитися, що нового набору даних буде просто недостатньо для якісного тренування моделі. Тому, ми вирішили скористатися методом over-sampling, зокрема технікою SMOTE, котра створює нові спостереження, які належать до класу меншості. Ця техніка була застосована окремо до кожної із залежних величин і відповідно створено кілька нових наборів даних різного розміру.

РОЗДІЛ 3. ПОБУДОВА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ТА ЇХ ПОРІВНЯННЯ

Ми поділили дані на 3 категорії: затримки до 60 хвилин, затримки від 60 до 120 хвилин, затримки понад 120 хвилин. За такого поділу немає сенсу розрізняти затримки вильоту та прильоту: якщо виліт затриманий більш аніж на годину, то і приліт буде з суттєвою затримкою. Як було видно з даних раніше, протягом польоту можна нівелювати лиш короткі затримки вильоту. Тому надалі будуватимемо передбачення лише для затримок вильоту. Нашою метою було побудувати та порівняти моделі, які класифікували б інтервал затримки, тобто виходом моделі було б «затримки не буде або буде коротка», «буде затримка середньої тривалості», «буде довга затримка». Проте, у ході роботи було помічено, що точність такого передбачення у багатьох моделях низька. Тим не менш, багато моделей доволі точно (близько 80%) можуть виконувати бінарну класифікацію, тобто давати відповідь на питання «Чи буде затримка тривалістю хоча б годину?». Тому, для кожної із моделей ми наводитимемо точність як для передбачення інтервалів затримки до години/від години до двох/більше двох годин так і точність відповіді на питання «чи буде затримка тривалістю хоча б 60 хвилин».

Метрики точності, якими ми користуватимемося для оцінки якості моделей:

- Точність – кількість правильно передбачених спостережень, поділена на загальну кількість зроблених прогнозів.
- Влучність та повнота (англ. Precision and recall, Рисунок 3.1).
- F1 score – середнє гармонійне влучності та повноти.
- Площа під кривою (англ. AUC) – площа, обмежена ROC-кривою. Ця крива, також відома як крива похибок – графік, що відображає співвідношення між часткою об'єктів від загальної кількості носіїв ознаки, правильно класифікованих до загальної

кількості об'єктів, що не несуть ознаки, помилково класифікованих, як такі, що мають ознаку.

- Матриця невідповідностей – матриця, у якій рядки представляють об’єкти спрогнозованого класу, тоді як кожен зі стовпців представляє об’єкти справжнього класу.

Чим ближче кожна із наведених числових метрик до 1, тим вища якість моделі. Розмір тестової вибірки становив 25% для всіх моделей.

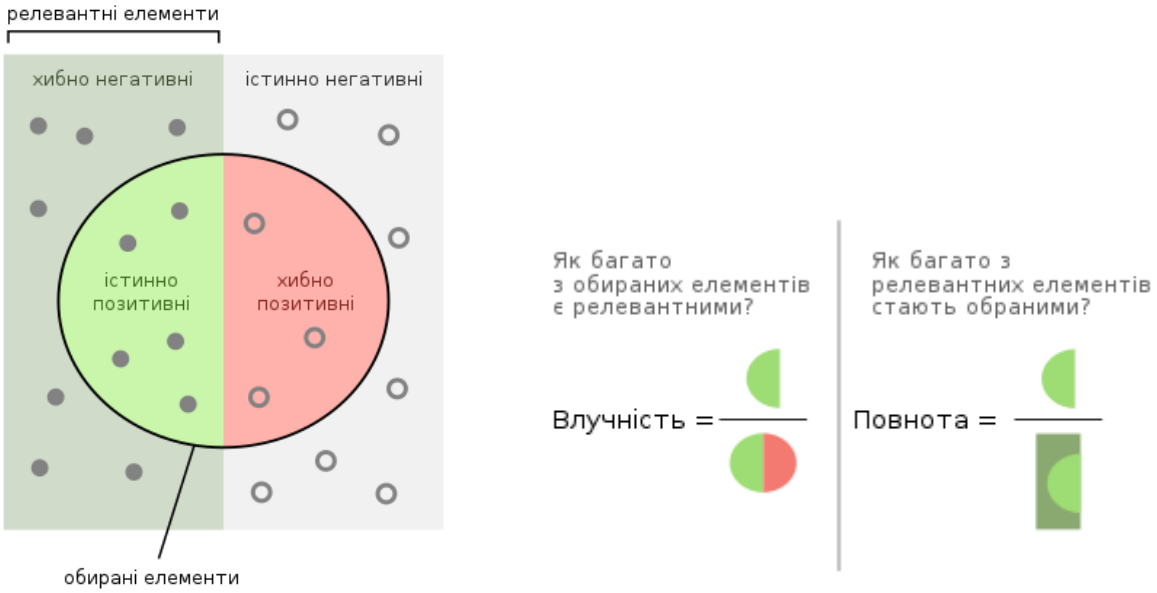


Рисунок 3.1 – Влучність та повнота

3.1 Наївний байєсів класифікатор

Наївний байєсів класифікатор — ймовірнісний класифікатор, що використовує теорему Байєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному) незалежності змінних. Теорема Байєса встановлює такий зв’язок між змінною класу y та вектором змінних x :

