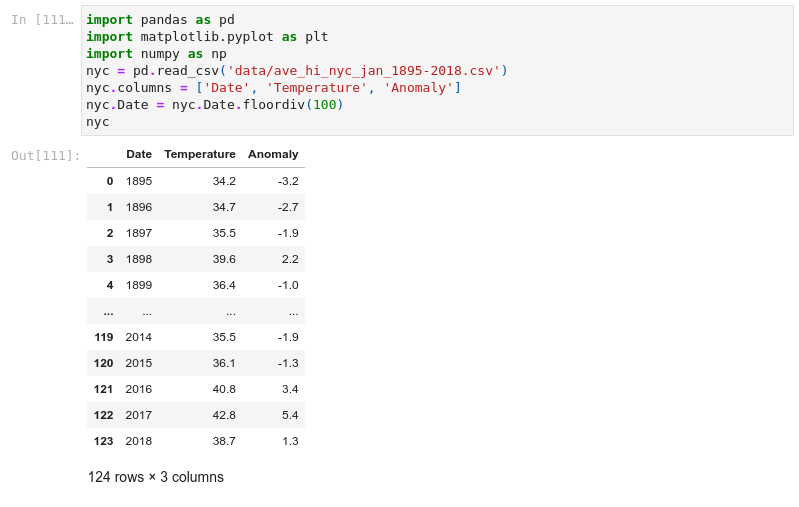
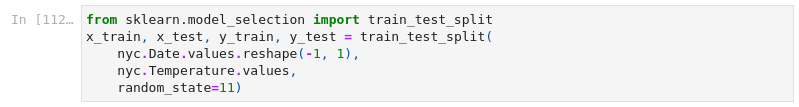
# Виконання

## Повторити дії описані в пункті «Часові ряди і проста лінійна регресія частина 2» даної лабораторної роботи та порівняти з результатом попередньої лабораторної роботи.

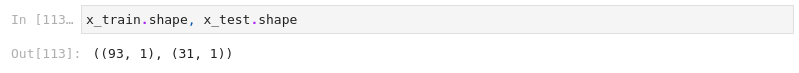
Завантажимо дані з ave\_hi\_nyc\_jan\_1895-2018.csv у датафрейм. Переназвемо колонки та застосуємо цілочисельне ділення, поділивши значенння років на 100.

  
  
Рисунок 3.1 - Завантаження датасету

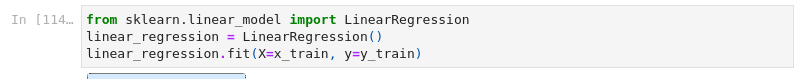
Розіб'ємо дані на навчальні та тестові. Оскільки оцінювачі scikit-learn вимагають, щоб в якості навчальних і тестових даних використовувалися двовимірні масиви, то застосуємо метод reshape та передамо в нього значення -1, 1б щоб перетворити їх з одновимірного масиву з n елементами в двовимірний масив з n рядками і одним стовпцем.

  
  
Рисунок 3.2 - Розбиття даних для навчання і тестування

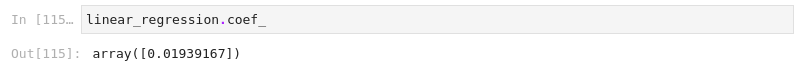
Для перевірки пропорції навчальних і тестових даних (75% до 25%) задамо розміри X\_train і X\_test.

  
  
Рисунок 3.3 - Роміри навчальних і тестових даних (75% до 25%)

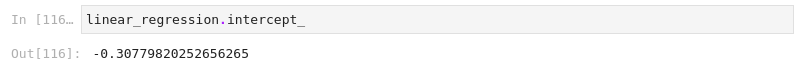
За допомогою оцінювача LinearRegression та простої лінійної регресії, що окремим випадком множинної лінійної регресії, навчимо модель.

