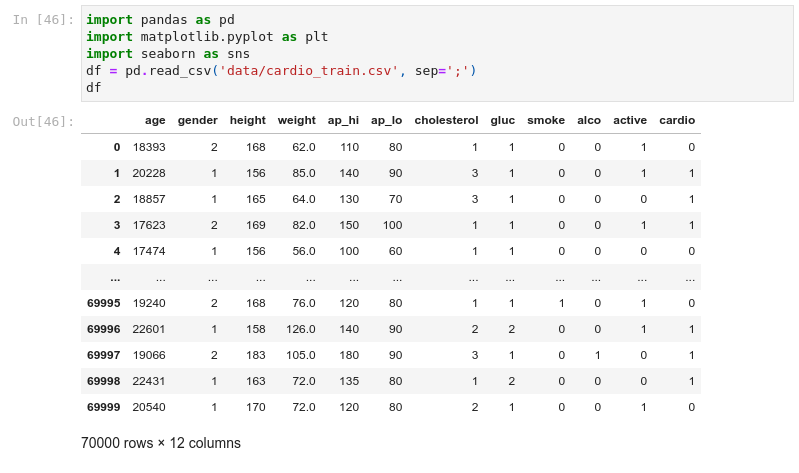
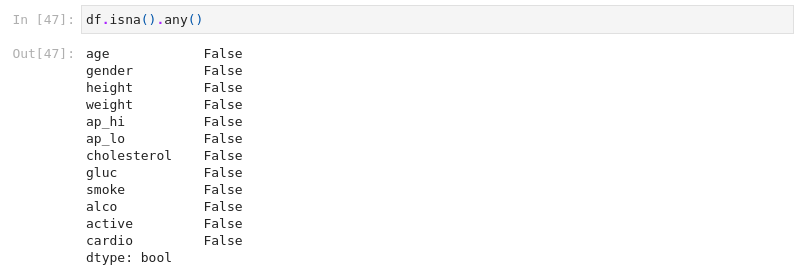
## Робота з даними

### Опис обраних даних

Імпортуємо пакет pandas, завантажимо дані в датафрейм, виводимо інформацію про нього. Для вирішення поставленої задачі було обрано "Cardiovascular Disease" датасет. Він складається з 70 000 пацієнтів. Даний датасет містить в собі такі колонки, як-от: "age" - вік, "height" - зріст, "weight" - вага, "gender" - стать, "ap\_hi" - систолічний кров’яний тиск, "ap\_lo" - діастолічний артеріальний тиск, "cholesterol" - холестерол, "gluc" - глюкоза, "smoke" - чи курить пацієнт, "alco" - чи вживає алкоголь, "active" - фізична активність, "cardio" - присутність захворювання. Додатково імпортуємо модулі matplotlib.pyplot та seaborn для відображення графіків.

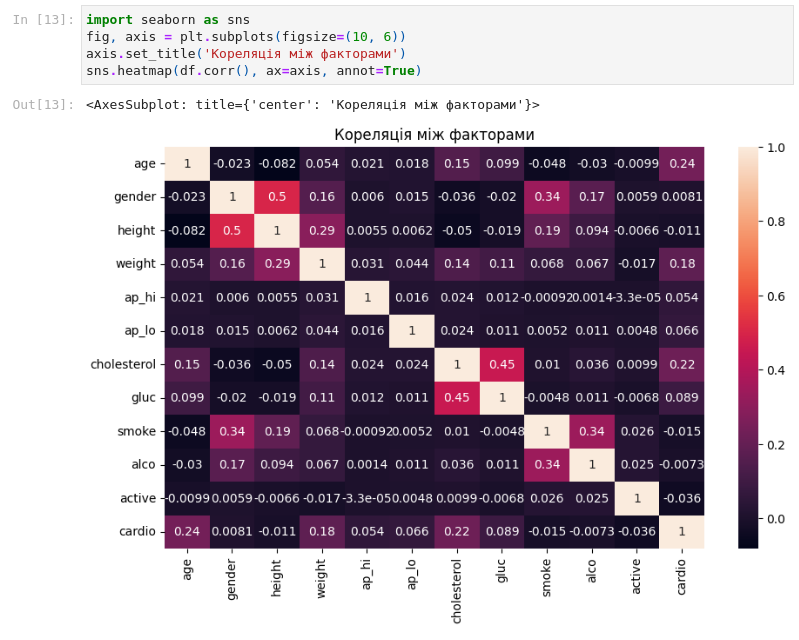
  
  
Рисунок 3.1 - Завантажений датафрейм

Видалимо колонку id за допомогою методу pandas.Dataframe.drop, передавши в нього список з єдиним елементом id та параметром за замовчуванням inplace=True, щоб зміна відбулася в самому датафреймі, а не видало новий.

