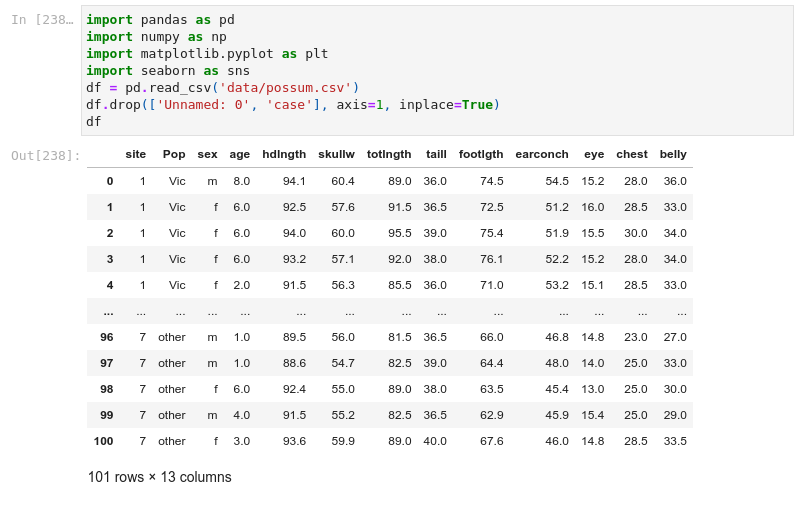
# Виконання

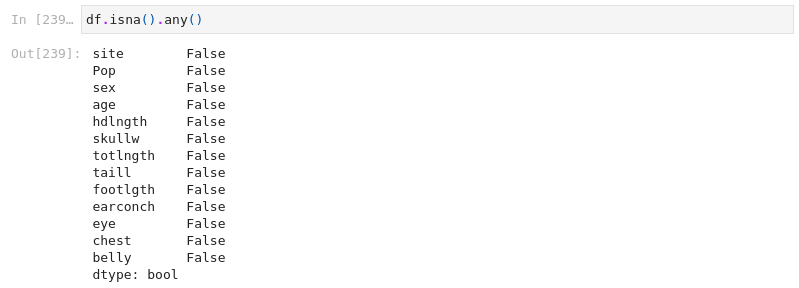
## Завантаження даних

Імпортуємо модулі pandas, numpy, matplotlib.pyplot, seaborn. Завантажимо датафрейм та виведемо інформацію про нього. Видалимо колонку "Unnamed: 0" та "case", що позначає індекси, оскільки вони вже були автоматично створені.

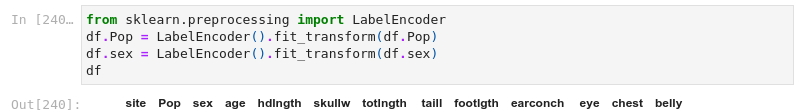
  
  
Рисунок 3.1 - Завантажений датафрейм

## Перетворення даних

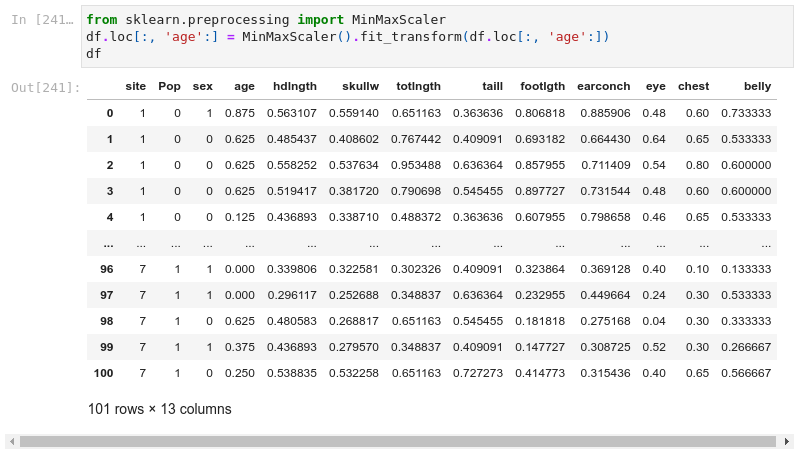
Перевіримо датасет на наявність пустих значень. Як побачимо, пустих значень нема.