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Оскільки $P(x_1, \dots, x_n)$ є незмінним виразом, можна скористатися таким правилом класифікації

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Різні наївні класифікатори Байєса відрізняються головним чином припущеннями, які вони роблять щодо розподілу $P(x_i \mid y)$. Оскільки деякі із незалежних змінних в нашому наборі даних є неперервними, у цій роботі ми використовуємо Гаусівський наївний байєсів класифікатор, у якому відповідний розподіл вважається гаусівським.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Наш набір даних задовольняє усі припущення цієї моделі окрім одного – відсутності мультиколінеарності. Щоб її позбутися, ми використали Метод головних компонент (зазвичай його використовують для зменшення розмірності даних, проте ми не зменшували кількість незалежних змінних, а лише позбулися кореляції між ними).

За використання цього методу точність класифікації затримок на інтервали становить 63% на тренувальних даних та 63% на тестових; точність поділу затримок на до 60 хвилин/понад 60 хвилин становить 84% на тренувальних даних та 83% на тестових. Байєсів класифікатор – це швидкий і простий метод, що вважається базовим у машинному навчанні. При побудові інших моделей їхню точність часто порівнюють саме з точністю цієї моделі.

3.2 Логістична регресія

Логістична регресія – це статистична модель, що використовується для передбачення ймовірності виникнення події за допомогою логістичної функції. ЇЇ застосовують у випадку, коли залежна змінна є категорійною, тобто може набувати скінченної множини значень.

Для логістичної регресії точність класифікації затримок на інтервали становить 64% на тренувальних даних та 64% на тестових; точність поділу затримок на до 60 хвилин/понад 60 хвилин становить 85% на тренувальних даних та 85% на тестових (Рисунок 3.2.1).

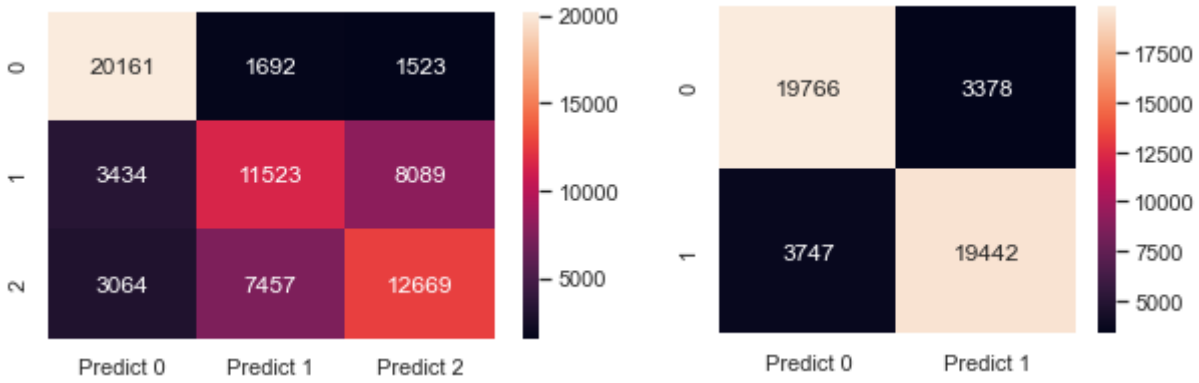


Рисунок 3.2.1 – матриці невідповідності для класифікації на інтервали та бінарної класифікації

3.3 Random forest

Random forest — ансамблевий метод машинного навчання, який працює за допомогою побудови численних дерев прийняття рішень під час тренування моделі й продукує моду для класів (класифікація) або усереднений прогноз (регресія) побудованих дерев. Навчальна вибірка складається з N прикладів, розмірність простору ознак дорівнює M, і заданий параметр m. Усі дерева будуються незалежно один від одного за такою процедурою:

1. Генерується випадкова підвибірка з повторенням із N.
2. Будується дерево рішень, яке класифікує приклади даної підвибірки, причому в ході створення чергового вузла дерева будемо вибирати ознаку, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних.

Класифікація об'єктів проводиться шляхом голосування: кожне дерево відносить об'єкт, який класифікується, до одного з класів, і перемагає клас, за який проголосувало найбільше число дерев.

Параметрами методу слугують кількість дерев та параметри для кожного із них. (максимальна глибина, мінімальна кількість елементів у листку тощо). Для підбору оптимальних значень параметрів було кілька тренувань моделей із різними параметрами. Сумарно тренування цих моделей зайняло близько восьми годин. З Рисунку 3.3.1, де зображено значення метрики f1 score для тестових даних, можна зрозуміти які параметри є найважливішими для досягнення високих значень цієї метрики. Вісь *y* відповідає мінімальній кількості спостережень у листку, це найсуттєвіший параметр. Розмір кружечка відповідає за мінімальну кількість спостережень у вузлі дерева, щоб його можна було ділити далі: що менше спостережень то вище значення f1 score. Цифри біля кружечків – максимальна глибина дерева: із збільшенням глибини покращуються якість моделі. Колір кодує кількість випадково вибраних ознак, між якими обирається найкраща для розбиття в ході створення чергового вузла.

