АНСАМБЛІ МОДЕЛЕЙ (ПРОДОВЖЕННЯ)

Надія І. Недашківська n.nedashkivska@gmail.com

Види ансамблів моделей

Ансамбль – набір моделей, які сумісно застосовуються для розв'язання єдиної задачі.

Мета – підвищити якість (точність) рішень

1. Вибір базової моделі

Однорідний ансамбль

Складається з базових моделей **одного типу**, наприклад тільки моделей нейронних мереж або дерев рішень.

Види ансамблів моделей

Ансамбль з моделей різного типу

Складається з моделей різного типу, наприклад, нейронних мереж, дерев рішень, регресійних моделей тощо.

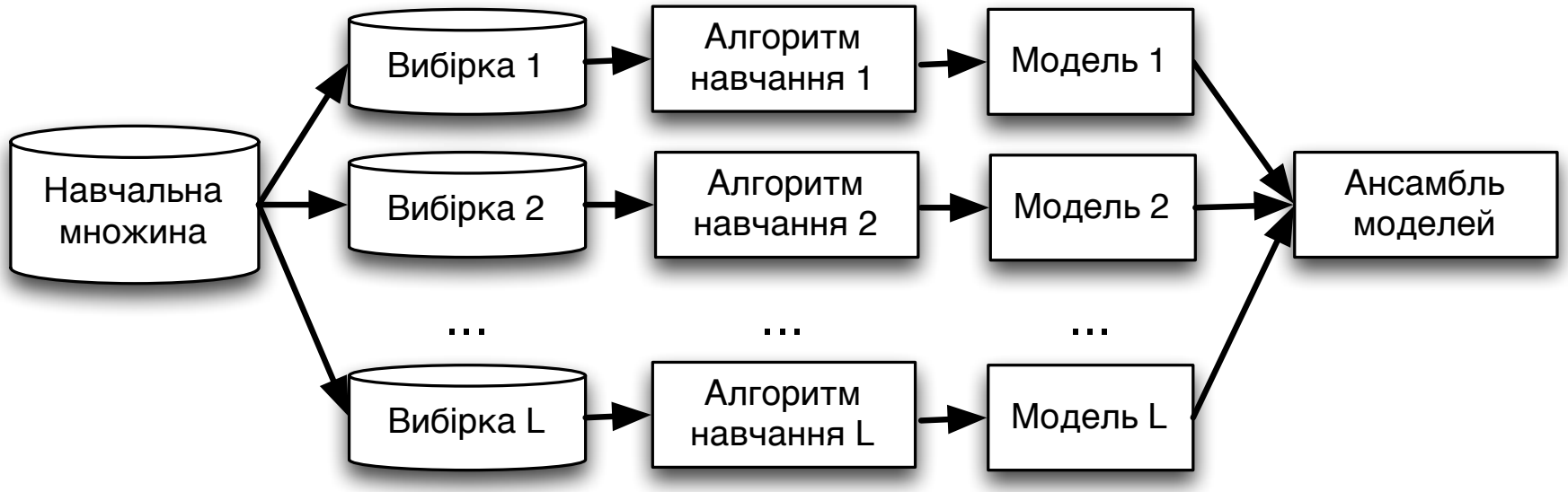
Перевага – додаткова гнучкість.

Недолік – необхідні додаткові перетворення для узгодження входів і виходів моделей різного типу.

Види ансамблів моделей

2. Використання навчальної множини

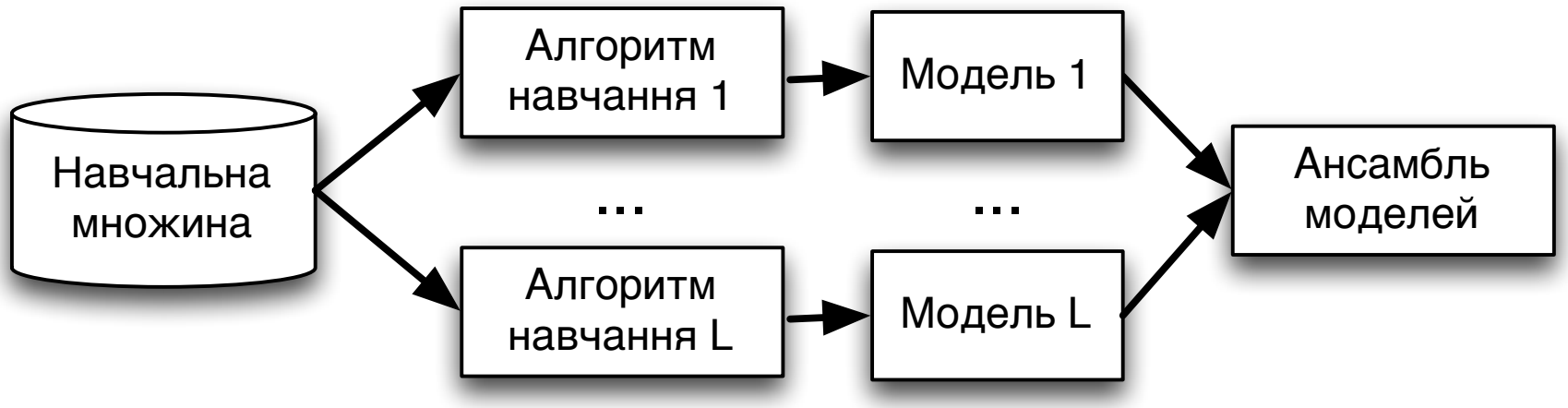
Перевибірка (resampling)