  
  
Рисунок 3.2 - Перевірка на присутність пустих значень

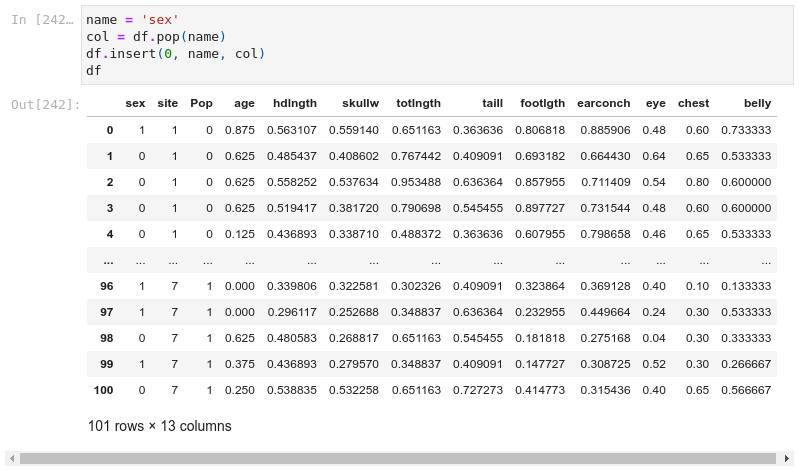
Перетворимо категоріальні змінні у числові за допомогою класу LabelEncoder з модуля sklearn.preprocessing.

  
  
Рисунок 3.3 - Перетворення категоріальних змінних в числові

Виконаємо масштабування даних. Для цього імпортуємо з sklearn.preprocessing клас MinMaxScaler.

  
  
Рисунок 3.4 - Масштабування даних

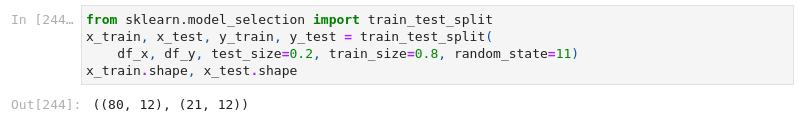
Переставимо колонку "sex" на перше місце.

  
  
Рисунок 3.5 - Переміщення колонки

Розділимо дані на аргументи та значення.

  
  
Рисунок 3.6 - Розділення даних на аргументи та значення

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

  
  
Рисунок 3.7 - Поділ інформації на тренувальну та тестову

## KNN

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

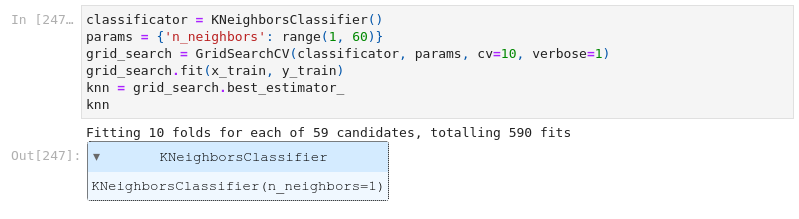
Зберігатимемо результати тестування моделей у списку results.

  
  
Рисунок 3.8 - Список результатів

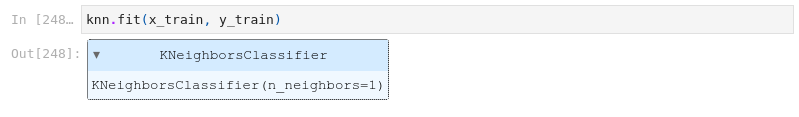
Для виконання роботи методу KNN імпортуємо sklearn.neighbors.KNeighborsClassifier та sklearn.model\_selection.GridSearchCV.

