Uczenie maszynowe

Sekrety doboru odpowiedniego algorytmu

Mateusz Grzyb

Spis treści

1 Wstęp

Dobór algorytmu to w przypadku Machine Learningu jedna z kluczowych kwestii. Niekiedy może ona zaważyć nad powodzeniem całego projektu, a w większości przypadków będzie mieć kluczowy wpływ na osiągane przez nasz model wyniki. Źle dobrany algorytm może spowodować wyciąganie niewłaściwych wniosków z danych, czego następstwem mogą być fatalne w skutkach decyzje biznesowe. Właśnie dlatego postanowiłem dogłębnie rozpracować temat i podzielić się moimi przemyśleniami na łamach bloga.

1.1 Problem doboru algorytmu

To jaki algorytm należy wybrać, zależy od wielu zmiennych. Nawet najbardziej doświadczeni badacze danych mają z tym problem i często nie są w stanie wskazać najlepszego algorytmu przed jego przetestowaniem na wskazanym zbiorze danych. W tym miejscu muszę zaznaczyć jedną bardzo istotną rzecz: 99,999% specjalistów nie jest w stanie wskazać algorytmu, który w danych przypadku przyniesie najlepsze rezultaty. Bez uprzedniego przetestowania kilku algorytmów na zadanym zbiorze danych, wybór jest to w dużej mierze spekulacją. Jest jednak pewien zbiór reguł, który na podstawie kilka zmiennych pomoże zawęzić Ci poszukiwania do 2-3 algorytmów najlepiej pasujących do konkretnego przypadku. Wskazane algorytmy będziesz mógł przetestować na rzeczywistym zbiorze danych, tak by podjęcie właściwej decyzji było formalnością.

W tym poradniku znajdziesz całą masę wskazówek, podpowiedzi i informacji, które mają jeden cel: ułatwić Ci dobór odpowiedniego algorytmu, poprzez lepsze zrozumienie problemów (a także algorytmów), z jakimi najczęściej spotykamy się w Machine Learning.

1.2 Podział algorytmów uczenia maszynowego

Najogólniejszego podziału algorytmów można dokonać, bazując na typie uczenia: **nadzorowanym i nienadzorowanym**. **Uczenie nadzorowane** opiera się na danych mających mniej lub bardziej jasną strukturę. Kolumny posiadają etykiety opisujące ich zawartość. Przykładem może być tu zbiór danych klientów banku, którzy są opisani za pomocą zmiennych takich jak: data urodzenia, numer dowodu osobistego, saldo konta, miejsce zamieszkania, dane dotyczące historii kredytowej, historia transakcji, etc.

Algorytmu używane przy uczeniu nadzorowanym szukają zależności pomiędzy opisanymi zmiennymi, w celu wyznaczenia wskazanej wartości. Wartością tą może być np. estymowana maksymalna zdolność kredytowa, ryzyko wystąpienia oszustwa dla wybranej transakcji, lub też wartość binarna wskazująca czy dany klient banku będzie dobrym, czy złym kredytobiorcą. Podsumowując, w tym przypadku dokładnie wiemy, czego szukamy i na czym będziemy opierać swe decyzje.

W przypadku **uczenia nienadzorowanego** dane wejściowe nie posiadają żadnych etykiet. Nie jest również do końca jasny wynik jaki mamy uzyskać. **Celem jest tu zatem organizacja danych lub wyjaśnienie struktury danych wejściowych.** Zazwyczaj sprowadza się to do grupowania danych w celu zbudowania nowego spojrzenia na dane.

Prócz podziała względem typu uczenia, zdecydowaną większość algorytmów możemy podzielić, bazując na typie problemu z jakim się borykamy. Podstawowy podział wygląda następująco:

1. **Algorytmy klasyfikacyjne** – wskazują jedną z wartości kategorycznych. Jeśli w danym problemie algorytm prognozuje jedną z dwóch wartości (prawda/fałsz, 0/1, tak/nie), mówimy o klasyfikacji dwuklasowej (ang. two-class classification). W przypadku problemów, w których zmienna predykcyjna może przyjąć więcej niż dwie wartości, mówimy o klasyfikacji wieloklasowej (ang. multiclass classification).
2. **Algorytmy regresyjne** – są to algorytmy, których celem jest estymacja wartości numerycznej, np. wartość akcji danej spółki, wartość portfela inwestycyjnego złożonego z wybranych produktów, etc.
3. **Algorytmy grupujące** – jest to typ algorytmów, których zadaniem jest pogrupowanie zmiennych wejściowych w klastry, np. grupowanie klientów banku na podstawie ich historii kredytowej i produktów, z jakich korzystali.

4. **Inne** – zawrzeć tu można wszystkie algorytmy, których nie jesteśmy w stanie przypisać do żadnej powyższych kategorii. Będą to m.in. algorytmy wspomagające wykrywanie anomalii, algorytmy rekomendacyjne, czy też algorytmy służące do analizy tekstu. By nie komplikować zbytnio tematu, dla potrzeb tego poradnika zamknę je wszystkie w jednej kategorii - „Inne".

1.3 Kryteria doboru algorytmu

Istnieje kilka podstawowych zmiennych, które są istotne przy wyborze algorytmu. Już teraz możesz dokonać pierwszej selekcji, bazując na podziale, który przedstawiłem wcześniej (klasyfikacja, regresja, grupowanie). Oczywiście to dopiero początek zabawy. Drugą kluczową kwestią jest matematyczna charakterystyka danego algorytmu. Należy tu rozważyć typ zmiennych i format danych wejściowych.

Przy wyborze algorytmu warto również zwrócić uwagę na takie elementy jak:

- liczba obserwacji,
- liczba zmiennych,
- wartości, jakie przyjmują zmienne kategoryczne (liczba etykiet) w danych wejściowych.

Są algorytmy, które spisują się znakomicie w przypadku niewielkiej liczby parametrów uczących. Inne posiadają ograniczenia charakterystyczne dla danej platformy, np. w R jedna z bibliotek posiada

ograniczenie, przez które zmienna kategoryczna w Random Forest może przyjmować maksymalnie do 32 unikalnych wartości. Gdy którakolwiek z naszych zmiennych przyjmuje więcej niż 32 unikalne wartości, wtedy mamy 3 możliwości:

1. Zmienić algorytm uczący.
2. Zrezygnować ze zmiennej
3. Zmniejszyć liczbę wartości do maksymalnie 32 (np. poprzez pogrupowanie podobnych wartości).

Nie bez znaczenia pozostają również **kwestie optymalizacji** i **dokładność poszczególnych algorytmów**. Gdy znamy wymagania dotyczące projektu i mamy do dyspozycji duży zbiór danych należy wziąć pod uwagę dokładność i **czas uczenia**. Dla przykładu: przyjmuje się, że w przypadku problemów klasyfikacyjnych drzewa decyzyjne są dosyć dokładne, kosztem nieco dłuższego czasu uczenia. Z kolei regresja logistyczna charakteryzuje się relatywnie niskim czasem uczenia przy nieco gorszej dokładności. Coś za coś.

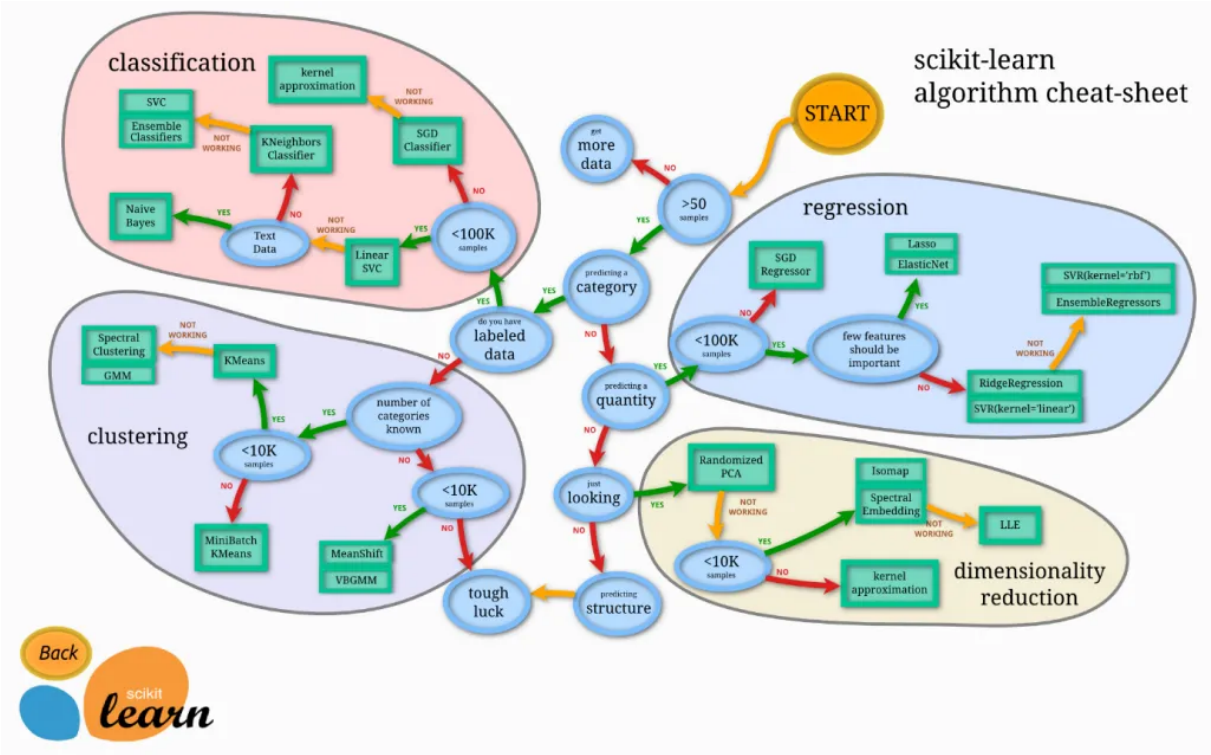
Podsumowując, przy wyborze algorytmu należy zatem wziąć pod uwagę:

- Typ problemu (klasyfikacja, regresja, grupowanie).
- Ograniczenia biznesowe.
- Interpretowalność.
- Charakterystykę algorytmu.
- Liczbę obserwacji.
- Liczbę zmiennych w zbiorze.
- Liczbę etykiet (wartości przyjmowanych przez zmienne kategoryczne).
- Dokładność algorytmu.
- Czas uczenia.
- Inne kwestie, których duża część jest zależna od typu problemu, z którym się mierzymy (np. liniowość, lub jej brak).

1.4 Ogólny schemat doboru algorytmów

Być może zastanawiasz się, czy istnieje jakiś ogólny schemat doboru algorytmu. Otóż w internecie można znaleźć kilka tego typu schematów. Ich celem jest pomoc w dokonaniu wyboru.

Niestety żaden z nich (a przynajmniej żaden z tych, które są mi znane) nie wyczerpuje tematu choćby w niewielkim stopniu. Poniżej chciałbym przedstawić jeden, który uznałem za najbardziej wartościowy. Udostępnia doskonale znany użytkownikom Pythona zespół odpowiedzialny za bibliotekę scikit-learn. Większy format tej ściągawki jest dostępny pod tym linkiem.



Rysunek 1: Schemat doboru algorytmu - by scikit-learn

1.5 Zanim zaczniemy

Czuję się zobowiązany, by Cię uprzedzić drogi czytelniku o kilku założeniach.