З початкової навчальної множини формуються декілька підвибірок, кожна з яких використовується для навчання однієї з моделей ансамбля.

Види ансамблів моделей

Використання єдиної навчальної множини для всіх моделей ансамбля



Види ансамблів моделей

3. Вибір методу комбінування результатів окремих моделей

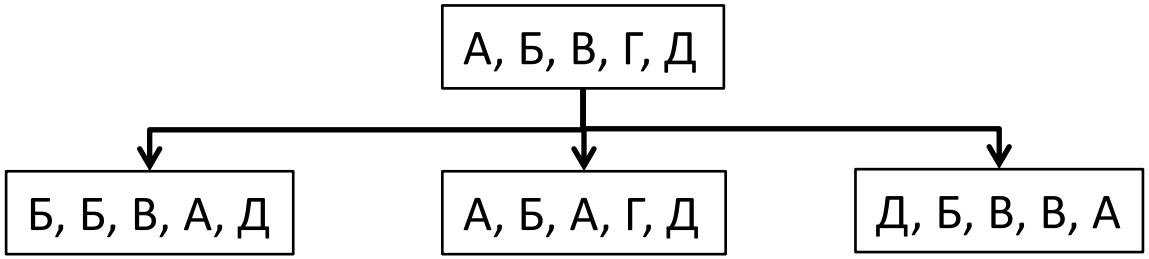
- **Голосування** – застосовується в задачах класифікації, для категоріальних цільових змінних. Вибирається клас, виданий більшістю моделей ансамбля.
- **Зважене голосування** – кожній моделі ансамбля назначається своя вага, яка відображає рівень довіри до результатів моделі.
- **Середнє значення** – застосовується в задачах регресії, для числових цільових змінних.
- **Зважене середнє значення** – в задачах регресії аналогічно зваженому голосуванню.

Беггінг (покращуюче об'єднання, bootstrap aggregating)

В основі беггінгу – технологія "збурення та комбінування"

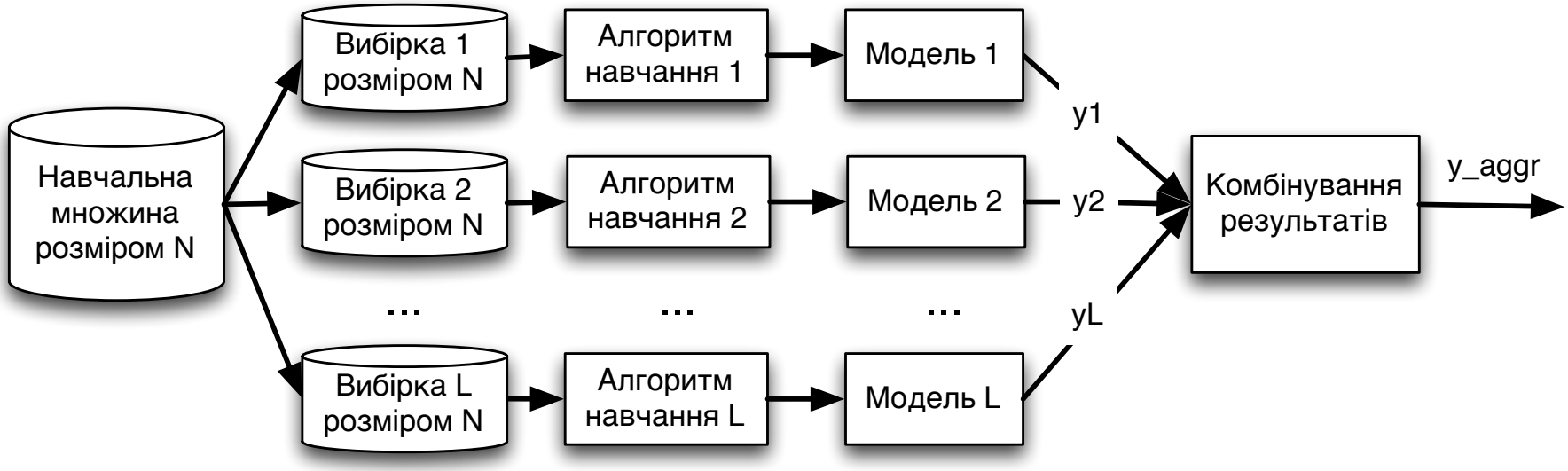
Внесення змін випадкового характеру в навчальні дані і побудова декількох альтернативних моделей на змінених даних з наступним комбінування результатів