  
  
Рисунок 3.4 - Тренування моделі

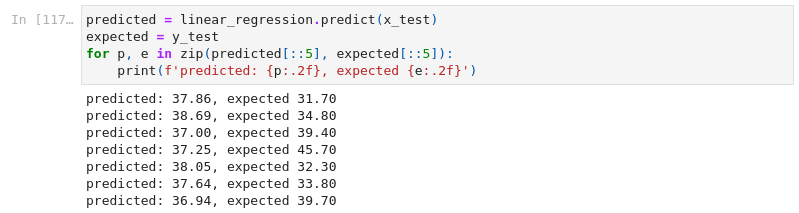
Виведемо кут нахилу, який зберігається в атрибуті coeff\_ оцінювача.

  
  
Рисунок 3.5 - Кут нахилу

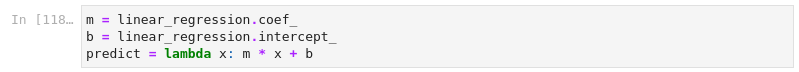
Виведемо точку перетину, яка зберігається в атрибуті intercept\_ оцінювача (m у формулі).

  
  
Рисунок 3.6 - Точка перетину

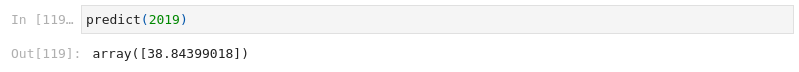
Проведемо тестування моделі за даними з X\_test і перевіримо прогнози по набору даних, виводячи прогнозовані і очікувані значення для кожного п'ятого елементу.

  
  
Рисунок 3.7 - Тестування моделі

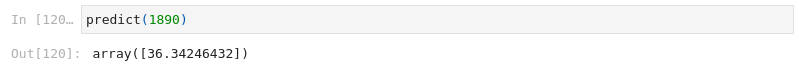
За допомогою кута нахилу і точки перетину зробимо прогнози для середньої температури в січні 2019 року, а також оцінки середньої температури в січні 1890 року.

  
  
Рисунок 3.8 - Лямбда вираз формули y = mx + b

Спрогнозуємо значення за 2019 рік.

  
  
Рисунок 3.9 - Прогноз на 2019 рік

Спрогнозуємо значення за 1890 рік.

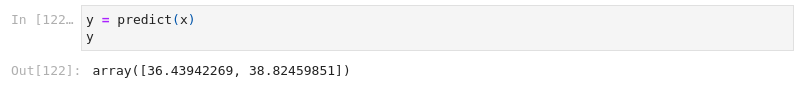
  
  
Рисунок 3.10 - Прогноз на 1890 рік

Візуалізуємо набір даних з регресійними прямими.

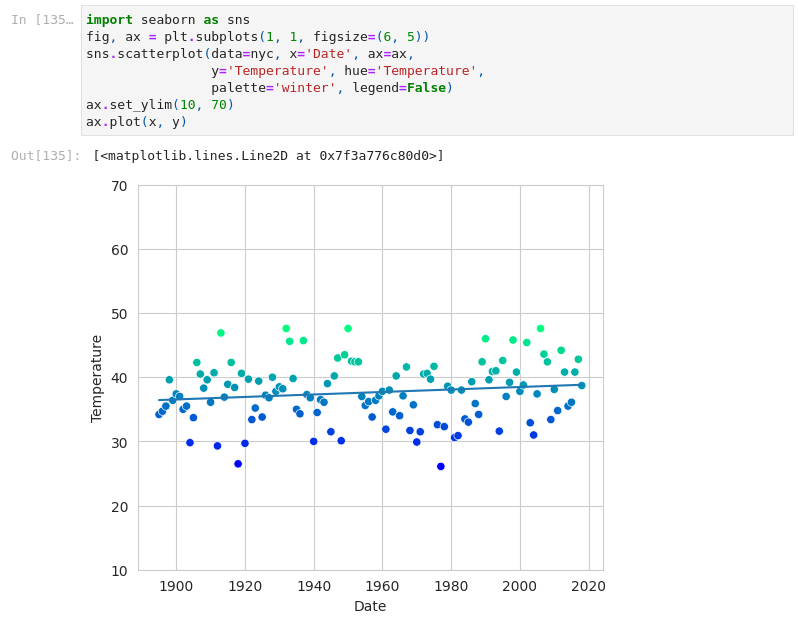
Почнемо зі створення масиву, що містить мінімальні і максимальні значення дати з nyc.Date. Вони стануть координатами x початкової і кінцевої точок регресійної прямої.