- W poradniku znajduje się wiele elementów wynikających z mojego, unikalnego doświadczenia. To, co się sprawdza u mnie, nie musi się sprawdzać u wszystkich.
- Wszystko, o czym piszę, to moje **subiektywne postrzeganie problemów** i ich rozwiązań. Bazuje to zarówno na moim prywatnym doświadczeniu, jak i ogólnie przyjętych zasad "poprawnego modelowania".
- W poradniku **przyjmuję pewne uproszczenia** i nie wchodzę głęboko w szczegóły. W prawdziwym życiu należy starannie przeanalizować problem, który rozpatrujemy. Tym bardziej, jeżeli w grę wchodzą duże pieniądze, a tak zazwyczaj się dzieje przy projektach z branży IT.
- Teoria jest tylko teorią. W praktyce może dojść do sytuacji, w której mając dwa pozornie identyczne problemy i zestawy danych, najlepszy wynik może być uzyskany przy pomocy różnych algorytmów. **Nie ma dwóch identycznych przypadków.**
- Przedstawiam jedynie **zestaw reguł**, pomagający „przefiltrować" najpopularniejsze algorytmy i zawęzić poszukiwania.

- Głównym celem jest tu uwidocznienie **procesu myślowego** przebiegającego w głowie doświadczonego badacza danych, wraz ze stosownym uzasadnieniem podejmowanych wyborów.
- W poradniku nie opisuję wszystkich algorytmów. Jest ich po prostu zbyt wiele i nieustannie powstają nowe. Ja przybliżam jedynie te najpopularniejsze.
- Poradnik ten należy traktować raczej jak **kompas i mapę**, niż GPS, który doprowadzi Cię zakręt po zakręcie do celu.

Reasumując: wszystko, o czym piszę, pomoże Ci **lepiej zrozumieć działanie i charakterystykę** najpopularniejszych algorytmów. Wartością dodaną będzie również zaoszczędzony na poszukiwaniach czas. Przy podejmowaniu ostatecznej decyzji zawsze:

1. Zrozum cel biznesowy.
2. Dobrze zdefiniuj problem.
3. Zapoznaj się z procesem generowania danych.
4. Zrozum dane.
5. Wybierz algorytm.

No to zaczynajmy! :)

2 Algorytmy klasyfikacyjne

Pierwsza grupa, to bodaj najpopularniejsza kategoria algorytmów – algorytmy klasyfikacyjne. Rozwiązują one problemy klasyfikacji danych i należą do moich ulubionych algorytmów. Jeśli chcesz wiedzieć, jak działają najpopularniejsze klasyfikatory i czym się kierować przy ich wyborze, to zapraszam do czytania.

2.1 Opis problemu klasyfikacji

W codziennym życiu spotykamy się z dziesiątkami problemów o podłożu biznesowym. W większości przypadków stoimy po „drugiej stronie". Chcąc, czy nie jesteśmy klientami instytucji finansowych zaciągającymi pożyczki i lokującymi swe oszczędności w bankach, odbiorcami reklam w internecie, czy też klientami sklepów internetowych, którym „podsuwa" się potencjalnie interesujące produkty. We wszystkich wymienionych sytuacjach w tle istnieje problem, który można rozwiązać z użyciem danych i uczenia maszynowego. Jest to problem klasyfikacji bądź prognozowania (jeśli tylko go odpowiednio sformułujemy).

Z problemem klasyfikacji (tudzież rekomendacji) spotykamy się wszędzie tak, gdzie z wykorzystaniem zbioru **zmiennych objaśniających** musimy wskazać wartość przyjmowaną przez **zmienną objaśnianą**. W przeciwieństwie do problemów regresyjnych, w problemach klasyfikacji zmienna modelowana nie przyjmuje wartości ciągłych, lecz wartości binarne (**klasyfikacja dwuklasowa**, ang. *two-class classification*), ew. jedną z kilku etykiet (**klasyfikacja wieloklasowa**, ang. *multiclass classification*).

2.1.1 W jakich przypadkach zastosowanie mają algorytmy klasyfikacyjne?

1. Korzystasz ze skrzynki mailowej? Na 99,99% wykorzystuje ona filtr spamu, będący przykładem systemu klasyfikacyjnego (czy dany e-mail jest spamem, czy też nie?).
2. Jesteś klientem banku i starałeś się o dowolny kredyt? Twój profil klienta również był analizowany z użyciem algorytmy uczenia maszynowego, wyznaczający ryzyko kredytowe (czy dana osoba jest dobrym, czy też złym kredytobiorcą?).
3. Masz profil na Filmweb lub Netflix i oceniasz obejrzane filmy? Każdy kolejny film, który widzisz na liście podpowiedzi, został wytypowany na podstawie Twojej historii oceniania. Użyto tu systemu rekomendacyjnego, zbudowanego na podstawie algorytmu klasyfikacyjnego (bądź też wielu algorytmów). Więcej o systemach rekomendacji, ich budowie i zastosowaniu możesz przeczytać w jednym z moich projektów.
4. Masz abonament telefoniczny i przed zakończeniem okresu umowy otrzymujesz propozycję jej przedłużenia na „superatrakcyjnych" warunkach? Za tym również stoi mniej lub bardziej złożony algorytm klasyfikacyjny, użyty w systemie, którego głównym celem jest retencja klientów.

5. Szukasz nowego produktu x w wyszukiwarce internetowej i nagle zdajesz sobie sprawę, że wszystkie strony wyświetlają Ci reklamy trafiające w Twoje potrzeby? Za tym również stoi algorytm klasyfikacji reklam (ew. silnik rekomendacji).

2.1.2 Na jakie pytania odpowiadają algorytmy klasyfikacyjne?

1. Czy klient będzie dobrym kredytobiorcą? (spłaci kredyt w całości, bez większych opóźnień) | 0, 1 (tak lub nie).
2. Czy dany klient będzie chciał zrezygnować z naszych usług? | 0, 1 (tak lub nie).
3. Czy dana transakcja jest fraudem? | 0, 1 (tak lub nie).
4. Czy użytkownik polubi dany film? | 0, 1 (tak lub nie).
5. Czy to jest spam? | 0, 1 (tak lub nie).
6. Jakie są nastroje ludzi używających Twittera? | pozytywne, negatywne, neutralne, etc.
7. Która reklama przypadnie do gustu użytkownikowi i przyniesie dla nas i naszych klientów największe zyski? A, B, C lub D.

Zgodnie z powyższych algorytmy klasyfikacyjne dzielimy zatem na dwie grupy: **dwuklasowe** i **wieloklasowe**. Myślę, że w zdecydowanej większości przypadków spotykamy się z problemami binarnymi, dlatego też im poświęcam największą uwagę. Wybór odpowiedniego klasyfikatora

Kiedy mamy zdefiniowany problem, znamy strukturę danych i cel biznesowy projektu, należy uzasadnić wybór algorytmu. Jest na to kilka sposobów. Pierwszy z nich zakłada pominięcie wszelkich ograniczeń i przetestowanie wszystkich algorytmów, w różnych konfiguracjach, z użyciem różnych predyktorów, a następnie porównanie wyników. Zajmie to jednak miesiące, jeśli nie lata. Bardziej praktyczne jest podejście bazujące na analizie 8 atrybutów problemu, o których to wspomniałem w części pierwszej.

Ważną kwestią są **ograniczenia, jakie napotykamy**. Zbyt mała (lub zbyt duża) liczba danych, brak czasu na zbudowanie odpowiedniego modelu czy czas wykonywania algorytmu przekładający się na działanie systemu to tylko niektóre z nich. Warto mieć świadomość wszystkich ograniczeń jeszcze przed przystąpieniem do budowania modelu. Chyba nikt z nas nie chciałby znaleźć się w sytuacji, w której po miesiącach starań, uzyskuje się model dający świetne rezultaty, ale jednocześnie taki, który nie może być produkcyjnie zaimplementowany ze względu np. na aspekty prawne (w kolejnych akapitach rozwinę ten temat).

W opisie algorytmów biorę pod uwagę wszystkie ich zalety, jak i również wady. Poniżej przedstawiam dodatkowo listę ograniczeń, z którymi możecie się spotkać przy problemach klasyfikacyjnych.

1. **Kwestie szybkościowe**:

    - **Czas uczenia i aktualizacji modelu**. W przypadku systemów nieustannie udoskonalanych, które są uaktualniane, jest to duże ograniczenie. Dotyka ono większości nieliniowych

rozwiązań (jak np. KNN).

- **Czas wykonywania modelu**. W szczególności systemy czasu rzeczywistego, w których kwestie szybkościowe nabierają pierwszorzędnego znaczenia. Tu zdecydowaną przewagę mają algorytmy liniowe, które są duże szybsze zarówno w fazie uczenia, jak i wykonywania. Krócej trwa również proces ich uaktualniania, dlatego są one tak popularne w silnikach rekomendacji. Nie wymagają również przetrzymywania dużej ilości danych w pamięci. W praktyce sprowadza się to do przechowywania w pamięci wektor współczynników, pozwalającego uzyskać odpowiedź.

2. **Interpretowalność**. Chodzi o interpretowanie działania algorytmu na potrzeby np. regulatora (w Polsce jest to KNF). Wspomniałem wcześniej o specyficznych sytuacjach, w których należy brać pod uwagę aspekty prawne charakterystyczne dla danej branży. Przykładem sektora, który jest mocno obarczony ryzykiem związanym z interpretacją decyzji pod kątem prawa, jest bankowość. W wielu krajach wszystkie decyzje dotyczące kredytów muszą mieć swoje uzasadnienie. Żeby lepiej zrozumieć, co mam na myśli, wyobraź sobie, że jesteś analitykiem dużego banku w USA. Przed amerykańskim odpowiednikiem KNF musisz wytłumaczyć się z decyzji o nieudzielenie kredytu dla Jana Kowalskiego. Decyzja ta była podjęta na podstawie danych z system, który korzysta z dobrodziejstw uczenia maszynowego. Jeśli wykorzystuje on algorytm sieci neuronowych, to masz duży problem. Istnieje duże prawdopodobieństwo, że bank będzie zmuszony wypłacić odszkodowanie klientowi, a Ty stracisz pracę.

   Pomyśl teraz o analogicznej sytuacji. Jedyną różnicą jest to, że został użyty inny algorytm. Tym razem system bazuje na drzewie decyzyjnym. Z wykorzystaniem odpowiedniej biblioteki R lub gotowych komponentów np. z Microsoft SQL Server Analysis Services jesteś w stanie w prosty sposób zobrazować i wytłumaczyć proces stojący za decyzją banku. Powyższa sytuacja to jedynie przykład. Podobne schematy można mnożyć.

   Analitycy danych w spółkach akcyjnych również musisz tłumaczyć się zarządowi i inwestorom z podejmowane decyzje. Gdyby używali algorytmów, których działanie ciężko zinterpretować, to mieliby duży problem.Podsumowując: wśród algorytmów klasyfikacyjnych, możemy wyróżnić takie, których działanie można opisać, spoglądając np. na wygenerowany schemat (np. wspomniane drzewa decyzyjne) – łatwo interpretowalne, jak i również inne, niemalże nieinterpretowalne, do których wyniku ciężko się odnieść.

3. **Skalowalność**. Mam tu na myśli wszystkie zasoby, jakie zużywa dany algorytm. Wyróżnić tu możemy czas uczenia, czas wykonywania, po jakim algorytm zwraca wynik użytkownikowi, czy ilość pamięci, która jest potrzebna do poprawnego wykonania.

4. **Czynnik ludzki**. Czy naprawdę dobrze znasz i rozumiesz wybrany algorytm? Pisząc „naprawdę dobrze”, mam na myśli bardzo, bardzo dobrze. Oczywiście, by zostać specjalistą w kwestii uczenia maszynowego i eksploracji danych, nie trzeba znać dogłębnie wszystkim algorytmów.