- Формування декількох вибірок на основі навчальної множини



- Додавання шуму
- Адаптивне зважування
- Випадковий вибір між конкуруючими вузлами (розбиттями)

Беггінг (bootstrap aggregating - покращуюче об'єднання)



Етапи беггінгу:

- Формування вибірок однакового розміру випадковим чином на основі навчальної множини
- Побудова моделі на основі кожної вибірки
- Комбінування результатів

Бустінг (boosting - підвищення)

Ідея:

- Моделі створюються послідовно. Створення ансамбля починається на основі єдиної навчальної множини.
- Кожна нова модель ансамбля будується на основі результатів раніше побудованих моделей.
- Нові моделі будуються таким чином, щоб вони доповнювали раніше побудовані моделі, виконували ту роботу, яку інші моделі не змогли зробити на попередніх кроках.
- Кожній моделі ансамбля залежно від її точності присвоюється вага.

Бустінг

Недоліки бустінгу:

- Приклади з низькими вагами не потрапляють на наступну ітерацію. Наслідок - втрата частини корисної інформації, виродження навчальних вибірок, на основі яких будуються більш пізні моделі.
- Більша схильність до перенавчання в порівнянні з беггінгом. Кількість ітерацій має бути якомога меншою без втрати точності.
- Низька прозорість ансамблю моделей для аналітика (недолік властивий і беггінгу).
- Складність інтерпретації результатів (недолік властивий і беггінгу).

ВИПАДКОВИЙ ЛІС В SCIKIT-LEARN *VERSION* 0.23

Надія І. Недашківська n.nedashkivska@gmail.com

Недоліки дерев рішень

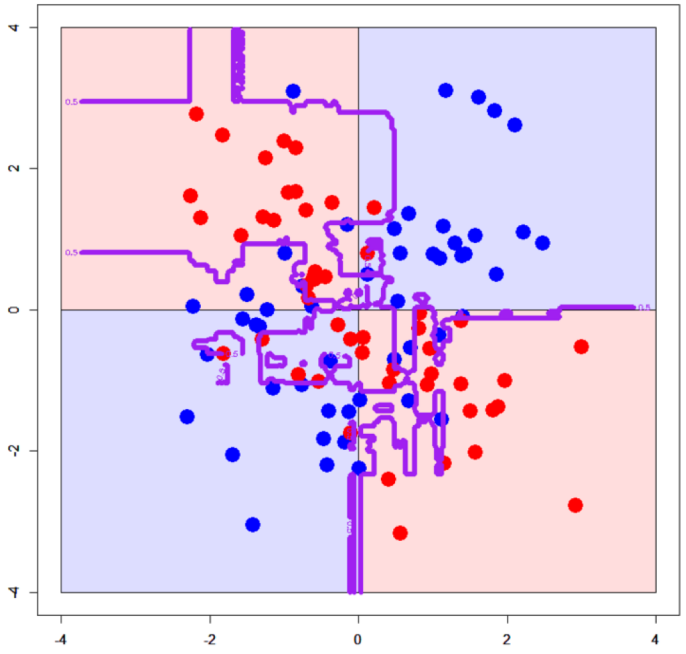
- дерева рішень можуть відновлювати дуже складні закономірності, тому вони схильні до перенавчання

Боротьба з перенавчанням – використання:

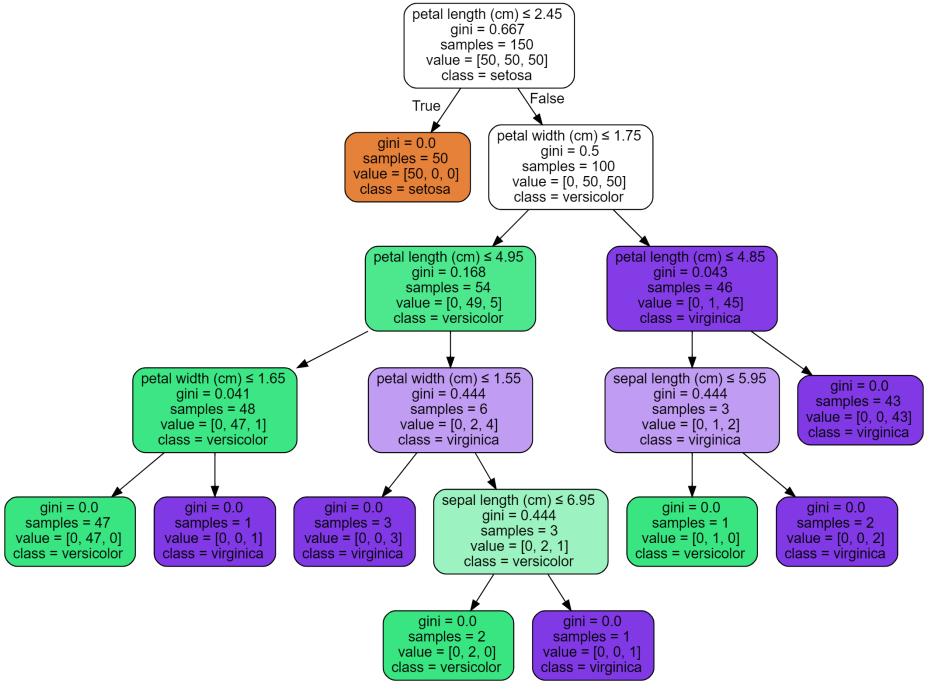
- *критеріїв зупинок*, які вважаються простими і не завжди допомагають,
- *стрижки дерев*, яка достатньо складна

Недоліки дерев рішень

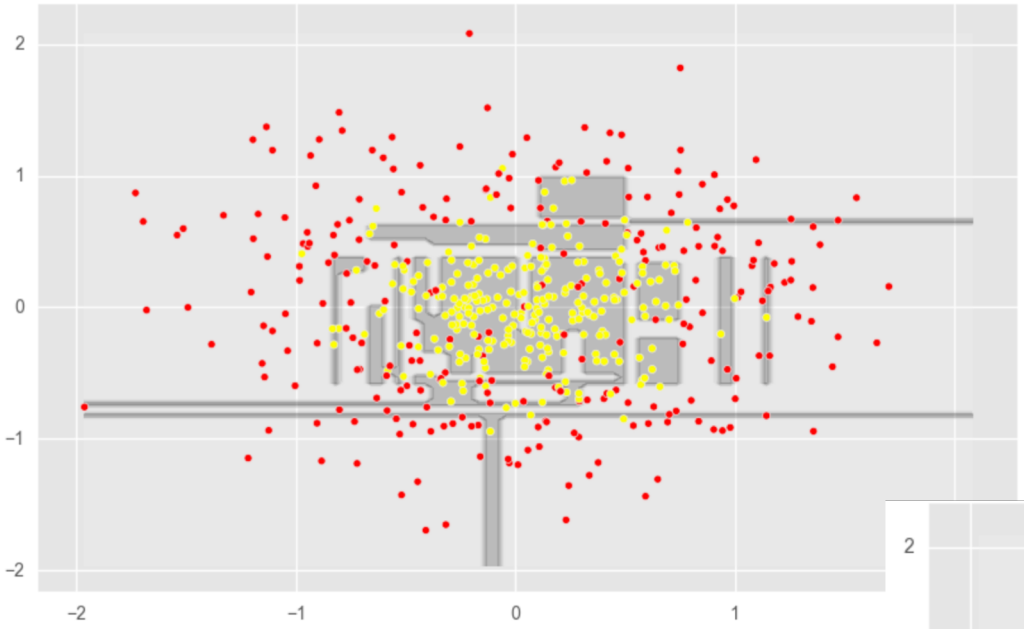
- Для навченого ДР, де в кожному листовому вузлі є по одному об'єкту, розділяюча поверхня буде дуже розрізаною – це дуже погано



