Wykrywanie występowanie chorób serca,porównanie algorytów uczenia maszynowego nadzorowanego na podstawie zbioru danych dotyczących chorób układu krążenia z repozytorium UCI

Magdalena Szulc

Toruń,2022-05-01

Contents

Abstrakt

The aim of the work is to compare selected algorithms of supervised machine learning and build a model based on medical data, which diagnoses the presence or absence of cardiovascular disorders.

Medical data is distinguished by the fact that it is difficult to access, in most cases it is restricted information not available for public use. Therefore, a key step is to choose the features taken into account when creating the model. The data obtained from the UCI repository has already undergone pre-processing. The dataset itself, due to its small size, allows checking effects of algorithms without getting rid of redundant and insignificant features.

The main motive is to answer the question of how data deficiency strongly influences the outcome and whether there is a difference between the use of selected supervised learning algorithms requires a comparison of the difficulty level of creating a model, accuracy, complexity and time to obtain an answer.

Wstęp

Uczenie maszynowe jako spopularyzowana dziedzna metod uczenia , zainteresowanie sobą w dużej mierze zawdzięcza rozwoju procesorów graficznych pozwalających na wykorzystanie algorytmów w optymalnym czasie. Ze względu na chodliwość tematu , powstało nowe oprogramowanie lub przystosowania ułatwiające wydajną pracę z obszernymi zasobami danych. Pojęcie sztucznej inteligencji pochodzi od próby odtowrzenia ludzkiego sposobu myślenia, jedną z bardziej znanych historycznie postacią z tym związaną jest psycholog Frank Rosenblatt z Cornell University. Badacz przyczynił się do powstania projektu zbudowania maszyny o nazwie "perceptron" mającej za zadnie rozpoznawać litery jest prawzorem nowoczesnych sztucznych sieci neuronowych. To dzięki jego badaniach nad perceptronem rozpropagowany został koncept algorytmu uczenia skończonej liczbie wywołań. Z czasem liczba przetwarzanych informacji stopniowo się zwiększała dzięki zastosowaniu procesów równoległych oraz pamięci. Wartą wpomnienia datą jest rok 2006 , w tym roku zaprezentowano opensource'owy odpowiednik MapReduce od Google o który dał sposobnść do przenoszenia między procesorami obróbki Big Data. Rok ten jest również znaczący ze względu na wydanie przez Nividia procesora graficzngeo monopolizującego rynek uczenia maszynowego. Zmiejszenie kosztów pamięci RAM zaoowocowała powstaniem kolejnych algorytmów uczenia,a istniejące podejścia są sukcesywnie ulepszane.[1] W całym ogromie powstałych już metod uczenia się, z których każda ma swoje zalety i niedogodności.[2]

Diagnnoza medycznej to rozległy temat i pod kątem uczenia maszynowego może być rozpatrywany na podstawie danych tekstowych czy obrazów. Zadanie postawienia diagnozy podlega to typowe zagadnienie klasyfikacji ,ale nie jedyny sposób wykorzystania. Przykładem systemu uczącego może być wycena akcji na giełdzie na podstawie danych z poprzedniego kwartału lub optymalizacja strony na podstawie historii odwiedzeń .**confusion?**

Sztuczna inteligencja wśród szerokiego zakresu swoich zastosowań może zostać wykorzytana do analizy bardziej lub mniej złożonych danych medycznych, w celu przewidzenia wystąpienia choroby u konkretnej osoby, bez udziału procesu myślowego od stony specjalisty.

Do tego przeznaczenia istnieje możliwość zastosowania uczenia nadzorowanego (ang. *supervised learning*) tj. rodzaj uczenia maszynowego zakładający istnienie zbioru danych testowych zawierających odpowiedzi, na których podstawie wyszukiwane są zależności, cechy znaczące oraz budowany jest w ten sposób model służący przykładowo do przewidywania przyszłych wartości.

W przypadku danych dotyczących chorób zależności typujące występowanie choroby, bazują na podstawie konkretnych wyników badań zgormadzonych w repozytorium UCI.

W dzisiejszych czasach choroby sercowo-naczyniowe stanowią najczęstszą przyczynę zgonów, a liczba osób cierpiących na te dolegliwości stale rośnie. Głównymi przyczynami zachorowalności diagnozowanymi przez specjalistów są niski poziom świadomości i profilaktyki chorób serca.Dlatego prowadzone są intensywne prace nad zwiększeniem dostępności badań, które wspomogą diagnostykę kardiologiczną na jak najwcześniejszym etapie [3].

Powodem szukania dokładniejszych sposobów diagnozowania są również wysokie koszty leczenia generowane przez choroby układu krwionośnego. Według analityków firmy konsultingowej KPMG [4] w 2011 r. koszty diagnostyki i terapii chorób serca wyniosły ponad 15 miliardów polskich złotych.

Uczenie maszynowe poprzez przetwarzanie dużych zasobów klinicznych danych historycznych pod kątem zależności przyczynowo skutkowych, może zostać wykorzystane do wczesnej diagnostyki lub wspomagania leczenia pacjentów [5].

Słowa kluczowe: uczenie maszynowe, uczenie nadzorowane, lasy losowe, maszyna wektorów nośnych, k-najbliższych sąsiadów

Cel i zakres pracy

Celem pracy jest porównanie wybranych algorytmów uczenia maszynowego nadzorowanego, przy założeniu że dane wejściowe są wybrakowane, a w rezultacie zbudowanie modelu który na podstawie danych medycznych wystawia diagnozę o występowaniu zaburzeń sercowo-naczyniowych lub ich braku.

Dane medyczne wyróżniają się tym, że trudno uzyskać do nich dostęp, najczęściej nie są to informacje, które się udostępnia do użytku publicznego, z tego powodu, kluczowym krokiem jest wybór cech branych pod uwagę przy tworzeniu modelu. Dane pozyskane z repozytorium UCI przeszły już wstępną obróbkę , sam dataset ze względu na swoje niewielkie rozmiary pozwala na sprawdzenie działań algorytmów bez pozbywania się nadmiarowych i mało znaczących cech.

Zatem odpowiedź na pytanie jak wybrakowanie danych mocno wpływa na rezultat i czy istnieją róznicę między zastosowaniem wybranych algorytmów nauczania nadzorowanego wymaga przedstawienia porównania łatwości tworzenia modelu, dokładności, złożoności oraz czasu uzyskania odpowiedzi.

W pracy opisano następujące algorytmu uczenia nadzorowanego:

- losowe lasy decyzyjne (ang. *random decision forests*)
- maszyna wektorów nośnych (ang. *support vector machines*, SVM)
- k-najbliższych sąsiadów (ang. *k-neares neighbours*, KNN)

Wykrywanie występowanie chorób serca,porównanie algorytów uczenia maszynowego nadzorowanego na podstawie zbioru danych dotyczących chorób układu krążenia z repozytorium UCI

Wprowadzenie teorertyczne

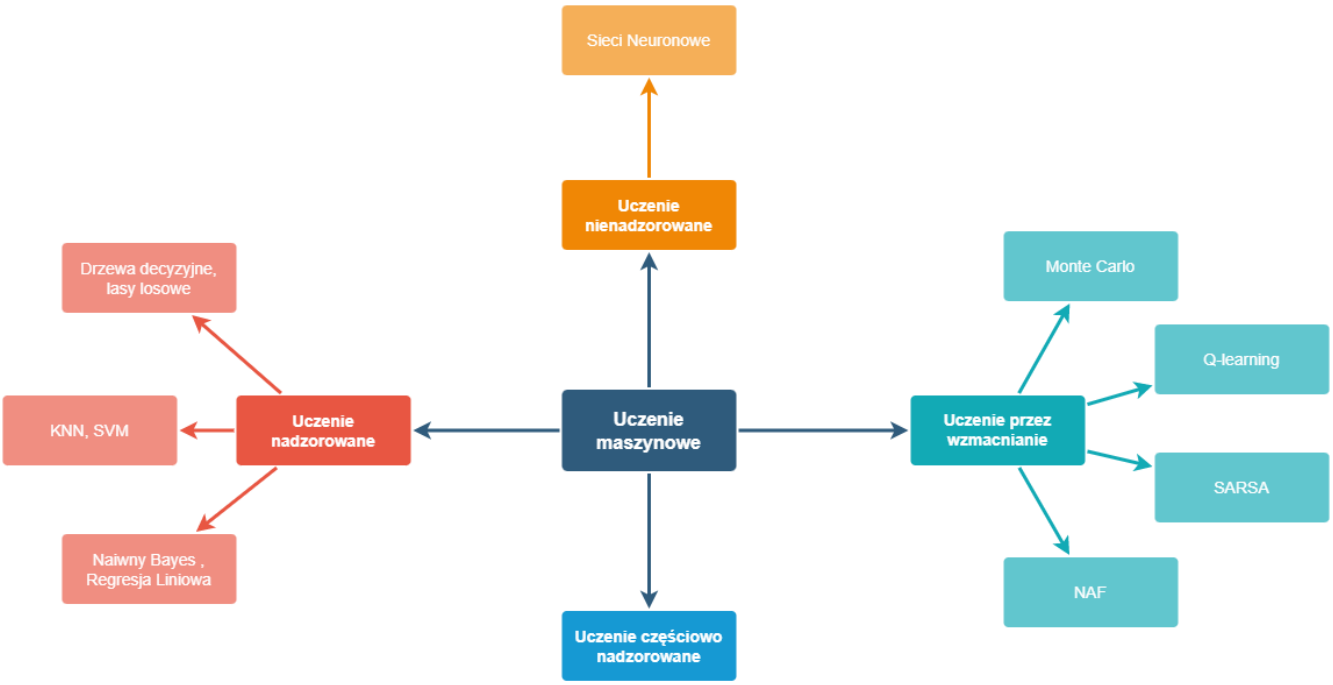
Omówienie teoretyczne rozpoczynają defninicje podstawowych pojęć wykorzystywanych w dalszych częściach, zaczynając od Algorytmu.

Algorytm to pojęcie matematyczne odpowiadające za szereg działań prowadzących do uzyskania żądanego rozwiązania, bazując na wynikach działań krokami opisujemy np problem znajdowania najktórszej drogi jak i zarówno tłumaczenie języka na podstawie analizy mowy.

Uczenie maszynowe (ang. *machine learning*, ML) to dziedzina zajmująca się tworzeniem modeli do analizy bardzo obszernych zasobów danch. Modele utworzone za pomocą algorytmów uczenia maszynowego są w stanie z wysokim prawdopodobieństwem wystawić predyckję lub dokonać klasyfikacji na temat zadanego problemu.

Model *klasyfikacjny* służy do przewidzenia etykiety klasy poprzez mapowanie na już z góry ustalony jednowymiarowy pozdział, model *regresywny* natomiast mapuje przestrzeń ustalając liczbę klas podziału oraz grupując wartości. [6] Istnieje możliwość przekształcenia problemu regresywnego na klasyfikacyje i na odwrót poprzez zamiane wartości oczekiwanego wyniku. Taką modyfikację zastosowano w praktycznej częsci projektu. Wyniki dla danych występowały w wartościach od 0 do 4 , dla wartości <1,4> przypadek testowy uznawany był za sklasyfikowany pozytywny (chory), dlatego przekształcenie z modelu regresywnego do modelu klasyfikacyjnego polega na konwersji wyników do wartości liczbowych 0 - brak stwierdzenia stanu choroboweo oraz 1 - stwierdzenie o chorobie układu krążenia.

Sposób wykorzystania segreguje alorytmy uczenia maszynowego na dwie kategorie, jednak powszechnie stosowanym podziałem jest podział zależnie od sposobu *trenowania* algorytmu. Algorytmy dzieli się na min.: uczenie nadzorowane, uczenie częściowo nadzorowane, uczenie bez nadzoru oraz uczenie przez wzmacnianie [7] .



Dobór typu uczenia oraz algorytmu uzależniony jest od danych wejściowych oraz oczekiwanego rezultatu. Dane

wyjściowe mogą przyjmować format odpowiedzi TAK/NIE , klasyfikacji do danego zbioru czy np procentowej oceny ryzyka.

Uczenie maszynowe nadzorowane (ang. *supervised learning*) to klasa algorytmów uczenia maszynowego, która bazuje na poetykietowanych danych. Nadzór polega na porównaniu rezultatów działania modelu z wynikami które są zawarte w danych wejściowych (*dane oznaczone*) [8]. Algorytm po osiągnięciu żądanej efektywnosci jest w stanie dokonać klasyfikacji przykładu dla którego nie posiada odpowiedzi. Sprawdza się to obecnie w rekomendacji produktów oraz diagnozie chorób. Z matematycznego puntu widzenia dopasowanie danych oznaczonych nazywane jest aproksymacją funkcji [7] .

Uczenie maszynowe bez nadzoru (ang. *unsupervised learning*) to klasa algroytmów uczenia maszynowego która wiodąco rozwiązuje problemy grupowania. Dane dostarczane do modelu nie zawierają *oznaczeń*, zatem nauczanie polega na wyciąganiu konkluzji z poprzednio wykonanych iteracji. Na skuteczność modeli budownych w oparciu o uczenie bez nadzoru wpływ ma rozmiar dostarczonego do nauki zbioru danch, im jest on większy tym bardziej wzrasta effektywność. Takie zbiory można uzyskać rejestrując dane na bierząco dlatego do najczęstrzych zastosowań tej klasy algorytmów, można zaliczyć rozpoznawanie mowy czy obrazu [7] .

Uczenie maszynowe przez wzmacnianie (ang. *reinforcement learning*) to klasa algorytmów uczenia maszynowego której nauczanie nie opiera się na danych wejściowych czy wyjściowych a rezultatach otrzymanych podczas testu nazywanych tzw. sygnałami wzmocnienia który może przyjmować wartość pozytwną lub negatywną. Algorytm generując dane wejściowe dostosowuje reguły by uzyskać zwrotnie sygnał pozytywny w jak największej liczbie przypadków. [9] .

Uczenie częściowo nadzorowane (ang. *semi-supervised learning*) to klasa algorytmów uczenia maszynowego która wykorzystuje zbór danych w większości niepoetykietowany na podstawie których tworzony jest model [10] , w wykorzystywana głównie w przypadkach niewydajności zastosowania osobno modeli nadzorowanych i nienadzorowanych. Zastosowanie tej klasy algorytmów pozwala również na maksymalizację wykorzystania zebranych informacji [2] .

Uczenie nadzorowane przedstawiając oficjalną matematyczą definicje :

$$DL = ((xi, yi))\,l$$
$$gdzie : i=1$$

(xi, yi) to punkt z zakresu xi X oznaczony etykietą yi dla danych wejściowych oznaczonych jako X.

Zbiór (xi,yi) to tgzw. dane uczące na podstawie których metody uczenia próbują wywnioskować funkcję która ustali y dla nieoznakowanego x. Większy zasób punktów sprawdzający działanie ,czyli dane testowe definiujemy następująco :

$$DU = (xi)\,l+u$$
$$gdzie : i=l+1$$

[11]

Klasyfikacja a Regresja

Oba przedstawione poniżej typy sysytematyzją w oparciu o dostarczone dane wejściowe i mają one wspólną część polegającą na budowaniu modelu separującego kategorie docelowe w użyteczny i dokładny sposób.[2]

Pod ogólnym pojęciem nadzorowanego uczenia się z przykładów rozróżnić można przewidywanie wartości i przewidywanie kategorii. Czy próbujemy (1) skorelować dane wejściowe do jednej z możliwych kategorii, określonej przez dyskretne symbole, czy (2) skorelować dane wejściowe do mniej lub bardziej ciągłego zakresu wartości numerycznych? W skrócie, czy cel jest kategoryczny, czy numeryczny? Jak już wspomniałem, predykcja kategorii nazywa się klasyfikacją. Predykcja wartości liczbowej nazywana jest regresją. Przyjrzymy się przykładom każdej z nich.

