**ДАТАСЕТ**

**DIABETES DATASET**

Цей набір даних отримано від Національного інституту діабету, захворювань органів травлення та нирок. Мета полягає в тому, щоб передбачити на основі діагностичних вимірювань, чи є у пацієнта діабет.

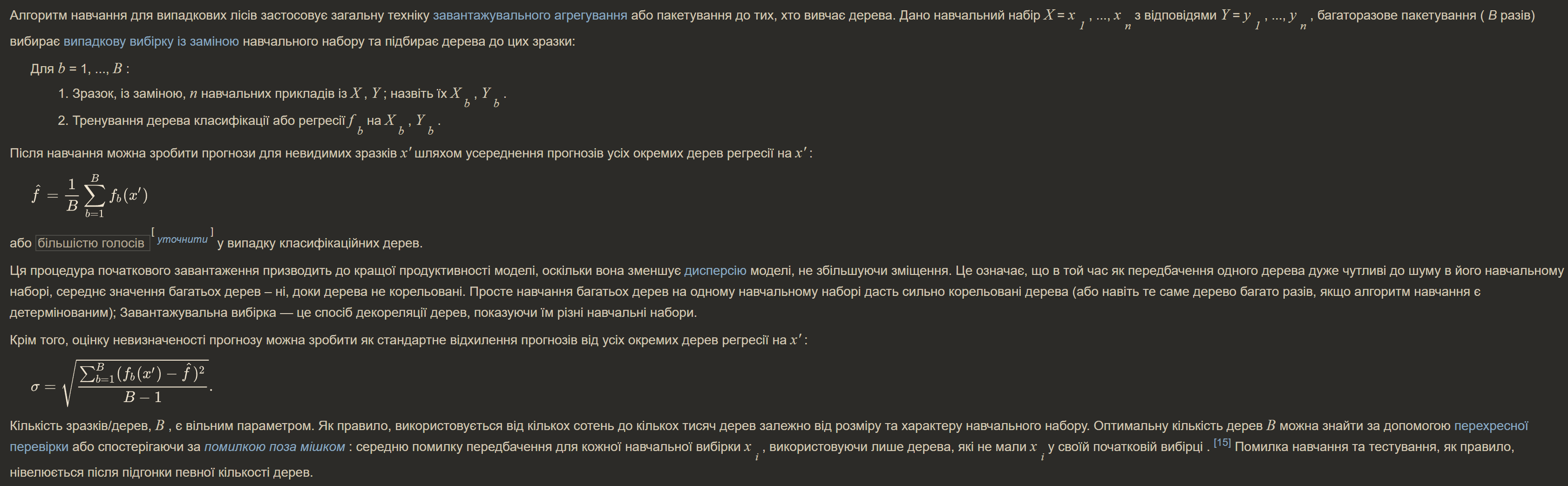
На вибір цих екземплярів із більшої бази даних було накладено кілька обмежень. Зокрема, усі пацієнти тут — жінки віком не менше 21 року, індіанці піма.

* Вагітності: кількість вагітностей
* Глюкоза: концентрація глюкози в плазмі протягом 2 годин під час перорального тесту на толерантність до глюкози
* Артеріальний тиск: діастолічний артеріальний тиск (мм рт. ст.)
* SkinThickness: Товщина шкірної складки трицепса (мм)
* Інсулін: 2-годинний сироватковий інсулін (мкОд/мл)
* ІМТ: індекс маси тіла (вага в кг/(зріст у м)^2)
* DiabetesPedigreeFunction: функція діабету
* Вік: Вік (років)
* Результат: змінна класу (0 або 1)

**НЕЙРОННА МЕРЕЖА**

**RANDOM FORESTS**

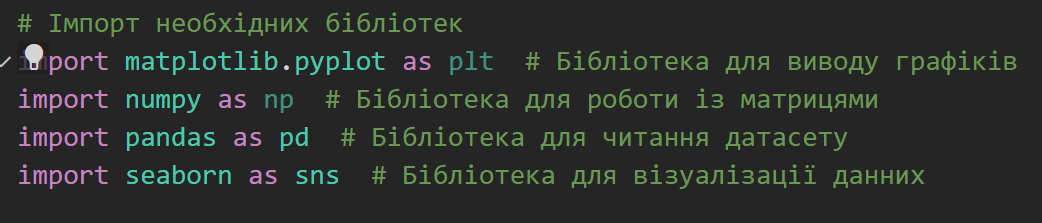
Random forests - це метод ансамблевого навчання для класифікації , регресії та інших завдань, який працює шляхом побудови безлічі дерев рішень під час навчання. Для класифікаційних завдань результатом випадкового лісу є клас, вибраний більшістю дерев. Для завдань регресії повертається середнє або середнє прогнозування окремих дерев. [1] [2] Випадкові ліси рішень виправляють звичку дерев рішень переобладнувати свій навчальний набір . [3] : 587–588  Випадкові ліси загалом перевершують дерева рішень, але їхня точність нижча, ніж дерева з посиленням градієнта. [ потрібне цитування ] Однак характеристики даних можуть впливати на їх ефективність.