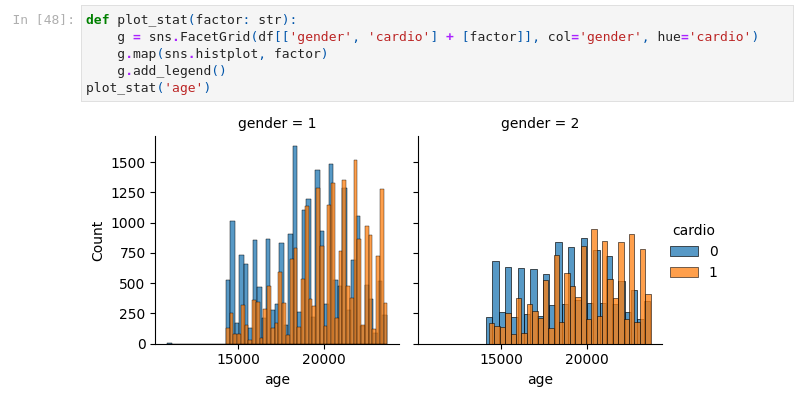
  
  
Рисунок 3.3 - Колонки, де всі значення ініціалізовані

### Вибір ознак для аналізу

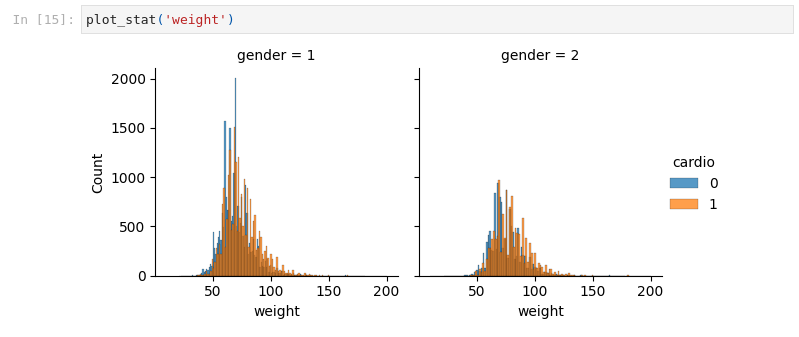
Побудуємо матрицю кореляції та визначимо, наскільки кожен фактор впливає на якість води. Для цього імпортуємо модуль seaborn та застосуємо функцію heatmap, де передаємо йому вище зазначену матрицю, використавши pandas.Dataframe.corr метод датафрейму.

  
  
Рисунок 3.4 - Матриця кореляцій

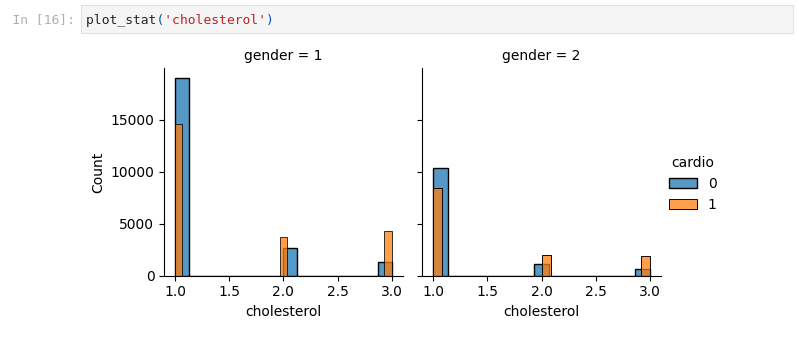
Бачимо, що найбільше на виникнення серцево-судинних захворювань впливає вік, вага, холестерол. Доволі цікавим фактами є кореляція між віком та холестеролом, статтю та курінням, статтю та алкоголем, холестеролом та глюкозою. Залежність між статтю та вагою і висотою доволі очевидна, що пояснюється звичайною різницею у фізичних показниках між чоловіком та жінокю. Зобразимо статистику факторів та виникненням захворювання, розділену за статтю. За допомогою функції seaborn.FacetGrid згрупуємо значення та за допомогою seaborn.histplot зобразимо їх у вигляді гістограми. Побудуємо статистику за віком.

  
  
Рисунок 3.5 - Статистика захворювання від віку між чоловіками та жінками

Зобразимо статистику за вагою.

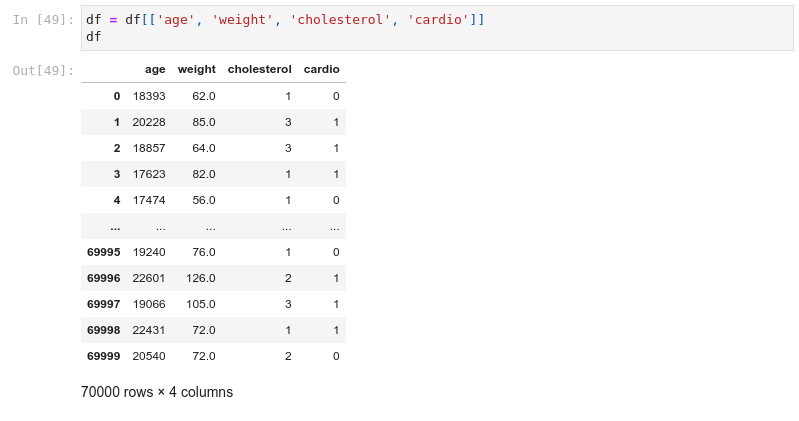
  
  
Рисунок 3.6 - Статистика захворювання від ваги між чоловіками та жінками

Зобразимо статистику за холестеролом.