Klasyfikacja - decyduje o przynależności do zbioru grupy lub klasy. *Regresja* - daję ciągłą prognoze korelacji między zmiennymi, standardowym przykładem zastosowania jest prognoza pogody. Realne pomiary temperatury, prędkości wiatru , ciśnienia wpływają na finalną odpowiedź. Sama regresja dzieli się równiez na kategorie ze względu na skomplikowania, najprostrzym przykładem jest oczywiście regresja liniowa.

Analogiczne typy istanieją dla uczenia bez nadzoru jak na przykład *grupowanie*, które klasyfikuje dane w zbiory. Rozbieżność z klasyfikacją polaga na wykorzystaniu do wykonania oceny korelacji podobnych cech, a nie wsad danych testowych.

Redukcja wymiarowa to jak nazwa wskazuje pozbycie się nieistotnych atrybutów i odrzuceniu duplikatów, a co za tym idzie wymiaru data set'u. Dobrym przykładem byłoby tutaj analizy zawartości skrzynki pocztowej i szukanie spamu.

Podział osób na kategorie cierpiące na choroby sercowo-naczyniwe oraz zdrowe, to dylemat klasyfikayjny nadający się do rozwiązania za pomoca algorytmów uczenia maszynowego nadzorowanego i na nich skupia się dalsza część pracy.

Ścieżka działania algorytmów uczenia maszynowego nadzorowanego

Bazowy schemat budowania i oceniania modeli uczenia nadzorowanego to kolejno:

1. Przygotowanie danych.
2. Implementacja modelu.
3. Trening oraz ocena precykcji.

Wykorzystanie utworzonego modelu wymaga:

1. Zbudowania modelu oraz jego wytrenowania
2. Wykonania predykcji na danych które nie posiadają oznaczenia[12]

Ten schemat można zastosować do dowolnego modelu uczenia, wykonanie kolejnych bloków zadaniowych różnić się będzie specyfiką dla danego modelu:

- przygotowanie danych dla algorytmów uczenia nadzorowanego musi zawierać również zebranie odpowiedzi/wyników dla danych testowych
- implementacja modelu za każdym razem jest specyficzna dla zastosowanego algorytmu który równiez zależy od typu uczenia
- ewaluacja modeli regresywnych różni się od ewluacji modeli klasyfikacyjnych ze względu na wykorzystanie innych miar oceny dokładności
- zastosowanie modeli może posiadać wielorakie formy realizacji.

Dane

Terminologia

Analiza uczenia maszynowego wymusza stosowanie rozróżnienia przy pojęciach parametru i argumentu. Cechy dla których szukamy optymalnych wartości , które stanowią podstawe modelu i ich dostrajanie wykonywane jest podczas treningu nazywane są parametrami i hiperparametrami. Argumenty natomiast to liczby znajdujące się w wierszach zbioru danych które podlgają zmianie tylko podczas pre-procesingu Nazewnictwo hiperparametrów wykorzystywane jest w przypadku zastosoawnia dla nich walidacji krzyżowej.

[13]

Repozytorium uczenia maszynowego UCI

Sensem wykorzystania uczenia maszynowego jest prognoza lub klasyfikacja rzeczywistych wartości z dużego zbioru danych które mogą znaleść zastosowanie w praktycznych dziedzinach. Im bardziej dokładne i rzeczywiste dane do testowania i tworzenia modelu tym większe prawdopodobieństwo otrzymania realnych wyników na końcu ścieżki

uczenia. W celu gromadzenia miarodajnej bazy dostępnych zbiorów danych testowych powstało repozytorium uczenia maszynowego UCI. Jak podaje strona informacyjna :

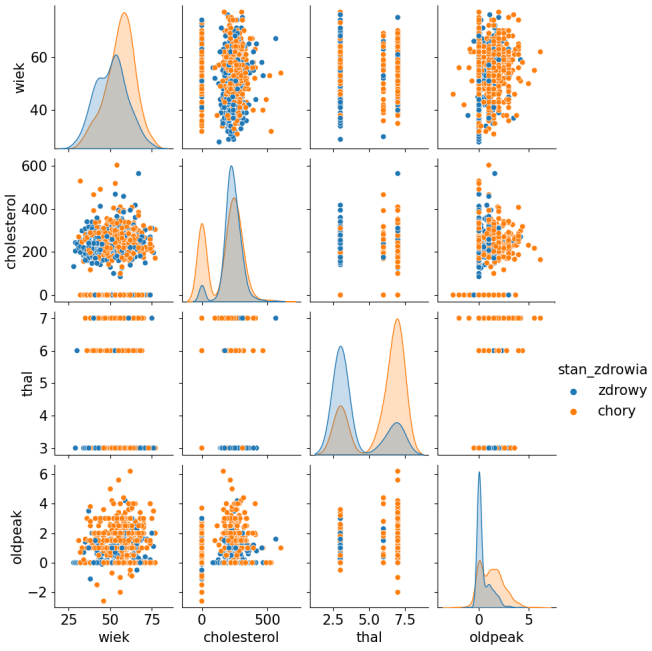
> … było ono cytowane ponad 1000 razy, co czyni je jednym ze 100 najczęściej cytowanych „artykułów" w całej informatyce … [13]

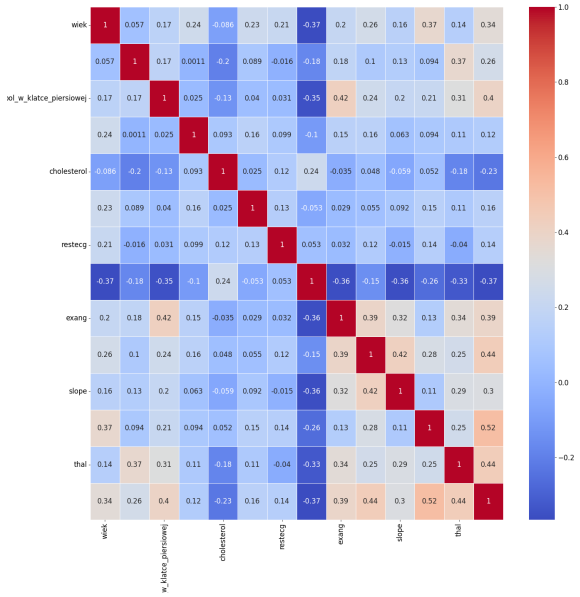
Repozytorium gromadzi dane z wielu rozbieżnych dziedzin , dane medyczne umieszczone w repozytorium nie zawierają wrażliwych danych pacjentów , a niektóre zbiory są poddane już wstępnej obróbce tak jak zbiór danych "Heart Disease Databases" wykorzystany w tym dokumencie, który powstał na podstawie realnych danych medycznych zebrany z lokalizacji

1. Fundacja Cleveland Clinic [14]
2. Węgierski Instytut Kardiologii, Budapeszt [15]
3. V.A. Centrum medyczne, Long Beach, Kalifornia [14]
4. Szpital Uniwersytecki, Zurych, Szwajcaria [16].

Stratyfikacja

Wyróżniono 14 atrybutów spośród 76 zebrancyh do wykorzystania w algorytmach uczenia maszynowego, wszystkie z nich mają wartośi liczbowe.





Rozkład chorób serca w danych testowych to 44.67% chorych czyli 509 prób pozytywnych oraz 411 negatywnych. W danych testowych znajduje się 726 przypadków osób płci męskiej oraz 194 żeńskiej. Dla zachorowań widać nierówność ale jest ona spowodowana rzeczywistą statystyką. Tylko u 25.77% badanych kobiet stwierdzono występowanie chorób wieńcowych, natomiast wśród badanych mężczyzn jest to aż 63.22%. [13]

Lista atrybutów wykorzystanych w algorytmie:

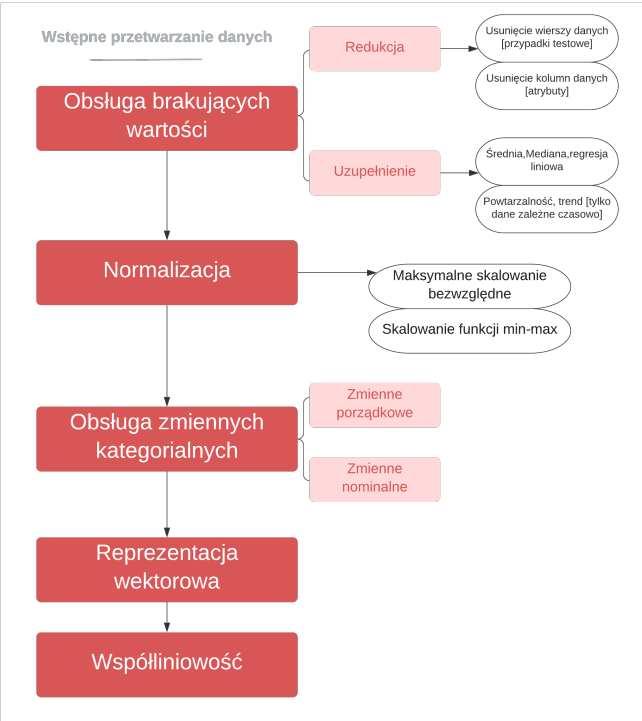
- wiek
- płeć
- rodzaj bólu w klatce piersiowej
- spoczynkowe ciśnienie krwi
- cholesterol w surowicy w mg/dl
- poziom cukru we krwi na czczo > 120 mg/dl
- spoczynkowe wyniki elektrokardiograficzne
- osiągnięto maksymalne tętno
- dławica piersiowa wywołana wysiłkiem fizycznym
- obniżenie odcinka ST wywołane wysiłkiem fizycznym w stosunku do odpoczynku

- nachylenie szczytowego odcinka ST ćwiczenia
- liczba głównych naczyń pokolorowanych fluorozopią
- skan serca z talem lub test wysiłkowy
- stan (brak choroby serca/choroba serca)

W przypadku danych testowych z repozytorium UCI, fakt iż dane pochodziły z różnych lokalizacji ma duże znaczenie ,gdyż od placówki medycznej zależy jakim badaniom poddani zostali pacjęci a co za tym idzie w jakich kolumnach tabelarycznego przedstawienia będą mieć uzupełnione bądź puste wartości. Scalenie ze sobą wyników badań dostarcza większej różnorodności również dzięki temu że dane pochodzą z wielu krajów. Jeżeli zestaw wejściowy zostałby ograniczony do jednej lokalizacji to cecha dla której nie uzupełniono wartości zostałaby pominięta podczas treningu ze względu na brak danych, co skutowało by uboższym modelem i możliwe że pominięciem kluczowej cechy wpływającej na działanie.

Wstępna obróbka danych

Proces przetwarzania danych może skladać sie z wielu różnych kroków zależenie od typu, w uczeniu nadzorowanym operującym na danych tekstowo-liczbowych poprawnym będzie zastosowanie schematu przedstawionego poniżej:



Po złączeniu można przeprowadzić szereg działań w celu sztucznego uzupełnienia pustych wartości bazując na wartościach które już istnieją.

Obsługa brakujących wartości

Możliwościami obsługi brakujących wartości są : mniej polecana ze względu na utrate danych, redukcja zestawu danych lub uzupełnienie go zgodnie z wybrany przez siebie założeniem. Biblioteki do nauczania maszynowego dostarczają już gotowe rozwiązania do upuszczenie wierszy lub kolumn zawierających wartości *null*. Uzupełnienie danych inaczej *imputacja*, rozwiązuje problem w mniej stratny sposób i tak samo jak do redukcji są już gotowe rozwiązania w bibliotece sklearn. Istnieją 4 różne strategie uzupełniania wykorzystujące proste matematyczne obliczenia takie jak :

- średnia,
- mediana,
- stała,
- najczęściej występująca wartość.

Do wyznaczenia wartości uzupełniających można również użyć regresji liniowej.

Standaryzacja

Przekształcenie danych również bazujące na statystycznych założeniach i również ustandaryzowane w popularnych bibliotekach. Dążymy aby średnia wartoś wynosiła 0, a odchylenie standardowe 1 dla liczbowyh reprezentacji danych. Z matematyczne punktu widzenia wykonujemy działanie

$$\frac{\bar{X}}{\sqrt{\frac{\sum_{i=1}^{n}(X-\bar{X})^2}{N-1}}}$$

[17]

Obsługa zmiennych kategorialnych

Cechy kategorialne dzielą się na dwie zasadnicze grupy ze względu na mozliwość uprorządkowania , dane takie jak wyksztaucenie , rozmiar podlegają mapowaniu , dane typu kolor lub płeć podlegaja kodowaniu. W ten sposób dane kategoryczne staja się wartosciami liczbowymi.

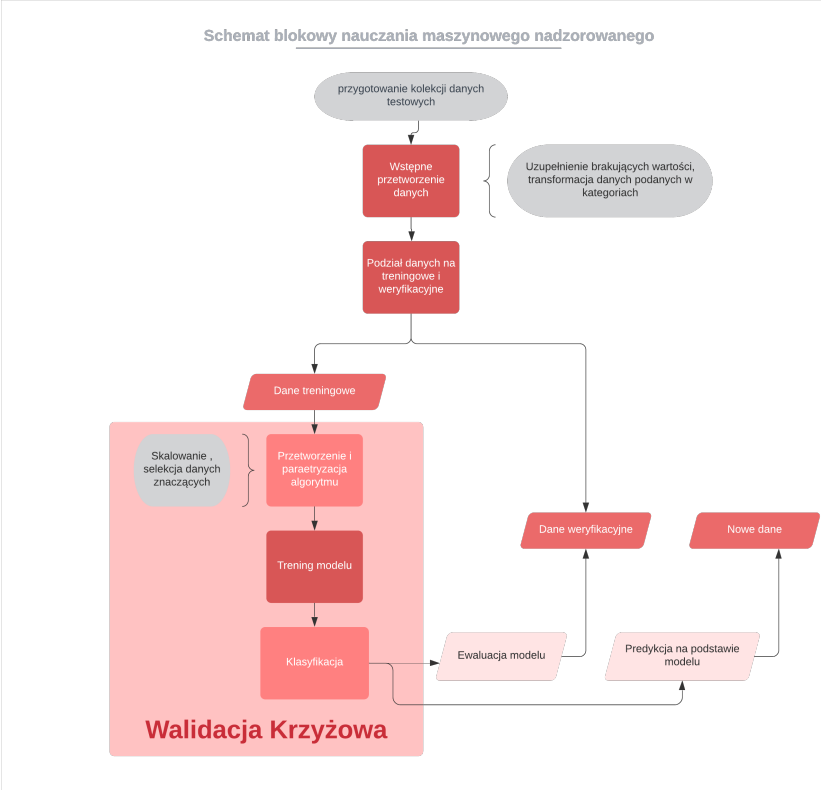
Reporezentacja wektorowa

Obsługa danych kategorialnych pozwoliła zmapować/zakodować je w postaci liczbowej, ale można pójść o krok dalej i te same dane mieć w postaci 0 lub 1 na odpowiedniej kolumnie. Rozwiązanie reprezentacji wektorowej polega na utworzeniu tylu kolumn ile jest unikalnych wartości dla kategorii i wpisanie 0 lub 1 dla każdego rekordu danych [18] .

Współliniowość cech

Aby znaleść korelacje współliniowości należy szukać liniowej zależności pomiędzy danymi, najłatwiej zauważyć to tworząc wykresy z danych testowych dla każdej pary [18].