Należy jednak mieć świadomość ich ograniczeń i ogólnej charakterystyki. Warto mieć swoje ulubione algorytmy, które znamy znakomicie i wykorzystujemy w konkretnych sytuacjach. Ma to służyć radzeniu sobie w sytuacjach, gdy musimy dopasować wybrany algorytm i jego działanie do problemu, tak by uzyskać, jak najlepszy wynik. Trzeba mieć wiedzę potrzebną do odpowiedniej **manipulacji hiperparametrami** danego algorytmu. Jeśli w nie masz praktycznej wiedzy, to istnieje ryzyko, że:

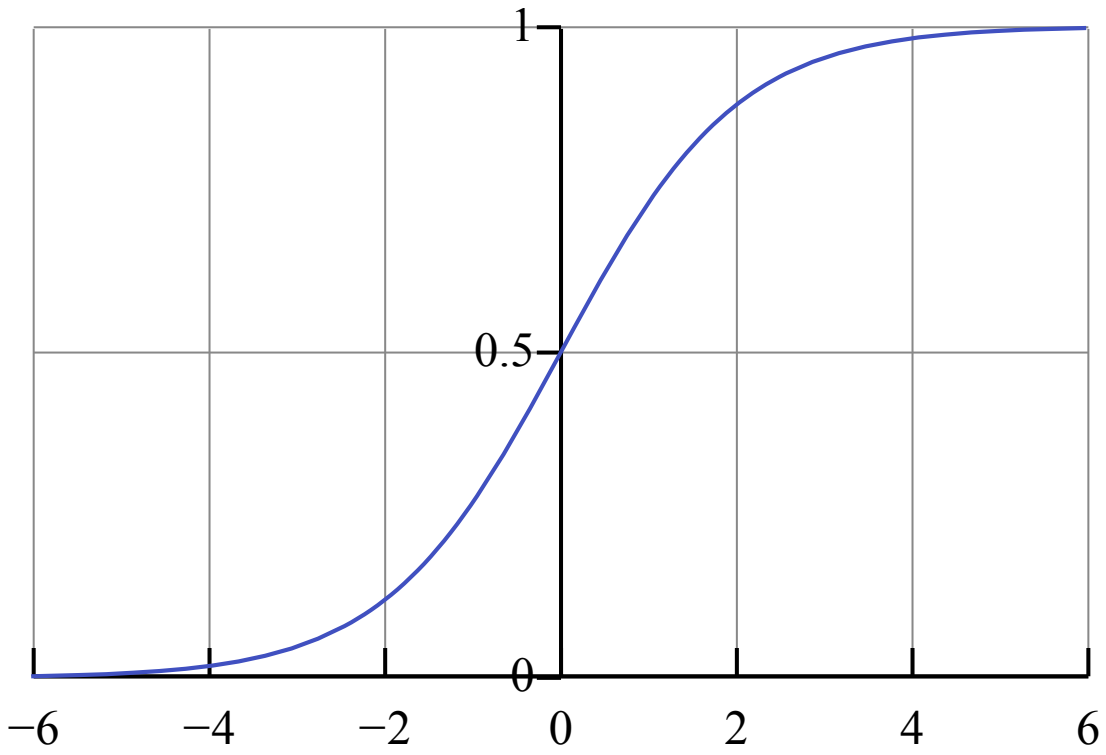
- Osiągnięty przez Ciebie wynik będzie niesatysfakcjonujący, bo nie dopasowałeś w sposób satysfakcjonujący algorytmu do danych (**problem niedouczenia modelu**).
- Osiągniesz zbyt duże dopasowanie algorytmu, a więc model będzie przeuczony. Uzyskasz przez to fatalny wynik na danych produkcyjnych (**problem zbyt wysokiego dopasowania do danych uczących**, ang. *overfitting*).

2.2 Opis poszczególnych algorytmów

2.2.1 Naiwny klasyfikator Bayesa

1. **Opis algorytmu**. Metoda klasyfikacji dwu i wieloklasowej. Została oparta na teorii Bayesa, dotyczącej przewidywania kategorii w nieznanym zbiorze danych. Zakłada zupełną niezależność poszczególnych zmiennych (co rzadko ma odzwierciedlenie w świecie rzeczywistym). Stąd też jego nazwa – naiwny klasyfikator. Jego działanie opiera się na prostym prawdopodobieństwie warunkowym zdarzeń. Zaliczamy go do jednego z najprostszych algorytmów. Pomimo prostoty potrafi dawać lepsze rezultaty niż bardziej skomplikowane metody klasyfikacji.
2. **Przykładowe zastosowanie**:
    - Analiza tekstu (filtry spamu, analiza danych z Twitter, etc., analiza opinii).
    - Przewidywanie w czasie rzeczywistym.
    - Inne (wszędzie tam, gdzie poszukuje się niskiego zużycia pamięci i CPU).
3. **Liczba obserwacji**: ze względu na liniowość dobrze sprawdza się przy dużych zbiorach. Jednocześnie nie wymaga dużej liczby danych uczących do poprawnego działania.
4. **Dokładność algorytmu**: średnia.
5. **Czas uczenia**: krótki - dzięki prostej strukturze i niskiej złożoności obliczeniowej.
6. **Liniowość**: tak.

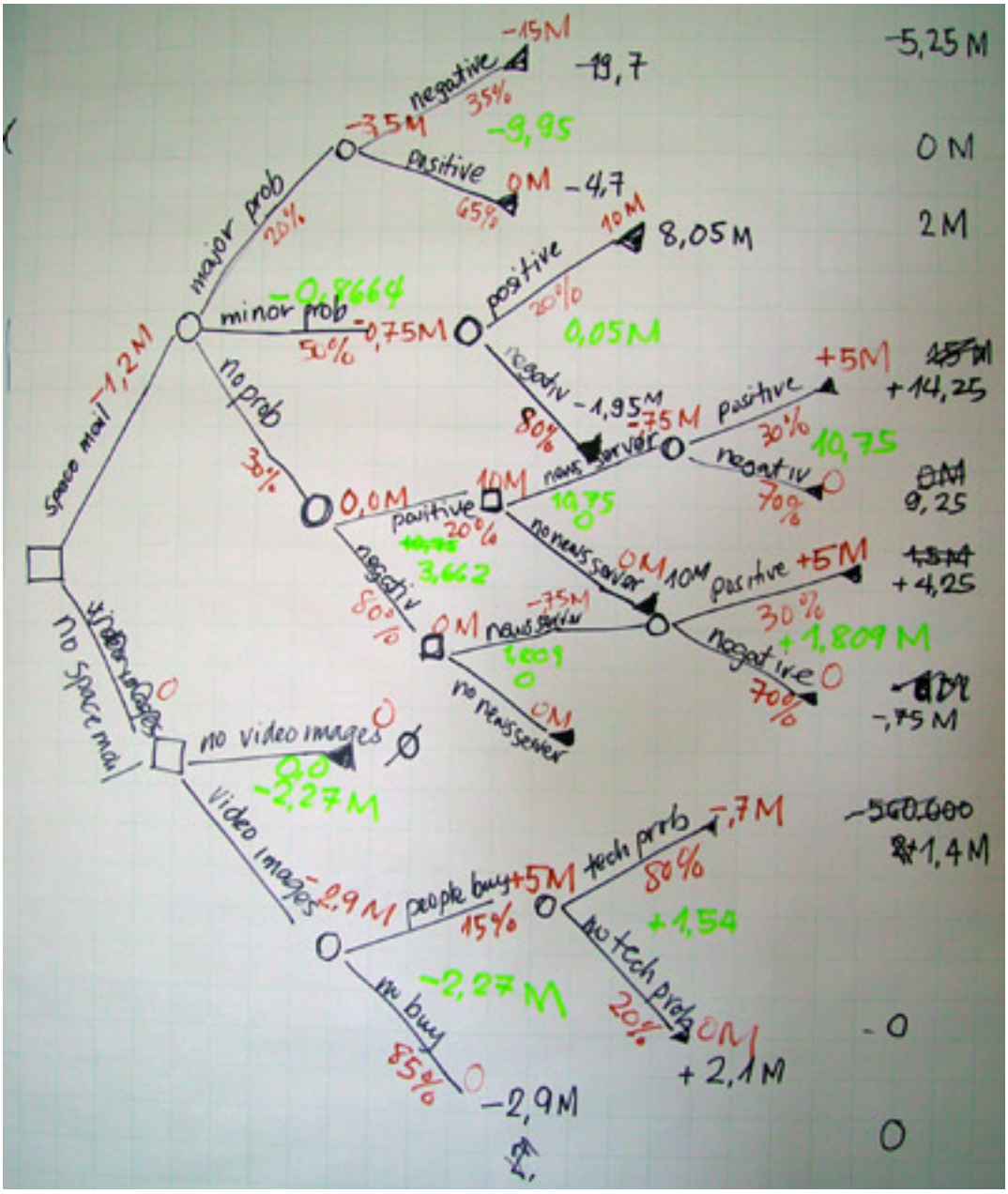
2.2.2 Regresja logistyczna



Rysunek 2: Krzywa regresji logistycznej

1. **Opis algorytmu**. Metoda klasyfikacji dwu i wieloklasowej. Szybka i relatywnie prosta. Oparta jest o specyficzny system prawdopodobieństw i (tzw. szansach wystąpienia zdarzenia) i funkcji logitowej. Celem owej funkcji jest przekształcenie prawdopodobieństwa w logarytm szansy, tzw. logit. Regresja logistyczna daje wynik ciągły, sprowadzany do binarnego. Z powodzeniem zatem może być stosowana w systemach rekomendacji, z dowolną metodą sortowania wyników.
2. **Przykładowe zastosowanie**:
    - Systemy oceny ryzyka kredytowego.
    - Detekcja fraudów.
    - Systemy rekomendacyjne.
3. **Liczba obserwacji**: dobrze radzi sobie nawet przy dużej liczbie obserwacji.
4. **Dokładność algorytmu**: średnia/wysoka.
5. **Czas uczenia**: krótki.
6. **Liniowość**: tak.

2.2.3 Drzewo decyzyjne



Rysunek 3: Drzewo decyzyjne generowane w sposób tradycyjny - metodą ręczną :)

1. **Opis algorytmu**. Mój ulubiony algorytm :) Nazywany jest również drzewem klasyfikacyjnym. Oparty na strukturach o drzewiastym kształcie. Proces klasyfikacyjny zaczyna się w korzeniu drzewa i postępuje (w dół) aż do osiągnięcia liści, czyli tzw. klas terminalnych.O procesie klasyfikacyjnym opartym na drzewie decyzyjnym możemy myśleć jak o rozbijaniu ważnej decyzji

na ciąg pytań. **Przedstawiane rezultaty są łatwo interpretowalne nawet przez osoby bez doświadczenia analitycznego**. Jest to niewątpliwa zaleta wszędzie tam, gdzie wynik uzyskany przez system decyzyjny musi zostać uzasadniony przez człowieka.

Drzewo decyzyjne jest budowane w sposób iteracyjny, począwszy od korzenia, aż do liści. W kolejnych iteracjach dodawane są węzły składające się z odpowiednio dobranych atrybutów. Atrybuty są wybierane przez tzw. algorytm wyboru cech, w kolejności mającej zmaksymalizować zysk informacyjny z danego węzła. Cały proces ma swój koniec w momencie, gdy wszystkie liście są w tej samej klasie lub gdy zabranie klas do podziału.Budując pełne drzewo istnieje duże prawdopodobieństwo **nadmiernego dopasowania algorytmu**. Jest ono częstym problemem w przypadku podstawowej wersji drzewa decyzyjnego. Może mieć to negatywny wpływ na wynik, jaki osiąga algorytm, dlatego dobrą praktyką jest przycinanie drzewa od pewnej, ustalonej głębokości.

„Przeuczenia" można uniknąć również w inny sposób, budując zmodyfikowaną wersję algorytmu. Piszę o tym w kolejnych punktach. Jest jeszcze jedna ważna rzecz, którą warto zaznaczyć. Mimo iż drzewo decyzyjne budowane jest na podstawie decyzji binarnych, to obecnie używane pakiety implementujące algorytm **radzą sobie również ze zmiennymi ciągłymi**. Zawierają one mechanizm przekształcający zmienne ciągłe w predyktor binarny. Dla przykładu, w przypadku zmiennej „Spalanie" mówiącej nam o średnim spalaniu danego auta, można sklasyfikować ilość zużytej benzyny na 100 km: „mniej niż 10" i „więcej niż 10".