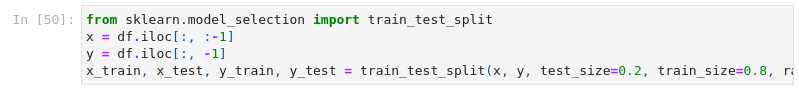
  
  
Рисунок 3.7 - Статистика захворювання від холестеролу між чоловіками та жінками

### Поділ даних

Для збільшення якості моделі залишимо лише чотири колонки: age, weight, cholesterol, cardio.

  
  
Рисунок 3.8 - Відфільтрований датафрейм

Ділимо дані на тренувальні та тестові для подальшох роботи. Імпортуємо модуль sklearn.model\_selection та застосуємо функцію train\_test\_split. Розділимо набір даних на 80% навчальних та 20% тестових.

  
  
Рисунок 3.9 - Поділ інформації на інформацію та результат

Зберігатимемо результати тестування моделей у списку results

  
  
Рисунок 3.10 - Список результатів

## Інтелектуальний аналіз даних

### Обґрунтування вибору методів інтелектуального аналізу даних

Було обрано чотири методи K-Nearest Neighbors, Logistic Regression, Random Forest, SVM.

K-Nearest Neighbors (KNN) — це простий, але ефективний алгоритм машинного навчання, який використовується для завдань класифікації та регресії. Я обрав, щоб порівняти наскільки він швидший за інші алгоритми, оскільки він не робить жодних припущень щодо розподілу даних.

До того ж однією з сильних сторін алгоритму KNN є те, що він легко інтерпретується. Прогнози можна пояснити, просто подивившись на K найближчих сусідів даної точки даних. Крім того, KNN може обробляти нелінійні межі рішень і може використовуватися як для бінарних, так і для багатокласових задач класифікації. Тому він чудово підійде для прогнозування хвороб, де треба враховувати багато факторів.

Однак, якщо датасети великі, то KNN може бути дорогим з обчислювальної точки зору. Крім того, продуктивність KNN значною мірою залежить від значення K, яке потрібно ретельно вибирати для досягнення оптимальних результатів.

Logistic Regression – це статистичний метод, який використовується для прогнозування ймовірності події. Оскільки її залежна змінна є двійковою, то це дуже підходить для виявлення хвороби: вона або є, або її нема.

Logistic Regression має кілька переваг перед іншими алгоритмами класифікації. По-перше, її легко реалізувати та інтерпретувати. По-друге, вона може обробляти як категоричні, так і безперервні незалежні змінні. Тобто можна обробляти як дискретні дані, які лекго визначити: вага, зріст - так і виявлення в організмі сполук, які складно визначити точно, і де робляться припущення про інтервали: рівень глюкози, рівень холестиролу, вітамінів в організмі тощо. По-третє, він може надати ймовірність події, що станеться, що корисно під час медичної діагностики.

Random Forest — це популярний алгоритм машинного навчання, який використовується для завдань класифікації та регресії. Це метод ансамблю, який поєднує передбачення кількох дерев рішень для отримання більш точних результатів.

Я його обрав, бо є стійким до шуму. До того ж цікаво, як він буде передбачувати наявність хвороби в пацієнта. Варто зауважити, що Random Forest може бути дорогим з точки зору обчислень, особливо при роботі з великими наборами даних і багатьма деревами. Також може бути важко інтерпретувати результати випадкового лісу, оскільки процес прийняття рішення розподіляється між кількома деревами.

SVM - це популярний і потужний клас алгоритмів керованого навчання, які використовуються для завдань класифікації та регресії. SVM особливо добре підходять для проблем, де існують складні межі між класами. У медичній сфері це часто використовується для прогнузування пацієту декількох хвороб, де окреме захворювання - це клас, де варто мати чіткі межі для поставлення правильного діагнозу.

Основна ідея SVM — знайти гіперплощину, яка найкраще розділяє дані на різні класи. Гіперплощина — це лінійна межа рішення, яка розділяє два класи в просторі ознак.

Однією з сильних сторін SVM є те, що вони нечутливі до викидів, оскільки на гіперплощину впливають лише найближчі до неї точки. Це робить SVM дуже стійкими до зашумлених даних, де зазвичай пацієнтів дуже багато, а тому ця особливість є важливою.

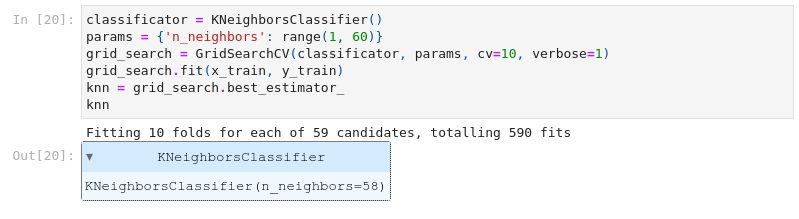
Однак SVM мають деякі обмеження. Навчання їх може бути дорогим з обчислювальної точки зору, особливо з великими наборами даних або просторами функцій великої розмірності. Крім того, вибір правильної функції ядра може бути складним, а продуктивність SVM залежить від вибору гіперпараметрів.