  
  
Рисунок 3.11 - Масив даних

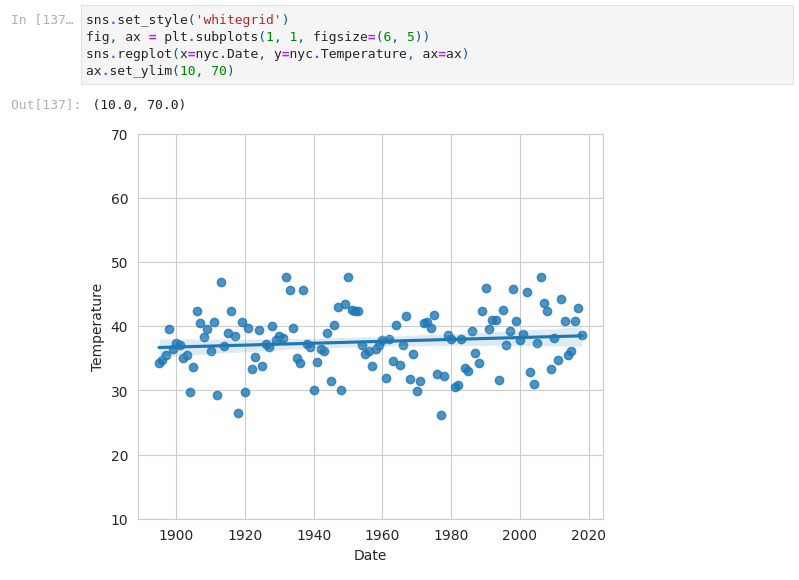
Передамо масив x функції predict та отримаємо пронозовані значення, які будуть використовуватися в якості координат y.

  
  
Рисунок 3.12 - Спрогнозовані значення

Побудуємо діаграму розкиду даних за допомогою функції scatterplot бібліотеки Seaborn і функції plot бібліотеки Matplotlib. Для виведення точок даних скористаємося методом scatterplot з колекцією DataFrame з ім'ям nyc. Змінимо масштаб осі. Зобразимо регресію.

  
  
Рисунок 3.13 - Візуалізація з допомогою LinearRegression

Поглянемо на візуалізацію з минулої лабораторної та переконаємося, що обидві ідентичні.

  
  
Рисунок 3.14 - Візуалізація минулої лабораторної

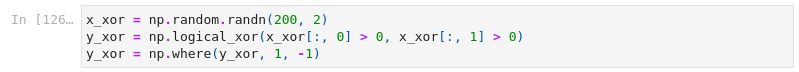
Як бачимо, візуалізації практично ідентичні.