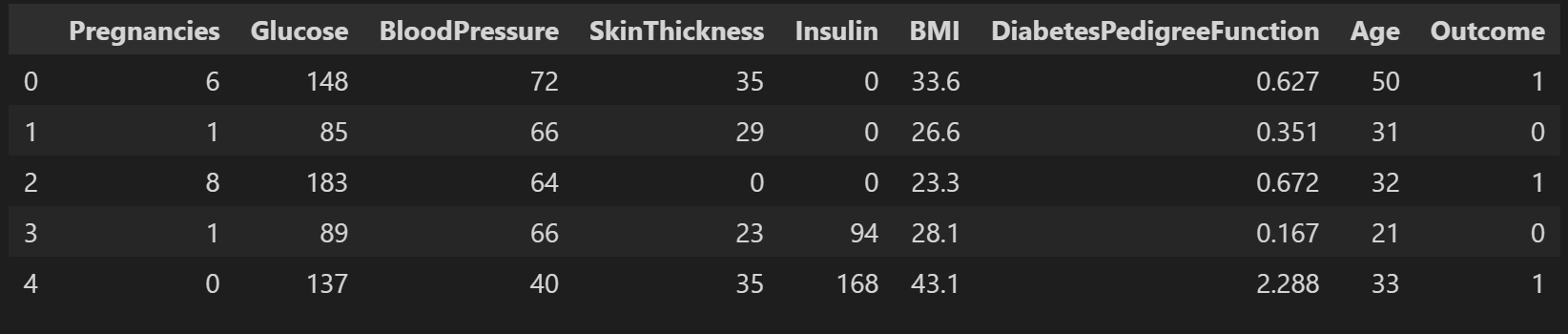
**ПРОЕКТ**

**Файл diabetics-pred-xgb.ipynb**

**Бібліотеки**



**Читаємо датасет**



**Модель**

from sklearn.model\_selection import train\_test\_split

Розбиваємо датасет або матриці на випадкові потяги та тестові підмножини.

train\_test\_split - Швидка утиліта, яка об’єднує перевірку введення та програму для введення даних в один виклик для розділення (і додаткової вибірки) даних у єдиному рядку

**Конфігурація моделі**

from xgboost.sklearn import XGBClassifier

Менеджер контексту для глобальної конфігурації XGBoost.

Глобальна конфігурація складається з набору параметрів, які можна застосувати в глобальній області. Повний список параметрів, які підтримуються в глобальній конфігурації.

Матриця даних, яка використовується в XGBoost.

DMatrix — це внутрішня структура даних, яка використовується XGBoost і оптимізована як для ефективності пам’яті, так і для швидкості навчання. Ви можете створити DMatrix з кількох різних джерел даних.

**Параметри**

дані ( os.PathLike/string/numpy.array/scipy.sparse/pd.DataFrame/ ) –

dt.Frame/cudf.DataFrame/cupy.array/dlpack/arrow.Table

Джерело даних DMatrix.

Якщо дані мають тип рядка або os.PathLike, вони представляють шлях до txt-файлу формату libsvm, файлу csv (за допомогою параметра URI 'path\_to\_csv?format=csv') або двійкового файлу, з якого xgboost може читати.

* label ( array\_like ) – Мітка навчальних даних.
* вага ( array\_like ) – Вага кожного екземпляра.

Примітка

Для завдання ранжирування ваги є для кожної групи.

У завданні ранжирування кожній групі (а не кожній точці даних) призначається одна вага. Це тому, що ми дбаємо лише про відносне впорядкування точок даних у кожній групі, тому немає сенсу призначати ваги окремим точкам даних.