2. **Przykładowe zastosowanie**:
    - Bankowość inwestycyjna.
    - Systemy oceny ryzyka kredytowego.
3. **Liczba obserwacji**: duża. Szybki czas uczenia.
4. **Dokładność algorytmu**: średnia/wysoka (UWAGA: przy podstawowej wersji drzewa decyzyjnego nietrudno o przeuczenie modelu).
5. **Czas uczenia**: krótki. Niestety nie wspiera uczenia ad-hoc. Jeśli mamy nowe obserwacje w naszym zbiorze, jesteśmy zmuszeni do przebudowania modelu.
6. **Liniowość**: nie (ze względu na strukturę algorytmu, nie obserwujemy tu liniowej zależności pomiędzy atrybutami).

2.2.4 Wzmocnione drzewo decyzyjne (ang. *Boosted Decision Tree*)

1. **Opis algorytmu**. Dwuklasowa wariacja drzewa decyzyjnego. Dzięki „niekończeniu" drzewa, **unika ono największej bolączki podstawowej wersji drzewa – przeuczenia**. Jest to możliwe

dzięki ograniczeniu liczby podziałów gałęzi drzewa i jego głębokości. Dodatkowym elementem jest tu wzmocnienie (ang. *boosting*) polegające na losowaniu nowych ciągów uczących służących do uczenia kolejnych wersji klasyfikatorów. W praktyce algorytm buduje sekwencję drzew. Każde nowe drzewo uczy się kompensować błąd pozostawiony przez poprzednie drzewo. **W rezultacie otrzymujemy bardzo dokładny algorytm, który niestety wymaga dużej ilości pamięci.**

2. **Przykładowe zastosowanie**:
    - Bankowość inwestycyjna.
    - Mechanizmy wykorzystywane w zespołach CRM (np. *next best offer*).
    - Inne – wszędzie tam, gdzie liczy się dokładność, nie ma ograniczających zasobów, ew., gdy algorytm nie musi być często uaktualniany.
3. **Liczba obserwacji**: średnia. Średni czas uczenia.
4. **Dokładność algorytmu**: wysoka/bardzo wysoka.
5. **Czas uczenia**: średni/długi (przyczyną jest większa złożoność obliczeniowa niż w przypadku podstawowej wersji drzewa decyzyjnego).
6. **Liniowość**: nie.

2.2.5 Lasy losowe (ang. *Random Forest*)

1. **Opis algorytmu**. Algorytm klasyfikacji dwu i wieloklasowej. Można spotkać się z różnym nazewnictwem: Random Forest i Decision Forest. Idea działania w obu przypadkach jest niemal identyczna. Działanie lasów losowych opiera się na klasyfikacji z użyciem grupy drzew decyzyjnych i największą różnicą w budowie obu algorytmów jest tzw. bootstrap.

    Algorytm rozpoczyna swe działanie od zbudowaniu wielu drzew decyzyjnych (liczba drzew jest definiowana przez użytkownika). Dla każdego drzewa wybierana jest losowa próba obserwacji, składająca się z kilku zmiennych objaśniających (liczba zmiennych w każdym z drzew jest drugim parametrem definiowanym przez użytkownika).

    Następnie mechanizm maksymalizacji zysku informacyjnego (o którym pisałem więcej w przypadku drzewa decyzyjnego) wskazuje kolejne atrybuty do podziału. Końcowa decyzja jest podejmowana w wyniku większościowego głosowania nad klasami wskazanymi przez poszczególne drzewa.

    Główną zaletą lasów losowych jest większa dokładność modelu niż w przypadku drzewa decyzyjnego. Nie ma oczywiście nic za darmo. Użyty mechanizm bootstrap ma negatywny wpływ na interpretowalność działania algorytmu.

2. **Przykładowe zastosowanie**:
    - Bankowość inwestycyjna.

- Mechanizmy wykorzystywane w zespołach CRM (np. *next best offer*).
- Inne – wszędzie tam, gdzie liczy się dokładność, nie ma ograniczających zasobów, ew., gdy algorytm nie musi być często uaktualniany.

3. **Liczba obserwacji**: mała/średnia (przy założeniu, że liczy się czas uczenia modelu).
4. **Dokładność algorytmu**: wysoka.
5. **Czas uczenia**: średni (przyczyną jest większa złożoność obliczeniowa niż w przypadku podstawowej wersji drzewa decyzyjnego; choć sam czas wykonywania będzie mniejszy niż w przypadku wzmocnionego drzewa decyzyjnego - wykonywanie równoległe vs sekwencyjne).
6. **Liniowość**: nie.

2.2.6 Sieć neuronowa

1. **Opis algorytmu**. Algorytm klasyfikacji dwu i wieloklasowej. Bodajże najbardziej wyrafinowany algorytm spośród wszystkich, które przedstawiam w tym poradniku. Inspirowany działaniem ludzkiego mózgu. Swoje zastosowanie ma również w problemach regresyjnych.

   Schemat działania sieci neuronowej jest opisywany za pomocą acyklicznego grafu skierowanego. Głównym elementem sieci neuronowej jest neuron przetwarzający. W sieci znajduje się wiele neuronów, które posiadają dowolną liczbę wejść i wyjść. Neurony pogrupowane są w warstwy, w których każdy neuron jest połączony z każdym neuronem warstwy poprzedzającej.

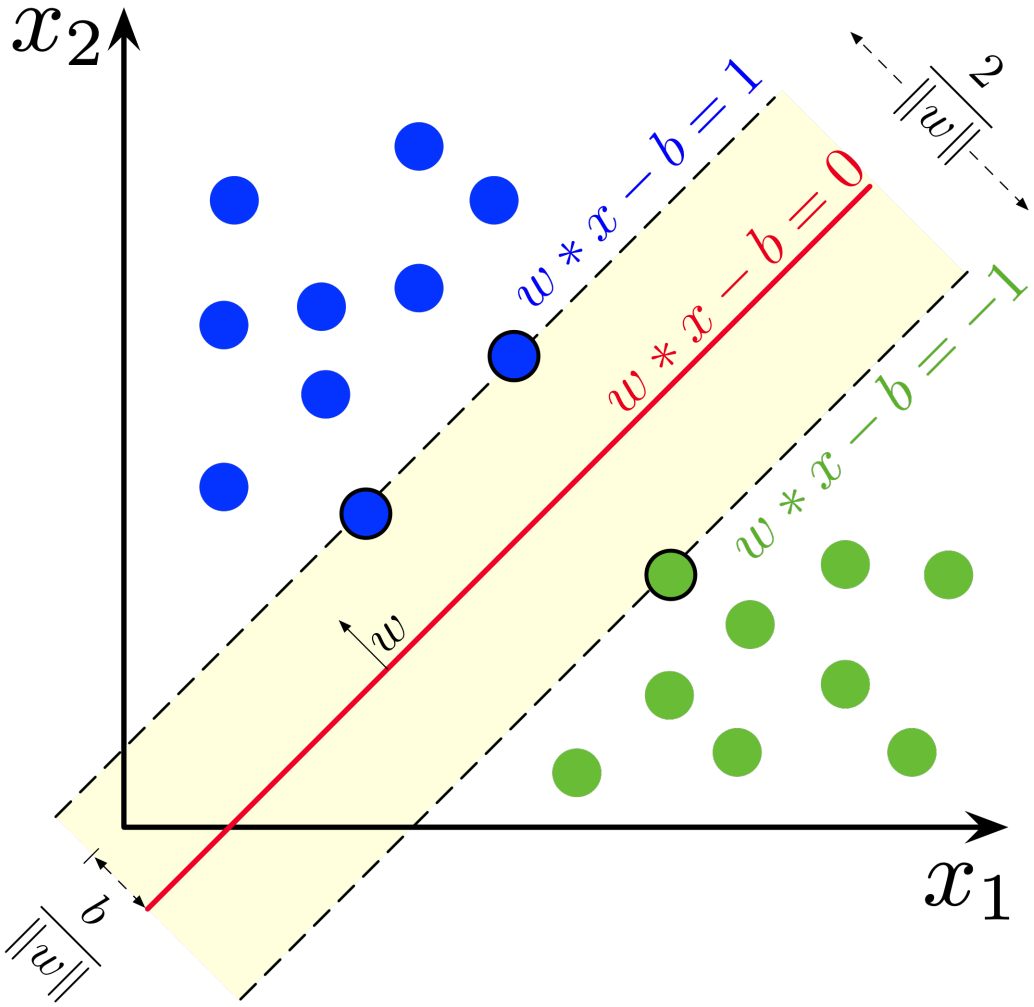
   Wartości zmiennych są zatem przekazywane w sposób postępujący, pomiędzy poszczególnymi warstwami sieci neuronowej. W kolejnym warstwach są wykonywane operacje na zmiennych, aż do osiągnięcia wartości wynikowej na końcu grafu.

2. **Przykładowe zastosowanie**:

   - Problemy, w których ciężko dopatrzeć się prostych wzorców.

3. **Liczba obserwacji**: mała (ze względu na długi czas uczenia).
4. **Dokładność algorytmu**: wysoka.
5. **Czas uczenia**: długi.
6. **Liniowość**: nie.

2.2.7 Maszyna wektorów nośnych (ang. *Support Vector Machines*)



Rysunek 4: SVM - przykład separacji obserwacji

1. **Opis algorytmu**. Algorytm opiera swe działanie na modelu abstrakcyjnej maszyny działającej jak klasyfikator. Jej celem jest oddzielenie obserwacji „granicą" o możliwie największym marginesie. Jeśli klasyfikator nie może znaleźć idealnego podziału, szuka rozwiązania dającego najlepsze rezultaty.

   SVM najlepiej spisuje się przy problemach dotyczących analizy tekstu i wykrywaniu obrazów. Wymaga niewielkiej ilości pamięci.

2. **Przykładowe zastosowanie**:

   - Analiza tekstu (filtry spamu, analiza danych z Twitter, etc., analiza opinii).

3. **Liczba obserwacji**: duża.
4. **Dokładność algorytmu**: średnia/wysoka (w zależności od typu problemu).
5. **Czas uczenia**: krótki.
6. **Liniowość**: tak.

2.2.8 Uśredniony perceptron (ang. *Averaged Perceptron*)

1. **Opis algorytmu**. Uproszczona wersja klasycznej sieci neuronowej. Perceptron składa się z pojedynczego neuronu z regulacją wag. Dzięki swej prostocie eliminuje największą wadę sieci neuronowej - długi czas uczenia. Mimo iż nieco prymitywny, to charakteryzuje się wysoką stabilnością, szybkością oraz liniowością.

   Kiedy zatem stosować AP? Otóż wszędzie tam, gdzie spotykamy się z mniej skomplikowanymi problemami, przy których zależności pomiędzy poszczególnymi zmiennymi jesteśmy w stanie zdefiniować za pomocą prostych wzorców. Przy bardziej wyrafinowanych problemach warto skorzystać z sieci neuronowej (o ile oczywiście nie odstrasza nas długi czas uczenia).
2. **Przykładowe zastosowanie**:
   - Wszędzie tam, gdzie sieć neuronowa jest za wolna, a regresja logistyczna daje niesatysfakcjonujące wyniki.
   - Tam, gdzie mamy do czynienia z dużymi strukturami danych.
3. **Liczba obserwacji**: duża.
4. **Dokładność algorytmu**: średnia.
5. **Czas uczenia**: krótki/średni.
6. **Liniowość**: tak

2.2.9 Pozostałe algorytmy

Powyższe zestawienie oczywiście nie wyczerpuje tematu algorytmów klasyfikacyjnych. Cały czas powstają nowe, z których część zdobywa popularność (np. LightGBM). W większości czerpią jednak one ze znanych już technik, dlatego starałem się przybliżyć Ci drogi czytelniku te najpopularniejsze, które jednocześnie są powszechnie dostępne i używane w większości przypadków.