  
  
Рисунок 3.9 - Імпортування модулів

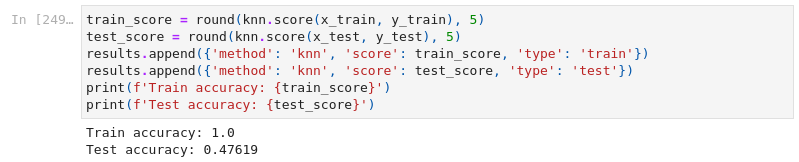
Визначимо, які варіанти параметрів найкраще вирішують дану задачу.

  
  
Рисунок 3.10 - Визначення найкращого параметра

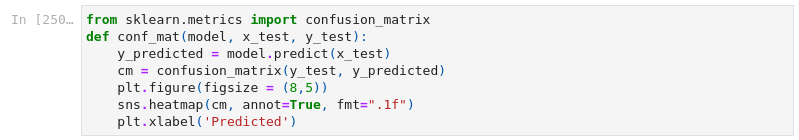
Натренуємо модель з найкращим параметром.

  
  
Рисунок 3.11 - Тренування моделі K-Nearest Neighbors

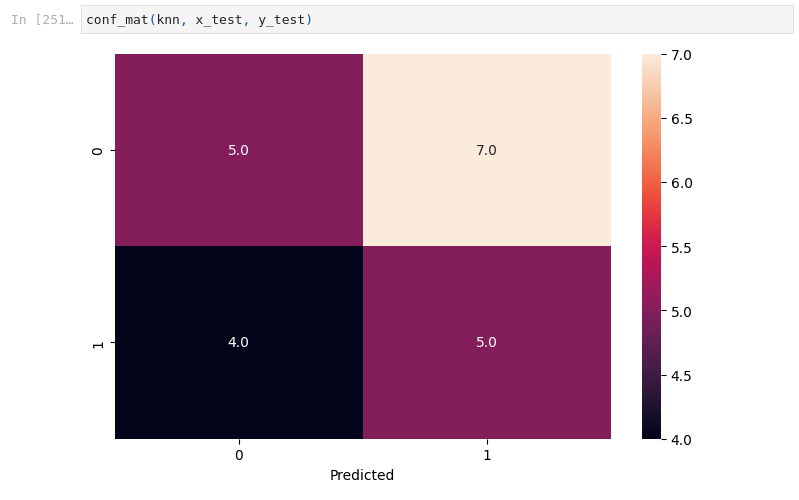
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.12 - Точність моделі K-Nearest Neighbors

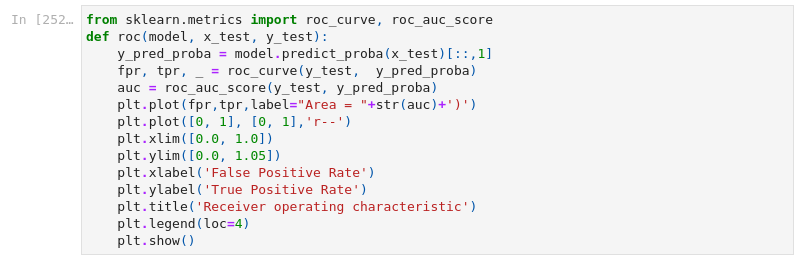
Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей. Для цього застосуємо sklearn.metrics.plot\_confusion\_matrix.

  
  
Рисунок 3.13 - Функція побудови матриці невідповідностей

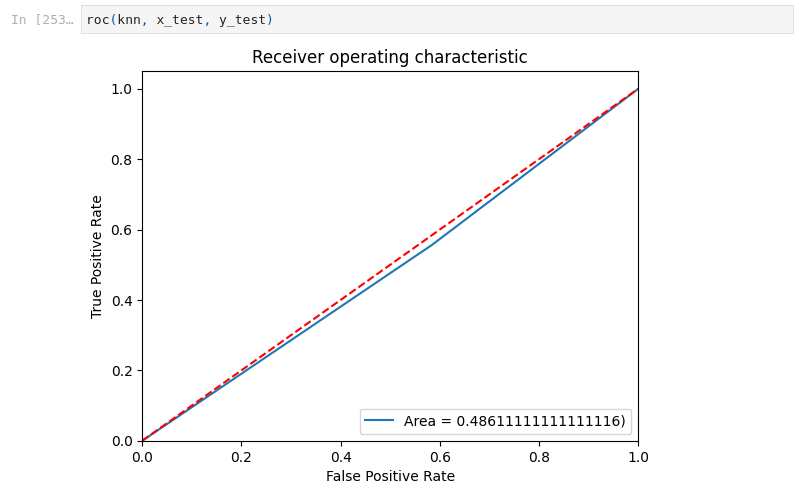
Матриця має два рядки та дві колонки: перший ряд і перша колонка - це істинно позитивні значення: опосум має стать A, і модель визначила цю стать. Перший ряд і друга колонка - хибно позитивні, тобто опосум має стать A, але модель вказала на стать B; другий ряд і перша колонка - хибно негативні, тобто опосум має стать B, але модель вказала на стать A; другий ряд і друга колонка - істинно негативні, тобто опосум має стать B, і модель вказала на стать B.