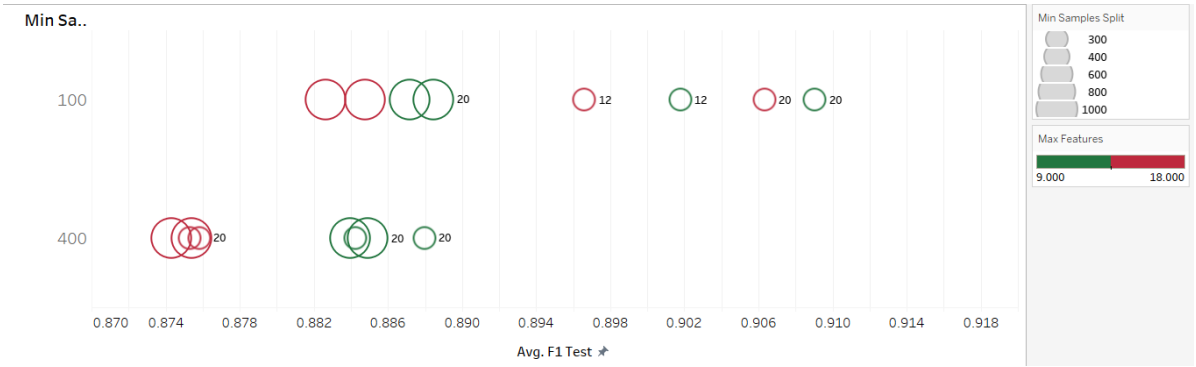


Рисунок 3.3.1 – значення f1 score на тестових даних моделі Random forest залежно від параметрів дерев прийняття рішень цієї моделі

У цьому методі точність класифікації затримок на інтервали становить 79% на тренувальних даних та 78% на тестових; точність поділу

затримок на до 60 хвилин/понад 60 хвилин становить 88% на тренувальних даних та 88% на тестових.

3.4 KNN

Метод k-найближчих сусідів — метричний алгоритм для класифікації об'єктів. Основним принципом методу найближчих сусідів є те, що об'єкт присвоюється тому класу, до якого належить найбільша кількість сусідів даного елемента. Щоб знайти сусідів, обчислюються відстані до усіх інших об’єктів. Вибираються k об’єктів, відстань до яких найменша, і вираховується, який клас є найчисленнішим серед них.

Найефективнішою кількістю сусідів для наших даних виявилися 4 та 6, ми використовували 4 під час тренування, оскільки так модель тренується та тестується швидше. Оскільки метод обчислює відстані, було змінено діапазон значень даних технікою Z Score.

Для KNN точність класифікації затримок на інтервали становить 97% на тренувальних даних та 94% на тестових; точність поділу затримок на до 60 хвилин/понад 60 хвилин становить 95% на тренувальних даних та 92% на тестових.

3.5 Boosting та Bagging

Підсилювання (англ. boosting) – це ансамблевий алгоритм машинного навчання, у якому набір слабких класифікаторів утворює один сильний класифікатор. Більшість алгоритмів підсилювання полягають в ітеративному навчанні слабких класифікаторів та додавання їх до остаточного сильного класифікатора. Слабким класифікаторам приписується вага, яка залежить від їхньої точності. Протягом ітеративного процесу навчання ті спостереження, що були класифіковані неправильно, набирають «вагу» (більше впливають на функцію втрат), а правильно класифіковані – втрачають. Ми використовуємо Adaptive Boosting та Gradient Boosting – подібні одне до одного алгоритми підсилювання, які

незначною мірою відрізняються функцією втрат [27]. В обох випадках слабкими класифікаторами є дерева рішень, а кількість цих дерев – гіперпараметром.