W literaturze natknąłem się na ciekawy koncept dotyczący użycia metod regresyjnych przy problemach klasyfikacji. Bo skoro w przypadku regresji logistycznej używamy wartości ciągłej do podjęcia decyzji

binarnej, to czemu nie można użyć do tego regresji liniowej? W teorii jest to możliwe (btw. czyż nie tym było zastosowanie funkcji logitowej do rozwinięcia regresji linii? :)).

Przykładowa regresja logistyczna zwraca nam wartość z przedziału [0,1] będącą prawdopodobieństwem wystąpienia danego zdarzenia. Najprostszy model zakłada, że jeżeli p(x) > 0.5, to wynik = 1. W przeciwnym wypadku wynik = 0. Teoretycznie można zatem zbudować model oparty o regresję liniową, który będzie wyznaczać wartość „zbliżoną" do [0, 1]. Celowo napisałem „zbliżoną", gdyż algorytmy regresyjne służą do estymacji wartości numerycznych, a nie wyznaczania prawdopodobieństwa. Wyniki wykraczałyby zatem poza zamknięty przedział [0, 1].

Ciężko byłoby nazwać wynik rzędu p(x) = 1.33 prawdopodobieństwem. Dociekliwe osoby powiedzą: przecież otrzymane wyniki dla badanej próby można znormalizować do wartości [0, 1]. Jest to oczywiście prawdą, ale w rezultacie otrzymamy model zawierający całą masę szumu. Można to porównać do wbijania gwoździ z użyciem kombinerek: teoretycznie da się to zrobić, ale po co, skoro mamy młotek (w tym wypadku algorytmu klasyfikacyjne)?

3 Algorytmy regresyjne

3.1 Opis problemu

Z regresją mamy do czynienia wszędzie tam, gdzie zmienna modelowana (opisywana, czy też zmienna celu) przyjmuje wartości ciągłe. Przy określaniu typu problemu wartośSieć neuci przyjmowane przez predyktory nie odgrywają żadnej roli (mogą być to zmienne kategoryczne lub numeryczne, ciągłe czy dyskretne). To zmienna celu definiuje nam typ problemu z jakim się borykamy.

Mamy zatem jedną zmienną typu ciągłego (np. wielkość sprzedaży w nadchodzącym roku, wyrażona w złotówkach), którą modelujemy w oparciu o dostępne predyktory. Do przykładowych problemów tej klasy możemy zaliczyć:

- Prognozowanie wielkości sprzedaży w danym roku fiskalnym. Na podstawie szeregu zmiennych dyskretnych (nazwy klientów, lista nowych produktów, historyczne dane dotyczące produktów, etc.) oraz ciągłych (sprzedaż w ubiegłych latach, wielkość „lejka" na bieżący rok) można prognozować sprzedaż w nadchodzących dniach i miesiącach. Te kilka dni marginesu daje czas na reakcję kierownictwu.
- Wycena danej spółki. Na podstawie danych z raportów kwartalnych analitycy i gracze giełdowi są w stanie podać realną wycenę firmy, będącą alternatywą dla wyceny widniejącej na GPW (liczba wyemitowanych akcji * cena akcji). Dzięki temu można podejmować decyzje o sprzedaży bądź kupnie akcji danej spółki. Jeżeli spółka jest przewartościowana, podejmujemy decyzję o sprzedaży. Jeśli spółka jest znacznie niedowartościowana, warto rozważyć zakup akcji.
- Potencjalny zysk ze sprzedaży danego produktu. Bazując na opisie produktu oraz danych historycznych sprzedaży produktów podobnych można estymować jego przyszłą sprzedaż.
- Wyznaczenie wartości danej nieruchomości. Mając takie informacje jak: położenie nieruchomości, typ zabudowy, rok budowy, standard wykończenia, liczba pokoi, piętro oraz dane historyczne nt. sprzedaży nieruchomości w okolicy, można przeprowadzić analizę której zmienną modelowaną jest cena nieruchomości.
- Obliczanie zdolności kredytowej danej osoby. Znając podstawowe informacje o danej osobie (wiek, miejsce zatrudnienia, zarobki roczne, okres zatrudnienia, etc.), można estymować jej zdolność kredytową.

Oczywiście powyższe zestawienie nie wyczerpuje tematu i przykłady można szerzyć. Celem jest tu jednak zobrazowanie problemu niż przekrojowa analiza zastosowań.

3.1.1 Mierzenie dokładności algorytmów decyzyjnych

Wyznaczanie wartości ciągłych wydaje się już na pierwszy rzut oka problemem znacznie trudniejszym niż problemy klasyfikacji. Znacznie trudniej jest poprawnie wytypować wartość ciągłą, nawet gdy

znamy medianę i odchylenie standardowe próby z populacji, niż wskazać jedną z dwóch wartości znanych wartości (tak jak to jest w przypadku problemów klasyfikacji). Wobec tego zmierzenie wyników działania algorytmu jest też nieco trudniejsze.

Do podstawowych miar służących do oceny jakości modelu możemy zaliczyć:

- **Błąd średniokwadratowy** – MSE (ang. *Mean Squared Error*). Jest wartością oczekiwaną kwadratu błędów. Błędem określamy tu różnicę pomiędzy wartością uzyskaną za pomocą algorytmu a wartością rzeczywistą. Wraz ze spadkiem MSE rośnie dopasowanie modelu.
- **Pierwiastek błędu średniokwadratowego** – RMSE (ang. *Root Mean Squared Error*). Już sama nazwa wskazuje nam, czym jest RMSE i jak się go liczy.
- **Średni błąd bezwzględny** – MAE (ang. *Mean Absolute Error*). Warto dodać, że dzięki MAE możemy uzyskać jeszcze jedną pomocniczą miarę dopasowania algorytmu: MAPE.
- **Średni bezwzględny błąd procentowy** – MAPE (ang. *Mean Absolute Percentage Error*). Mówi nam ona o procentowym obciążenie algorytmu błędem (MAE/średnia wartość zmiennej modelowanej ze zbioru walidującego).

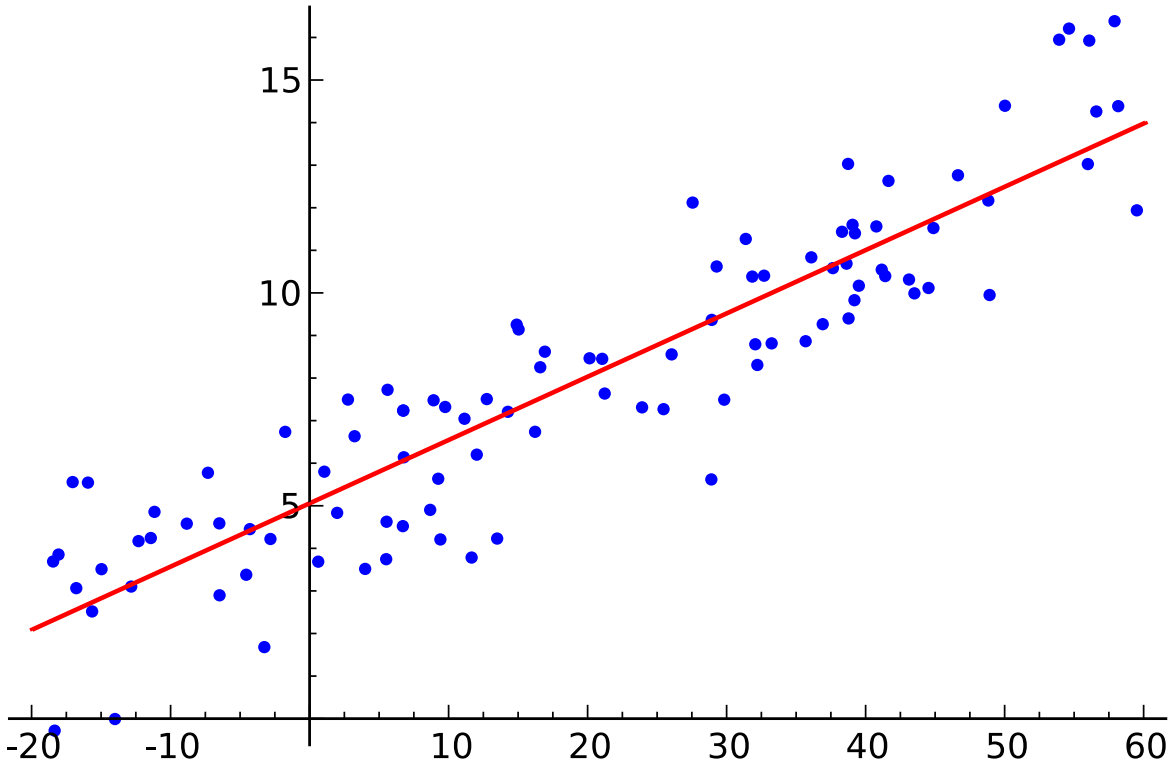
3.1.2 Wybór odpowiedniego algorytmu regresyjnego

Podobnie jak w przypadku algorytmów klasyfikacyjnych przy wyborze optymalnego algorytmu należy się kierować kilkoma podstawowymi kryteriami:

- **Szybkość działania** – w przypadku algorytmów liniowych, takich jak np. regresja liniowa zarówno czas wykonywania, jak i uczenia będzie relatywnie krótki w porównaniu do np. sieci neuronowej.
- **Interpretowalność** – cecha mówiąca o tym, czy działanie danego algorytmu może być łatwo interpretowane przez człowieka.
- **Skalowalność** – ilość zasobów, jakich potrzebuje wybrany algorytm do pełnego wykonania.
- **Liniowość** – cecha mocno skorelowana z szybkością. Z definicji algorytmy liniowe będą działać szybciej niż nieliniowe.
- **Jakość predykcji** – mierzona za pomocą miar takich jak: RMSE, czy też MAE.

3.2 Opis poszczególnych algorytmów

3.2.1 Regresja liniowa



Rysunek 5: Funkcja regresji liniowej z jedną zmienną

1. **Opis algorytmu**. Algorytm pierwszego wyboru przy prognozowaniu wartości ciągłych. Jeden z najpopularniejszych i najprostszych algorytmów. Zakłada on istnienie zależności liniowej pomiędzy zmienną modelowaną a predyktorem/-ami. Jest to być może pewne uproszczenie, ale w większości przypadków jest ono zasadne, np. im więcej produktów sprzeda dana firma, tym większy osiągnie przychód. W najprostszym przypadku regresji liniowej przedstawia ona jedną zmienną modelowaną i jeden predyktor. Zależność pomiędzy nimi jest modelowana na dwuwymiarowym układzie współrzędnych za pomocą prostej o odpowiednim nachyleniu. Odzwierciedla ona trend i zmienność danych.