Перенавчена поверхня

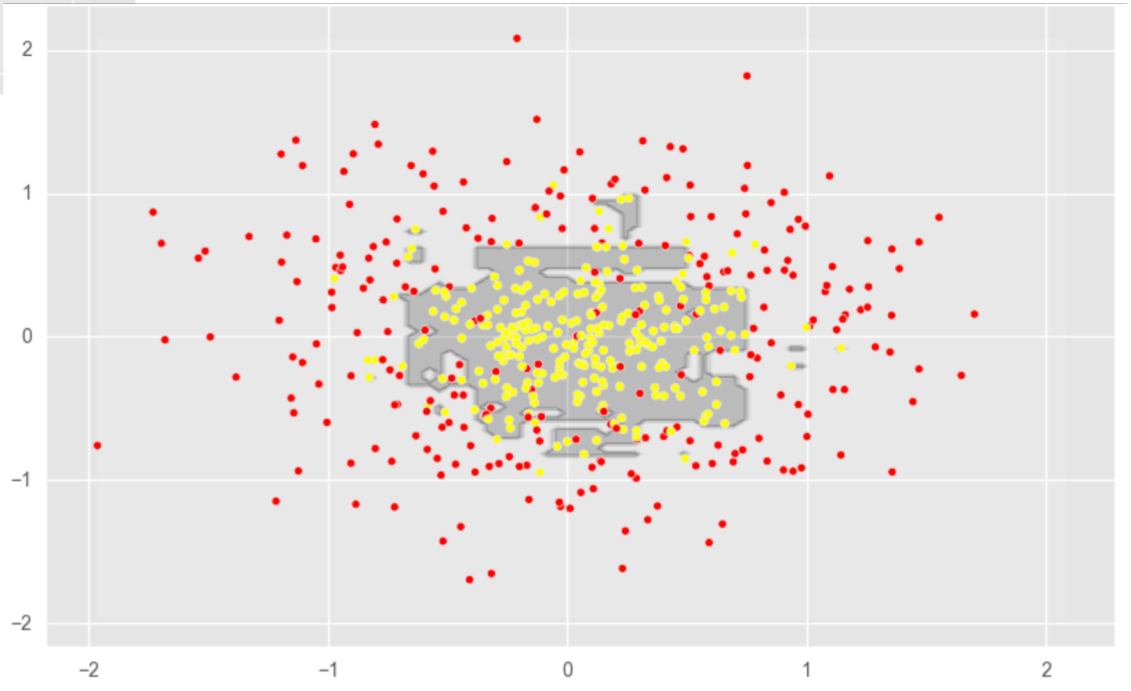


Порівняння розділяючих поверхонь

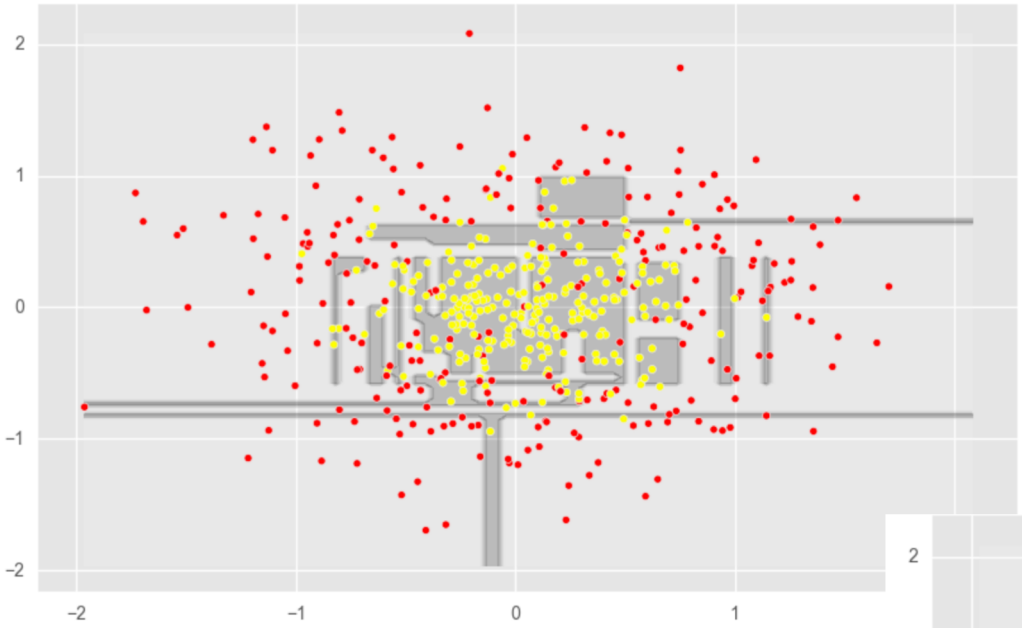


Decision Tree

Bagging & Decision Tree

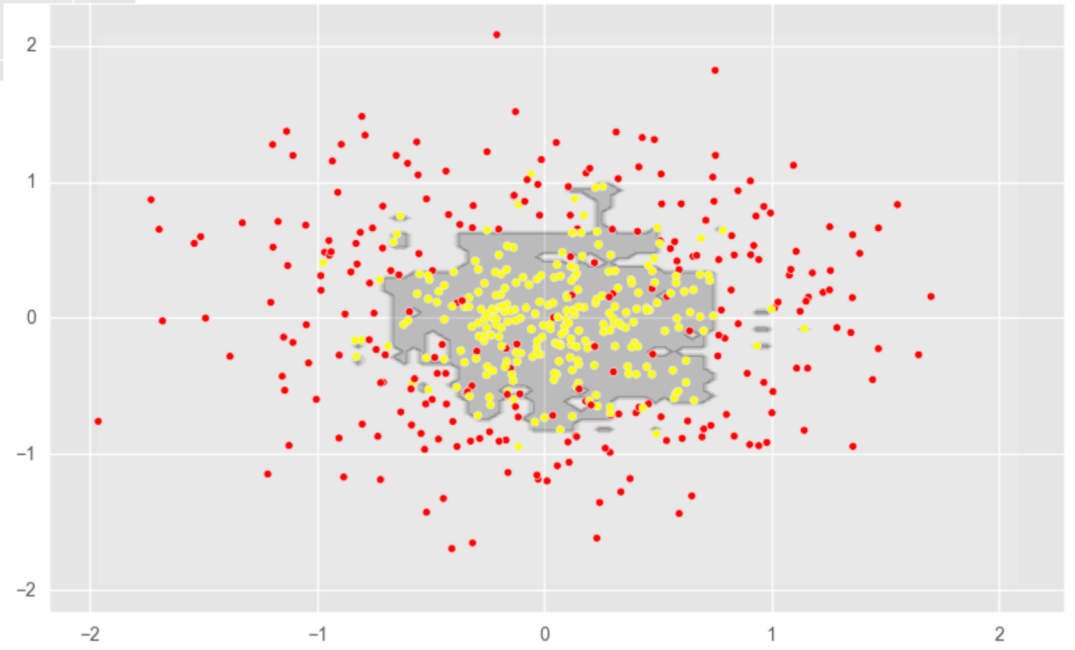


Порівняння розділяючих поверхонь



Decision Tree

Random Forest



Недоліки дерев рішень

- вони сильно перенавчаються,
- вони дуже нестійкі, сильно змінюються навіть при невеликих змінах у навчальній вибірці.

Другий недолік можна перетворити на їх перевагу за допомогою композиції дерев.

Випадковий ліс (Random Forest)

- Ансамбль дерев прийняття рішень, які навчаються зазвичай методом беггінга або іноді вставки і, як правило, з параметром max_samples рівним розміру навчальної вибірки.

```
class sklearn.ensemble.RandomForestClassifier ()

class sklearn.ensemble. RandomForestRegressor ()
```

Метод випадкового лісу

```
class sklearn.ensemble.RandomForestClassifier ()
```

n_estimators – число дерев у лісі, default=100

max_depth – максимальна глибина дерева,

default=None – дерево будується, поки всі листи не стануть чистими або поки всі листи не містять менше ніж min_samples_split прикладів,