## з прикладом з лекції 7 згенеруйте набір даних та класифікуйте його використавши класифікатор SVC (слайд 95)

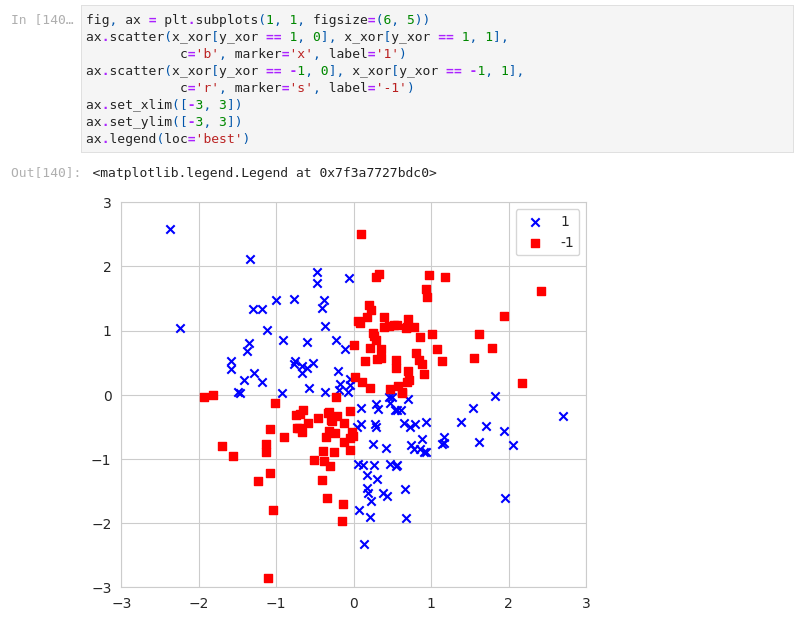
Для початку імпортуємо ListedColormap з matplotlib.colors та класифікатор SVC з sklearn.svm.

  
  
Рисунок 3.15 - Імпортування модулів

За допомогою бібліотеки NumPy згенеруємо набір даних, передавши початковий seed 1 та використавши функції np.random.randn( згенерує матрицю з 200-ми рядками та 2-ма стовпцями ), np.logical\_xor( застосовує операцію виключного або ), np.where( приймає в себе логічну операцію xor та повертає значення 1, якщо True, -1, якщо False).

  
  
Рисунок 3.16 - Генерація даних

Зобразимо згенеровані дані. За допомогою функції scatter, спочатку синіми хрестиками зобразимо ті точки, для яких y\_xor == 1. Потім ті точки, для яких y\_xor == -1. Значення 0 та 1 у других індексах - це відповідно порядкові номери стовпчиків.

  
  
Рисунок 3.17 - Візуалізація згенерованих даних

Запишемо функцію plot\_decision\_regions, яка приймає в параметри: X - двовимірний масив аргуметів, y - одновимірний масив значень, test\_idx=None - тестовий індекс, resolution=0.02 - масштаб.

Рефактиремо функцію, надану в лекції. "unq" - масив унікальних значень "y", а "unql" - довжина "unq".

Викличемо IndexError, якщо передані масиви "markers" або "colors" менші за разміром ніж "unq".

cmap - екземпляр класу ListedColormap, у який ми передали кольори до індекса "unql"; x1\_min, x1\_max - відповідно мінімальне і максимальне значення значення першого стовпчика, зміщені на 1 для кращої видимості; x2\_min, x2\_max - для другого стовпчика відповідно.

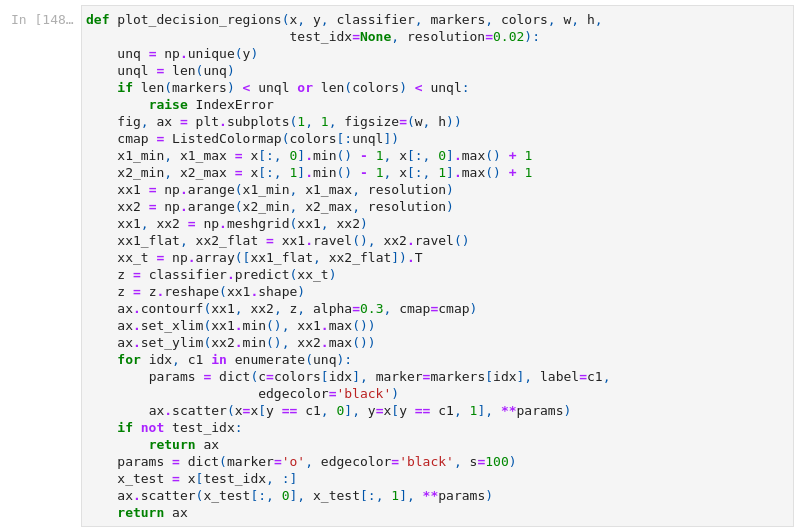
xx1, xx2 - масиви рівномірно розподілених значень між мінімальним та максимальним з кроком в resolution за допомогою функції numpy.arange; xx1, xx2 - це перетворені попередні xx1 та xx2 за допомогою функції numpy.meshgrid, яка робить сітку індексів з одновимірних масивів; xx1\_flat, xx2\_flat - сплющені до одновимірних двовимірні масив xx1, xx2 за допомогою методу ravel; xx\_t - матриця, стовпчиками якої є xx1\_flat та xx2\_flat.