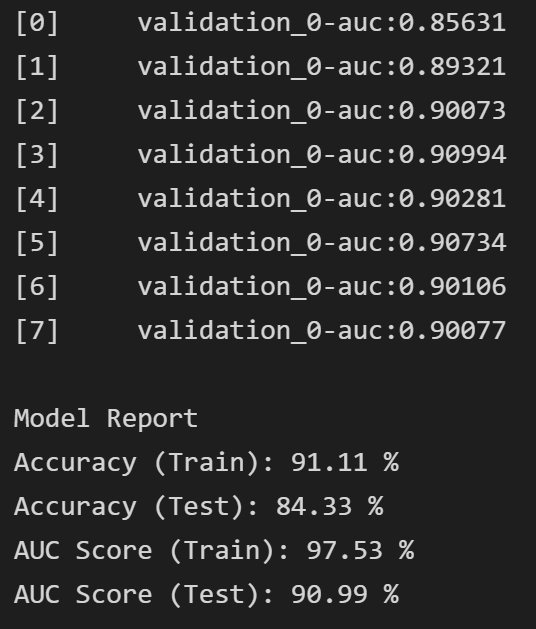
* base\_margin ( array\_like ) – базова маржа, яка використовується для збільшення від існуючої моделі.
* missing ( float , необов’язковий ) – значення у вхідних даних, яке повинно бути присутнім як відсутнє значення. Якщо немає, за замовчуванням np.nan.
* silent ( boolean , необов’язковий ) – чи друкувати повідомлення під час створення
* feature\_names ( список , необов’язково ) – Установіть назви для функцій.
* feature\_types ( FeatureTypes ) – установіть типи для функцій. Коли enable\_categorical має значення True , рядок «c» представляє категоріальний тип даних, тоді як «q» представляє числовий тип ознаки. Для категоріальних ознак передбачається, що вхідні дані попередньо оброблені та закодовані користувачами. Кодування можна виконати за допомогою методу .cat.codes sklearn.preprocessing.OrdinalEncoderабо pandas dataframe . Це корисно, коли користувачі хочуть вказати категоріальні ознаки без необхідності створювати фрейм даних як вхідні дані.
* nthread ( integer , необов’язковий ) – кількість потоків для завантаження даних, якщо застосовне розпаралелювання. Якщо -1, використовується максимальна кількість потоків, доступних у системі.
* group ( array\_like ) – розмір групи для всіх ранжируваних груп.
* qid ( array\_like ) – ідентифікатор запиту для зразків даних, що використовується для ранжирування.
* label\_lower\_bound ( array\_like ) – нижня межа для навчання виживання.
* label\_upper\_bound ( array\_like ) – верхня межа для навчання виживання.
* feature\_weights ( array\_like , необов’язково ) – встановіть ваги ознак для вибірки стовпців.
* enable\_categorical ( логічне значення , необов’язково ) – Нове у версії 1.3.0.

Примітка

Цей параметр є експериментальним

Експериментальна підтримка спеціалізації за категоріальними ознаками. Не встановлюйте значення True, якщо ви не зацікавлені в розвитку. Також потрібен формат серіалізації JSON/UBJSON.

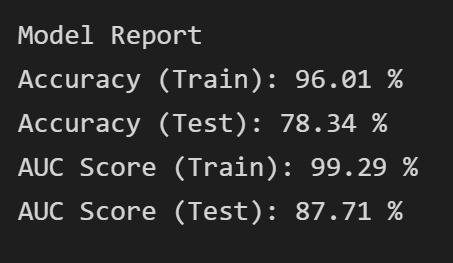
Змінюючи ці параметри пробуємо досягнути максимального результату, у нашому випадку це вийшло



**Оптимізація**

Optuna — це програма для автоматичної оптимізації гіперпараметрів, спеціально розроблена для машинного навчання. Він має імперативний API користувача в стилі визначення за виконанням.

Після оптимізації результати нейронної мережі наступні:

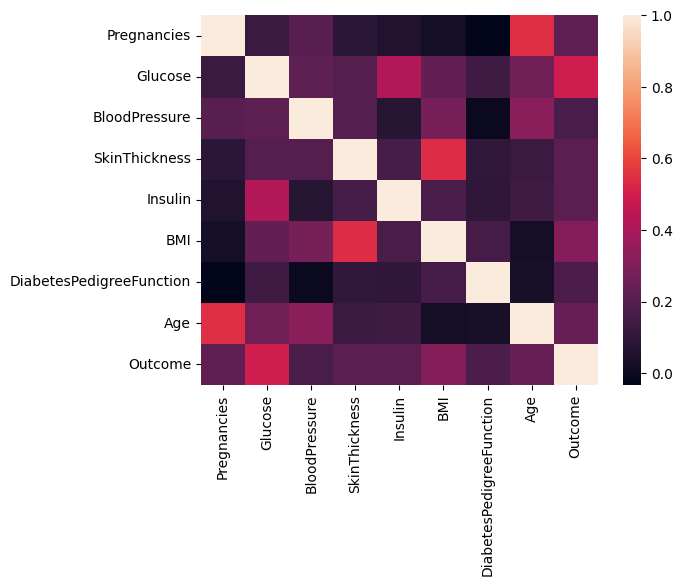


**Файл diabetics-prediction.ipynb**

**Додаткова бібліотека Seaborn**

Seaborn — це бібліотека візуалізації даних Python на основі matplotlib . Він забезпечує інтерфейс високого рівня для малювання привабливої ​​та інформативної статистичної графіки.

Аналіз данних (кореляція стовпців) датасету



**seaborn.pairplot**

Побудуємо парні зв’язки в наборі даних.

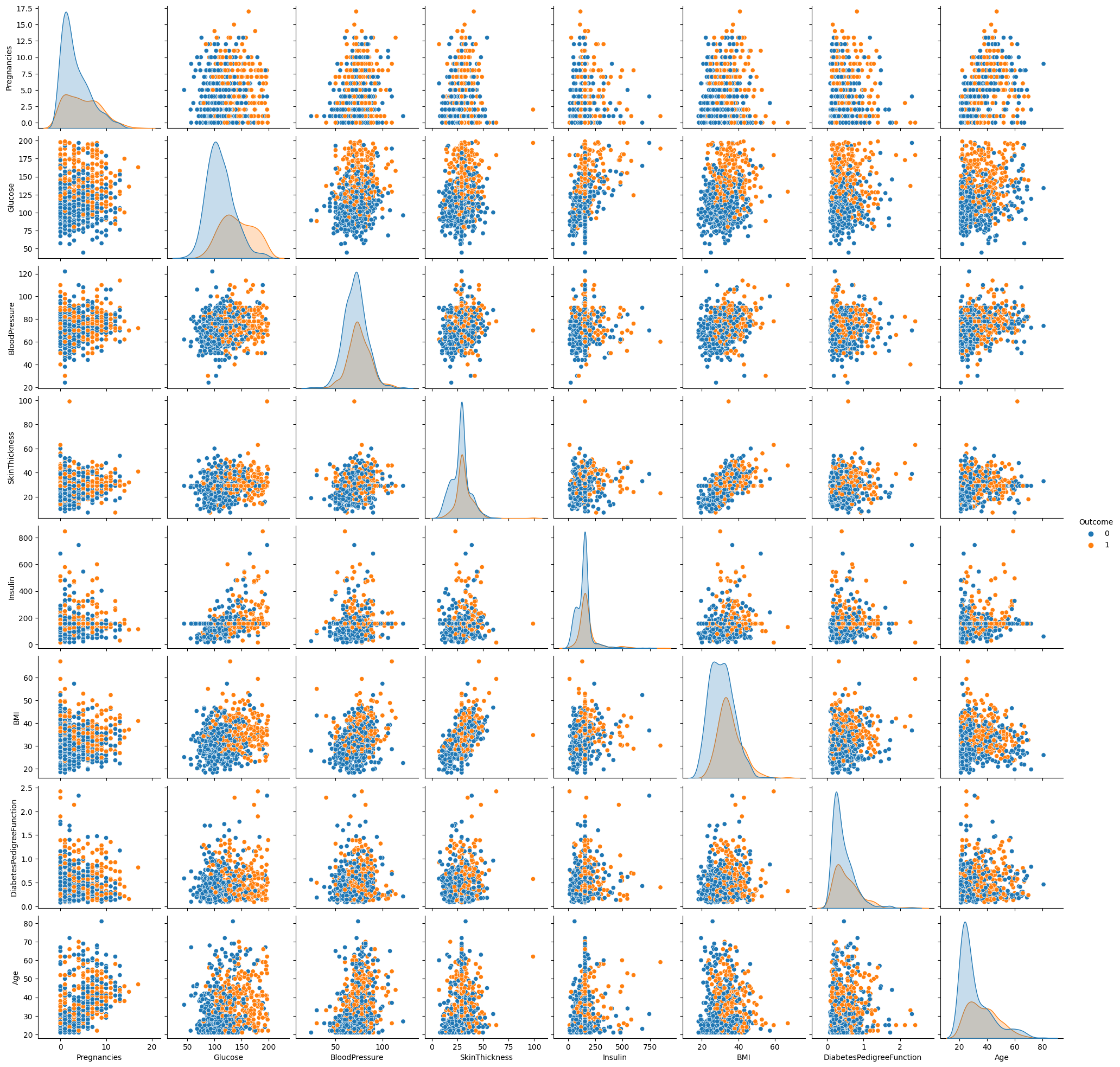
За замовчуванням ця функція створить сітку осей так, що кожна числова змінна data розподілятиметься між осями y в одному рядку та осями x в одному стовпці. Діагональні графіки обробляються по-іншому: малюється однофакторний графік розподілу, щоб показати граничний розподіл даних у кожному стовпці.