min_samples_split – мінімальна кількість об'єктів, необхідна для поділу внутрішнього вузла. Можна задати числом або відсотком від загального числа об'єктів, default=2,

Метод випадкового лісу

```
class sklearn.ensemble.RandomForestClassifier ()
```

criterion – функція, яка вимірює якість розбиття гілки дерева,
criterion={"gini", "entropy”}, по дефолту – "gini"

min_samples_leaf – мінімальне число об'єктів у листі. Можна задати числом або відсотком від загального числа об'єктів, default=1,

min_weight_fraction_leaf – мінімальна зважена доля від загальної суми ваг по всім вхідним об’єктам повинна бути в листі, по умовчанню мають однакову вагу,
max_leaf_nodes – максимальна кількість листів, по умовчанню немає обмежень,

Метод випадкового лісу

```
class sklearn.ensemble.RandomForestClassifier ()
```

max_features – число ознак, за якими шукається розбиття. Можна вказати:
- конкретне число або відсоток ознак,
- "auto" – всі ознаки,
- «sqrt», тоді max_features=sqrt(n_features) (те саме, що і “auto”),
- "log2", тоді max_features=log2(n_features),

По дефолту "auto".

bootstrap – чи застосовувати бустреп для побудови дерева, default= True.
Якщо значення False, для побудови кожного дерева використовується весь набір даних.