### Аналіз отриманих результатів для методу K-Nearest Neighbors

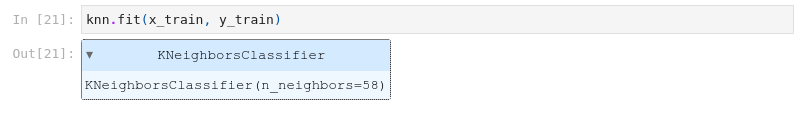
Для виконання роботи методу KNN імпортуємо sklearn.neighbors.KNeighborsClassifier та sklearn.model\_selection.GridSearchCV.

  
  
Рисунок 4.11 - Імпортування модулів

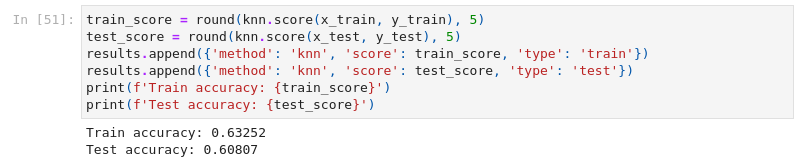
Визначимо, які варіанти параметрів найкраще вирішують дану задачу.

  
  
Рисунок 4.12 - Визначення найкращого параметра

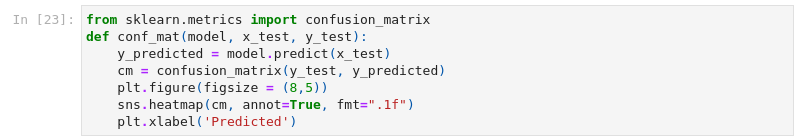
Натренуємо модель з найкращим параметром.

  
  
Рисунок 4.13 - Тренування моделі K-Nearest Neighbors

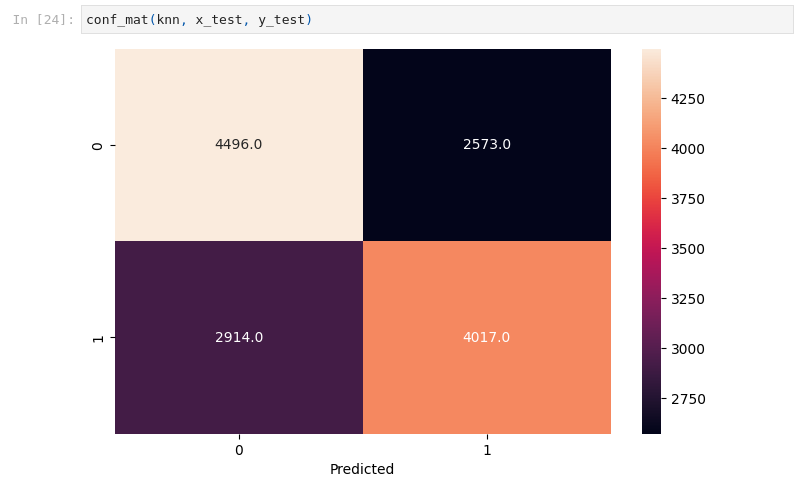
Визначимо точність моделі на тренувальних та тестових даних

  
  
Рисунок 4.14 - Точність моделі K-Nearest Neighbors

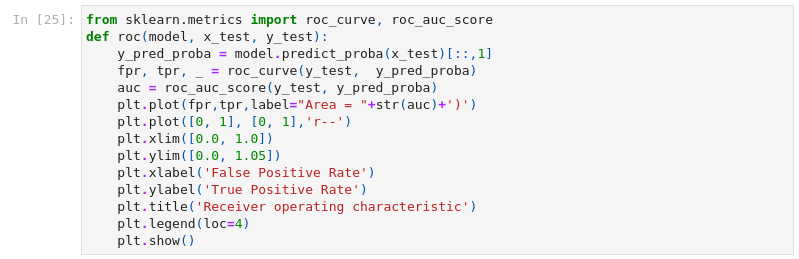
Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей. Для цього застосуємо sklearn.metrics.plot\_confusion\_matrix.

  
  
Рисунок 4.15 - Імпортування модуля

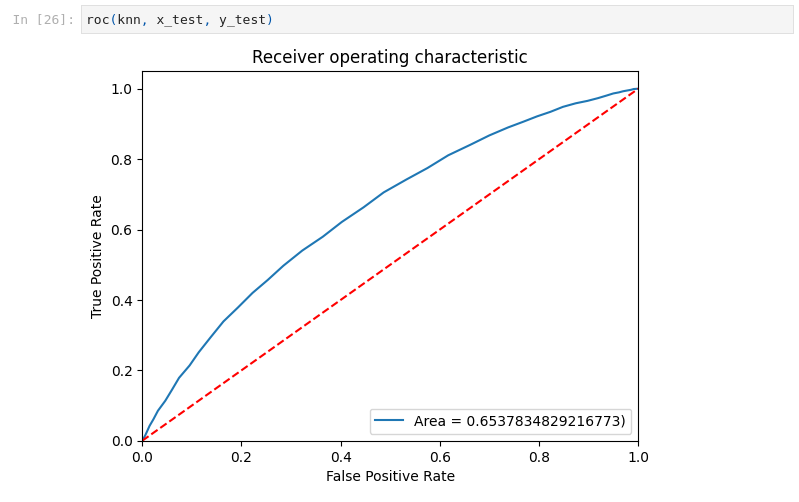
Матриця має два рядки та дві колонки: перший ряд і перша колонка - це істинно позитивні значення, тобто людина здорова і модель не визначила захворювання; першмй ряд і друга колонка - хибно позитивні, тобто людина здорова, а модель сказала, що є захворювання; другий ряд і перша колонка - хибно негативні, тобто людина хвора, а модель сказала, що здорова; другий ряд і друга колонка - істинно негативні, тобто людина хвора і модель сказала, що хвора.