Параметри даних pandas.DataFrame

Охайний (довгий) фрейм даних, де кожен стовпець є змінною, а кожен рядок – спостереженням.

* hue ім'я змінної в data
* Змінна data для відображення аспектів графіка в різних кольорах.

Результат



Створюємо нову мережу та тренеруємо її за допомогою

from sklearn.preprocessing import м

from sklearn.neighbors import KNeighborsClassifier

StandardScaler - Стандартизування функції, видаливши середнє значення та масштабуючи дисперсію одиниць.

Стандартний бал зразка розраховується як:

z = (x - u) / с

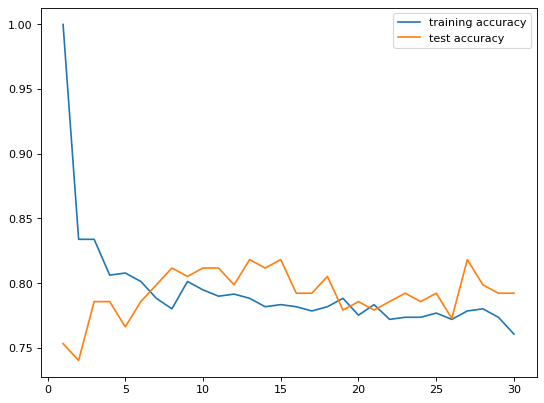
де u – середнє значення навчальних вибірок або нуль, якщо with\_mean=False, і sстандартне відхилення навчальних вибірок або одиниця, якщо with\_std=False.

Центрування та масштабування відбуваються незалежно для кожної функції шляхом обчислення відповідних статистичних даних на зразках у навчальному наборі. Середнє значення та стандартне відхилення потім зберігаються для подальшого використання в даних за допомогою transform.

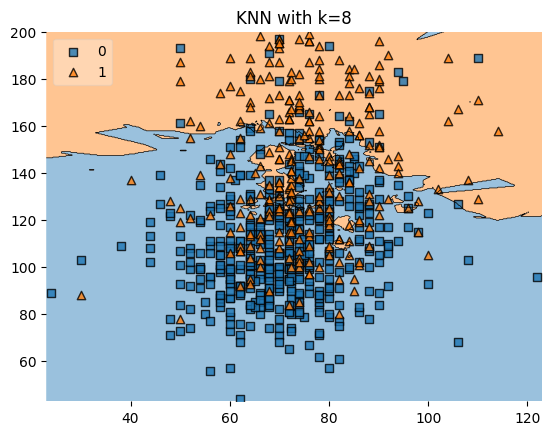
Стандартизація набору даних є загальною вимогою для багатьох оцінювачів машинного навчання: вони можуть поводитися погано, якщо окремі функції більш-менш не схожі на стандартні дані з нормальним розподілом (наприклад, гауссове значення з нульовим середнім і одиничною дисперсією).

KNeighborsClassifier - Класифікатор, що реалізує k-голосування найближчих сусідів.

Графік тренування моделі:



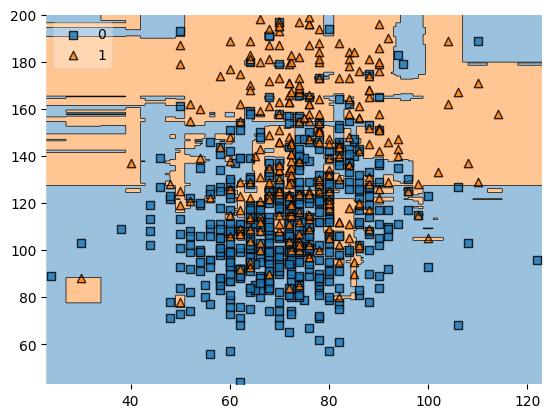
plot\_decision\_regions: Візуалізація області прийняття рішень класифікатора