Метод випадкового лісу

```
class sklearn.ensemble.RandomForestClassifier ()
```

oob_score – чи використовувати out-of-bag об’єкти для оцінки узагальненої точності accuracy, default=False

n_jobs – кількість задач, які будуть виконуватися паралельно - кількість ядер для побудови моделі і прогнозів, default=1,
якщо n_jobs= -1, то будуть використовуватися всі ядра

verbose – вивід логів з побудови дерев, default= 0

max_samples – якщо bootstrap=True, кількість прикладів для відбору з X для навчання кожного базового оцінювача, default=X.shape[0] прикладів. Можна задати числом або відсотком від загального числа прикладів.

Метод випадкового лісу

```
class sklearn.ensemble.RandomForestClassifier ()
```

warm_start – повторно використовує рішення попереднього виклику, використовує вже натреновану модель і додає більше оцінювачів до ансамблю, default= False – навчає весь новий ліс

class_weight – вага кожного класу, по дефолту всі ваги дорівнюють 1. Можна вказати:

- "balanced" – ваги класів дорівнюють їх вихідним частинам в навчальній вибірці,
- "balanced_subsample" – ваги на кожній підвибірці будуть змінюватися залежно від розподілу класів на цій підвибірці,
- передача списку словників з вагами

Extremely Randomized Trees

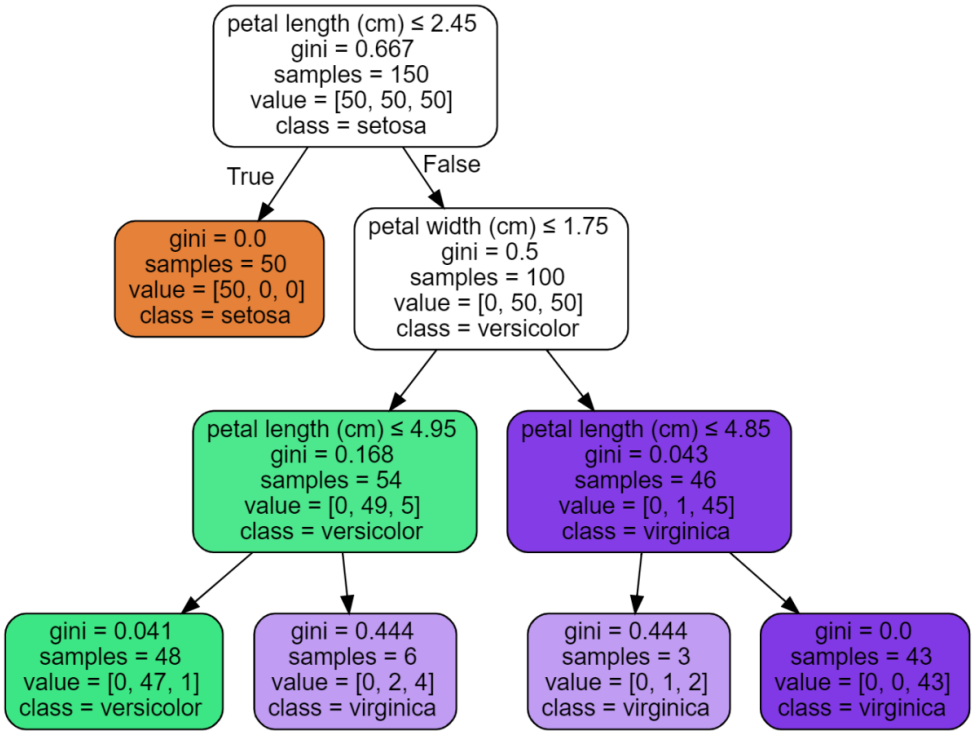
`ExtraTreesClassifier i ExtraTreesRegressor`

- Застосовуються випадкові пороги для кожної ознаки замість пошуку найкращих можливих порогів, як це робиться у звичайних деревах прийняття рішень.
- Має місце більш низька дисперсія при трохи вищому значенні зміщення.
- Вони навчаються набагато швидше, ніж традиційні випадкові ліси.
- ExtraTrees слід використовувати у випадку сильного перенавчання на випадковому лісі або градієнтному бустингу.