z - спрогнозовані класифікатором значення, у метод predict якого передаємо xx\_t.

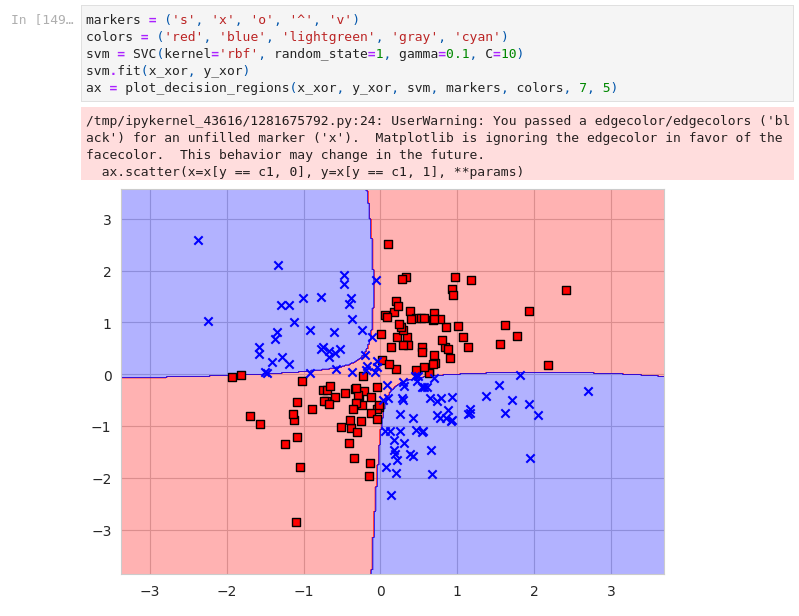
За допомогою plt.contourf зображаємо контури класів, передаючи у функцію xx1, xx2 та кольорову мапу. За допомогою xlim та ylim, у які передаємо мінімальні та максимальні значення, обмежуємо візуалізацію для зручного сприйняття.

У циклі для кожного класу вимальовуємо свої точки, передаючи у функцію scatter їхню позицію, колір, маркер, кольор контуру, позначку.

Якщо тестовий індекс не пустий, то вимальовуємо точки з x\_test.

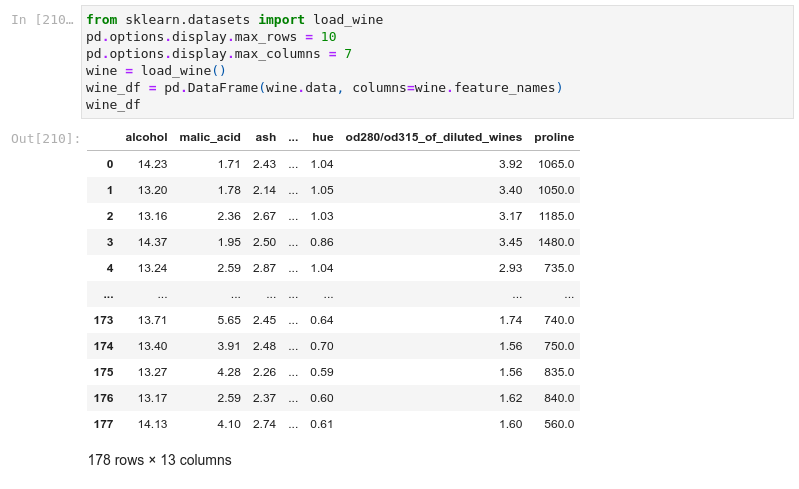
  
  
Рисунок 3.18 - Функція plot\_decision\_regions

Зобразимо результати.