  
  
Рисунок 4.16 - Матриця невідповідностей для K-Nearest Neighbors

Побудуємо графік ROC( Receiver Operating Characteristic ), що є графіком істинно позитивної відносної частоти проти хибно позитивної частоти. Це показує компроміс між чутливістю та специфічністю. Для цього імпортуємо sklearn.metrics.roc\_curve та sklearn.metrics.roc\_auc\_score. До того ж визначимо AUC( Area Under the ROC Curve ), що є мірою того, наскільки добре модель може розрізняти позитивні та негативні рузультати. Він коливається від 0 до 1, де 1 є найкращим класифікатором, а 0,5 – випадковим класифікатором. AUC корисний під час порівняння продуктивності різних класифікаторів на одному наборі даних, бо даж єдине число, яке підсумовує загальну продуктивність.

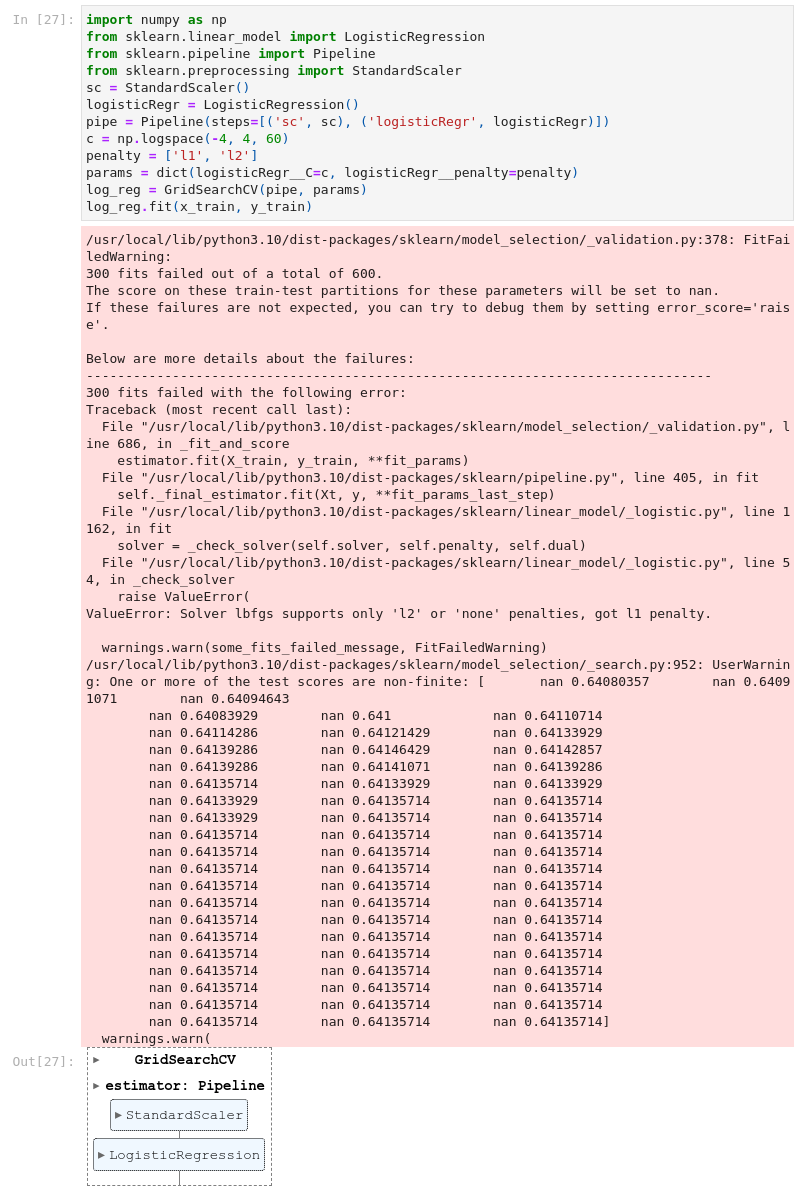
  
  
Рисунок 4.17 - Імпортування модуля та визначення функції roc

Побудуємо ROC для K-Nearest Neighbors.