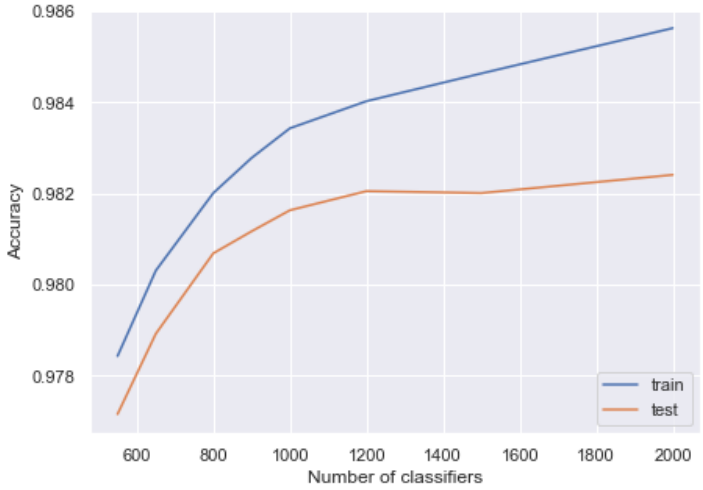


Рисунок 3.5.1а) – Gradient Boosting

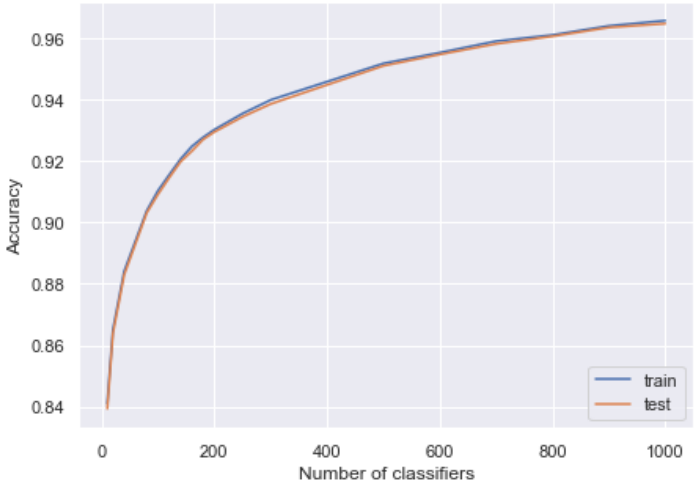
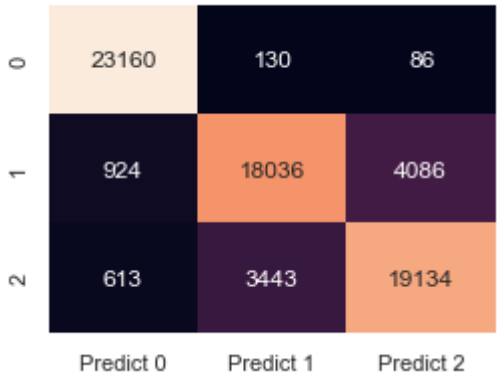


Рисунок 3.5.1б) – Adaptive Boosting

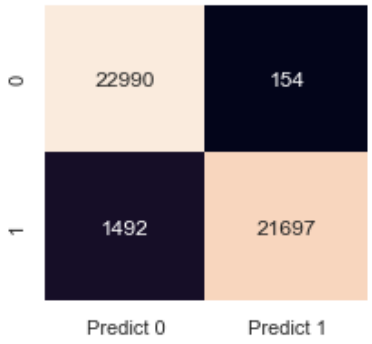
На Рисунках 3.5.1а) та 3.5.1б) зображено точність методів Gradient Boosting та Adaptive Boosting для бінарної класифікації на тренувальній та тестувальній вибірках в залежності від кількості слабких класифікаторів. Із збільшенням кількості класифікаторів зростає і точність, проте значне збільшення кількості класифікаторів, від 800 до 1000, дає дуже незначне підвищення точності, менш аніж на 0.5%. Оскільки зі збільшенням кількості класифікаторів суттєво зростає тривалість навчання моделі, ми вирішили обмежитися кількістю класифікаторів 800 для обох моделей.

Для Gradient Boosting точність класифікації затримок на інтервали становить 88% на тренувальних даних та 87% на тестових (Рисунок 3.5.2); точність поділу затримок на до 60 хвилин/понад 60 хвилин становить 97% на тренувальних даних та 96% на тестових (Рисунки 3.5.3, 3.5.4).



	precision	recall	f1-score
0	0.94	0.99	0.96
1	0.83	0.78	0.81
2	0.82	0.83	0.82
accuracy			0.87
macro avg	0.86	0.87	0.86
weighted avg	0.86	0.87	0.87

Рисунок 3.5.2 – Матриця невідповідностей та метрики точності, Gradient Boosting, класифікація на інтервали



	precision	recall	f1-score
False	0.94	0.99	0.97
True	0.99	0.94	0.96
accuracy			0.96
macro avg	0.97	0.96	0.96
weighted avg	0.97	0.96	0.96

Рисунок 3.5.3 – Матриця невідповідностей та метрики точності, Gradient Boosting, бінарна класифікація