Zgodnie z poniższym schematem po przeprocesowaniu wejściowego zbioru danych, należy go podzielić na dane treingowe oraz ewaluacyjne. Powszechnie stosowana K krzyżowa walidacja umożliwia maksymalne wykorzystanie dostarczonego wejścia do dostrajania parametrów modelu, ponieważ optymalizacja hiperparametrów połączone z ciągłą weryfikacją poprawności to sedno treningu.



K-krotna walidacja krzyżowa (ang. *K-fold Cross Validation*, KCV) - metoda weryfikacji działająca poprzez podział zbioru danych na k podzbiorów z których każdy przynajmniej raz jest zbiorem oceniającym wydajność , zaznaczając że K musi być równe lub mniejsze niż liczba elementów w zbiorze [19], [20] .

Kluczowym elementem jest ewaluacja która odbywa się na końcu każdej z k-1 iteracji w celu dostosowania parametrów, po osiągnięciu wymaganych lub ustalonych wartości dokładności modelu lub weryfikacji wszystkich możliwych opcji i znalezienie najlepszego modelu można go wykorzystać do weryfikacji na danych spoza zestawu testowego.

Wybrane algorytmy uczenia maszynowego nadzorowanego

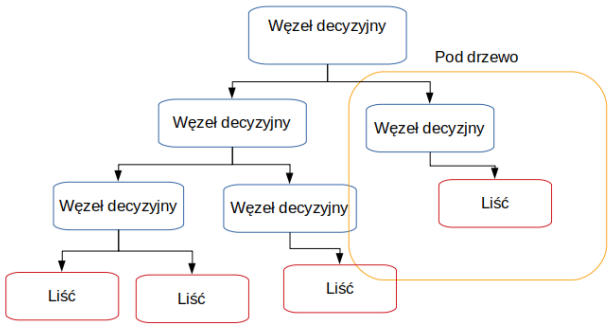
Wstęp

Nadzorowana klasyfikacja obrazów wymaga uczącego zestawu danych składającego się z pikseli o różnych sygnaturach widmowych oznaczonych w ramach jednej klasy lub więcej w przypadku klasyfikatory rozmyte. Etykietowanie może pochodzić z wizualnej interpretacji obrazów do sklasyfikowania lub z innych referencyjnych zbiorów danych pochodzących z bardziej czułych czujników lub zaobserwowane/zmierzone in situ. Wizualna interpretacja obrazu jest zwykle wykonywana przez: oprogramowanie systemu informacji geograficznej (GIS) i obejmuje ręczne wyznaczanie i znakowanie wielokątów pokrywających obszary jednorodne. Korzystanie z innego pilota referencyjnego wykrywanie zbiorów danych często wymaga kolokacji przestrzenno-czasowej między dwoma czujnikami za pomocą środków interpolacji lub dopasowania najbliższego sąsiada. Należy zastosować oba podejścia do etykietowania wspólne zasady tworzenia optymalnego zestawu danych treningowych: 1. Referencyjny zbiór danych powinien być wystarczająco duży, aby wypełnić przestrzeń funkcji informacjami wystarczającymi do prawidłowego wywnioskowania etykiet dla nowych przypadków testowych, które się nie pojawiły w uczącym zbiorze danych. Oznacza to, że nowe próbki powinny być bardzo podobne do te używane do treningu klasyfikatorów. Taka sytuacja zdarza się rzadko dla prostych zadania klasyfikacyjne oparte na kilku cechach. Wieloczasowy i wielospektralny obrazy teledetekcyjne składające się z heterogenicznych pikseli o mieszanych klasach wymagają ogromnego zestawu danych treningowych, aby przedstawić wszystkie potencjalne relacje między funkcjami. Na treningowe zbiory danych zwykle mają wpływ problemy z próbkowaniem związane z nierównomiernym rozkładem obserwacji in situ ze względu na ograniczoną dostępność (np. odległe obszary, bagna, wysokie góry), ograniczenia czasowe i przestrzenne, podczas gdy kolokacji różnych źródeł danych i subiektywnej selekcji poligonów uczących. W związku z tym treningowy zestaw danych

może być rzadki i niedostatecznie reprezentowany. W związku z tym, solidny klasyfikator obrazu powinien być w stanie uogólnić dostępne informacje a priori do niewidzianych przypadków bez nadmiernego dopasowania zestawu danych uczących. Overfitting oznacza, że Remote Sens. 2022, 14, 378 4 z 36 klasyfikator zbyt dobrze pasuje do zestawu danych treningowych (w tym hałaśliwe przypadki zamiast tylko „prawdziwy" sygnał). Wręcz przeciwnie, niedopasowanie oznacza, że klasyfikator wykonuje: słabo na danych treningowych, prawdopodobnie z powodu prymitywnych założeń lub błędnie wybrane hiperparametry lub cechy klasyfikacyjne. 2. Zbiór danych uczących powinien charakteryzować się niską autokorelacją przestrzenną i czasową, aby maksymalizować informacje w przestrzeni funkcji i obiektywnie oceniać jakość wygenerowanej klasyfikacji. Jeśli piksele są silnie skorelowane, to ich widmowe podpisy są w większości zbędne. W konsekwencji klasyfikator ma niewiele informacji, aby: uczyć się na podstawie nowych przypadków testowych i uogólniać je. Taka sytuacja często ma miejsce w przypadek wizualnej interpretacji obrazu, gdzie operator zaznacza kilka dużych wielokątów obejmujących tę samą klasę. Mimo zaznaczenia dużej liczby pikseli, bardzo ograniczone informacje spektralne są w rzeczywistości wykorzystywane do trenowania klasyfikatora. Co więcej, jeśli a podzbiór tych pikseli jest używany do oceny jakości klasyfikacji, a następnie statystyki pochodne są zawyżone i niewiarygodne. Tak więc, jak sugeruje [36], najbardziej solidnym rozwiązaniem tego problemu jest losowy dobór trenujących pikseli i zbiory danych walidacyjnych. Ponadto korzystne jest tymczasowe odłączenie obu zestawy danych, dzięki czemu treningowy zestaw danych jest pozyskiwany w innym czasie (np. inny rok) niż zbiór danych walidacyjnych. 3. Referencyjny zbiór danych powinien odzwierciedlać taką samą proporcję klas, jak ta występująca na obrazie niejawnym. Jest to szczególnie ważne w przypadku klasyfikatorów bayesowskich, które polegają o prawdopodobieństwie wystąpienia klasy a priori [37]. Donoszono o podobnych wnioskach przez [36] w przypadku algorytmów drzew decyzyjnych. 4. Dane referencyjne powinny być wysokiej jakości, z jak najmniejszą liczbą błędnie oznakowanych pikseli. To oczywiste stwierdzenie jest trudne do zrealizowania w rzeczywistości, gdzie dane referencyjne mogą pochodzić z: (1) innych produktów klasyfikacyjnych charakteryzujących się własną niepewnością; (2) obserwacje in situ zmienione przez niesprawność przyrządów lub błędy typograficzne; (3) dane crowdsourcingowe zawierające wiele subiektywnych obserwacji; oraz (4) słabo wytyczone wielokąty uczące. Ponadto w przypadku krycia obrazów o niskiej rozdzielczości niejednorodny krajobraz, piksele składają się zwykle z mieszanych klas, które ostatecznie muszą zostać zaklasyfikowane do jednej klasy. Alternatywnie, niektóre subpiksele techniki klasyfikacji [38–40] można zastosować do wywnioskowania ułamkowej proporcji klasy w takich pikselach. Niemniej jednak, z perspektywy klasyfikatora obrazu, błędnie oznakowane piksele wprowadzają szum, który należy oddzielić od prawdziwego sygnatura widmowa powiązana z określoną klasą. Ograniczona liczba badań ma omówiono wrażliwość klasyfikatorów obrazu na błędnie oznakowane dane wejściowe. W [34], okazuje się, że algorytm losowego lasu jest mniej czuły[21]

Losowe lasy decyzyjne

Drzewa decyzyjne (ang. *decisions trees*) są uznawane za najprostyszy i najbliższy ludzkiemu zrozumieniu algorytm uczenia, który swoją nazwę zawdzięcza graficznej reprezentacji w postaci drzewa. Każdy węzeł oznacza atrybut, na podstawie którego następuje rozróżnienie. W modelu kluczowa jest kolejnośc cech, które występują po sobie ponieważ determinuje to otrzymany rezultat [7], [22] .



Prawie każdy algorytm uczenia maszynowego nadzorowanego można podzielić na dwa etapy. W pierwszym opracowywany jest wzorzec, na którym bazuję późniejsza predykcja. Etap nauki dla drzewa decyzyjnego polega na typowaniu atrybutów, które staja się węzłami decyzyjnymi, dzielącymi rekordy na dwa mniejsze zestawy i tak aż nie ma możliwości dalszego podziału.

Rozpoczęcie od otwartego drzewa bez ograniczenia głębokości daje bardzo elastyczny model, pozwalający uchwycić

dowolny skończony wzór, który nie jest losowy na poziomie rzutów monetą. Choć może się to wydawać idealnym rozwiązaniem, wiesz już wystarczająco dużo, aby stwierdzić, że muszą tu występować kompromisy. Użycie drzew nieograniczonych zapewne doprowadzi do nadmiernego dopasowania, dlatego przy próbie uogólnienia rozwiązania wyniki dla danych testowych będą kiepskie. Jeśli chcesz przypomnieć sobie ten kompromis, zajrzyj do podrozdziału 5.6. Jak więc dodać obciążenie do drzew i dodać ograniczenia, aby zapobiec nadmiernemu dopasowaniu? Można tu wykonać kilka kroków. 1. Można ograniczyć głębokość drzew. Wtedy dozwolonych jest mniej pytań przed określeniem klasy przykładu. 2. Można wymagać większej liczby przykładów w liściach. To ograniczenie wymaga grupowania przykładów, które mogą różnić się między sobą. Ponieważ ich rozdzielanie jest niedozwolone, skutkuje to wygładzeniem niektórych granic. 3. Można ograniczyć liczbę uwzględnianych cech przy zadawaniu pytań o przykłady. To ograniczenie ma dodatkową korzyść w postaci przyspieszenia procesu uczen 4. [12]

Na metodologie drzew decyzyjnych oparta jest dokładniejsza forma nauczania nadzorowanego: *losowe lasy decyzyjne.*

Losowe lasy decyzyjne (ang. *random decision forests*) to technika polegająca na połączeniu wielu drzew decyzyjnych w celu uniknięcia problemu z *nadmiernym dopasowaniem* do treningowego zestawu danych na którym został przeszkolony.

Utworzony szablon aby poprawnie działać na danych testowych i służacych weryfikacji, nie może stać się charakterystycznym przypadkiem rozwiązującym przypadek testowy [7], [22]. W tym celu dla loswoych lasów decyzyjnych najpierw stosuję się **agregację bootstrap'ową**. Z treningowego zestawu danch losuję się, z możliwymi powtorzeniami, wiersze danych dla których trenowany będzie model. Jako rezutat brana jest większość lub średnia wartości uzyskanych wyników dla poszczególnych drzew dezycyjnch. Dodatkowo dla drzew decyzyjnych w lasach losowych, atrybuty odpowiadające za kategoryzację są wybierane z wylosowanego podzbioru.[23]

Przejdźmy teraz to bardziej skomplikowanych sposobów łączenia modeli. Pierwszy z nich to lasy losowe. Metoda ta jest oparta na technice o nazwie bagging. Nazwa ta nie pochodzi od angielskiego bag, czyli torba. Jest to sztuczne słowo, zbitek dwóch wyrazów bootstrap aggregation (agregacja bootstrap). Agregacja oznacza łączenie, nadal jednak trzeba wyjaśnić, czym jest bootstrap. W tym celu opiszę, jak obliczyć prostą statystykę (średnią) za pomocą techniki bootstrap. 12.3.1. Technika bootstrap Podstawowe zadanie techniki bootstrap polega na znalezieniu rozkładu (przedziału możliwości) na podstawie jednego zbioru danych. Jeśli myślisz teraz o sprawdzianie krzyżowym, gdzie też jeden zbiór danych jest dzielony w celu uzyskania wielu wyników, 4c2836bbe011706b10838f45bf5d766f 12.3. Bagging i lasy losowe 405 jesteś blisko prawdy. Obie te metody to techniki wielokrotnego pobierania próbek. Więcej o powiązaniach między tymi metodami dowiesz się za chwilę. Wróćmy teraz do techniki bootstrap. Wyobraź sobie, że masz zbiór danych i chcesz obliczyć na jego podstawie statystykę. Dla wygody niech będzie to średnia. Oczywiście istnieje prosty, bezpośredni wzór na średnią. Wystarczy zsumować wartości i podzielić wynik przez ich liczbę. Matematycznie zapis wygląda tak: n x średnia(X) . Po co bawić się w bardziej skomplikowane rozwiązania? Ponieważ średnia oszacowana za pomocą jednego punktu, jednej obliczonej wartości, może być myląca. Nie wiadomo, jak zmienne są szacunki średniej. W wielu sytuacjach chcemy wiedzieć, o ile możemy się mylić. Dlatego zamiast zadowalać się jedną wartością, możemy zażądać rozkładu średnich, jakie można obliczyć na podstawie danych podobnych do analizowanego zbioru. Istnieją teoretyczne techniki opisu rozkładu średnich. Znajdziesz je w każdym podręczniku z wprowadzeniem do statystyki. Jednak w przypadku innych statystyk, także czegoś tak wyrafinowanego jak wyuczone klasyfikatory, nie istnieją wygodne, gotowe odpowiedzi obliczane za pomocą prostych wzorów. Zamiast nich można zastosować wtedy technikę bootstrap. Aby ustalić, jak obliczyć wartość za pomocą tej techniki, przyjrzyj się kodowi i rysunkom obrazującym ten proces. Pobieranie próbek ze zwracaniem i bez zwracania. Przed omówieniem procesu bootstrap trzeba objaśnić jeszcze jedną kwestię. Istnieją dwa sposoby pobierania próbek wartości z kolekcji danych. Pierwsze podejście to pobieranie próbek bez zwracania. Wyobraź sobie, że umieszczasz wszystkie dane w worku i chcesz pobrać ze zbioru danych jedną próbkę z pięcioma przykładami. Możesz włożyć rękę do worka, wyciągnąć przykład, zapisać go i odłożyć na bok. W tej technice — bez zwracania — przykład nie wraca do worka. Następnie należy wziąć z worka drugi przykład, zapisać go i odłożyć. Także tego przykładu nie należy wrzucać z powrotem do worka. Proces wyciągania należy powtarzać do momentu zapisania pięciu przykładów. Jest tu oczywiste, że to, który przykład zostanie wyciągnięty za pierwszym razem, ma wpływ na drugą, trzecią i kolejne operacje pobierania. Jeśli dany przykład został zabrany z worka, nie można go ponownie wyciągnąć. Pobieranie próbek bez zwracania sprawia, że wybory są zależne od tego, co stało się wcześniej. Przykłady nie są niezależne od siebie. To z kolei wpływa na to, jak statystycy traktują daną próbkę. Inne podejście to pobieranie próbek ze zwracaniem. Należy wyciągnąć przykład z worka ze zbiorem danych, zapisać wartość i zwrócić przykład do worka. Następnie pobieranie przykładów jest powtarzane do momentu uzyskania potrzebnej ich liczby. Pobieranie próbek ze zwracaniem (zobacz rysunek 12.3) daje sekwencję niezależnych przykładów. Jak więc zastosować technikę boot-