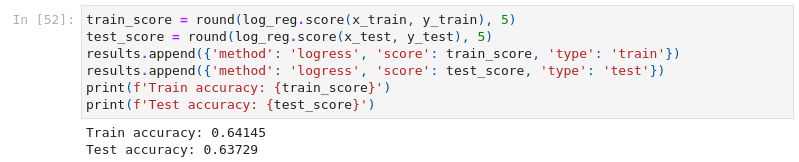
  
  
Рисунок 4.18 - Графік ROC для K-Nearest Neighbors

### Аналіз отриманих результатів для методу Logistic Regression

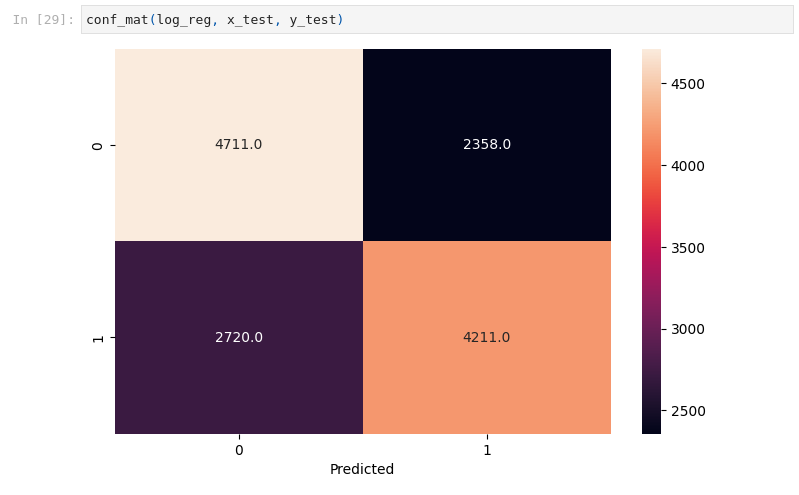
Для виконання роботи методу KNN імпортуємо sklearn.linear\_model.LogisticRegression; sklearn.pipeline.Pipeline для побудови пайплайну, щоб у зручному вигляді передавати параметри до декількох функцій; sklearn.preprocessing.StandardScaler для скейлингу даних до проміжку від 0 до 1. Визначимо найкращі параметри моделі, передавши в неї параметри регуляризації.

  
  
Рисунок 4.19 - Тренування моделі Logistic Regression

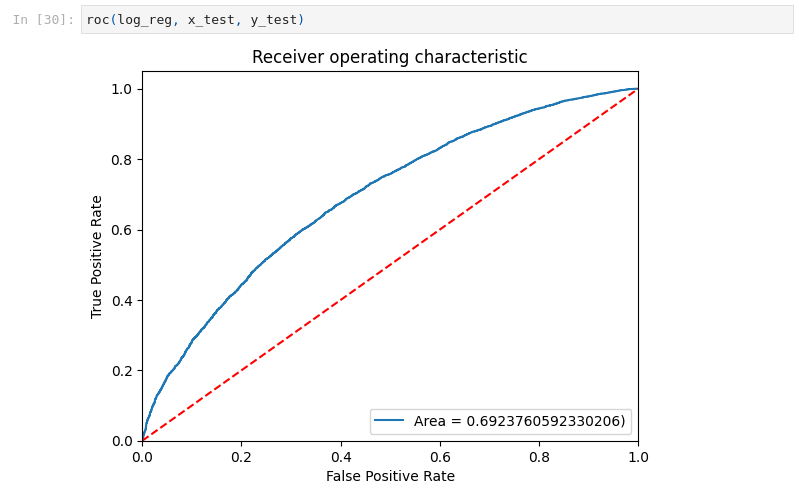
Визначимо точність моделі на тренувальних та тестових даних

  
  
Рисунок 4.20 - Точність моделі Logistic Regression

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

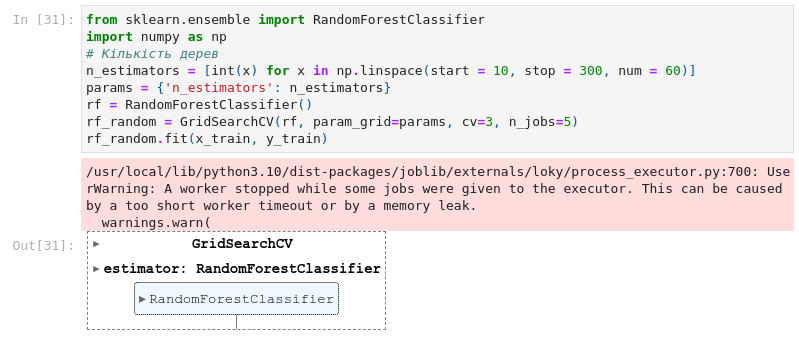
  
  
Рисунок 4.21 - Матриця невідповідностей для Logistic Regression

Побудуємо графік ROC для Logistic Regression.