Оцінка важливості ознак

- чим вище знаходиться ознака в дереві рішень, тим вона є важливішою в даній задачі,

- при кожному розбитті в кожному дереві покращення критерію поділу (індексу Джині) - це показник важливості, пов'язаний зі змінною поділу, і накопичується він за всіма деревами лісу окремо для кожної змінної



Оцінка важливості ознак (продовження)

- Ознаки з більшим середнім зменшенням точності важливіші для класифікації /регресії (визначається під час обчислення out-of-bag помилки).
- Середнє зменшення індексу Джині (або помилки mse в задачах регресії) – це міра того, як кожна ознака сприяє однорідності вузлів і листів у лісі:

  - значення 0 відповідає повній однорідності,

  - значення 1 – повній неоднорідності.

- Ознаки, які призводять до вузлів з більш високою чистотою, мають більш високе зниження індексу Джині.

Оцінка важливості ознак (продовження)

```
>>> from sklearn.datasets  import load_iris
>>> iris = load_iris ( )
>>> model = RandomForestClassifier (n_estimators=500, n_jobs=-1)
>>> model. fit(iris["data"], iris["target"])
>>> for name, score in zip (iris["feature_names"], model. feature
importances_) :
        print (name, score)
```

```
sepal length (cm) 0 . 112492250999
sepal width (cm) 0 . 0231192882825
petal length (cm) 0.441030464364
petal width (cm) 0.423357996355
```

Перетворення ознак в багатовимір-ний простір

`RandomTreesEmbedding`

- Перетворення набору даних в багатовимірне розріджене його представлення.
- Будуються випадкові дерева, і індекс листа, в якому опинився об’єкт даних, вважаємо за нову ознаку.
- Бінарне кодування – якщо в лист потрапив об'єкт, то ставимо 1, якщо не потрапив, то 0 .
- Контролювати кількість змінних і ступінь розрідженості такого представлення можна збільшуючи або зменшуючи кількість дерев та їх глибину.

Зміщення і дисперсія для випадкового лісу

```
Tree: 0.0255 (error) = 0.0003 (bias^2) + 0.0152 (var)  +
0.0098 (noise)

Bagging(Tree): 0.0196 (error)= 0.0004 (bias^2) + 0.0092(var) +
0.0098 (noise)
```

Зміщення для моделі випадкового лісу таке ж, як і зміщення в окремому дереві $T(x, \Theta(Z))$:

$$Bias = E(x) - E_z f(x) = E(x) - E_z E_{\Theta|Z} T(x, \Theta(Z))$$

Зміщення випадкового лісу зазвичай більше, ніж зміщення «неусіченого» (unprunned) дерева, внаслідок рандомізації і скорочення простору вибірки.

Переваги випадкового лісу

- має високу точність прогнозування, точність порівнянна з бустінгом
- здатний ефективно обробляти дані з великим числом ознак і класів
- рідко перенавчається, але після досягнення певної кількості дерев крива навчання наближається до асимптоти
- практично не чутливий до викидів в даних, оскільки кожне дерево навчається на іншому випадковому піднаборі навчального набору
- не чутливий до будь-яких монотонних перетворювань значень ознак, напр, масштабування, це пов'язано з вибором випадкових підпросторів,

Переваги випадкового лісу (продовження)

- добре працює з пропущеними даними; зберігає гарну точність, навіть якщо велика частина даних пропущена
- можна збалансувати вагу кожного класу на всій вибірці, або на підвибірці кожного дерева
- не вимагає ретельної настройки параметрів
- існують методи оцінювання значущості окремих ознак в моделі
- допускає розпаралелювання, має високу масштабованість

Недоліки випадкового лісу

- схильний до перенавчання на деяких задачах, особливо на зашумлених даних
- працює гірше багатьох лінійних методів, коли у вибірці дуже багато розріджених ознак (напр., тексти)
- більший розмір результуючих моделей: потрібно O (nM) пам'яті для зберігання моделі, де M - число дерев
- не може виконувати екстраполяцію

Недоліки випадкового лісу (продовження)

- для даних, що мають категоріальні змінні з різною кількістю рівнів, випадкові ліси упереджені на користь ознак з великою кількістю рівнів: дерево сильніше підлаштовується саме під такі ознаки, так як на них можна отримати більш високе значення функціоналу, що оптимізується (індекс Джині, приросту інформації)
- якщо дані містять групи корельованих ознак, що мають схожу значущість для міток, то перевага віддається невеликим групам перед великими
- результати випадкового лісу складніше інтерпретувати, порівняно з одним деревом.