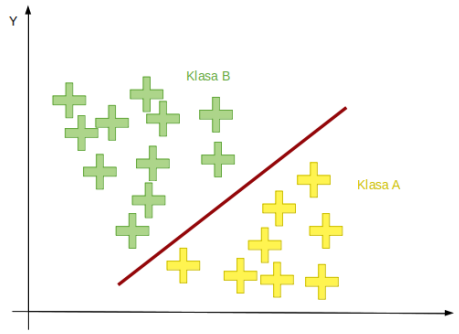
strap do obliczenia statystyki dla próbki bootstrap? Należy losowo pobierać przykłady ze zwracaniem ze źródłowego zbioru danych do momentu uzyskania nowej próbki o tej samej liczbie elementów co pierwotny zbiór. Następnie można obliczyć potrzebną statystykę dla próbki bootstrap. Proces ten jest powtarzany kilkakrotnie, po czym należy obliczyć średnią ze statystyk dla poszczególnych próbek. Na rysunku 12.4 zobaczysz wizualnie, jak wygląda proces obliczania średniej z użyciem techniki bootstrap.[12]

Lasy losowe (LL) to specjalny typ mechanizmów uczących się opartych na baggingu. Wykorzystuje się tu specyficzną wersję drzew decyzyjnych. Standardowe drzewa decyzyjne do utworzenia węzła wykorzystują tę spośród wszystkich cech, która daje najlepsze wyniki. W LL wybierany jest podzbiór cech, po czym podejmowane są decyzje optymalne na podstawie tego podzbioru. Celem jest wymuszenie zróżnicowania drzew. Możliwe jest, że nawet przy losowym doborze przykładów jedna cecha będzie ściśle powiązana z wartościami docelowymi. Taka najważniejsza cecha zapewne zostanie wybrana jako pierwszy punkt podziału we wszystkich drzewach. Pora zakończyć hegemonię takich cech. Gdy stosowana jest technika bootstrap, pierwszy punkt podziału w drzewach zwykle nie jest oparty na jednej cesze. Zamiast tego selektywnie ignorowane są niektóre cechy i wprowadzana jest losowość, aby uzyskać różne drzewa. Ta losowość powoduje, że w różnych drzewach uwzględniane są różne cechy. Pierwotny algorytm drzew losowych wykorzystywał głosowanie większościowe podobne jak w baggingu. Jednak algorytm LL 4c2836bbe011706b10838f45bf5d766f 12.3. Bagging i lasy losowe 411 w bibliotece sklearn oblicza prawdopodobieństwo wyboru danej klasy przez każde drzewo z lasu, a następnie uśrednia wyniki, aby uzyskać ostateczną odpowiedź. Klasa o najwyższym średnim prawdopodobieństwie jest „zwycięzcą". Jeśli zaczniesz od kodu przedstawionej wcześniej funkcji bagged_learner, musisz jeszcze zmodyfikować kod tworzenia drzewa (etap budowania modelu składowego) z podrozdziału 8.2. Jest to prosta modyfikacja, którą wyróżniłem kursywą: 0. Losowy wybór podzbioru cech uwzględnianych w danym drzewie. 1. Analiza wybranych cech i podziałów oraz wybór najlepszej kombinacji cecha-podział. 2. Dodanie do drzewa węzła reprezentującego tę kombinację cecha-podział. 3. W każdym podwęźle należy uwzględnić powiązane z nim dane i: jeśli wartości docelowe są wystarczająco podobne, zwrócić prognozowaną wartość; w przeciwnym razie wrócić do kroku 1. i powtórzyć proces. Używając tego pseudokodu jako bazowego estymatora, można w połączeniu z kodem funkcji bagged_learner szybko zbudować prototyp systemu uczącego się wykorzystującego las losowy. Ekstremalne lasy losowe i podział cech. Następna wersja to ekstremalne drzewa losowe. Nie, nie chodzi tu o igrzyska sportów ekstremalnych dla drzew losowych. Słowo ekstremalne dotyczy tu dodatkowego poziomu losowości w procesie tworzenia modelu. Przedstaw informatykowi jakąś ideę (zastąpienie deterministycznej techniki losowością), a zacznie ją stosować wszędzie. W ekstremalnych lasach losowych w procesie budowania składowych drzew wprowadzana jest następująca zmiana: 1. Losowe generowanie punktów podziału i wybór najlepszego z nich. Tak więc obok losowego wyboru uwzględnianych cech losowo ustalane są też punkty podziału. Zaskakujące jest, że ta technika się sprawdza — a może działać naprawdę dobrze. Do tej pory nie opisywałem szczegółowo procesu wyboru punktów podziału. Nie muszę go tu omawiać, ponieważ zrobili to inni autorzy. Ja cenię przede wszystkim podejście stosowane przez Fostera i Provosta w ich książce, którą podaję w końcowej części rozdziału. Proces wyboru dobrych punktów podziału przedstawię na przykładzie. Załóżmy, że chcę na podstawie prostych wskaźników biometrycznych przewidzieć, kto będzie dobrym koszykarzem. Wzrost — niestety dla osób takich jak ja — jest naprawdę dobrym prognostykiem sukcesów w koszykówce. Dlatego można zacząć od analizy tego, jak wzrost jest powiązany ze zbiorem danych opisującym dobrych i — tak jak w moim przypadku — słabych koszykarzy. Po uporządkowaniu danych według wzrostu może się okazać, że w koszykówkę próbowali grać ludzie o wzroście od 1,35 m do 2,10 m. Jeśli wprowadzę punkt podziału na poziomie 1,52 m, w zbiorze osób z niższym wzrostem zapewne znajdzie się wielu słabych koszykarzy. Jest to stosunkowo jednorodna grupa, jeśli chodzi o sukcesy w tym sporcie. Jednak zbiór osób wyższych niż 1,52 m prawdopodobnie będzie obejmował obie klasy. Występuje więc niska czystość (ang. purity) powyżej punktu podziału i wysoka czystość poniżej punktu podziału. Podobnie jeśli jako punkt podziału 4c2836bbe011706b10838f45bf5d766f 412 Rozdział 12. Łączenie mechanizmów uczących się wybiorę wartość 1,95 m, wśród wyższych osób większość zapewne będzie odnosić sukcesy, jednak wśród niższych graczy wyniki będą mieszane. Celem jest znalezienie odpowiedniego punktu podziału dla wzrostu, aby uzyskać możliwie dużo informacji na temat sukcesów — zarówno wśród niższych, jak i wśród wyższych osób. Wydzielenie graczy o niskim i wysokim wzroście daje dużo informacji. W środkowym segmencie ważne są inne czynniki, na przykład ilość czasu poświęcanego na grę. Tak więc w ekstremalnych lasach losowych nie uwzględnia się wszystkich możliwych punktów podziału. Zamiast tego wybierany jest losowy podzbiór analizowanych punktów. Następnie te punkty są oceniane i z tego ograniczonego zbioru wybierane są najlepsze punkty.

Wśrod zalet lasów losowych nalezy wyróżnić iż potrafią one trafnie wykalkulować brakujace wartości cech. Idealnie znajdują zastosowanie dla realnych danych, których zasadniczym problemem jest ich niekompletność.
Dane medyczne posiadają szeroką wariację zmiennych z dużym prawdopodobieństwem wybrakownia, zastosowanie do nich lasów decyzyjnych ma potencjał na pozytywne rezulataty.

Maszyna wektorów nośnych

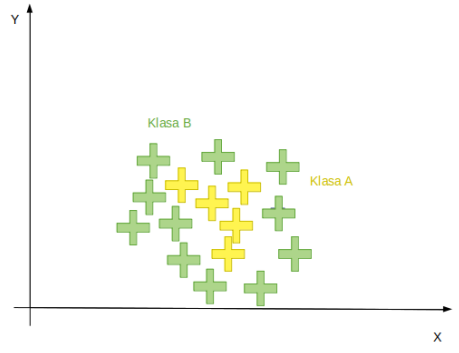
Metoda wektorów nośnych (ang. *support vector machines* , skr. ***SVM***) to algorytm uczenia maszynowego nadzorowanego, który każdy parametr z dostępnych cech dla danych wejściowych, traktuje jako punkt w przestrzeni. Na podstawie ułożenia punktów dzieli się je na 2 klasy. Graficznie jest to reprezentowane przez prostą dla której odległość między najbliższymi dwoma punktami dla wektorów jest możliwie największa.



[^schemat_wzorowany]

Taka prosta nazywana jest *prostą marginalną* i powstaje ona poprzez generowanie i selekcję tych prostych które rzetelnie szufladkują klasy danych [7], [22].

Techinka ta gwarantuje precyzyjniejsze rezulatay niż drzewa decyzyjne, niestety dla dużych zbiorów danych czas trwania szkolenia znacznie się wydłuża oraz istnieją przypadki dla których podział jedną prostą jest niewykonalny, taki przypadek reprezentuje rozkład na schemacie nr. 2.



[^schemat_wzorowany]

Z powyższego schematu widać że prosta marginalna ma zastosowanie w przypadku dwóch wymiarów, dla większej ilości stosowane jest przekształcenie do innego systemu współrzędnych i szukanie hiperpłaszczyzny brzegowej dzielącej tak samo jak prosta punkty w przestrzeni na dwa zbiory.[24]

Wyszukiwanie podziału

Idea działania maszyny wektorów nośnych opiera się na wyznaczenia minimalnej wartości wektora wag oraz przesunięcia (ang. *bias*) który geometrycznie opisuje współrzędne hiperpłaszczyzny.

$$\min_{w,b} \frac{1}{2}||w||^2$$

$$\text{pod warunkiem } y_i(w \cdot x_i + b) - 1 \geqslant 0, \ i = 1, \ldots, n.$$

[25]

Na szczęście sklearn zapewnia nam pewne wsparcie w tym względzie. Wykorzystamy narzędzie z biblioteki sklearn, aby uniknąć uczenia się do testu. Funkcja train_test_split podzieli zestaw danych znajdujący się w Pythonie pod zmienną iris. Pamiętaj, zestaw danych ma już dwie składowe: cechy i cel. Nasz nowy podział podzieli dane na dwa zbiory przykładów: 1. Część danych, które zostaną wykorzystane do zbadania i zbudowania zrozumienia. 2. Część danych, które wykorzystamy do sprawdzenia. Będziemy badać dane treningowe (czyli uczyć się na ich podstawie). Aby otrzymać miarodajną ocenę, do sprawdzenia wykorzystamy tylko dane testowe. Obiecujemy nie podglądać

danych testowych. Zaczęliśmy od rozbicia danych na dwie części: cechy i cel. Teraz każdą z nich rozbijemy na dwie grupy: 1. Cechy → cechy treningowe i cechy testowe. 2. Cele → cele treningowe i cele testowe. [12]

Można tu wygenerować nieskończoną liczbę linii, stopniowo rotując je od tej z górnym krańcem po lewej stronie do prawej kreski (z górnym krańcem po prawej stronie). Wszystkie te linie zapewniają idealny podział i doskonałe wyniki z użyciem różnych wskaźników (dla danych treningowych). Można też stopniowo przesuwać środkową linię nieco w lewo i nieco w prawo przy zachowaniu jej pionowego ustawienia — tak dużo linii, a tak mało czasu. Może istnieje inny standard (oprócz samej poprawności), który można wykorzystać do porównywania linii dzielących? 4c2836bbe011706b10838f45bf5d766f 270 Rozdział 8. Inne metody klasyfikacji Dla której linii można podać mocny argument na rzecz tego, że jest najlepsza? Włącz ścieżkę dźwiękową z filmu Karate Kid. Ja jestem wielkim zwolennikiem środkowej linii B. Dlaczego? Ponieważ jej odległość od danych jest największa. Pod niektórymi względami jest to „najbezpieczniejsza" linia — zawsze rozdziela przestrzeń między znakami X i O. Pusty obszar między klasami możesz traktować jak ziemię niczyją lub rzekę między dwoma państwami. W społeczności zainteresowanej uczeniem maszynowym ten pusty obszar ma specjalną nazwę. Jest to margines (ang. margin) między dwoma państwami, to znaczy klasami. Linia B też ma specjalną nazwę. Jest to separator o maksymalnym marginesie dzielący klasy, ponieważ obie klasy znajdują się od niego tak daleko, jak to możliwe. Oto powiązane zagadnienie. Po niewielkim zmodyfikowaniu problemu przez zachowanie tylko tych punktów, które znajdują się wzdłuż krawędzi obu klastrów, uzyskamy następujący Można stwierdzić, że nie utracono w ten sposób żadnych danych. Obrys klas (kadłub) pozostaje nietknięty. Dalej można tu redukować dane potrzebne do uzyskania linii podziału o maksymalnym marginesie. Nie jest tu potrzebna cała krawędź obu klas. Ważne są tylko punkty graniczne leżące naprzeciw punktów z innej klasy. Dlatego pozostałe punkty można usunąć. Pozostają wtedy dwie linie przeciwników stojących naprzeciw siebie przed grą Zadaniem klasyfikatorów SVC jest znalezienie kompromisu między dwoma konkurencyjnymi wymogami: uzyskaniem maksymalnego marginesu między klasami przykładów i zminimalizowaniem liczby błędów w czasie treningu. W pokazanych danych nie występują trudne przykłady treningowe — wszystkie punkty znajdują się po swoich stronach linii. Jednak niedługo zetkniesz się z bardziej skomplikowanymi scenariuszami. Oto dwie dodatkowe uwagi. Po pierwsze duże marginesy są pożądane, ponieważ — przy określonych założeniach — ułatwiają uogólnianie i zapewniają niewielki błąd dla zbioru danych testowych. Po drugie dwa czynniki związane są z potrzebą tworzenia dodatkowych wektorów nośnych: zwiększona złożoność granicy między klasami i przykłady, które nie wpasowują się w proponowane granice. Dotychczasowe omówienie jest koncepcyjnym wprowadzeniem do klasyfikatorów SVC i ich starszych braci, maszyn wektorów nośnych (ang. support vector machines — SVM). Więcej o maszynach SVM dowiesz się z punktu 13.2.4. Choć unikam szczegółów matematycznych w omówieniu klasyfikatorów SVC, klasyfikatory te mają kilka bardzo atrakcyjnych właściwości matematycznych i praktycznych: 4c2836bbe011706b10838f45bf5d766f 272 Rozdział 8. Inne metody klasyfikacji 1. Działają dla podzbiorów danych — nacisk położony jest na trudne przykłady treningowe leżące blisko marginesów. W razie potrzeby klasyfikatory te dostosowują się do dodatkowych przykładów. Porównaj to z metodą k-NN, która zawsze wymaga zapisania wszystkich przykładów treningowych, aby mogła generować predykcje. 2. Granice klasyfikacji w klasyfikatorach SVC są proste i mają postać linii. Można też w wygodny sposób przekształcić ją na bardziej skomplikowaną postać, co wyjaśniam przy omawianiu maszyn SVM. 3. Klasyfikatory SVC zapewniają dobre uogólnianie i są skuteczne dla nowych danych testowych, ponieważ starają się zachowywać maksymalny odstęp między klasami. Pomocna jest tu zasada maksymalizacji marginesu. 4. Podstawowy problem optymalizacji związany ze znalezieniem najlepszej linii prowadzi tu do znalezienia rzeczywiście najlepszej linii przy danych ograniczeniach. Metoda ta nie zwraca nieoptymalnych linii. 8.3.1. Stosowanie klasyfikatorów SVC Oto kilka szczegółów praktycznych związanych z używaniem klasyfikatorów SVC: Klasyfikatory SVC z natury nie są dostosowane do klasyfikacji z użyciem wielu klas. Przeważnie używa się ich w systemach typu OvO (ang. one-versus-one, czyli jedna lub druga) i OvR (ang. one-versus-rest, czyli jedna lub pozostałe). Więcej o różnicach między tymi podejściami dowiesz się z punktu 6.3.4 i podrozdziału 6.4. Warto zauważyć, że nie musisz sam dodawać tego mechanizmu do maszyn SVM. Odbywa się to na zapleczu. W scikit-learn w wersji 0.19 wszystkie metody standardowo używają odmiany OvR. Wcześniej stosowane było połączenie odmian OvO i OvR. W sklearn dostępne są przynajmniej cztery sposoby na utworzenie klasyfikatora SVC: (1) użycie klasy LinearSVC, (2) użycie klasy SVC z jądrem liniowym, (3) użycie klasy SVC z jądrem wielomianowym stopnia 1 (jest to linia „w przebraniu") i (4) użycie klasy NuSVC z jądrem liniowym. Te cztery sposoby nie zawsze dają identyczne rezultaty. Jest to spowodowane różnicami w obliczeniach matematycznych, implementacji i argumentach domyślnych. Pogodzenie tych różnic sprawia trudności. Na końcu rozdziału przedstawiam uwagi na ten temat. Ponadto na razie nie omówiłem jeszcze jąder. Ich opis znajduje się dopiero w punkcie 13.2.4. Na razie traktuj je jak szczegół implementacji, choć w rzeczywistości są czymś znacznie ważniejszym

K najbliższych sąsiadów

K najbliższych sąsiadów (ang. *k nearest neighbours*, skr. ***KNN***) to algorytm uczenia maszynowego nadzorowanego operający swoje estymacje dla konkretnego przypadku danch na wartościach jego K najbliższych sąsiadów(punktów) liczonych min. dla przestrzeni Euklidesowej [7].