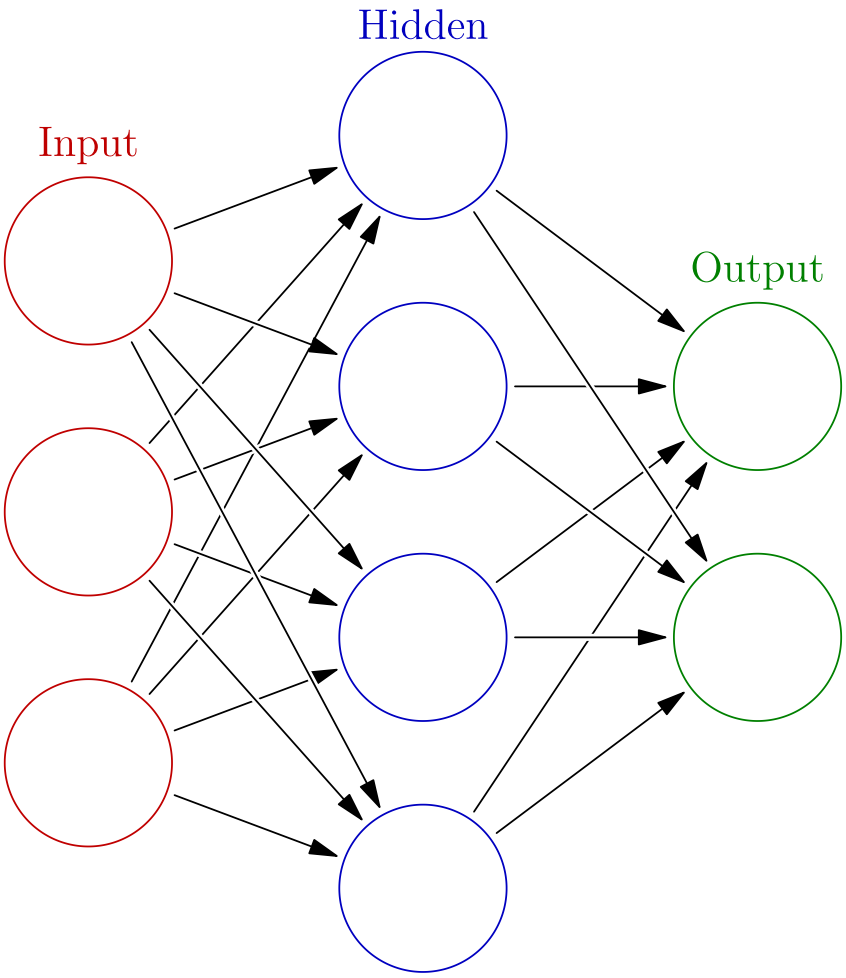
   Rozwiązaniem problemu prognozowania z użyciem regresji liniowej będą odpowiednio dobrane parametry. To one definiują, jak bardzo prosta jest dopasowana do danych. Oczywiście celem jest tu minimalizowanie sumy pionowych odległości wszystkich obserwacji od prostej. Wykorzystuje się do tego m.in. metodę estymacji najmniejszych kwadratów.

Regresja liniowa jest prosta w zrozumieniu i użyciu. Zapewnia szybkie uczenie, a czas wykonywania algorytmu jest krótki. Ma jednak jedną dużą wadę, o której należy pamiętać: jest bardzo wrażliwa na wartości odstające. Jedna odstająca obserwacja może nam znacznie zaburzyć proces optymalizacji współczynników modelu. Dane wejściowe powinny być zatem dobrze przygotowane: usunięte wartości odstające, braki oraz duplikaty.

UWAGA: algorytm jest bardzo wrażliwy na wartości odstające obserwacji!

2. **Liczba obserwacji**: duża.
3. **Dokładność algorytmu**: niska/średnia.
4. **Czas uczenia**: krótki – dzięki prostej strukturze i niskiej złożoności obliczeniowej.
5. **Czas wykonywania**: krótki.
6. **Liniowość**: tak.

3.2.2 Sieć neuronowa



Rysunek 6: Schemat sieci neuronowej z jedną warstwą ukrytą

1. **Opis algorytmu**: Algorytm używany zarówno w problemach klasyfikacji, jak i regresji. Ostatnio przeżywający drugą młodość ze względu na popularność deep learningu. Jest jednym z bardziej wyrafinowanych algorytmów uczenia maszynowego. Inspirowany działaniem ludzkiego mózgu.

   Schemat działania sieci neuronowej jest opisywany za pomocą acyklicznego grafu skierowanego. Głównym elementem sieci neuronowej jest neuron przetwarzający. W sieci znajduje się wiele neuronów, które posiadają dowolną liczbę wejść i wyjść.

   Neurony pogrupowane są w warstwy, w których każdy neuron jest połączony z każdym neuronem warstwy poprzedzającej. Wartości zmiennych są zatem przekazywane w sposób postępujący, pomiędzy poszczególnymi warstwami sieci neuronowej. W kolejnym warstwach są wykonywane

operacje na zmiennych, aż do osiągnięcia wartości wynikowej na końcu grafu.

2. **Liczba obserwacji**: mała/średnia (ze względu na długi czas uczenia. Co ciekawe sieć neuronowa znakomicie radzi sobie z większą liczbą predyktorów.)
3. **Dokładność algorytmu**: wysoka.
4. **Czas uczenia**: długi.
5. **Czas wykonywania**: średni/długi.
6. **Liniowość**: nie.

3.2.3 Drzewo decyzyjne – regresyjne

1. **Opis algorytmu**. Podstawowa wersja drzewa decyzyjnego. Mimo iż drzewo decyzyjne budowane jest na podstawie decyzji binarnych, to obecnie używane pakiety implementujące algorytm radzą sobie również ze zmiennymi ciągłymi. Zawierają one mechanizm przekształcający zmienne ciągłe w predyktor binarny. Dla przykładu, w przypadku zmiennej „Spalanie" mówiącej nam o średnim spalaniu danego auta, można sklasyfikować ilość zużytej benzyny na 100 km: „mniej niż 10" i „więcej niż 10".
2. **Liczba obserwacji**: duża.
3. **Dokładność algorytmu**: średnia/wysoka.
4. **Czas uczenia**: krótki.
5. **Czas wykonywania**: średni.
6. **Liniowość**: nie.

3.2.4 Regresyjny las losowy (ang. *Regression forest*)

1. **Opis algorytmu**. Algorytm buduje *n*-drzew (*n* jest parametrem wejściowym algorytmu). Ostateczna decyzja co do wyznaczenia predykcji jest podejmowana za pomocą głosowania większościowego. Każde kolejne drzewo jest budowane na podstawie **próby bootstrapowej**. Polega ona na wylosowaniu ze zwracaniem k obserwacji z ciągu uczącego o długości K. Co więcej, w każdym drzewie podział odbywa się za pomocą m wylosowanych cech (atrybutów) danej obserwacji. Liczba wylosowanych cech jest mniejsza niż liczba wszystkich dostępnych cech.

   W lesie losowym powstaje zatem *n*-drzew, składających się z *k*-obserwacji (każde drzewo może mieć unikalny zestaw cech), bazujących na *m* spośród wszystkich cech.
2. **Liczba obserwacji**: duża.
3. **Dokładność algorytmu**: wysoka.

4. **Czas uczenia**: krótki.
5. **Czas wykonywania**: średni/długi.
6. **Liniowość**: nie.

3.2.5 Drzewo decyzyjne wzmocnione (ang. *Boosted decision tree*) – regresyjne

1. **Opis algorytmu**. Odmiana drzewa decyzyjnego. Wykorzystuje procedurę wzmacniającą AdaBoost (ang. *Adaptive Boosting*) (ew. inną odmianę wzmocnienia).

   Podobnie jak w przypadku lasu losowego algorytm losuje nowe ciągi uczące. Główna różnica polega na tym, iż każde kolejne budowane drzewo stara się kompensować błędy swojego poprzednika. Owo „ulepszanie" jest możliwe dzięki przypisaniu wag do obserwacji. W przypadku popełnienia błędu zwiększana jest waga danego elementu i jest on częściej losowany w kolejnym algorytmie.
2. **Liczba obserwacji**: duża.
3. **Dokładność algorytmu**: wysoka.
4. **Czas uczenia**: średni.
5. **Czas wykonywania**: średni.
6. **Liniowość**: nie.

> Jeśli chodzi o zastosowanie drzew, jeśli głównym celem jest dokładność jaką osiągnie model predykcyjny, to ogólna zasada wygląda następująco:
> Boosted Decission Tree > RandomForest > Bagging Decission Tree > Decission Tree

3.2.6 Pozostałe algorytmy

Oczywiście powyższe zestawienie nie wyczerpuje tematu. Jest jeszcze co najmniej kilka algorytmów regresyjnych którym warto się przyjrzeć ale przecież nie o mnożenie przykładów chodzi.

Jeśli mierzysz wysoko i Twoim celem jest szlifowanie wyniku na Kaggle, to pewni skłonisz się ku bibliotece XGBoost i dostępnym tam algorytmom wykorzystującym „wzmocnienie". Jeśli natomiast chcesz rozwiązać podstawowy problem i bardziej zależy Ci na szybkości i prostocie to w zupełności wystarczy Ci podstawowa regresja liniowa (w wersji spełniającej założenia projektu nad którym pracujesz), która doskonale się sprawdzi w większości przypadków.

4 Algorytmy grupujące

4.1 Opis problemu grupowania

Obiektywnie, w kwestii doboru odpowiedniego algorytmu grupowanie jest znacznie bardziej złożone niż regresja, czy klasyfikacja. W przypadku grupowania trudno o jedną, spójną statystykę, którą można porównać działanie różnych algorytmów. Sytuacja komplikuje się jeszcze bardziej, gdy weźmiemy pod uwagę transformacje, które musimy wykonać na zbiorze, by dostosować go do wymagań danego algorytmu.

W tej części spróbuję zmierzyć się z powyższym wyzwaniem i przedstawić zestaw wskazówek, które być może pomogą Ci w wyborze metody klastrowania dostosowanej do zestawu danych, którymi dysponujesz.

Pomimo że grupowanie nie jest najpoczytniejszym tematem w uczeniu maszynowym, to chciałem Ci nieco przybliżyć jego ideę. Od początku moim celem było kompleksowe opisanie problemu i zainspirowanie Cię do zgłębiania tego niedocenianego tematu. Mam nadzieję, że przynajmniej w niewielkim stopniu mi się to udało. :)

4.1.1 Działanie algorytmów grupujących

Problemem grupowania w uczeniu maszynowym nazywamy podział próby (zbioru) na grupy obserwacji, które cechują się podobnymi właściwościami. Algorytmy używane w grupowaniu należą do rodziny określanej uczeniem bez nadzoru. W przeciwieństwie do klasyfikacji, czy regresji nie uświadczymy tu zmiennej celu. Podział dokonywany jest zatem na podstawie wszystkich wybranych zmiennych.

Wynikiem działania algorytmów grupujących są grupy obserwacji. Nie wszystkie jednak obserwacje zostają przydzielone do grup. W zależności od wybranej metody grupowania może się zdarzyć, że niektóre obserwacje nie zostanę przypisane do żadnej grupy. Powodem jest fakt, że na płaszczyźnie wielowymiarowej są one odległe od wszystkich pozostałych obserwacji. Nazywamy je wtedy obserwacjami nietypowymi.

4.1.2 Miary używane w grupowaniu

Zwykło się mówić o problemach, które rozwiązują algorytmy segmentacyjne – uczenie bez nadzoru. Nie jest to do końca trafne określenie. O ile same algorytmy segmentacyjne nie posiadają wbudowanej metody optymalizacji pod kątem statystyki bazującej na którejkolwiek zmiennej, to już ich kalibracja, jak i interpretacja wyników wymagają nadzoru eksperta.

Poniżej prezentuję trzy najpopularniejsze miary używane w grupowaniu:

1. **Miara wewnętrzna – wskaźnik sylwetkowy (ang. *silhouette score*).**

   Silhouette nazywany jest miarą wewnętrzną, ponieważ wykorzystuje w swej ocenie średnią odległość pomiędzy obserwacjami wewnątrz grupy (a) i średnią odległość obserwacji do najbliższej „obcej" grupy (b). Silhouette obliczany jest dla każdej obserwacji w następujący sposób: (a – b)/max(a, b). W zależnosci od implementacji funkcja zwraca średnią wartość dla każdej obserwacji lub średnią dla całego zbioru.

2. **Miara zewnętrzna – wskaźnik czystości (ang. *purity score*)**

   Do wyznaczenia tej miary niezbędna jest znajomość prawdziwych (lub oczekiwanych) klas poszczególnych obserwacji. Wykorzystujemy ją przy problemach semi-supervised i w problemach klasyfikacji wieloklasowej. W uproszczeniu: mierzy ona spójność w przydziale obserwacji do grup. W idealnej sytuacji wszystkie obserwacje oznaczone tym samym numerem trafiają do tej samej grupy.