  
  
Рисунок 3.19 - Візуалізація роботи класифікатора SVC

## декілька класифікаційних оцінювачів наприклад KNeighborsClassifier, SVC та GaussianNB для вбудованого в scikit-learn одного набору даних (вибрати довільний за бажанням)

Обиремо датасет вин та виведемо його.

  
  
Рисунок 3.20 - Датасет вин

Розділемо датасет на тренувальну та тестові частини. Нехай буде 80% тренувальних та 20% тестових даних.

  
  
Рисунок 3.21 - Навчальні та тестові дані

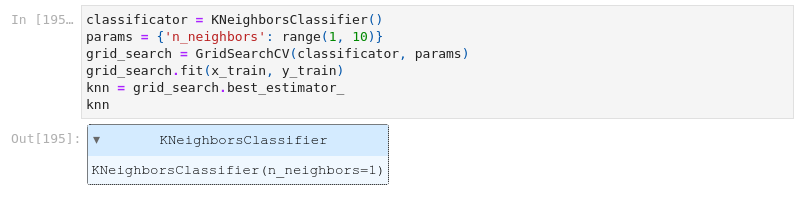
Ініціалізуємо список results, у який будемо додавати результати моделей.

  
  
Рисунок 3.22 - Список результатів моделей

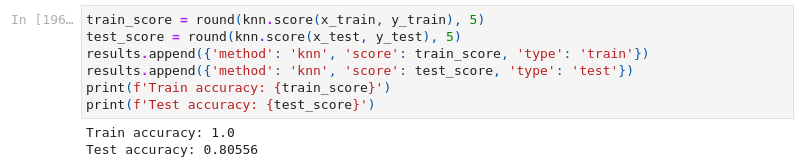
Спочатку застосуємо модель K-Nearest Neighbors та за допомогою GridSearchCV знайдемо оптимальну модель.

  
  
Рисунок 3.23 - Імпортування модулів

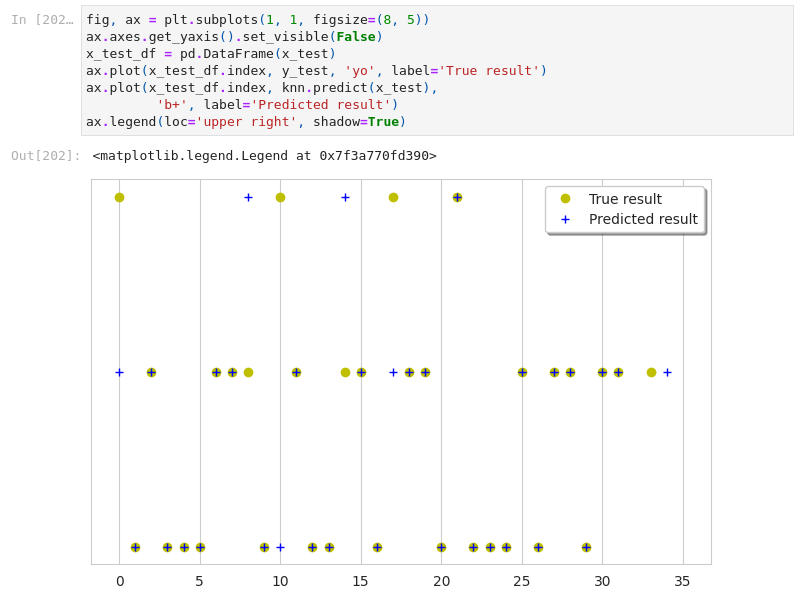
Визначимо, які варіанти параметрів найкраще вирішують дану задачу, підбираючи оптимальну кількість сусідів.