3.5. Prosty klasyfikator nr 1: najbliżsi sąsiedzi, związki na odległość i założenia Oto jedna z najprostszych koncepcji budowania predykcji z oznaczonego zbioru danych: 1. Znajdź sposób na opisanie podobieństw dwóch różnych próbek. 2. Kiedy konieczne jest stworzenie predykcji dla nowego, nieznanego przykładu, po prostu weź wartość z najbardziej podobnej znanej próbki. Ten proces to skrócony opis algorytmu „najbliżsi sąsiedzi". Znam ulubione przekąski moich trzech przyjaciół — Marcina, Barbary i Emila. Nowy przyjaciel, Andrzej, jest bardzo podobny do Marcina. Ulubiona przekąska Marcina to Cheetos. Myślę, że ulubiona przekąska Andrzeja będzie taka sama jak Marcina. Jest wiele sposobów na modyfikację tego podstawowego schematu. Możemy rozważyć więcej niż tylko jeden najbardziej podobny przykład: 1. Opisz podobieństwa pomiędzy parami przykładów. 2. Wybierz kilka najbardziej podobnych przykładów. 3. Połącz wybrane przykłady, aby otrzymać jedną odpowiedź.3.5. Prosty klasyfikator nr 1: najbliżsi sąsiedzi, związki na odległość i założenia Oto jedna z najprostszych koncepcji budowania predykcji z oznaczonego zbioru danych: 1. Znajdź sposób na opisanie podobieństw dwóch różnych próbek. 2. Kiedy konieczne jest stworzenie predykcji dla nowego, nieznanego przykładu, po prostu weź wartość z najbardziej podobnej znanej próbki. Ten proces to skrócony opis algorytmu „najbliżsi sąsiedzi". Znam ulubione przekąski moich trzech przyjaciół — Marcina, Barbary i Emila. Nowy przyjaciel, Andrzej, jest bardzo podobny do Marcina. Ulubiona przekąska Marcina to Cheetos. Myślę, że ulubiona przekąska Andrzeja będzie taka sama jak Marcina. Jest wiele sposobów na modyfikację tego podstawowego schematu. Możemy rozważyć więcej niż tylko jeden najbardziej podobny przykład: 1. Opisz podobieństwa pomiędzy parami przykładów. 2. Wybierz kilka najbardziej podobnych przykładów. 3. Połącz wybrane przykłady, aby otrzymać jedną odpowiedź.[12]

Do wyznaczenia odległości w metryce Euklidesowej stosowany jest wzór:

$$d_{(x,y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad [26]$$

popularne są również przestrzenie Manhattan:

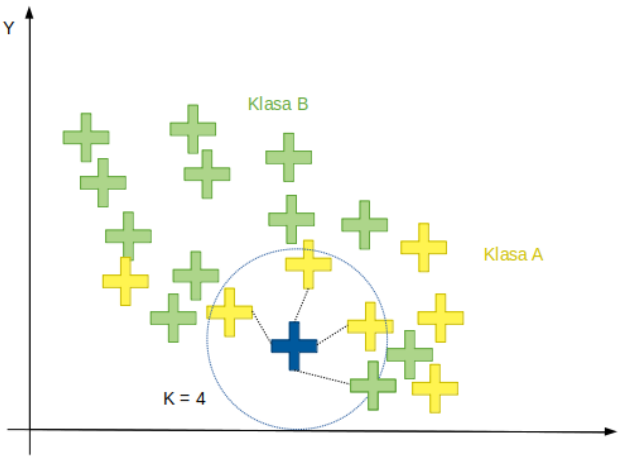
$$d_{(x,y)} = \sum_{i=1}^{n} |(x_i - y_i)| \quad [26]$$

oraz Mińkowskiego:

$$d = \left(\sum_{i=1}^{m} |u_i - v_i|^p\right)^{1/p} \quad [27]$$

Atrybut który nastraja proces uczenia się modelu i ma na niego największy wpływ określany jest jako hiperparametr. Dla KNN jest to liczba sąsiadów i może przyjmować maksymalnie wartości do rozmiaru zbioru cech. Im większa ilość jednostek mających wpływ, tym potęguje się niestety złożoność czasowa algorytmu, znacząco już większa od przedstawionych powyżej innych algoryrtmów,[7] oraz tym bardziej wzrasta ryzyko nadmiernego dopasowania do modelu testowanego.

W celu przewidzenia wartości dla nowych danych, należy odnaleść K najbliższych punktów wyliczając odległości, a następnie przpisać odpowiedź implikowaną przez większość sąsiadów. Dla wartości K równej jeden, metoda



ta nazywana jest algorytmem najbliższego sąsiada. [ˆschemat_wzorowany]

Dla lekarza wartością dodatnią jest wykrycie zależności które decyzują o uznaniu lub zaprzeczeniu występowania choroby. Zastosowanie algorytmu KNN może nie tylko zakwalifikować osoby chorujące na serce, ale również ułatwić swoją graficzną reprezentacją wpływ cech na ostateczny osąd próbki.

3.5.3. Kombinacja odpowiedzi Mamy jeszcze jeden problem do rozwiązania. Musimy zdecydować, jak połączymy znane wartości (głosy) pobliskich lub podobnych sąsiadów. Jeżeli nasz problem dotyczy klasyfikacji zwierząt, czterech naszych najbliższych sąsiadów może głosować na kota, kota, psa i zebrę. Jak zinterpretować przykład testowy? Rozsądne wydaje się wybranie najczęściej pojawiającej się odpowiedzi. Co ciekawe, możemy wykorzystać tę samą bazującą na sąsiedztwie technikę w regresji problemów, w których próbujemy przewidzieć wartość numeryczną. Jedyną rzeczą, którą musimy zmienić, jest sposób łączenia cech docelowych sąsiadów. Jeżeli trzech najbliższych sąsiadów daje numeryczne wartości 3,1, 2,2 i 7,1, to jak je połączyć? Możemy używać dowolnej statystyki, ale dwie powszechne i użyteczne alternatywy to średnia i mediana. Do k-NN dla regresji wrócimy w następnym rozdziale. 3.5.4. k-NN, parametry i metody bezparametrowe Ponieważ k-NN jest pierwszym omawianym przez nas modelem, trudno będzie go porównać z jakąkolwiek inną metodą. Część tych porównań zostawimy na później, ale jest jedna znacząca różnica, której możemy przyjrzeć się już teraz. Mam nadzieję, że udało mi się przykuć Twoją uwagę. Przypomnij sobie porównanie modelu uczenia się do maszyny z pokrętłami i dźwigniami z boku. W przeciwieństwie do wielu innych modeli dane wyjściowe z k-NN — czyli predykcje — nie mogą być wyliczone na podstawie przykładu wejściowego i wartości niewielkiej, stałej liczby pokręteł, które można dostosowywać. Potrzebujemy wszystkich danych treningowych do znalezienia wartości wyjściowej. Naprawdę? Wyobraź sobie, że wyrzucamy tylko jeden 4c2836bbe011706b10838f45bf5d766f 3.5. Prosty klasyfikator nr 1: najbliżsi sąsiedzi, związki na odległość i założenia 91 z przykładów treningowych. Ten przykład może być najbliższym sąsiadem nowego przykładu. Z pewnością brak tego przykładu będzie miał wpływ na dane wyjściowe. Są inne metody uczenia maszynowego posiadające podobne wymagania. Te inne metody potrzebują części, a nie wszystkich danych treningowych podczas testowania. Można twierdzić, że dla ustalonej liczby danych treningowych moglibyśmy mieć ustaloną liczbę pokręteł: powiedzmy, że 100 przykładów i 1 pokrętło na przykład, czyli 100 pokręteł. Być może. Wystarczy jednak, że dodam jeden przykład i potrzebujesz 101 pokręteł, a to już inna maszyna. W tym sensie liczba pokręteł maszyny k-NN zależna jest od liczby przykładów w danych treningowych. Jest lepszy sposób na opisanie tej zależności. Nasza maszyna miała z boku tackę pozwalającą przesyłać dodatkowe informacje. Możemy traktować dane treningowe jako te dodatkowe informacje. Niezależnie od tego, co wybierzemy, jeżeli potrzebujemy (1) rosnącej liczby pokręteł lub (2) bocznej tacki do dosyłania informacji, mówimy, że taka maszyna jest nieparametryczna. k-NN to nieparametryczna metoda uczenia się. Nieparametryczne metody uczenia się mogą mieć parametry. (Dzięki za nic, formalna definicjo). O co chodzi? Kiedy nazywamy metodę nieparametryczną, oznacza to, że dla tej metody relacja pomiędzy cechami a celami nie może być odzwierciedlona tylko za pomocą ustalonej liczby parametrów. Dla statystyków koncepcja ta jest powiązana z koncepcją parametrycznej i nieparametrycznej statystyki: statystyka nieparametryczna wykorzystuje mniej założeń odnośnie do koszyka danych. Przypomnij sobie jednak, że nie robimy żadnych założeń odnośnie do tego, jak nasza maszyna będąca czarną skrzynką ma się do rzeczywistości. Modele parametryczne (1) wykorzystują założenia odnośnie do klasy modelu, a następnie (2) wybierają konkretny model, ustawiając parametry. Ma to związek z dwoma pytaniami: jakie pokrętła znajdują się na maszynie i na jakie wartości są ustawione? W k-NN nie robimy takich założeń, jednakże k-NN buduje i polega na założeniach. Najważniejszym założeniem jest to, że nasze obliczenia podobieństwa są związane z rzeczywistym podobieństwem,

które chcemy uchwycić pomiędzy przykładami. [12]

Opis praktycznej częsci projektu

Narzędzia i biblioteki zastosowane w pojekcie

Biblioteki w większości posiadają otwarty kod źródłowy, napisany w języku Python [28].

Python

Python to łatwy do nauczenia, potężny język programowania. Posiada wydajne struktury danych wysokiego poziomu i prosty ale efektywne podejście do programowania obiektowego. Elegancka składnia Pythona i dynamiczne pisanie wraz z jego interpretowany charakter sprawia, że jest to idealny język do pisania skryptów i szybkiego tworzenia aplikacji w wielu obszarach większość platform. Interpreter Pythona i obszerna biblioteka standardowa są bezpłatnie dostępne w postaci źródłowej lub binarnej dla wszystkich głównych platformy ze strony internetowej Pythona, https://www.python.org/ i mogą być swobodnie rozpowszechniane. Ta sama strona również zawiera dystrybucje i wskaźniki do wielu darmowych modułów Pythona, programów i narzędzi innych firm oraz dodatkowych dokumentacja. Interpreter Pythona można łatwo rozbudować o nowe funkcje i typy danych zaimplementowane w C lub C++ (lub w innych języki wywoływane z C). Python nadaje się również jako język rozszerzeń dla aplikacji, które można dostosowywać. Ten samouczek wprowadza czytelnika nieformalnie do podstawowych pojęć i funkcji języka i systemu Python. Ono pomaga mieć pod ręką interpreter Pythona do praktycznego doświadczenia, ale wszystkie przykłady są samowystarczalne, więc samouczek można czytać również w trybie offline. Aby uzyskać opis standardowych obiektów i modułów, zobacz indeks biblioteki. indeks referencyjny daje bardziej formalną definicję języka. Aby napisać rozszerzenia w C lub C++, przeczytaj extending-index i c-api-index. Jest też kilka książek omówienie Pythona dogłębnie. Ten samouczek nie stara się być wyczerpujący i obejmuje każdą pojedynczą funkcję, a nawet każdą powszechnie używaną funkcja. Zamiast tego wprowadza wiele najbardziej godnych uwagi funkcji Pythona i daje dobre pojęcie o smak i styl języka. Po jego przeczytaniu będziesz mógł czytać i pisać moduły i programy Pythona oraz będziesz gotowy, aby dowiedzieć się więcej o różnych modułach bibliotecznych Pythona opisanych w bibliotecznym indeksie. [10]

Scikit-learn

Praktyczna część pracy napisana została w języku Python z wykorzystaniem *scikit-learn*, obsługującym wiele algorytmów maszynowego uczenia się w tym uczenia nadzorowanego i docelowo wybranych algorytmów przedstawionych w teoretycznej części pracy.

Biblioteka opiera się o *Numpy* oraz *Scipy*, daje zestaw narzędzi do obliczeń na macierzach, wektorach oraz umożliwiający metody numeryczne takie jak całkowanie, różniczkowanie i temu podobne [29]. W rezultacie można za jej pomoca wykonać elementy procesu nauczania algorytmu, takie jak: przetwarzanie wstępne, redukcja wymiarowości, klasyfikacja, regresja. [28]

The scikit-learn library is very well established within the machine learning community (developed for >10 years). However, it has to be emphasized that the numerical implementation of some classifiers might still be suboptimal

or erroneous. Moreover, this library makes external calls to other libraries implemented in Python (i.e., Numpy version 1.21.1) or C (i.e., BLAS version 3.7.1), which adds another source of potential software bugs. Nevertheless, debugging of the scikit-learn classification routines is beyond the scope of this work and the library was used as it was. Besides classifiers, the scikit-learn library comes with a set of routines used within this study for (1) generation of synthetic datasets, (2) computation of classification performance metrics, (3) data pre-processing and standardization, (4) hyper-parameter optimization, (5) feature selection and (6) visualization of classifiers' performance (plot_classifier_comparison.py). In the following subsections, a short summary of each classifier is given in order to facilitate further understanding and interpretation of the results. [21]

Pandas Do przygotowania danych wykorzystano zestaw narzędzi *Pandas*, ułatwiający tworzenie struktur danych i ich analizę. [todo] *Matplotlib*

W celu wizualizacji wyników w postaci wykresów zastosowano, opartą na *Matplotlib*, bibliotekę *Seaborn* powszechnie stosowaną do rysowania estetycznej grfiki statystycznej. [todo] *Flask*

Część prezentacyjna czyli możliwość wprowadzenia danych w formularzu na stronie i weryfikacja wyniku dla wyuczonych już modeli wykorzystuje bibliotkę *Flask*. Framework Flask ułatwia pisanie aplikacji internetowch i jest rozwiązaniem które daje duży zakres dowolności oraz możliwości. Flask sam z siebie nie definiuje warstwy bazy danych czy formularzy, pozwala za to na obsługę rozszerzeń które ubogacają aplikację o wybraną funkcjonalność. [30]

[todo] *JsonPickle* Przekazywanie obiektów o bardziej skomplikowanej budowie i ich *serializacja* oraz *deserializacja* do formatu JSON wykonane są za pomocą biblioteki *jsonpickle*, a zapis modeli wykonano za pomocą *joblib* która zapewnia obsługę obiektów Pythona i jest zoptymalizowana pod kątem pracy na dużych tablicach Numpy. [28]

[todo] *JobLib*

[todo]

Środowisko wykonania

Wykonanie programu i analizę danych testowych wykonano na maszynie o parametrach :

```
Procesor      Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz   2.50 GHz
Zainstalowana ęćpami RAM    8,00 GB (ędostpne: 7,82 GB)
Typ systemu 64-bitowy system operacyjny, procesor x64
```

Wersjie bibliotek wykorzystanych w projekcie:

```
setuptools~=49.2.1
Flask~=2.0.1
matplotlib~=3.4.2
numpy~=1.20.3
pandas~=1.2.4
scikit-learn~=0.24.2
pytest~=6.2.5
joblib~=1.0.1
scipy~=1.6.3
seaborn~=0.11.2
jsonpickle~=2.1
```

Interpreter Python w wersji *3.9*.