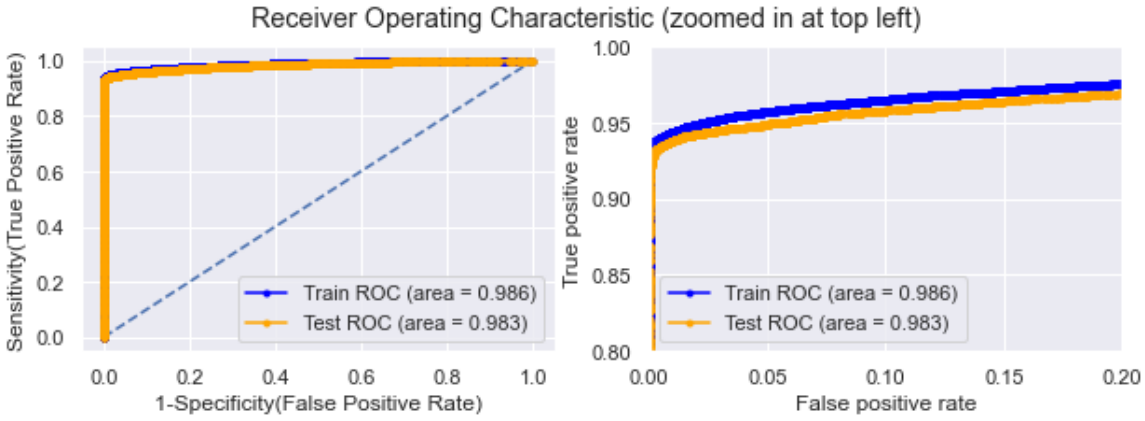


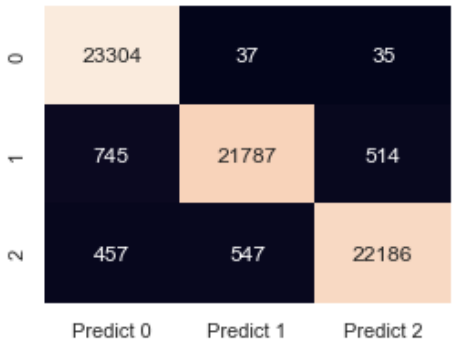
Рисунок 3.5.4 – Крива похибок, Gradient Boosting, бінарна класифікація

Для Adaptive Boosting точність класифікації затримок на інтервали становить 74% на тренувальних даних та 74% на тестових; точність поділу

затримок на до 60 хвилин/понад 60 хвилин становить 96% на тренувальних даних та 96% на тестових.

Бутстреп агрегація (англ. Bootstrap aggregating, bagging) – це ансамблевий алгоритм машинного навчання, у якому слабкі класифікатори тренуються на підмножині набору даних. Далі, остаточний прогноз формується на основі передбачень слабких класифікаторів із використанням методів голосування. Слабкі класифікатори можуть бути будь-якими моделями, ми обрали дерева рішень. Основним гіперпараметром методу є кількість класифікаторів. Окрім того, ми використовували додатковий параметр для регуляризації кожного із дерев-класифікаторів – кількість незалежних змінних, з яких обирається найкраща для поділу на кожній ітерації. Емпіричним шляхом визначено, що найкращі значення цих параметрів становлять 25 та 10 відповідно.

Для bagging точність класифікації затримок на інтервали становить 99% на тренувальних даних та 97% на тестових (Рисунок 3.5.5); точність поділу затримок на до 60 хвилин/понад 60 хвилин становить 99% на тренувальних даних та 97% на тестових.



	precision	recall	f1-score
0	0.95	1.00	0.97
1	0.97	0.95	0.96
2	0.98	0.96	0.97
accuracy			0.97
macro avg	0.97	0.97	0.97
weighted avg	0.97	0.97	0.97

Рисунок 3.5.5 – Матриця невідповідностей та метрики точності, bagging, класифікація на інтервали

3.6 Порівняння моделей

За Таблицею 3.5.1 можемо порівняти моделі за такими метриками: точність передбачення на тестових даних для трьох інтервалів затримок (0-

60/60-120/понад 120 хвилин), точність бінарної класифікації (затримка до 60 хвилин/понад 60 хвилин), площа під кривою похибок для бінарної класифікації та час, який знадобився для навчання моделі. Наведений час – час для тренування та передбачення інтервалів затримок. Як видно, з побудовою кожної наступної моделі вдавалося досягнути вищої точності на класифікації інтервалів.

Model	Accuracy multiclass	Accuracy binary	AUC binary	Time, seconds
Naïve bayes	0.633	0.832	0.893	4
Logistic regression	0.64	0.847	0.905	40
Adaptive Boosting	0.74	0.942	0.976	480
Random forest	0.79	0.88	0.953	35
Gradient Boosting	0.867	0.964	0.983	1,530
KNN	0.94	0.92	0.958	1,150
Bagging	0.967	0.967	0.991	30

Таблиця 3.5.1 – точність та час роботи моделей

Можемо зробити висновок, що для нашої задачі за таких наявних даних найкращою моделлю є bagging, що має високу точність і водночас не потребує тривалого часу для тренування, як KNN. Оскільки bagging модель змогла виявити три інтервали затримок з високою точністю, ми спробували застосувати цю ж модель для класифікації затримок на більшу кількість інтервалів: 0-15 хвилин (клас 0), 15-30 хвилин (клас 1), 30-60 хвилин (клас 2), 60-90 хвилин (клас 3), 90-120 хвилин (клас 4), понад 120 хвилин (клас 5).

Було побудовано дві моделі: для затримок вильотів та прильотів. Точність першої на тренувальних даних становить 99%, на тестових 81%. Для другої точність на тренувальних даних 99%, на тестових 95%. Як видно з Рисунків 3.5.1 та 3.5.2, де наведено матриці невідповідностей для класифікації вильотів та прильотів, найбільше помилок моделі припускаються за визначення класів 0, 1 та 2, які відповідають коротким затримкам.