  
  
Рисунок 3.14 - 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 результати. 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 дає єдине число, яке підсумовує загальну продуктивність.

  
  
Рисунок 3.15 - Імпортування модуля та визначення функції roc

Побудуємо ROC для K-Nearest Neighbors.

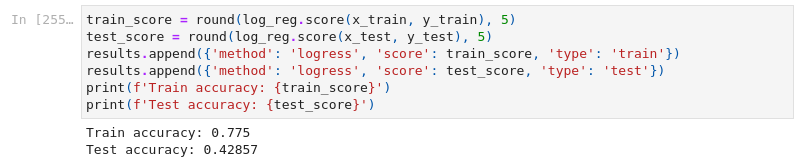
  
  
Рисунок 3.16 - Графік ROC для K-Nearest Neighbors

## Logistic Regression

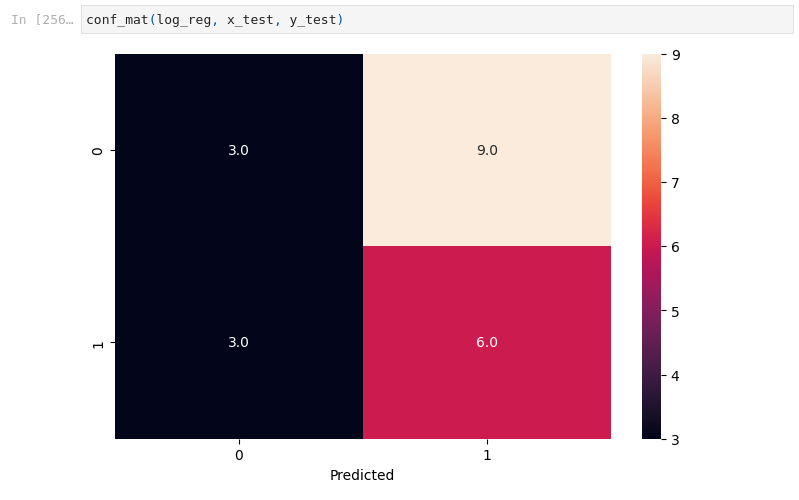
Для виконання роботи методу Logistic Regression імпортуємо sklearn.linear\_model.LogisticRegression. Визначимо найкращі параметри моделі, передавши в неї параметри регуляризації.

  
  
Рисунок 3.17 - Тренування моделі Logistic Regression

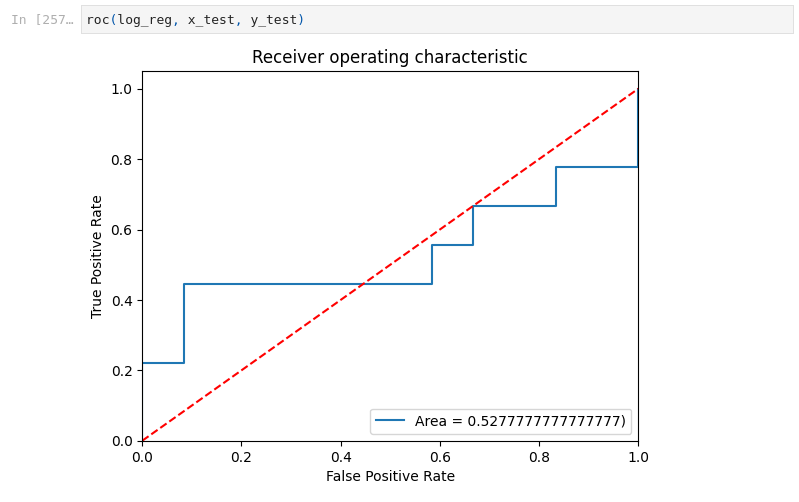
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.18 - Точність моделі Logistic Regression