Moduły projektu:

Wstęp

- Config - zawiera statyczne zasoby oraz konfigurację logowania projektu
- Data - moduł odpowiada za wczytywanie i obróbkę danych testowych, zawiera definicje objektów wykorzystywanych przy uczeniu oraz zapisu modelu oraz przekazywaniu wyników prezentowanych na stronie

- Management:
  - PlotGeneration - moduł odpowiedzialny za prezentację wyników w postaci wykresów porównujących algorytmy oraz odpowiedzi na zadany problem oraz przechowuje obiekty które przy wywoływaniu funkcji od strony www są jedynie formatowane i zwrcane
  - Prediction - przygotowanie parametryzacji oraz implepmentacja treningu dla każdego z 3 algorytmów :
    * RF - przygotowanie, trening i zapis modelu sprecyzowany dla klasyfikatora lasów losowych
    * KNN - przygotowanie, trening i zapis modelu sprecyzowany dla klasyfikatora k-najbliższych sąsiadów
    * SVM - przygotowanie, trening i zapis modelu sprecyzowany dla maszyny wektorów nośnych
- Static - folder z grafikami, plikami stylów, skryptami javascript oraz jQuerry
- Templates - folder z stronami html wykorzystującymi dyrektywy Flask

Projekt posiada dwa tryby pracy :

- tryb nauczania na podstawie danych testowych
  machine learning z wykorzystaniem 3 algorytmów (*Run_Learning_Proces.xml*) , musi być wykonany przynajmniej raz przed wykorzystaniem programu jako aplikacja

- tryb aplikacji web
  wykorzystanie Flask do prezentacji i wykorzystania utworzonych modeli (*Run_Web_Application.xml*) , strona prezentuje analizę danych oraz uczenia algorytmów , przy czym głównym zadaniem jest wykonanie predykcji na podstawie danych wpisanych do formularza.

Trening algorytmu

Wstęp

Głównym zadaniem trybu nauczania jest utworzenie i wytrenowanie modeli dla 3 algorytmów nauczania nienazdorowanego, w tym celu wykonywany jest preprocesing danych czyli kolejno:

Przygotowanie danych

{ Skalowanie i normalizacja polegają na dostosowywaniu przedziału i wartości centralnej danych, aby ułatwić uczenie i interpretację wyników. Może przypominasz sobie, że zbiór danych diabetes z biblioteki sklearn (podrozdział 4.1) jest wstępnie standaryzowany, co jest jedną z postaci skalowania. Uzupełnianie brakujących wartości. W rzeczywistych zbiorach danych może brakować wartości. Przyczyny to trudności z rejestrowaniem kompletnych zbiorów danych i błędy w procesie zbierania danych. Brakujące wartości mogą zostać uzupełnione na podstawie wiedzy eksperckiej, heurystyk lub technik uczenia maszynowego. Wybór cech polega na eliminowaniu cech, ponieważ są nieistotne, nadmiarowe lub nawet szkodliwe. Czasem cech jest zbyt dużo i trzeba ograniczyć ich listę. Przykładem szkodliwej cechy jest zmienna identyfikacyjna, która nie pomaga w uogólnianiu modelu. W podrozdziale 8.2 zobaczyłeś, że unikatowe identyfikatory mogą prowadzić do powstania drzewa decyzyjnego z unikatowymi liśćmi dla każdego przykładu treningowego. Problem polega na tym, że nowego unikatowego identyfikatora z przykładu testowego nie będzie w drzewie. No i mamy problem. Kodowanie cech polega na doborze zbioru wartości symbolicznych reprezentujących różne kategorie. Opisałem to w rozdziale 1. Informację WZwiązku można zapisać za pomocą jednej kolumny przyjmującej wartości WZwiązku i Singel. Można też użyć dwóch kolumn, WZwiązku i Singel, z których jedna ma wartość Prawda, a druga — Fałsz. Jest to jeden ze sposobów tworzenia cech. Tworzenie cech polega na tworzeniu nowych cech na podstawie innych. Na przykład na podstawie długości i szerokości płatków kwiatu można utworzyć cechę powierzchnia płatka. Wyodrębnianie cech oznacza przechodzenie od niskopoziomowych cech, które nie są przydatne w uczeniu (dają słabe wyniki testów), do bardziej wysokopoziomowych cech przydatnych w tym procesie. Wyodrębnianie cech jest przydatne, gdy używane są specjalne formaty danych (na przykład grafika lub tekst), które trzeba przekształcić na tabelowy format wiersz-kolumna z przykładami i cechami. Wyodrębnianie cech i tworzenie cech różnią się poziomem złożoności przekształceń, ale koncepcyjnie polegają na tym samym} [12]

Proces przygotowania danych zastosowany w prajekcie składa się z następujących kroków:

2. Załadowanie i konkatenacja dataset'u

3. Uzupełnienie pustych wartości - dla późniejszego porównania tworzone są imputery dla 4 różnych form uzupełnienia
4. Standaryzacja
5. Konwersja danych dla kategorii
6. Normalizacja z wykorzystaniem MinMaxScaler.

Standardowo przed zebraniem danych należy zastanowić się na jaki problem chcemy uzyskać odpowiedź oraz zebrać jak najliczniejszą grupę cech i danych. Następnie należy wykonać wyeliminowanie cech nie wpływających na odpowiedź i dopiero po dokonaniu selecji przystapić do przetwarzania zebranych informacji.

{ 10.3. Skalowanie cech Tu omawiam dwa sposoby zmiany skali i wartości centralnej cech bez uwzględniania relacji z innymi cechami lub wartościami docelowymi. Zmiana skali polega na przekształcaniu danych w taki sposób, aby zmienić wartości skrajne i przesunąć wartości pośrednie w spójny sposób. Zmiana wartości centralnej oznacza przekształcenie danych tak, że wartości skrajne się zmieniają, a wartości pośrednie są przesuwane w spójny sposób. Zmiana skali często skutkuje też zmianą wartości centralnej. Dwa podstawowe sposoby zmiany skali to stała zmiana skali i modyfikacja wartości według statystyk obliczonych na podstawie danych. Chwileczkę, czy Markowi coś się nie pomieszało? Wszystko w porządku. Zaraz to wyjaśnię. Oto przykład stałej zmiany skali. Jeśli przeliczasz temperaturę ze stopni Fahrenheita na stopnie Celsjusza, 220°F daje 100°C, a 32°F to 0°C. Temperatury pośrednie, na przykład 104°F równe 40°C, są przeliczane według stałej reguły. Jest to trochę bardziej konkretne wyjaśnienie, ale jaka stała wartość jest używana w tych przekształceniach? No cóż, wynika ona ze wzoru na przeliczanie temperatur: 9 5(32) F C . Krótkie przekształcenia pozwalają uzyskać postać 17,7 9 5 C F , będącą naszym dawnym znajomym y = ax + b. Jeśli jeszcze nie zauważyłeś, przekształcanie wartości za pomocą wzoru liniowego po prostu rozciąga lub kompresuje wartości i przesuwa je; a odpowiada za rozciąganie lub kompresję, a b — za przesunięcie. }

10.4. Dyskretyzacja Zetknąłeś się już z dyskretyzacją (jest to proces przekształcania ciągłego przedziału wartości na skończoną, czyli dyskretną, liczbę kubełków) w kontekście implementowania regresji segmentowej ze stałymi. Tam trzeba było wybrać wartość wyjściową (segment linii) na podstawie kubełka, do którego należy wartość wejściowa. Dyskretyzacja pojawia się też w ramach podziałów w drzewach decyzyjnych; równanie wzrost > 1,7 ma wartość prawda lub fałsz. Istnieją też różne automatyczne metody dyskretyzacji danych. Niektóre z nich szukają zależności między wartościami cech, idealnych punktów podziału lub punktów podziału prowadzących do skutecznej klasyfikacji. Bardzo ważne jest, aby stosować te metody w ramach sprawdzianu krzyżowego, by zapobiec nadmiernemu dopasowaniu modelu do danych treningowych. Możesz traktować zaawansowane strategie dyskretyzacji jak samodzielne mikromodele. Wyobraź sobie, że potrafiłyby one dzielić wartości na poprawne kubełki w ramach klasyfikacji! Jednak na razie rozważam ręczne sposoby dyskretyzacji zbioru danych iris.

[12] ### Trening algorytmu

Podział danych

[todo dlaczego] W niniejszej pracy proponujemy procedurę walidacji krzyżowej uśredniania K-krotnego do wyboru modelu i estymacja parametrów. Ustalamy jego teoretyczną własność i pokazujemy jej obietnicę poprzez: badanie empiryczne. Ponieważ walidacja krzyżowa jest aktywnie wykorzystywana w wielu obszarach statystyki, ACV można również zastosować do szerokiego zakresu procedur modelowania. Na przykład, ACV może być łatwo stosowany z karną metodą wyboru modelu, dla której wybór interesujące są parametry kary. Zademonstrowane poprzez symulacje z wykorzystaniem LASSO, metoda ACV przewyższa metodę CV pod względem błędu średniokwadratowego i selekcji dokładność. Badamy również jego zastosowanie w wygładzaniu kwantylowym splajnów i demonstrujemy jego zdolność do zapewniania bardziej płynnego dopasowania danych niż tradycyjna metoda CV. Należy zwrócić uwagę na wielkość zestawu testowego i dobór K przy wdrażaniu ACV. Duża wartość K w połączeniu z małą wielkością próbki może powodować niewystarczające dopasowanie dane, ponieważ K-fold ACV dokonuje wyboru modelu na podstawie 1/K danych. Zalecamy uwzględnienie w zestawie testowym wystarczającej liczby obserwacji. Zwróć uwagę, że pożądany rozmiar zestaw testowy zależy również od złożoności bazowego prawdziwego modelu. [31]

Następnie wykonywany jest podział na dane treningowe i testowe z wykorzystaniem zdefiniowanej w bibliotece sklearn predefiniowanej metody. Tak spreparowany zestaw danych poddawany jest treningowi modelu kolejno dla każdego z algorytmów. Do dostrojenia parametrów oraz znalezienia najlepszego modelu wykorzystuwany jest:

GridSearchCV

{ GridSearchCV jest alternatywą dla naiwnej metody, którą opisałem powyżej. Zamiast ręcznie modyfikować parametry i kilkakrotnie ponownie uruchamiać algorytm, możesz wyświetlić listę wszystkich wartości parametrów,

które algorytm powinien wypróbować i przekazać do GridSeachCV.

GridSearchCV wypróbuje wszystkie kombinacje tych parametrów, oceni wyniki za pomocą weryfikacji krzyżowej i podanej przez Ciebie metryki punktacji. W końcu wypluwa najlepsze parametry dla twojego zestawu danych.

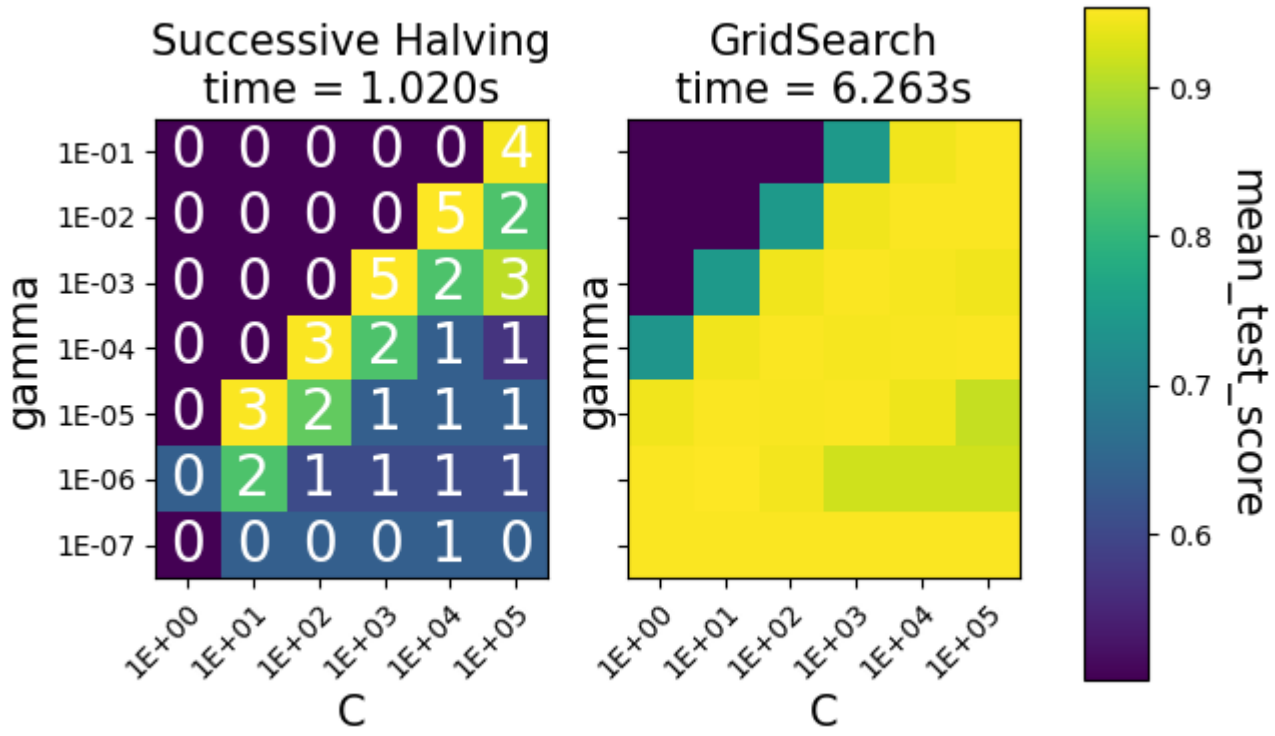
GridSearchCV może być używany z dowolnym algorytmem uczenia nadzorowanego uczenia maszynowego, który znajduje się w bibliotece nauki sci-kit. Będzie działać zarówno w przypadku regresji, jak i klasyfikacji, jeśli podasz odpowiednią metrykę. } ypowe modele, dla których wywołujemy funkcję fit, są w pełni zdefiniowane. Wybieramy model oraz jego hiperparametry, aby był konkretny. Co więc, do diabła, dzieje się po wywołaniu funkcji fit dla megamodelu z klasy GridSearchCV? Po wywołaniu funkcji fit dla modelu LinearRegression otrzymujemy zestaw wartości wag wewnętrznych parametrów. Są to preferowane, optymalne wartości dla treningowego zbioru danych. Po wywołaniu GridSearchCV z hiperparametrem n_neighbors dla algorytmu k-NN otrzymujemy wartość tego hiperparametru, która jest preferowanym jego ustawieniem dla danych treningowych. Te dwa podejścia robią to samo na różnych poziomach. Na rysunku 11.3 przedstawione jest graficzne ujęcie przeszukiwania siatki dla dwóch hiperparametrów z pokazanymi danymi wyjściowymi tej techniki: dopasowanym modelem ze skutecznymi hiperparametrami. Widoczny jest tu też sprawdzian krzyżowy będący wewnętrznym komponentem przeszukiwania siatki. Sprawdzian krzyżowy dla jednego hiperparametru pozwala ocenić połączenie tego hiperparametru i modelu. Następnie kilka takich ocen jest porównywanych, aby wybrać preferowany hiperparametr **confusion?**