  
  
Рисунок 3.24 - Визначення найкращого параметра

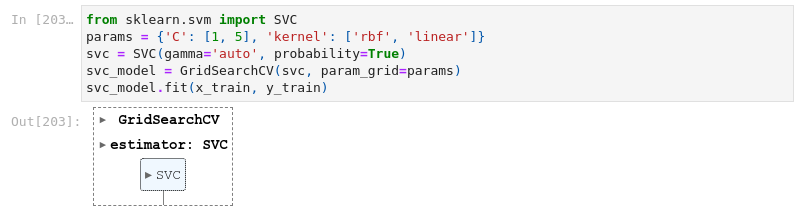
Визначимо точність моделі на тренувальних та тестових даних

  
  
Рисунок 3.25 - Точність моделі K-Nearest Neighbors

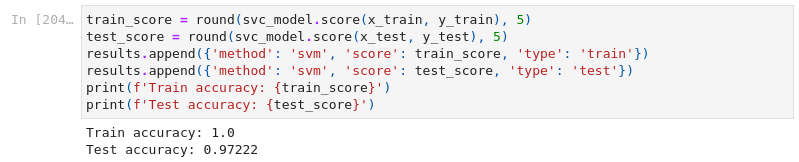
Візуалізуємо отримані результати. Побудуємо графік, де маємо cині плюси - прогнозовані значення, жовті кружечки - фактичні значення. Якщо жовті кружечки не закриваються синіми плюсами, то це означає, що в цих місцях модел допустила помилку.

  
  
Рисунок 3.26 - Візуалізація точності результатів K-Nearest Neigbors

Далі застосуємо модель SVC. Ядрами для SVC будуть "rbf", "linear". Задамо параметр C, який контролює трейд-офф між гладкістю вирішальної границі та правильною класифікацією точок. Де чим більше С, тим краще класифікуються точки.

  
  
Рисунок 3.27 - Тренування моделі SVM

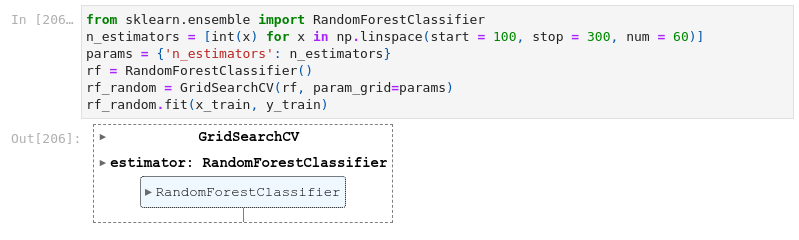
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.28 - Точність моделі SVM

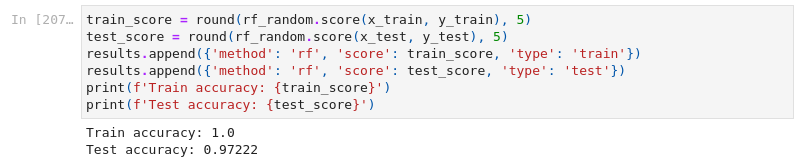
Візуалізуємо отримані результати.

  
  
Рисунок 3.29 - Візуалізація точності результатів SVC

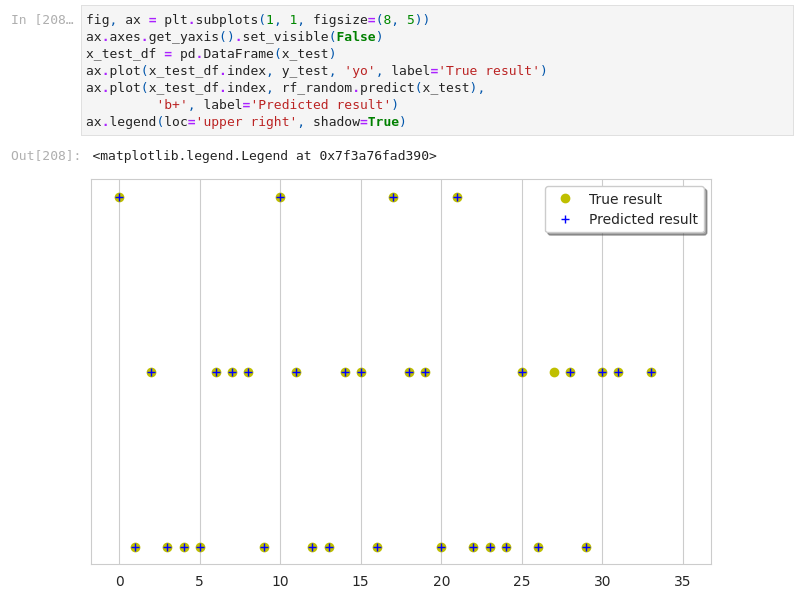
Далі застосуємо модель Random Forest. n-estimators - кількість дерев у лісі. За допомогою функції linspace утворимо рівномірно розподілений список кількость дерев у заданих межах.