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

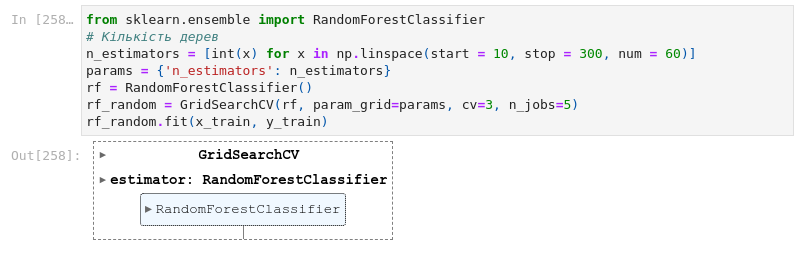
  
  
Рисунок 3.19 - Матриця невідповідностей для Logistic Regression

Побудуємо графік ROC для Logistic Regression.

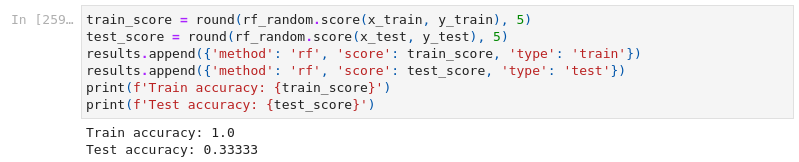
  
  
Рисунок 3.20 - Графік ROC для Logistic Regression

## Random Forest

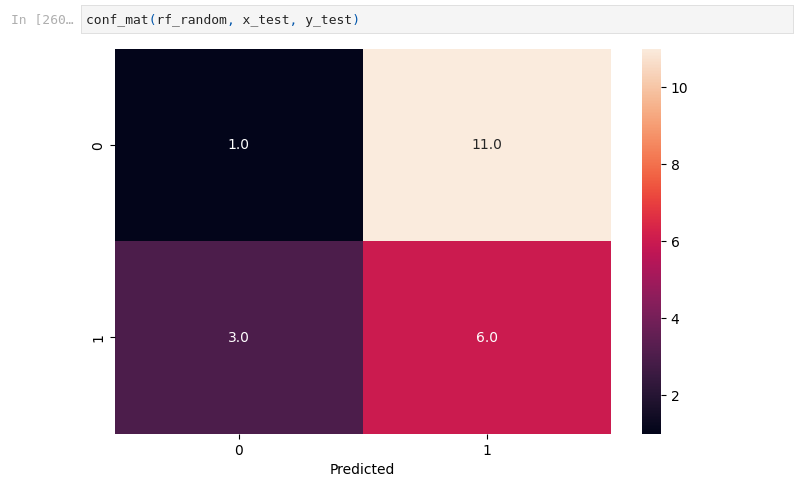
Для виконання роботи методу Random Forest імпортуємо sklearn.ensemble.RandomForestClassifier. Визначимо найкращі параметри для моделі. У випадку Random Forest параметри включають кількість дерев рішень та кількість характеристик, які враховуються кожним деревом під час поділу вузла і використовуються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

  
  