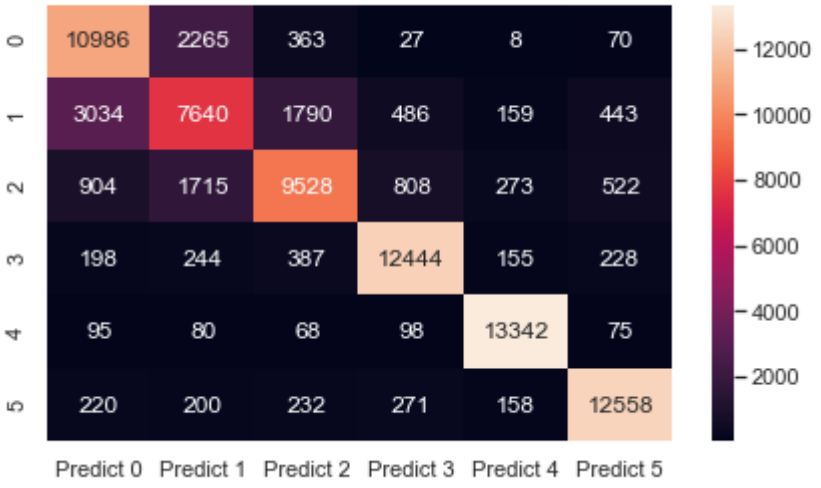


Рисунок 3.5.1 – Матриця невідповідностей для затримок вильотів

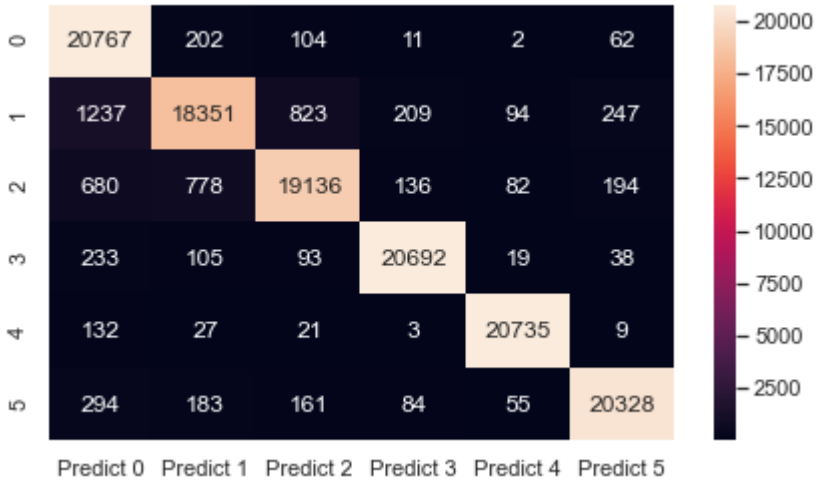


Рисунок 3.5.2 – Матриця невідповідностей для затримок вильотів

Зауважимо, що більшість із моделей доволі з високою точністю (близько 90%) можуть передбачати, чи буде хоча б якась затримка при вильоті. Для затримки прильоту ця точність менша – в середньому 75%. У Таблиці 3.5.2 наведено точності для різних моделей. Проте, ми не вважаємо таке передбачення достатньо інформативним, оскільки затримка може становити як 5, так і 105 хвилин.

Model	departure		arrival	
	train	test	train	test
Adaptive Boosting	0.9230	0.9120	0.7620	0.7530
Bagging	0.9990	0.9690	0.9980	0.7920
Gradient Boosting	0.9440	0.9400	0.8060	0.7910
KNN	0.9100	0.8680	0.8170	0.7230
Logistic regression	0.8350	0.8330	0.7060	0.7020
Naïve bayes	0.8230	0.8210	0.6980	0.6960
Random forest	0.8870	0.8810	0.7660	0.7530

Таблиця 3.5.2 – точність передбачення наявності затримки

Незважаючи на те, що побудовані моделі можуть давати доволі високу точність при бінарній класифікації (чи буде затримка тривалістю хоча б 60 хвилин?, чи буде затримка тривалістю хоча б 90 хвилин?), ці самі моделі значно гірше виконують задачу класифікації на декілька класів (чи затримка буде до 60/60-90/понад 90 хвилин?). Можливо, цю проблему можна вирішити створенням ансамблю із моделей, побудувати такий нам не вдалося, щоб його точність хоча б наблизилася до точності bagging класифікатору (який сам є ансамблем).

Побудовані моделі не схильні до перетренування, тобто точність на тестових даних не сильно нижча порівняно із точністю на тренувальних даних. Для деяких моделей це досягається вдалим підбором гіперпараметрів, для інших – регуляризацією

Зауважимо, що при побудові моделей ми вважали помилки першого та другого роду рівноцінними між собою і зважали на значення обох з них. У випадках, коли якась із цих помилок є більш важливою аніж інша, моделі можна коригувати, змінюючи поріг, по якому вони виносять рішення щодо належності об’єкту до одного чи іншого класу (для ймовірнісних моделей це легко, для інших застосовуються спеціальні методи [25], які використовуються зокрема і для побудови ROC-кривих). Таким чином можна зменшувати одну з цих помилок, проте це досягається за рахунок збільшення іншої.