3. **Inercja (ang. *inertia*).**

   Ostatnia i najbardziej intuicyjna miara. Dotyczy ona głównie algorytmu k-średnich. **Inercja jest sumą kwadratów odległości obserwacji wewnątrz segmentu od jego centrum.** W implementacji k-means w bibliotece sklearn inercją mierzona jest jakość segmentacji. Algorytm minimalizuje ją i spośród kilku wykonań wybierane jest to, w którym wartość inercji jest najmniejsza.

4.2 Opis poszczególnych algorytmów

4.2.1 Algorytm grupowania aglomeracyjnego

1. **Opis algorytmu**. Należy do rodziny algorytmów hierarchicznych. Swą nazwę bierze od hierarchii, którą budują algorytmy implementujące to podejście. Ma ona strukturę drzewa (wizualizowanego na wykresie zwanym dendrogramem), które przedstawia proces działania algorytmu i budowy grup (poprzez podział bądź scalanie). Na osi x uporządkowane są grupy, a z osi y w łatwy sposób można odczytać odległość pomiędzy łączonymi grupami.

   W podejściu aglomeracyjnym (wstępujące lub łączące) algorytm zaczyna swe działanie w momencie, gdy liczba grup równa się liczbie obserwacji – każda obserwacja stanowi odrębną grupę. Następnie w sposób iteracyjny grupy są scalane w taki sposób, by wariancja wewnątrz nich była możliwie najmniejsza, a pomiędzy grupami możliwie duża. Algorytmy te reprezentują podejście „od szczegółu do ogółu".

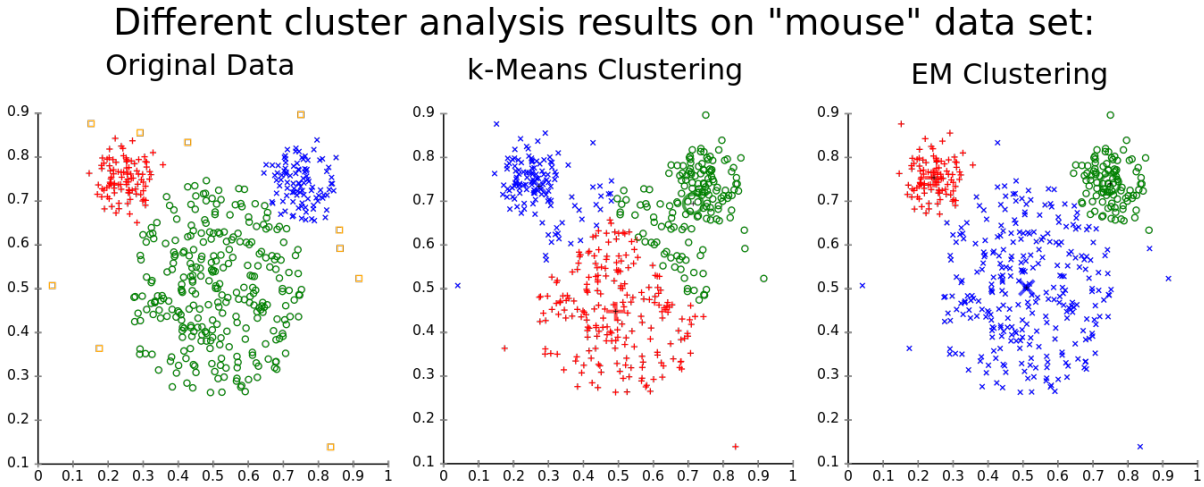
   Podejście zaimplementowane w algorytmach aglomeracyjnych sprawdza się, gdy decydujemy się na poszukiwanie małych grup (po kilku iteracjach proces może zostać przerwany).

2. **Rodzina algorytmów:** hierarchiczne.
3. **Założenia i ograniczenia:**
    - nie wymaga wcześniejszego podania docelowej liczby grup (choć w przypadku implementacji sklearn wymagane jest określenie liczby grup),
    - liczbę grup możemy określić dopiero po np. wnikliwej analizie dendrogramu,
    - wysoka złożoność obliczeniowa.
4. **Obsługiwane typy zmiennych:** tylko numeryczne.
5. **Szybkość:** niska.
6. **Więcej informacji:**
    - wprowadzenie teoretyczne,
    - praktyczne użycie.

4.2.2 Algorytm grupowania deglomeracyjnego

1. **Opis algorytmu**. Kolejny algorytm z rodziny hierarchicznych. Bardzo podobny do algorytmu aglomeracyjnego. Główna różnica polega na tym, że rozpoczyna on swe działanie, gdy wszystkie obserwacje znajdują się w jednej grupie. W kolejnych iteracjach działania algorytmu grupy są dzielone, mając na uwadze te same kryteria, co w przypadku algorytmów aglomeracyjnych: wariancję i miary odległości pomiędzy grupami. Tego typu algorytmy sprawdzają się, gdy zależy nam na znalezieniu dużych grup (podejście „od ogółu do szczegółu").
2. **Rodzina algorytmów:** hierarchiczne.
3. **Założenia i ograniczenia:** identyczne jak w przypadku algorytmu grupowania aglomeracyjnego.
4. **Obsługiwane typy zmiennych:** tylko numeryczne.
5. **Szybkość:** niska.
6. **Więcej informacji:**
    - wprowadzenie teoretyczne.

4.2.3 K-średnich



Rysunek 7: K-średnich - przykład segmentacji zbioru z obserwacjami ułożonymi w kształcie myszy

1. **Opis algorytmu**. O algorytmie k-średnich słyszał chyba każdy, kto zetknął się z problemem grupowania. Swą popularność zawdzięcza on prostocie i szybkości wykonywania.

   Swoje działanie k-średnich opiera na centroidach (utworzonych na podstawie średniej ze współrzędnych punktów w grupie), podobnie jak to miało miejsce w przypadku metod hierarchicznych. Jest on zaliczany do metod iteracyjno-optymalizacyjnych ze względu na schemat działania.

   W kolejnych iteracjach wykonywana jest optymalizacja wyniku działania algorytmu przedstawionego w postaci odległości wszystkich obserwacji danej grupy względem jej centroidu. Dąży on zatem do minimalizacji wariancji wewnątrz grup i jej maksymalizacji pomiędzy grupami.

2. **Rodzina algorytmów:** iteracyjno-optymalizacyjne.

3. **Założenia i ograniczenia:**

   - wymaga od użytkownika ustalenia liczby grup *a priori*,
   - wrażliwość na dobór punktów startowych,
   - wrażliwość na wpływ obserwacji odstających i szum,
   - każda powstała grupa posiada przybliżoną liczbę obserwacji,
   - zakłada sferyczność grup (tu znajduje wpis dedykowany ograniczeniom algorytmu k-średnich, wraz ze sposobami na ich obejście),

4. **Obsługiwane typy zmiennych:** tylko numeryczne.

5. **Szybkość:** wysoka. W zależności od implementacji i zbioru jest to jeden z najszybszych algorytmów.

6. Schemat działania algorytmu k-średnich:

    1. Losowanie k-punktów startowych w przestrzeni (losujemy k pierwszych centroidów).
    2. Przyporządkowanie wszystkich obserwacji do najbliższego centroidu z pomocą danej miary odległości (w przypadku k-średnich jest to odległość euklidesowa).
    3. Dla każdej z k-grup wyznaczamy nowy centroid.
    4. Powtarzamy krok 2 i 3 aż do osiągnięcia warunku stopu, którym może być:
        1. Osiągnięcie zbieżności, ew. „znacznej" poprawy względem wybranej miary jakości grupowania (w przypadku implementacji dostępnej w bibliotece sklearn jest to: identyczna jak w poprzedniej iteracji suma kwadratów odległości wszystkich obserwacji od centroidu).
        2. Osiągnięcie momentu, w którym przydział obserwacji do grup się nie zmienia.
        3. Osiągnięcie zakładanej liczby iteracji.

7. **Więcej informacji:**

    - wprowadzenie teoretyczne,
    - praktyczne użycie,
    - projekt segmentacji behawioralnej z użyciem algorytmu k-średnich.

4.2.4 K-median

1. **Opis algorytmu**. K-median jest algorytmem grupującym, należącym do rodziny algorytmów iteracyjno-optymalizacyjnych. W swoim działaniu jest on bardzo zbliżony do algorytmu k-średnich. Główna różnica polega na tym, że dla każdej z grup używa on mediany współrzędnych obserwacji zamiast centroidu (ze wszystkimi konsekwencjami takiego zabiegu, czyli np. **brakiem wrażliwości na wartości odstające**).
2. **Rodzina algorytmów:** iteracyjno-optymalizacyjne.
3. **Założenia i ograniczenia:** niemal identyczne, jak w przypadku k-średnich, z wyłączeniem wrażliwości na wartości odstające.
4. **Obsługiwane typy zmiennych:** tylko numeryczne.
5. **Szybkość:** wysoka.
6. **Więcej informacji:**

    - wprowadzenie teoretyczne i praktyka.

4.2.5 K-modes

1. **Opis algorytmu**. Algorytm bardzo podobny do algorytmu k-średnich, rozszerzający jego możliwości o wsparcie dla zmiennych kategorycznych. Nazwa algorytmu pochodzi od angielskiego

mode (moda/dominanta/wartość modalna/wartość najczęstsza) - użyta jest tu jako reprezentant tendencji centralnej danej grupy. Jest to obiekt będący reprezentantem danej grupy obserwacji. Jest odpowiednikiem centroidu z algorytmu k-średnich.

Zamiast dystansu (jak w k-średnich) używa on **miary odmienności**. Im mniejsza jej wartość, tym większe podobieństwo pomiędzy obserwacjami. Miara odmienności jest przedstawiona jako suma niedopasowań poszczególnych zmiennych kategorycznych pomiędzy obserwacjami.

Algorytm jest mało popularny. Ciężko znaleźć na jego temat wartościowe informacje w sieci. Być może wynika to z faktu, że wspiera on tylko i wyłącznie zmienne kategoryczne, a nieczęsto spotykamy się ze zbiorami zawierającymi same zmienne dyskretne.

2. **Rodzina algorytmów:** iteracyjno-optymalizacyjne.
3. **Założenia i ograniczenia:**
    - wymaga ustalenia liczby grup *a priori*,
    - wrażliwy na dobór punktów startowych,
    - używa jedynie zmiennych kategorycznych.
4. **Obsługiwane typy zmiennych:** tylko kategoryczne (obsługuje braki danych).
5. **Szybkość:** wysoka.
6. **Schemat działania algorytmu k-modes**:
    1. Losowe wybranie k unikalnych obserwacji ze zbioru jako inicjalnych k-dominant.
    2. Przeliczenie miary odmienności pomiędzy dominantami a wszystkimi obserwacjami.
    3. Przypisanie obserwacji do najbardziej podobnej dominanty.
    4. Wybranie nowych k-dominant dla każdej grupy obserwacji. Jeżeli dominanty się nie zmieniły, to algorytm kończy działanie. W przeciwnym wypadku powtarzane są kroki 2, 3 i 4.
7. **Więcej informacji:**
    - wprowadzenie teoretyczne i praktyka,
    - k-modes vs k-średnich - porównanie na przykładowym zbiorze.

4.2.6 K-prototypów

1. **Opis algorytmu**. Ze względu na schemat działania algorytm ten jest zaliczany do metod iteracyjno-optymalizacyjnych. Jest on kolejną ewolucją algorytmów k-średnich i k-modes. Zachowuje wszystkie zalety swoich protoplastów, eliminując przy tym kilka istotnych wad.

W kolejnych iteracjach działania algorytmu wykonywana jest optymalizacja wyniku przedstawionego w postaci odległości wszystkich obserwacji danej grupy względem jej "prototypu".

Nazwa k-prototypes odnosi się do *k*-prototypów będących reprezentantami tendencji centralnej danej grupy. Prototyp jest obiektem będącym reprezentantem danej grupy obserwacji. Jest odpowiednikiem centroidu z algorytmu k-średnich i mody z algorytmu k-modes. Algorytm ten jest połączeniem k-średnich i k-modes.