  
  
Рисунок 3.30 - Тренування моделі Random Forest

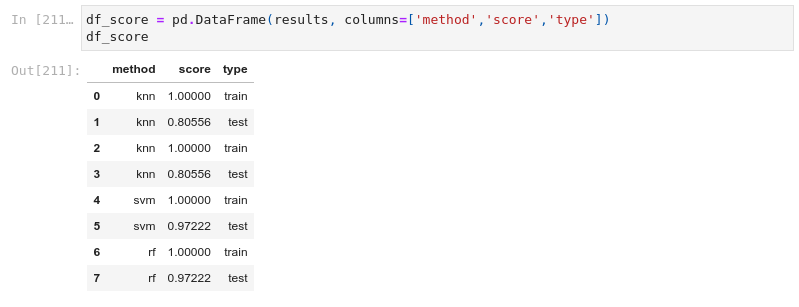
Визначимо точність моделі на тренувальних та тестових даних.

  
  
Рисунок 3.31 - Точність моделі Random Forest

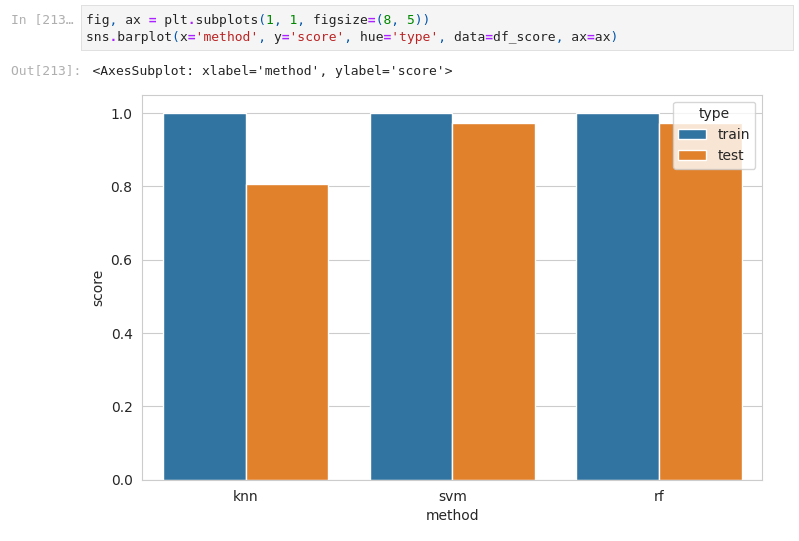
Візуалізуємо отримані результати.

  
  
Рисунок 3.32 - Візуалізація точності результатів Random Forest

Порівняємо отримані результати моделей. Утворимо датафрейм результатів.

  
  
Рисунок 3.33 - Датафрейм результатів

Для наочності побудуємо гістограму.

  
  
Рисунок 3.34 - Результати моделей

З огляду бачимо, що на тренувальних даних усі методи відпрацювали однаково. Однак на тестових даних KNN показує себе найгірше. Бачимо, що SVC та RandomForest показали себе однаково на тестових даних, але треба зауважити, що вибірка була доволі малою, тому може бути не видно суттєвої різниці.