W projekcie dla każdego algorytmy zapróbkowano większość dostępnych dla danego modelu klasyfikacji hiperparametrów przekazywane w param_grid. Wykorzystane parametry wykonania GridSearchCv [29]:

- estimator: implementacja interfejsu obiekt estymatora scikit-learn,
- param_grid: słownik parametrów które są potem testowane w dowolnej sekwencji ustawień,
- refit: dopasowanie best_estimator_ ,best_index_, best_score_ i best_params_ dla najlepszej sekwencji ustawień parametrów
- cv: parametr k dla KFold walidacji krzyżowej,
- verbose: obszerność logowanych informacji

HalvingGridSearchCV

Sciikit-learn udostępnia również inne implmentacj zastosowania walidacy krzyżowej np.: *HalvingGridSearchCV* lub *HalvingRandomSearchCV*. HalvingGridSearchCV polega na zmniejszaniu o połowe (z ang. *half*) zbioru parametrów po każdej iteracji algorytmu krzyżowego. Ta strategia wyszukiwania sukcesywnie zmniejsza ilość wymaganych iteracji dla danego zestawienia przez co wykonania jest szybsze niż w przypadku zwykłego GridSearchCv. Na poniższym wykresie przedstawiającym średni wynik dla algorytmu SVC widać że czas wykonania zmniejszył się ponad 6 krotnie w stosunku do GridSearch.



[29]

{ width=30% }

Umieszczone oznaczenia od 0 do 5 informują o tym w której iteracji kombinacja parmaretów została oznaczona jako najlepsze zestawienie. Implementacja ta nie została wykorzystana ze względu na nadal pozycjonowanie jej jako eksperymentalnej.

Estymatory to impementacja z skleran która powstała w oparciu o dokumntacje sklearn oraz dostępną dla nich parametryzacje. Zestawienie najlepszych osiągniętych estymatorów przedstawia się następująco:

KNeighborsClassifier [29] :

- n_neighbors: 8 - liczba sąsiadów z których wnioskowany jest jednostkowy resultat
- weights: distance - wagi na podsatwie których wyliczana jest predykcja , można zastosować wagę 1:1 lub nałożyć wagi zgodnie z dystansem.
- algorithm: auto - algorytm zastosowany do znalezienia najbliższych sąsiadów, w projekcie wykorzystano : brute-force oraz auto
- leaf_size: 1 - rozmiar liścia dla algorytmów BallTree or KDTree
- p: 1 - wykorzystanie miar odległości dla manhattan
- metric: canberra -metryka odległości

RandomForestClassifier [29] :

- criterion: [todo] - funkcja pomiaru dokładności rozgałęzienia
- min_samples_leaf: [todo] -minimalna liczba probek wymagana na liściu.
- min_weight_fraction_leaf: [todo] -minimalny ułamek sumy wag wymagany na liściu
- min_impurity_decrease: [todo] - większe lub równe zmniejszenie zanieczyszczenia powoduje podział danego węzła
  Zmniejszenie zanieczyszczenia liczone jest zgodnie z wzorem:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

gdzie N to całkowita liczba próbek, N_t to liczba próbek w bieżącym węźle, N_t_L to liczba próbek w lewym liściu, a N_t_R to liczba próbek w prawym liściu.

- max_features: [todo] - liczba funkcji najlepszego podziału

- random_state: [todo] - wykorzystywany przy próbkowaniu cech przy poszukiwaniu najlepszego podziału w węźle
- cpp_aplha: [todo] - zastosowanie to przycinanie drzewa o największej złożoności mniejszej niż cpp_alpha

SVC [29] :

- C: [todo]float, default=1.0 Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.
- kernel: [todo]- jądro wykorzystane w algorytmie
- degree: [todo] - stopień dla funkcji jądra *poly*
- gamma: [todo] - współczynnik jądra dla wartości *scale* parametr jądra ustawiany jest na wartość:

```
1 / (n * X.var())
```

dla wartości auto jest to :

```
1 / n
```

gdzie n to liczba cech.

- coef0: [todo] - niezależny parametr funkcji jądra , wykorzystywany tylko przy jądrach *poly* i *sigmoid*.
- shrinking: [todo] - heurystyka kurcząca
- cache_size: [todo] - cache jądra (w MB).

SVC używa jednego podstawowego parametru, C, do kontrolowania kompromisu między obciążeniem a wariancją. Bezpośrednia interpretacja tego parametru jest trudna. Klasa NuSVC wykonuje to samo zadanie co SVC, ale za pomocą innych obliczeń. Zwracam na to uwagę, ponieważ podstawowy parametr klasy NuSVC, (ny, ang. nu), ma proste znaczenie: przynajmniej procent danych jest zachowywanych jako wektory nośne. Ma to wpływ na błędy, przy czym są one określonego rodzaju — są to błędy marginesu. Błędy marginesu to punkty, które albo (1) znajdują się po złej stronie separatora (jest to błąd klasyfikacji), albo (2) znajdują się po właściwej stronie (zostały poprawnie sklasyfikowane), ale na marginesie. Inne znaczenie parametru jest takie, że dla danych treningowych akceptowany jest maksymalnie procent błędów marginesu. W określonych warunkach procent błędów marginesu rośnie do , a procent danych w wektorach nośnych spada do . Wartości znajdują się w przedziale [0, 1] i są interpretowane jako wartość procentowa od 0% do 100%. Choć klasa SVC jest trudniejsza do interpretacji, ma lepszą charakterystykę działania niż NuSVC. Parametry klasyfikatorów SVC są nieco magiczne. Są jak tajniki znane tylko alchemikom. Aby zrozumieć ich znaczenie, trzeba zagłębić się w zaawansowaną matematykę. Można jednak analizować skutki używania tych parametrów na niewielkich przykładach (napisałem „niewielkich", a nie „prostych"). Zobacz, jak zmiany wartości parametrów C i wpływają na granice między klasami.

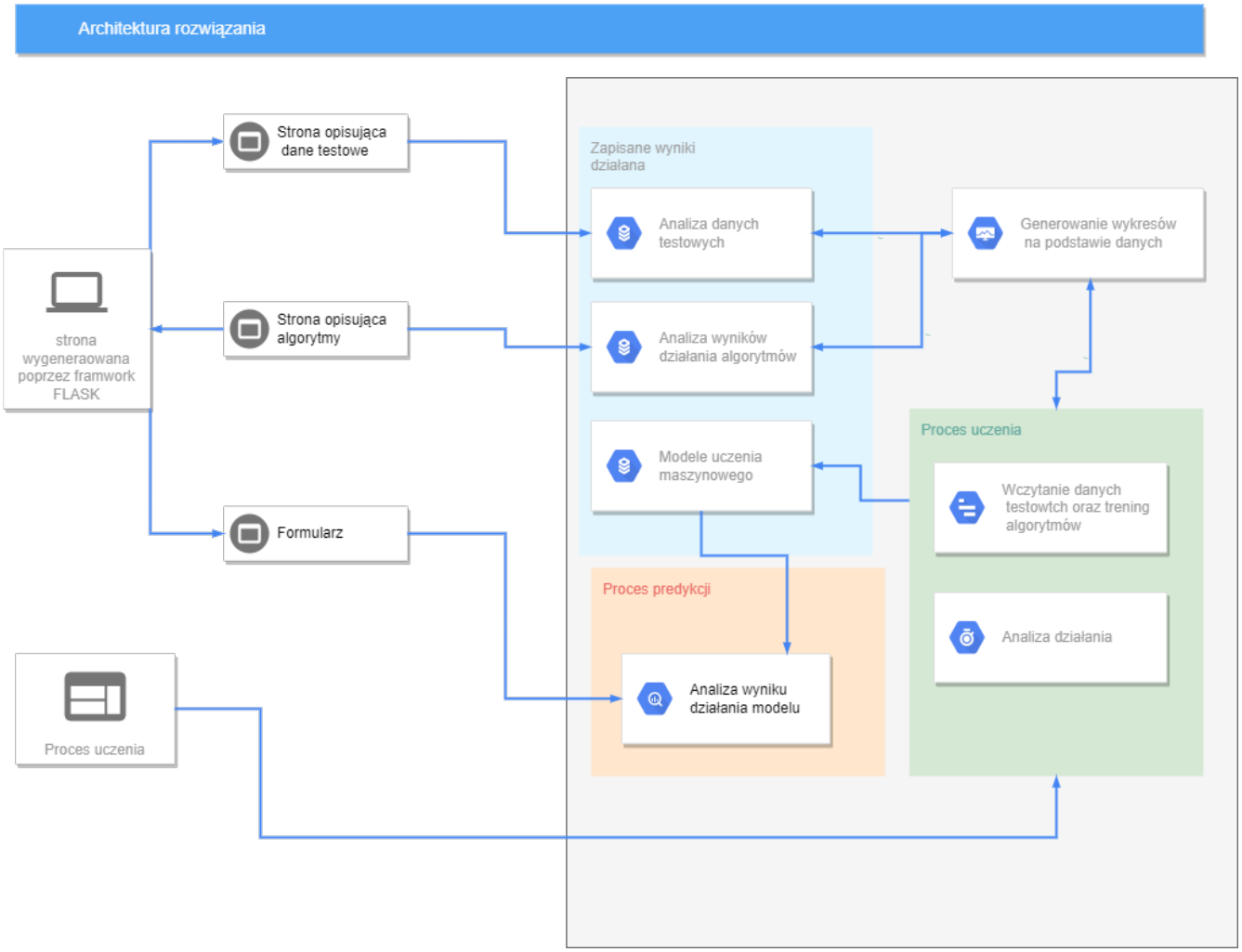
Z tego przykładu należy zapamiętać dwie rzeczy: 1. Zwiększanie i zmniejszanie C dają mniej więcej ten sam efekt. Jednak skale dla obu tych parametrów są wyraźnie różne. C zmienia się o rzędy wielkości, natomiast zmieniane jest liniowo w krokach o wielkości 1/10. 2. Duże i małe C mogą prowadzić do klasyfikatora SVC, który w pewnym zakresie ignoruje błędy klasyfikacj**confusion?**

Po odnalezieniu najlepszego estymatora model jest zapisywany oraz generowane sa wykresy dla trybu aplikacji webowej:

- wykresy modeli datasetu wejściowego i rozłożenia cech
- wykresy prezentujące zestawienia danych zebranych na temat algorytmu podczas wykonywania treningu.

Opis działania aplikacji webowej

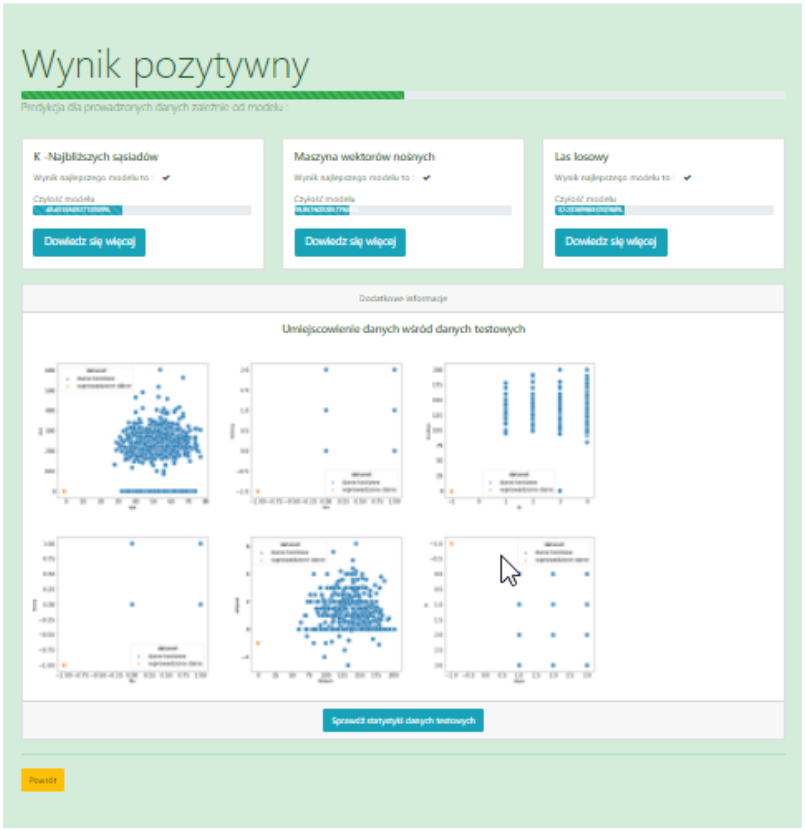
Poniżej przedstawiono architektówe działania:



Aplikacja posiada 4 widoki :

- widok głowny strony
- widok prezentacji danych wejściowych
- widok omówienia treningu algorytmów
- widok formularza pozwalającego na wykonanie predycji na wyuczonych modelach na podstawie własnych danych wejściowych

Zatwierdzenie formularza wyzwala odczytanie zapisanych modeli , iteracje i wykonanie predykcji na każdym z nich , następnie prezentowane są wyniki dla najlepszych estymatorów oraz wykresy wskazujące na umiejscowienie nowych danych na tle zbior testowego.



Porównanie działania modeli

Wstęp

Chodzi o to, że domyślna metoda oceny k-NN to średnia dokładność. Dokładność ma pewne fundamentalne ograniczenia i niebawem przejdziemy do tematu precision, recall, 4c2836bbe011706b10838f45bf5d766f 6.2. Więcej niż dokładność — wskaźniki dla klasyfikacji 187 roc_auc i f1 z rozległej listy powyżej. Po co omawiać te wskaźniki i co jest nie tak z dokładnością? Czy nie jest dokładna? Cieszę się, że pytasz. Przejdźmy od razu do odpowiedzi na Twoje pytania. Oto szybki przykład tego, jaki jest problem z dokładnością. Pamiętaj: dokładność polega w zasadzie na zliczeniu, ile razy odpowiedź jest poprawna. Wyobraźmy sobie zestaw danych, wśród których mamy 100 pacjentów i rzadką chorobę. Cieszymy się, że choroba jest rzadka, niemniej jest bardzo niebezpieczna. W naszym zestawie danych mamy 98 zdrowych pacjentów i 2 chore osoby. Weźmy prostą strategię bazową przepowiadającą, że wszyscy są zdrowi. Nasza dokładność wynosi 98%. To świetnie, prawda? Cóż, nie do końca. Znaleźliśmy dokładnie zero spośród ludzi, których mieliśmy znaleźć — tych, którzy wymagają leczenia. To poważny problem. Gdybyśmy mieli bardziej złożony system uczący się, który w taki sposób nie trafia z wynikami, bylibyśmy bardzo z niego niezadowoleni. W tym podrozdziale zamieszczone zostały wyniki oraz wykresy wygenerowane podczas treningu i weryfikacji danych testowych.