ВИСНОВКИ

У ході роботи було зібрано дані про перельоти, здійснені українськими авіалініями протягом останніх трьох років. На основі цих даних було побудовано сім моделей машинного навчання та проведено порівняльний аналіз їхньої якості. Вдалося побудувати моделі, які не перетреновуються, тобто точність на тестових даних майже не відрізняється від точності на тренувальних. Було визначено, що для нашої задачі найкращу класифікацію здійснює bagging модель, при застосуванні якої точність на тестових даних сягає 97%. Окрім того, було виявлено, що більшість рейсів затримуються на короткий проміжок часу при вильоті і значна частина з них прилітає вчасно. Моделі машинного навчання краще передбачають тривалі затримки і гірше – короткі. Це спричинено тим, що на короткі затримки впливають значно більше факторів, аніж наявні в нашому наборі даних. До таких факторів можуть належати завантаженість аеропорту, вчасність прильоту лайнеру з попереднього рейсу, непередбачувані поломки та інше. Звісно, всі подібні фактори врахувати неможливо. Проте, критично погана погода з високою ймовірністю гарантує тривалу затримку рейсу, тому і точність передбачення тривалої затримки більша. Застосовувати моделі для передбачення можна за наявності прогнозу погоди на час запланованого вильоту рейсу. Неточності прогнозу погоди погіршуватимуть якість прогнозу моделей, тому якість буде тим вища, коли передбачення робиться за короткий період перед запланованим вильотом.

Подальші дослідження та перспективи. Одним із основних варіантів покращення цієї роботи є розширення набору даних, зокрема додавання рейсів нових авіаліній, відповідно нових моделей та аеропортів. За такого сценарію бажаною є автоматизація цього процесу: що менше буде ручної роботи, то швидше відбуватиметься розширення набору даних.

Окремої уваги заслуговує частина із погодою: можна шукати нові джерела даних, де була б доступна інформація про рівень опадів та стан злітно-посадкової смуги; можна пришвидшувати існуючий в нашому проекті запис даних про погоду, використовуючи більш оптимізовані функції пошуку або запускаючи програму не на одному комп'ютері, а на віддалених сервісах із більшою потужністю ресурсів. Найкращий можливий варіант – знайти API, яке б надавало доступ до потрібної історичної інформації про погоду. Ще однією можливістю покращення є врахування додаткової інформації щодо завантаженості аеропорту, до прикладу, чи був затриманий попередній рейс і на скільки. Проте, у вільному доступі такої інформації вкрай мало.

Інший шлях покращення цього проекту – вдосконалення існуючих та додавання нових моделей. Зокрема, для існуючих моделей можна вдосконалювати підбір гіперпараметрів; новими моделями можуть бути нейронні мережі, приховані марковські моделі, екстремальний градієнтний спуск (XGBoost) тощо. Окрім того, можна вдосконалювати моделі для використання у реальному часі і покращувати передбачення із наближенням до часу вильоту рейсу (для цього, до прикладу, можна підключатися до сервісів погоди і оновлювати своє передбачення за зміни прогнозу).

Незважаючи на нинішню кризу цивільної авіації, спричинену пандемією вірусу COVID-19, з часом ця галузь відновиться, як вже неодноразово ставалося протягом її історії. Затримки польотів є однією з її найбільших проблем, яка спричиняє економічні та репутаційні втрати. Тому й надалі передбачення та усунення потенційних затримок залишатиметься актуальним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. U.S. Government Accountability Office, Airline Consumer Protections: Additional Actions Could Enhance DOT's Compliance and Education Efforts, 2018
2. J. J. Rebollo and H. Balakrishnan. Characterization and prediction of air traffic delays. Transportation Research Part C: Emerging Technologies, 44, 231–241, 2014
3. S. Khanmohammadi, C. A. Chou, H. W. Lewis, D. Elias. A systems approach for scheduling aircraft landings in JFK airport. 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), ст. 1578–1585
4. P. Balakrishna, R. Ganesan, L. Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. Transportation Research Part C: Emerging Technologies, 18(6):950–962, 2010
5. L. Zonglei, W. Jiandong, Z. Guansheng. A New Method to Alarm Large Scale of Flights Delay Based on Machine Learning. 2008 International Symposium on Knowledge Acquisition and Modeling, ст. 589–592, 2008.
6. Department of Transportation, 2015 Flight Delays and Cancellations, 2017-02-09, [Електронний ресурс] Режим доступу: https://www.kaggle.com/usdot/flight-delays
7. Manna, S., Biswas, S., Kundu, R., Rakshit, S., Gupta, P., & Barman, S. (2017). A statistical approach to predict flight delay using gradient boosted decision tree. 2017 International Conference on Computational Intelligence in Data Science(ICCIDS). doi:10.1109/iccids.2017.8272656
8. Shao W, et al. Flight delay prediction using airport situational awareness map. In Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2019
9. Flightradar24 (2018) Flightradar24 live flight tracker. [Електронний ресурс] Режим доступу: https://www.flightradar24.com