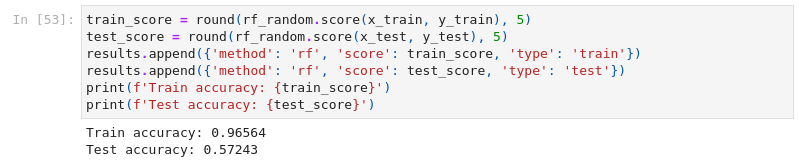
  
  
Рисунок 4.22 - Графік ROC для Logistic Regression

### Аналіз отриманих результатів для методу Random Forest

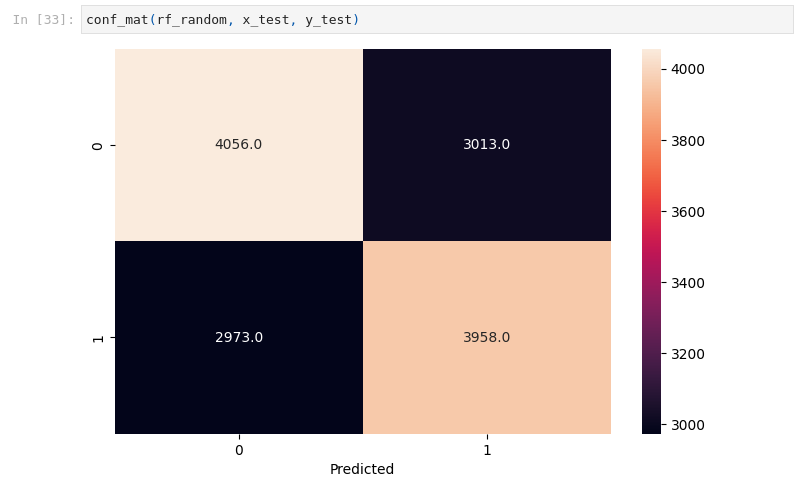
Для виконання роботи методу KNN імпортуємо sklearn.ensemble.RandomForestClassifier. Визначимо найкращі параметри для моделі. У випадку Random Forest параметри включають кількість дерев рішень та кількість характеристик, які враховуються кожним деревом під час поділу вузла.використовуються для поділу кожного вузла, отриманого під час навчання. Імпортуємо sklearn.model\_selection.RandomizedSearchCV.

  
  
Рисунок 4.23 - Тренування моделі Random Forest

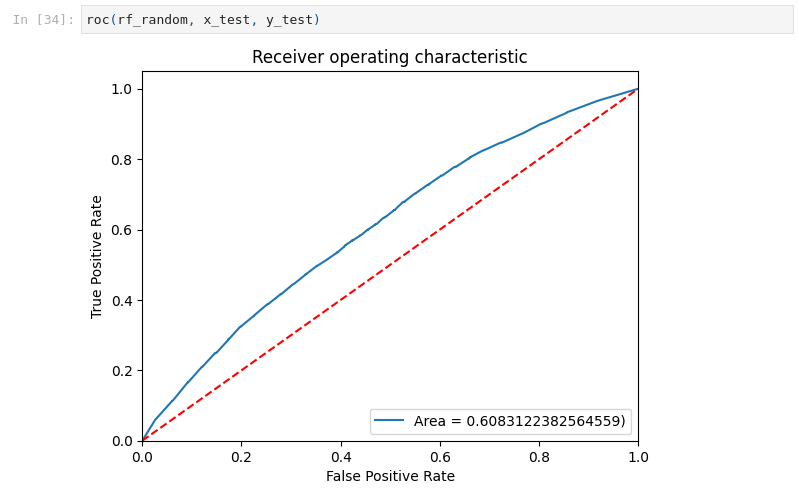
Визначимо точність моделі на тренувальних та тестових даних

  
  
Рисунок 4.24 - Точність моделі Random Forest

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

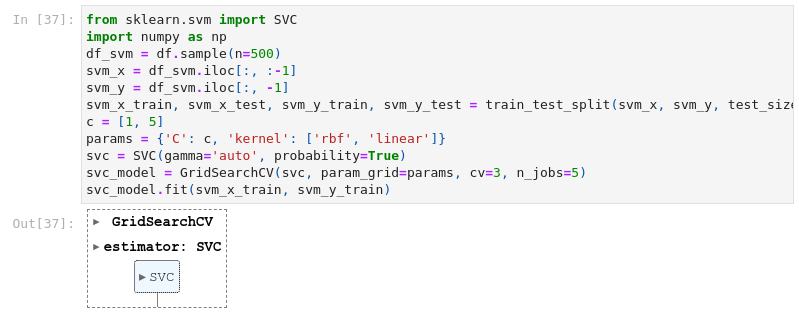
  
  
Рисунок 4.25 - Матриця невідповідностей для Random Forest

Побудуємо графік ROC для Logistic Regression.