Рисунок 3.21 - Тренування моделі Random Forest

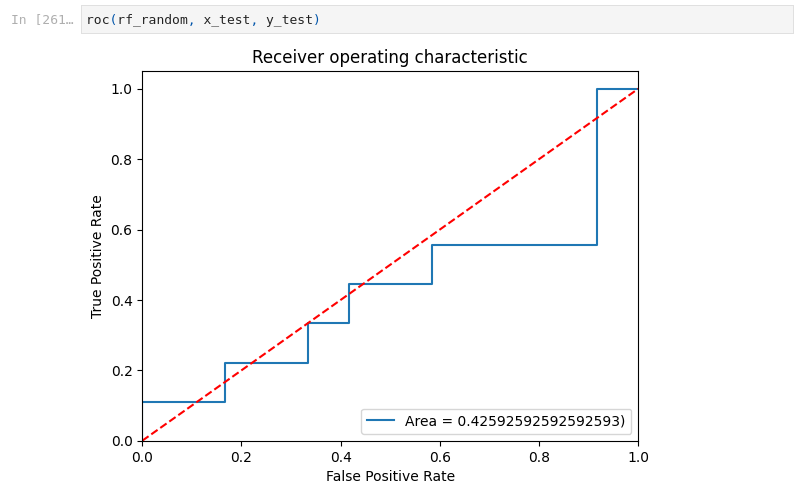
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.22 - Точність моделі Random Forest

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

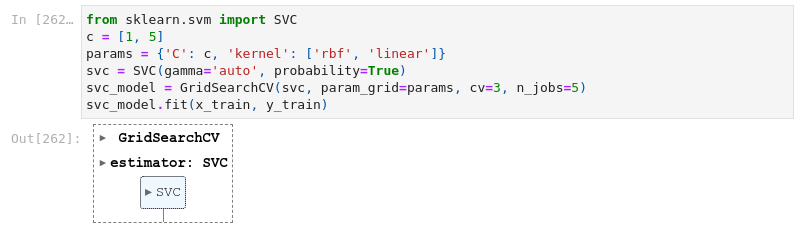
  
  
Рисунок 3.23 - Матриця невідповідностей для Random Forest

Побудуємо графік ROC для Logistic Regression.

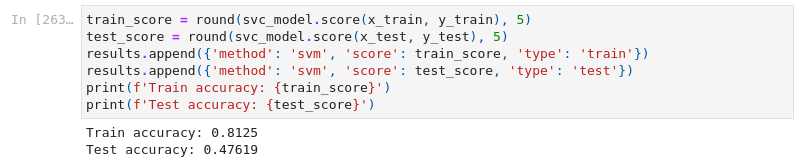
  
  
Рисунок 3.24 - Графік ROC для Random Forest

## SVM

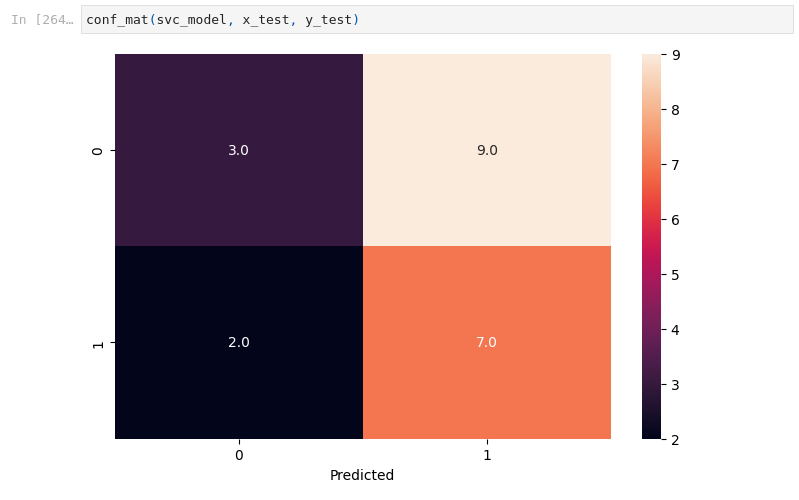
Для виконання роботи методу SVM імпортуємо sklearn.svm.SVC.