from sklearn.ensemble import RandomForestClassifier

Із бібліотеки sklearn імпортуємо уже готову модель andomForest

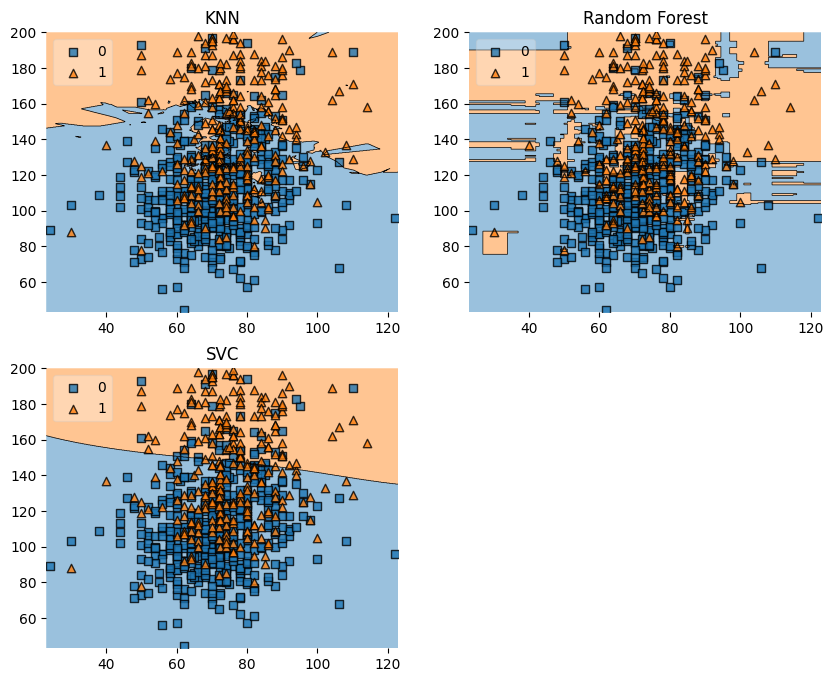
Візуалізуємо області прийняття рішень цієї моделі:



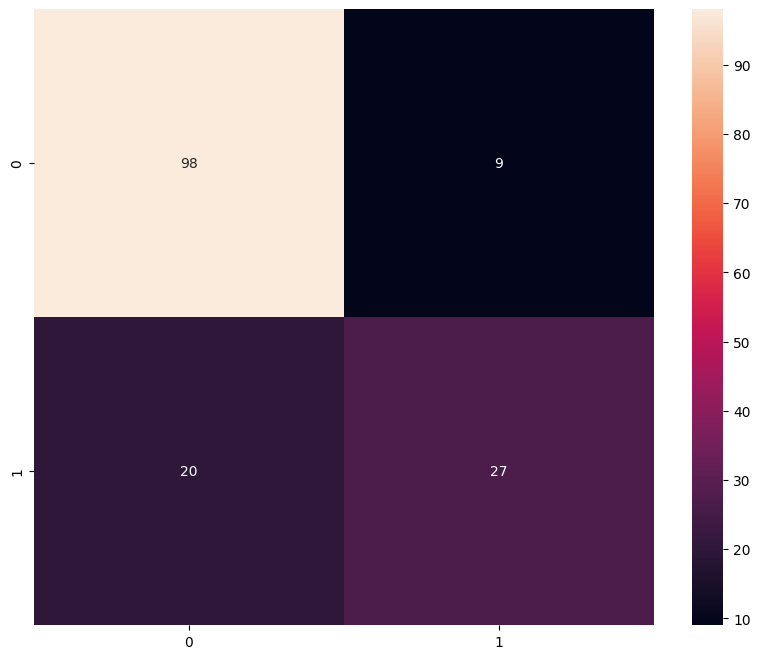
sklearn.svm.SVC - Опорний вектор класифікації.

Реалізація базується на libsvm. Час підгонки змінюється принаймні квадратично залежно від кількості зразків і може бути непрактичним за межі десятків тисяч зразків. Для великих наборів даних розгляньте можливість використання LinearSVCабо SGDClassifierзамість нього, можливо, після Nystroemтрансформатора.

Зрівняємо графіки цих 3 класифікаторів



Обчислюємо матрицю помилок, щоб оцінити точність класифікації.



Обчислюєио робочу характеристику приймача