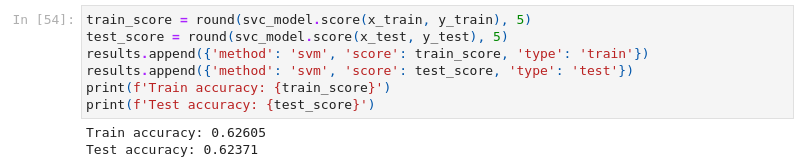
  
  
Рисунок 4.26 - Графік ROC для Random Forest

### Аналіз отриманих результатів для методу Support Vector Machines

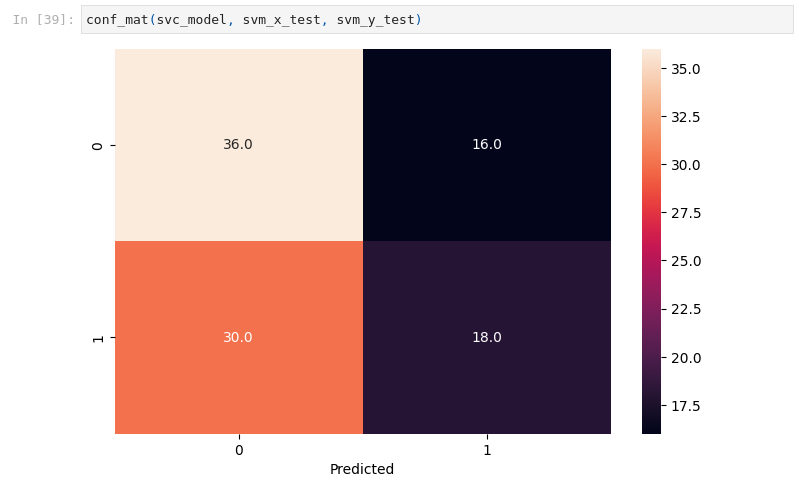
Для виконання роботи методу SVM імпортуємо sklearn.svm.SVC. Визначимо найкращі параметри для моделі. Оскільки складність SVM - це O(n\_samples^2 \* n\_features), тобто для якщо факторів 3 і 70 000 зразків, то маємо 1.47e10 ітерацій, що надзвичайно багато. Тому оберемо 1000 випадковиз зразків і отримаємо загальну кільскість ітерацій в 3e6.

  
  
Рисунок 4.27 - Тренування моделі SVM

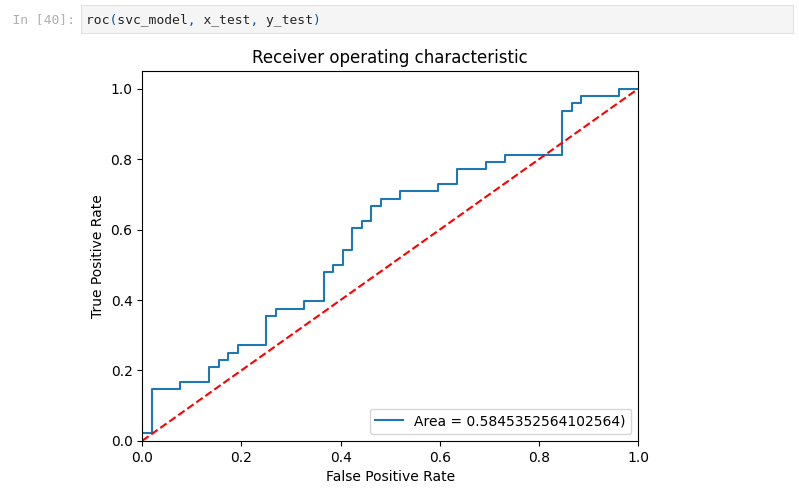
Визначимо точність моделі на тренувальних та тестових даних

  
  
Рисунок 4.28 - Точність моделі SVM

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

  
  
Рисунок 4.29 - Матриця невідповідностей для SVM

Побудуємо графік ROC для SVM.

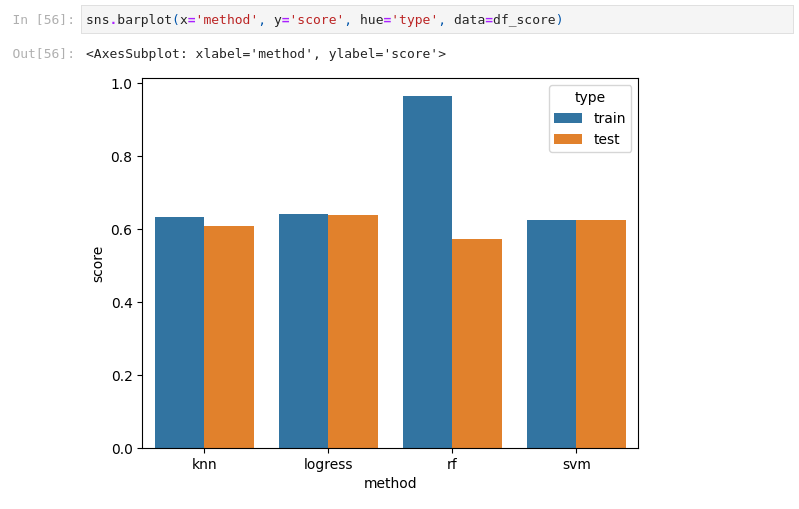
  
  
Рисунок 4.30 - Графік ROC для SVM

### Порівняння отриманих результатів методів

Проаналізувавши окремо кожен із методів, проведемо порівнянняданих методів.

  
  
Рисунок 4.31 - Датафрейм результатів

Для наочності побудуємо гістограму.

  
  
Рисунок 4.32 - Результати моделей

З огляду бачимо, що на тестових даних усі методи відпрацювали більш-менш однаково. Однак на тренувальних даних Random Forest показує себе найкраще.