  
  
Рисунок 3.25 - Тренування моделі SVM

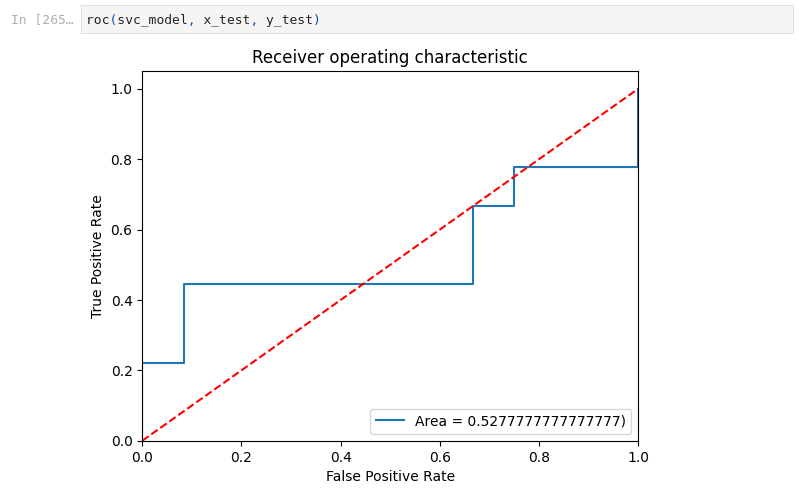
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.26 - Точність моделі SVM

Визначимо продуктивність роботи моделі на прикладі матриці невідповідностей.

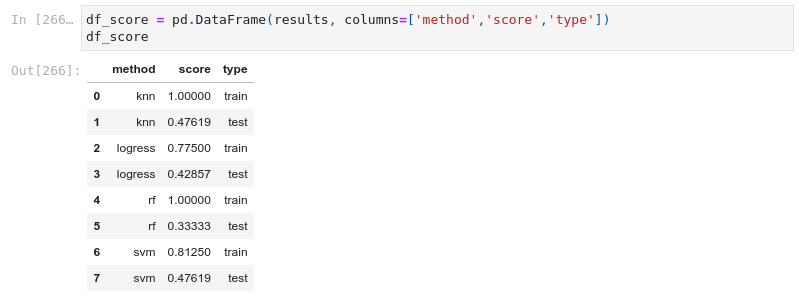
  
  
Рисунок 3.27 - Матриця невідповідностей для SVM

Побудуємо графік ROC для SVM.

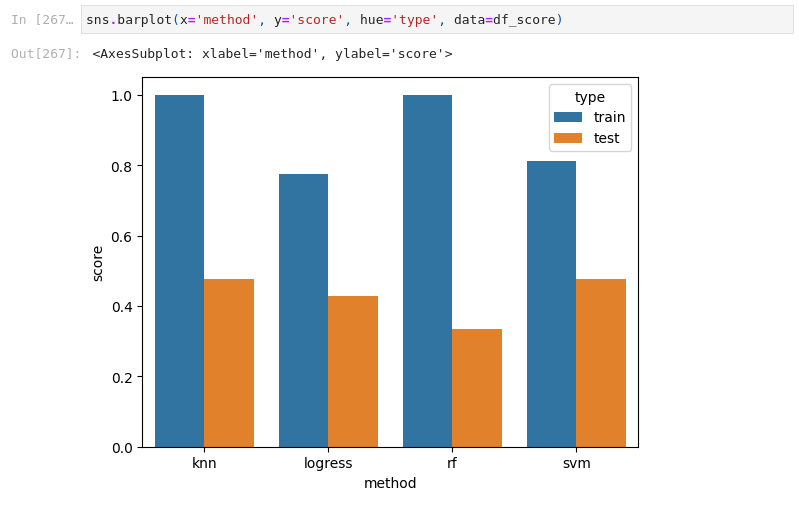
  
  
Рисунок 3.28 - Графік ROC для SVM

## Порівняння результатів

Проаналізувавши окремо кожен із методів, проведемо порівняння даних методів.

  
  
Рисунок 3.29 - Датафрейм результатів

Для наочності побудуємо гістограму.

  
  
Рисунок 3.30 - Результати моделей

Як бачимо, усіма методами отримали доволі низьку точність. Найкраще себе показали KNN та SVM. Найгірше - Random Forest. Такі результати пояснюються тим, що а датасеті надзвичайно мало тренувальних даних, і їхню кількість треба суттєво збільшувати для досягнення вищих показників точності.