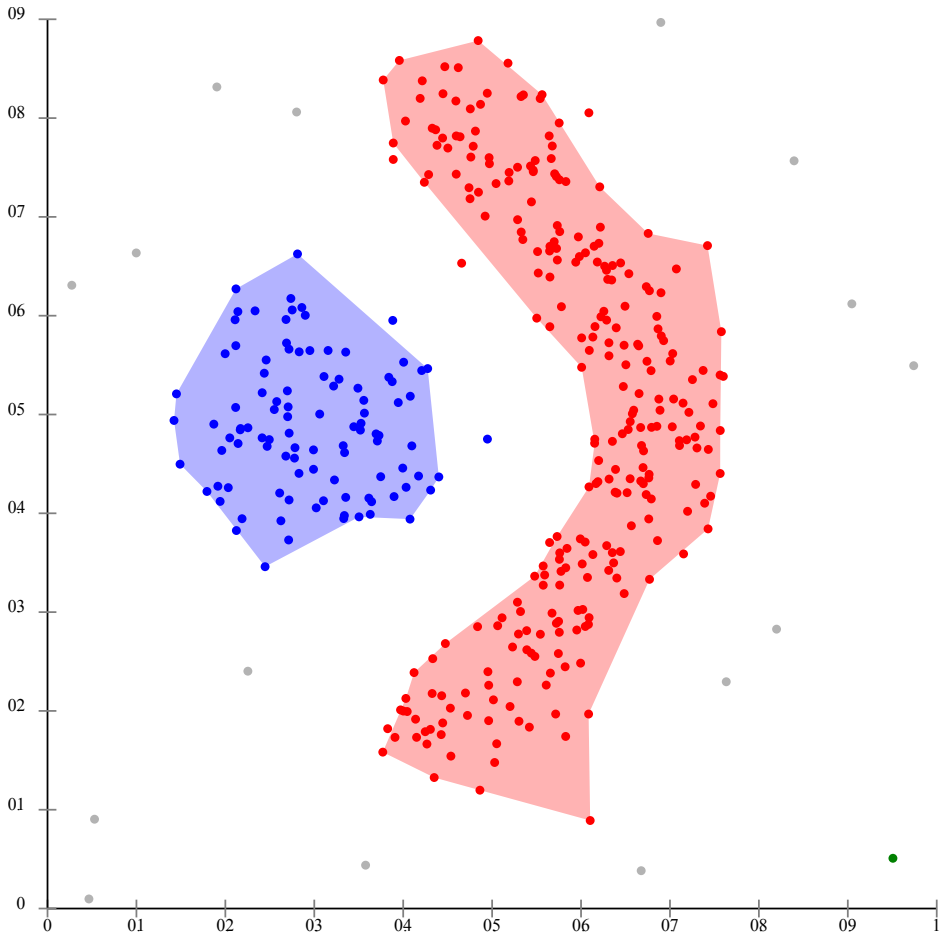
Zamiast dystansu (jak w k-średnich) używa on miary odmienności, będącej zmodyfikowaną wersją miary odmienności zaimplementowanej w algorytmie k-modes. Jest ona hybrydą odległości z k-średnich i miary odmienności k-modes.

Algorytm dąży do minimalizacji wariancji wewnątrz grup i jej maksymalizacji pomiędzy grupami. Dla zmiennych ciągłych algorytm bazuje na centroid, natomiast dla zmiennych kategorycznych algorytm bazuje na częstościach występowania kategorii.

Im mniejsza wartość miary odmienności, tym większe podobieństwo pomiędzy obserwacjami. Miara odmienności jest przedstawiona jako suma niedopasowań poszczególnych zmiennych kategorycznych pomiędzy obserwacjami.

2. **Rodzina algorytmów:** iteracyjno-optymalizacyjne.
3. **Założenia i ograniczenia:**
    - wymaga ustalenia liczby grup *a priori*,
    - wrażliwy na dobór punktów startowych.
4. **Obsługiwane typy zmiennych:** numeryczne i kategoryczne (obsługuje braki w danych).
5. **Szybkość:** średnia/wysoka (w zależności od zbioru).
6. **Więcej informacji:**
    - wprowadzenie teoretyczne,
    - projekt z użyciem algorytmu k-prototypów.

4.2.7 DBSCAN



Rysunek 8: DBSCAN - przykład segmentacji zbioru z grupami liniowo nieseparowalnymi

1. **Opis algorytmu**. Bodaj najpopularniejszy reprezentant rodziny algorytmów gęstościowych. Segmenty buduje na podstawie informacji o zagęszczeniu obserwacji w przestrzeni. Poprzez zagęszczenie należy rozumieć odległości dzielące poszczególne obserwacje w danym obszarze. W mojej ocenie jest ono najbardziej intuicyjną metodą grupowania.

   O odróżnieniu od algorytmów iteracyjno-optymalizacyjnych znakomicie radzi sobie przy identyfikacji segmentów o nieregularnych (niewypukłych) kształtach. Daje możliwość budowania grup o różnych kształtach (brak założenia o sferyczności grup, tak, jak w przypadku algorytmu k-średnich).

   W przypadku DBSCAN nie wszystkie obserwacje muszą zostać przypisane do grup. Obserwacje niespełniające zakładanych kryteriów są uznawane za obserwacje odstające. Dzięki temu bywa on używany przy problemach detekcji anomalii.

DBSCAN posiada jedynie dwa parametry, których optymalizację trzeba wykonać. Ich dobór w taki sposób, by powstała „odpowiednia” liczba grup i by obserwacje nietypowe nie stanowiły znaczącej części zbioru, jest jednak dosyć poważnym wyzwaniem.

2. **Rodzina algorytmów:** gęstościowe.
3. **Założenia i ograniczenia:**
    - nie daje możliwości definiowania *a priori* liczby grup,
    - dobór odpowiednich parametrów bywa dosyć problematyczny.
4. **Obsługiwane typy zmiennych:** tylko numeryczne.
5. **Szybkość:** średnia/wysoka (w zależności od zbioru i implementacji)
6. **Więcej informacji:**
    - wprowadzenie teoretyczne,
    - praktyczne użycie.

5 Podsumowanie i dodatkowe materiały

Mam nadzieję, że opisane w tym poradniku informacje okażą się dla Ciebie przydatne i będziesz z nich regularnie korzystać przy wykonywanych przez Ciebie projektach.

Jeśli spodobała Ci się lista i opisane w niej wskazówki, to zapraszam Cię na mojego bloga. Znajdziesz na nim więcej wartościowych wskazówek i porad.

Poniżej przedstawiam **listę linków do dodatkowych materiałów**, które chciałbym Ci polecić jako uzupełnienie wiedzy zawartej w poradniku. Podzieliłem je na 4 sekcje:

1. Materiały dotyczące algorytmów klasyfikacyjnych.
2. Materiały dotyczące algorytmów regresyjnych.
3. Materiały dotyczące algorytmów grupujących.
4. Wartościowe materiały, które publikuję na swoim blogu.

5.1 Algorytmy klasyfikacyjne

- Wybór punktu odcięcia w algorytmach klasyfikacyjnych
- Jak wybrać punkt odcięcia, bazując na uwarunkowaniach biznesowych?
- Poprawa działania regresji logistycznej - kategoryzacja zmiennych ciągłych

5.2 Algorytmy regresyjne

- Błąd średniokwadratowy
- Pierwiastek błędu średniokwadratowego
- Średni błąd bezwzględny
- Jak działa głęboka sieć neuronowa

5.3 Algorytmy grupujące

- Seria wpisów o algorytmach grupujących.
- Porównanie czasu wykonywania algorytmów grupujących.
- Porównanie czasu wykonywania i wyników algorytmów grupujących - sklearn.

5.4 Materiały z bloga MateuszGrzyb.pl

Prowadząc blog, publikuję na nim wiele artykułów, z których być może część Cię zainteresuje. Przykładowe wpisy, które opublikowałem na blogu:

- Jaki język programowania wybrać?
- 7 najczęstszych błędów w Data Science
- 2 proste i skuteczne metody optymalizacji parametrów modelu
- Jak wybrać punkt odcięcia, bazując na uwarunkowaniach biznesowych?
- Prosty sposób na wybór odpowiedniego punktu odcięcia
- Wybór liczby segmentów w algorytmie k-średnich
- Ocena jakości modelu segmentacyjnego
- Blogi, które polecam
- Książki, które polecam
- 3 najlepsze ściągawki z bibliotek Python

Na blogu publikuję również mini-projekty, w których biorę na tapetę jeden zbiór danych, jeden problem i staram się go rozwiązać, bazując na metodach i narzędziach używanych w Data Science.

- Przykładowe projekty, które opublikowałem na blogu
- Silnik rekomendacji filmów
- Segmentacja behawioralna klientów z użyciem metody RFM
- Przewidywanie defaultu wśród posiadaczy kart kredytowych
- Klasyfikacja wniosków o wydanie karty kredytowej
- Przepowiednie wyroczni z Omaha
- Największy przeciwnik króla boksu
- Segmentacja pokemonów, czyli wpis z okazji Międzynarodowego Dnia Dziecka

Prowadzę również kanał na Youtube: Data Science Plus. W kolejnych odcinkach staram się wychodzić poza sztywne ramy nauki o danych. Serdecznie zaprasza Cię do odwiedzenia kanału. :)

- Mój kanał na YouTube

Źródła zdjęć:

- Regresja logistyczna
- Regresja liniowa
- Sieć neuronowa
- Drzewo decyzyjne
- SVM
- K-średnich
- DBSCAN

5.5 Uwagi od autora

Czy poradnik jest w 100% darmowy? Tak i nie. Wszystkimi poradami w nim zawartymi dzielę się z Tobą, nie pobierając żadnej opłaty. Traktuj go zatem jako moje podziękowanie za zapisanie się do listy subskrybentów bloga. Jedyną "opłatą" jest Twój adres e-mail dodany do bazy subskrybentów bloga. UWAGA: Twój e-mail NIGDY nie zostanie nikomu przekazany. Nigdy nie zostanie "sprzedany", etc. Używam go tylko (albo aż ;-)) do realizowania swojej pasji - dzielenia się z innymi swoją wiedzą na temat Data Science.

Czy poradnik mogę udostępniać innym osobom? Nie. Chciałbym Cię prosić, być nie udostępniał go nikomu. Jeśli ktoś będzie potrzebować wiedzy w nim zawartej zawsze może zapisać się na newsletter i go otrzymać za darmo. W razie konieczności dopuszczam sytuację, w której można napisać do mnie (poprzez formularz kontaktowy znajdujący się na blogu) z prośbą o plik.

Czy mogę cytować zawartość poradnika, przekazywać jego treść dalej? Oczywiście. Jeśli chcesz wykorzystać fragmenty poradnika - rób to śmiało. Nie potrzebujesz do tego dodatkowej zgody. Wspomnij tylko o źródle. To wszystko. :)

W notatniku jest błąd. Co zrobić? Daj mi proszę znać: kontakt@mateuszgrzyb.pl. :)

Nie zgadzam się z tym co napisałeś. Widzę tu ewidentny błąd. Co zrobić? J.W. Proszę o e-mail na adres: kontakt@mateuszgrzyb.pl. :)

6 O autorze

Mateusz Grzyb. Z branżą IT związany od kilkunastu lat. Certyfikowany specjalista Microsoft w obszarze Data Science, entuzjasta zbiorów rozmytych, oraz wszystkiego co związane z predykcją i szeroko pojętą nauką o danych.

Zawodowo: Ekspert ds. zaawansowanej analityki. Na co dzień dostarczający rozwiązania biznesowe bazujące na szeroko pojętej nauce o danych dla Alior Banku.

Prywatnie: fan kolarstwa, miłośnik dobrej muzyki, kina i gór. Członek Stowarzyszenia Inwestorów Indywidualnych. Po godzinach łączę pasję do eksploracji danych, inwestowania i rynków kapitałowych.