10. Coy, S. (2006). A global model for estimating the block time of commercial passenger aircraft. Journal of Air Transport Management, 12(6), 300–305. doi:10.1016/j.jairtraman.2006.07.005
11. Milrad, S. (2018). METAR Code. Synoptic Analysis and Forecasting, 23–39. doi:10.1016/b978-0-12-809247-7.00003-x
12. METAR авіаційний метеорологічний код [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/METAR
13. FAA: Aviation weather services [Електронний ресурс]/ Date: 11/14/16 AC No: 00-45H Initiated by: AFS-400/ Режим доступу: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_00-45H.pdf
14. Iowa Environmental Mesonet web page of Iowa State University ASOS-AWOS-METAR [Електронний ресурс] / Режим доступу: http://mesonet.agron.iastate.edu/AWOS/
15. ASOS/AWOS Global METAR Archives [Електронний ресурс]/ Режим доступу: https://mesonet.agron.iastate.edu/info/datasets/metar.html
16. ASOS Network Ukraine ASOS [Електронний ресурс] / Режим доступу: https://mesonet.agron.iastate.edu/request/download.phtml?network=UA__ASOS#
17. Національне управління океанічних і атмосферних досліджень [Електронний ресурс] / Режим доступу: https://aviationweather.gov/metar
18. Weather scraper example [Електронний ресурс] / Режим доступу: https://github.com/akrherz/iem/blob/main/scripts/asos/iem_scraper_example.py
19. Denis Arnaud. 2017. Open travel data. Amadeus IT Group. [Електронний ресурс] / Режим доступу: https://github.com/opentraveldata/opentraveldata
20. Jani Patokallio, Open Flights, Airport database. [Електронний ресурс] / Режим доступу: https://openflights.org/data.html

21. Matejka, J., & Fitzmaurice, G. (2017). Same Stats, Different Graphs. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI ’17. doi:10.1145/3025453.3025912
22. Yen, S.-J., & Lee, Y.-S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. Expert Systems with Applications, 36(3), 5718–5727. doi:10.1016/j.eswa.2008.06.108
23. N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, “SMOTE: Synthetic Minority Over-Sampling Technique,” J. Artificial Intelligence Research, 16, ст. 321-357, 2002.
24. Код проекту// Режим доступу: https://github.com/Zhanna-Lopuliak/flight_delay
25. Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., & Ziegler, A. (2011). Probability Machines: Consistent Probability Estimation Using Nonparametric Learning// Machines Methods of Information in Medicine, 51(1), 74–81. doi:10.3414/me00-01-0052
26. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
27. Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.

ДОДАТОК А

Використані технології та структура проєкту

Повний код проєкту викладено у вільний доступ [24]. Увесь проєкт написаний мовою програмування Python3. Інтегрованими середовищами розробки на різних етапах роботи слугували PyCharm, Google Colaboratory та Jupyter Notebook.

У папках «data_csv» та «data_xlsx» зберігаються дані по авіарейсам кожної із моделей літаків, зібрані із html файлів, завантажених з джерела даних. Самі html не викладені, тому що після того, як на сайті джерела даних скінчився пробний період для облікового запису автора, ці html сторінки перестали бути доступними та відображати інформацію. У папці «unique» можна знайти файли з кодами аеропортів та їхніми часовими зонами; описами моделей; зібрані в один файл польоти; файл із польотами та записаною погодою. У папці «data_weather» можна знайти завантажену погоду для кожного із аеропортів (файли лише розміру менше ніж 2MB, оскільки більші не дозволяє платформа).

У файлі «scraping.py» наведено код, за допомогою якого було трансформовано html файли в систематизовані набори даних в форматі csv. Ключові бібліотеки, використані для цього: BeautifulSoup, pandas.

У файлі проєкту «weather.py» наведено код, який виконував збір даних про погоду: вибір аеропортів для яких потрібні дані, завантаження кожного із файлів. Ключові бібліотеки, використані для цього: urllib.request, datetime, time.

У файлі проєкту «time_transform_one_df.py» наведено код для об’єднання всіх польотів в один файл та роботи із часом: переведення в одну часову зону та запис нових стовпців, обчислення затримок, запис погоди. Ключові бібліотеки, використані для цього: pytz для часових зон, datetime, timedelta для роботи з часовими даними.

У папці «ipynb_files» наведено файли у форматі ipynb. Саме у такому форматі зберігаються файли із середовищ розробки Google Colaboratory (використання ресурсів Google) та Jupyter Notebook (використання ресурсів власного комп'ютера). Тут можна побачити файли із розвідувальним аналізом даних, попередньою обробкою даних та файл із побудованими моделями і результатами (models.ipynb). Ключові бібліотеки, використані для цього: pandas та numpy для збереження та обробки даних; matplotlib та seaborn для візуалізації; google.colab.drive для підключення до файлів із даними; imblearn.over_sampling.SMOTE для збалансування набору даних створенням штучних спостережень. Для побудови усіх моделей, передобробки даних для них та вимірювання усі метрик точності було використано бібліотеку scikit-learn [26].

У папці «other» можна знайти всі інші дані, що використовувалися в роботі.

Окрім того, для створення деяких із візуалізацій було використано програмне забезпечення для інтерактивної візуалізації даних Tableau Desktop.

На етапі збору даних для відкриття повної веб-сторінки ми скористалися розширенням для браузеру Google Chrome Tampermonkey. Щоб знайти елемент (кнопку), на який треба натискати, ми знайшли її ідентифікатор у вихідному коді. Нижче наведено текст скрипту. Додатково було встановлено затримку між натисканнями на кнопку, бо інакше веб-сайт переставав відповідати на запити (можливо, це певний захист від веб-скрапінгу).

```
(function() {
    'use strict';
    var delayInMiliseconds = 5000;
    var button = document.getElementById('btn-load-earlier-flights');
    setInterval(function(){button.click();
    }, delayInMiliseconds);
})();
```