Dokładność w %								
Parametryzacja		parametry domyślne			wyznaczanie parametrow			
Algorytm / Imputer	średnia	mediana	stała	naj. wartość	średnia	mediana	stała	naj. wartość
Losowe lasy decyzyjne	100	100	100	100	100	100	100	100
Maszyna wektorów nośnych	100	100	100	100	100	100	100	100

Dokładność w %									
K-najbliższych sąsiadów	100	100	100	100	100	100	100	100	

•

Możemy zadawać pytania i uzyskiwać odpowiedzi za pomocą macierzy błędu. Na przykład, jeżeli jesteśmy lekarzami, interesuje nas, jak dobrze jesteśmy w stanie identyfikować ludzi, którzy rzeczywiście są chorzy. Skoro zdefiniowaliśmy chorobę jako nasz wynik pozytywny, są to nasze osoby z pierwszego rzędu. Pytamy, jak dobrze nam poszło dla RzeczP: ile spośród rzeczywiście chorych osób wykryliśmy poprawnie: RzeczP TP TP FN TP . Określamy to terminem czułość. Można to określić następująco: „Jak dobrze test jest dopasowany do znajdowania chorych ludzi" — nacisk kładziony jest na chorych ludzi. Jest to też nazywane skutecznością wyszukiwania przez ludzi zawodowo zajmujących się wyszukiwaniem informacji. Różne społeczności wymyśliły tę koncepcję niezależnie, dlatego nadały jej inne nazwy. Aby zrozumieć skuteczność wyszukiwania, pomyśl o ruchu wyszukiwarki internetowej. Spośród interesującego nas ruchu, RzeczP, ile zapytań odnaleźliśmy lub wyszukaliśmy poprawnie? Jeszcze raz dla czułości i skuteczności wyszukiwania interesuje nas poprawność w przypadku rzeczywistych pozytywnych lub interesujących nas przykładów. Czułość ma jeszcze jeden powszechnie używany synonim: współczynnik poprawnie pozytywnych (TPR — ang. true positive rate). Nie szukaj ołówka, aby to zapisać; za moment przedstawię Ci tabelę terminów. TPR jest współczynnikiem przypadków poprawnie pozytywnych w odniesieniu do rzeczywistości. Jest jeszcze kwestia związana z chorymi osobami, które odnaleźliśmy poprawnie: chore osoby, które oceniliśmy błędnie. Taki błąd nazywany jest błędnym negatywnym (ang. false negative — FN). Zgodnie ze wzorem chorzy ludzie, w przypadku których się pomyliliśmy, to TP FN FN . Możemy dodać chorych ludzi, których oceniliśmy poprawnie, i chorych ludzi, których nie odnaleźliśmy, aby uzyskać liczbę wszystkich chorych ludzi. Matematycznie wygląda to tak: 1 100%

*TP FN TP FN TP FN FN TP FN TP . Te równania mówią, że mogę rozbić wszystkich chorych ludzi na (1) chorych, o których myślę, że są chorzy, i (2) chorych, o których myślę, że są zdrowi. Możemy również zapytać: „Jak dobrze idzie nam ze zdrowymi ludźmi?". Nasza uwaga przenosi się teraz do drugiego rzędu RzeczN. Jeżeli jesteśmy lekarzami, chcemy wiedzieć, jaka jest wartość naszego testu, kiedy ludzie są zdrowi. O ile ryzyko jest inne, wciąż występuje ryzyko błędnej diagnozy zdrowej osoby. My — bawiący się przez chwilę w lekarzy — nie chcemy mówić ludziom, że są chorzy, kiedy są zdrowi. Nie tylko straszymy ich i dajemy powód do zmartwienia, ale możemy przepisać im zabiegi i lekarstwa, których nie potrzebują! Taki przypadek nazywany jest błędnym pozytywnym (ang. false positive — FP). Możemy go ocenić, patrząc na to, jak dobrze poszło nam ze zdrowymi osobami: RzeczN TN FP TN TN .Terminem diagnostycznym jest tutaj specyficzność testu: czy test zgłasza alert tylko w specyficznych 4c2836bbe011706b10838f45bf5d766f 190 Rozdział 6. Ocena klasyfikatorów przypadkach, o które nam chodzi. Specyficzność jest także nazywana współczynnikiem poprawnie negatywnych (ang. true negative rate — TNR). W istocie jest to częstotliwość występowania przypadków prawdziwie negatywnych w odniesieniu do rzeczywistości. Ostatnia kombinacja macierzy błędu bierze predykcje jako część główną, a rzeczywistość jako drugorzędną. Ja widzę to jako odpowiedź na pytanie: „Jaka jest wartość naszego testu, kiedy wynik jest pozytywny?" lub bardziej zwięźle: „Jaka jest wartość trafienia?". Cóż, trafienia to PredP. W ramach PredP obliczymy liczbę poprawnych: TP. Wtedy otrzymamy TP FP TP PredP TP , nazywane dokładnością. Spróbuj dziesięć razy szybko powiedzieć: „Jak dokładne są nasze predykcje pozytywne?". Rysunek 6.1 przedstawia macierz błędu wraz ze wskaźnikami jej oceny. Rysunek 6.1. Macierz błędu poprawnych i niepoprawnych predykcji Jeszcze jeden komentarz odnośnie do macierzy błędu. Podczas czytania o niej możesz zauważyć, że niektórzy autorzy zamieniają osie: ustawiają rzeczywistość w kolumnach, a predykcje w wierszach. Wtedy TP i TN będą tymi samymi komórkami, ale FP i FN będą zamienione. Nieświadomy czytelnik może być bardzo, bardzo skonsternowany. Uznaj się za ostrzeżonego i bądź uważny, czytając inne omówienia macierzy błędu.***confusion?***

Zestawienie efektywności działania algorytmów

Konfrontacja technik ucznia maszynowego zaleznie od zestawu danch będzie dawała odmienne wyniki ze względu na ich predyspozycje do zajmowania się odpowiednimi zbiorami danych.

Potencjał algorytmów dla niewielkiego kompletu danych zawierającego wartości

Zczynając od drzew decyzyjnych,można od razu stwierdzić ich niski potencjał. Istnieje zbyt duże prawdopodobieństwo dopasowania się do modelu treningowego, gdyż wspomniany zbiór dancyh wejściowych nie jest wystarczająco liczny. Dlatego w pracy omówione zostały lasy decyzyjne.

Większej dokładności można się spodziewać po metodzie wektorów nośnych, ale jego złożoność czasowa oraz pamięciowa mogą zaniżyc jego ogólną klasyfikację.

K-najbliższego sąsiada może być przydatny w przypadku danych nieliniowych oraz łatwo wykorzystany w problemach regresji. Wartość wyjściowa obiektu jest obliczana przez średnią k wartości najbliższych sąsiadów. Niestety tak samo jak w przypadku maszyny wektorów nośnych jest wolniejsza i bardziej kosztowna pod względem czasu i pamięci. Wymaga dużej pamięci do przechowywania całego zestawu danych treningowych do przewidywania oraz nie nadaje się również do dużych danych wymiarowych.

Wskaźniki wydajności

Określenie stopnia, w jakim skonstruowany model z powodzeniem realizuje wyznaczone zadanie należy do wskaźnika wydajności. Przykładem nieprawidłowego wyboru może być próba przewidzenia wystąpienia rzadkiej choroby u pacjenta i określenie głownym miernikiem *dokładność*. W takim scenariuszu klasyfikacja wszystkich pacjentów jako zdrowych , daje niewiele odbiegającą od perfekcji dokładność, a jednocześnie błędnie osądzać każde wystąpienie choroby.

Losowe lasy decyzyjne

##todo liczenie błędów macież pomysłek

Losowe lasy decyzyjne

###OCENA PODELI ORAZ UŻYTYCH PARAMETRÓW -OCENA SZYBKOŚCI WYKONANIA -OCENA ZALEŻNIE OD UZUPELNIANIA DANYCH -OCENA ZALEŻNIE OD DOBRANEJ PARAMERYZACJI : - które parametry mają i wpływ i dlaczego: - ZALEŻNIE OD METRYKI(SHORT OPIS METRYK)

Maszyna wektorów nośnych

###OCENA PODELI ORAZ UŻYTYCH PARAMETRÓW -OCENA SZYBKOŚCI WYKONANIA -OCENA ZALEŻNIE OD UZUPELNIANIA DANYCH -OCENA ZALEŻNIE OD DOBRANEJ PARAMERYZACJI : - które parametry mają i wpływ i dlaczego: - ZALEŻNIE OD METRYKI(SHORT OPIS METRYK)

K-najbliższych sąsiadów

###OCENA PODELI ORAZ UŻYTYCH PARAMETRÓW -OCENA SZYBKOŚCI WYKONANIA -OCENA ZALEŻNIE OD UZUPELNIANIA DANYCH -OCENA ZALEŻNIE OD DOBRANEJ PARAMERYZACJI : - które parametry mają i wpływ i dlaczego: - ZALEŻNIE OD METRYKI(SHORT OPIS METRYK)

Porównianie całościowe algorytmów : złożoność czasowa , dokładność , złożoność implementacyjna , wpływ danych wykorzytywanych w modelu

Wybrane najlepsze modele klasyfikacji:

[CV 5/15] END C=0.1, cache_size=200, coef0=0.0, degree=1, gamma=scale, kernel=linear, shrinking=False;, score=0.982 total time= 0.0s [CV 4/15] END C=0.1, cache_size=200, coef0=0.0, degree=1, gamma=scale, kernel=poly, shrinking=True;, score=0.982 total time= 0.0s [CV 4/15] END C=0.1, cache_size=200, coef0=0.0, degree=1, gamma=scale, kernel=poly, shrinking=False;, score=0.982 total time= 0.0s [CV 4/15] END C=10, cache_size=500, coef0=0.3, degree=5, gamma=scale, kernel=linear, shrinking=True;, score=0.982 total time= 0.0s [CV 4/15] END C=10, cache_size=200, coef0=0.3, degree=5, gamma=auto, kernel=rbf, shrinking=True;, score=0.982 total time= 0.0s

Wybrane najgorsze modele klasyfikacji :

[CV 10/15] END C=100, cache_size=200, coef0=0.3, degree=5, gamma=scale, kernel=poly, shrinking=True;, score=0.800 total time= 0.0s [CV 10/15] END C=100, cache_size=500, coef0=0.3, degree=5, gamma=scale, kernel=poly, shrinking=False;, score=0.800 total time= 0.0s [CV 10/15] END C=100, cache_size=200, coef0=0.3, degree=5, gamma=scale, kernel=poly, shrinking=False;, score=0.800 total time= 0.0s

Najlepsze modele i wartości dla regresji :

Najgorsze modele i wartości dla regresji :

Maszyna wektorów nośnych

K-najbliższych sąsiadów

###OCENA PODELI ORAZ UŻYTYCH PARAMETRÓW

problem multiklasyfikacji - problem regresji kategrycznej - zwykła regresja , mierzyć będe metoda prównania - tzrea było wprowadzić reguły do float na int -> inne metody do liczenia błędów na dzień dobry widzimy nie dokładność ze względu na klasyfiakcję po przecinku regresja kategoryczna -> rzutowanie przedziału wartości na wartość graniczną

score method of classifiers Every estimator or model in Scikit-learn has a score method after being trained on the data, usually X_train, y_train.

When you call score on classifiers like LogisticRegression, RandomForestClassifier, etc. the method computes the accuracy score by default (accuracy is #correct_preds / #all_preds). By default, the score method does not need the actual predictions. So, when you call:

clf.score(X_test, y_test) it makes predictions using X_test under the hood and uses those predictions to calculate accuracy score. Think of score as a shorthand to calculate accuracy since it is such a common metric. It is also implemented to avoid calculating accuracy like this which involves more steps:

from sklearn.metrics import accuracy score

preds = clf.predict(X_test)

accuracy_score(y_test, preds) When using accuracy_score you need ready predictions, i.e. the function does not generate prediction using the test set under the hood.

For classifiers, accuracy_score and score are both the same - they are just different ways of calculating the same thing.

score method of regressors When score is called on regressors, the coefficient of determination - R2 is calculated by default. As in classifiers, the score method is simply a shorthand to calculate R2 since it is commonly used to assess the performance of a regressor.

reg.score(X_test, y_test) As you see, you have to pass just the test sets to score and it is done. However, there is another way of calculating R2 which is:

from sklearn.metrics import r2_score

preds = reg.predict(X_test)

r2_score(y_test, preds) Unlike the simple score, r2_score requires ready predictions - it does not calculate them under the hood.

So, again the takeaway is r2_score and score for regressors are the same - they are just different ways of calculating the coefficient of determination.score method of classifiers Every estimator or model in Scikit-learn has a score method after being trained on the data, usually X_train, y_train.

When you call score on classifiers like LogisticRegression, RandomForestClassifier, etc. the method computes the accuracy score by default (accuracy is #correct_preds / #all_preds). By default, the score method does not need the actual predictions. So, when you call:

clf.score(X_test, y_test) it makes predictions using X_test under the hood and uses those predictions to calculate accuracy score. Think of score as a shorthand to calculate accuracy since it is such a common metric. It is also implemented to avoid calculating accuracy like this which involves more steps:

from sklearn.metrics import accuracy score

preds = clf.predict(X_test)

accuracy_score(y_test, preds) When using accuracy_score you need ready predictions, i.e. the function does not generate prediction using the test set under the hood.

For classifiers, accuracy_score and score are both the same - they are just different ways of calculating the same thing.

score method of regressors When score is called on regressors, the coefficient of determination - R2 is calculated by default. As in classifiers, the score method is simply a shorthand to calculate R2 since it is commonly used to assess the performance of a regressor.

reg.score(X_test, y_test) As you see, you have to pass just the test sets to score and it is done. However, there is another way of calculating R2 which is:

from sklearn.metrics import r2_score

preds = reg.predict(X_test)

r2_score(y_test, preds) Unlike the simple score, r2_score requires ready predictions - it does not calculate them under the hood.

So, again the takeaway is r2_score and score for regressors are the same - they are just different ways of calculating the coefficient of determination. ++++++++++ Is it possible that after running the optimization my score won't get better (and even worse?) ?

Yes, theoretically, by pure luck, it is possible that your initial guess, before optimization of hyper-parameters, provides better results than the best of parameter combination found in the parameters grid. However, assuming you have enough data and your parameter grid is wide enough it is rather unlikely that the tuning of hyper-parameters would not be able to find better results. Such behavior rather indicates that something is wrong with your approach or your data.

If understand correctly, the choice of the best parameters is based on the cv results on training data, while in your final run the performance is assessed based on test dataset. If the distribution of training and test data differ significantly it could lead to the situation when the parameters providing the best results on the training data perform poorly on test data.

Where is my mistake?

As already mentioned by others, the parameters you are testing after the tuning were not included in the parameter grid. In this case it is incorrect to talk about the model performance "after running the optimization".

I suggest the following in order to investigate and fix the problem

Instead of using the hard-coded parameters in the XGBClassifier call, use the optimal parameters found by tuning process, i.e. grid_search.best_params_. Furthermore, if you think that subsample and cilsample_bytree (a typo?) are relevant parameters include them in the parameters grid. Increase the cv parameter to e.g. 5-10, the results with cv = 3 might be very unstable. You can assess the stability of your current results by using different random seeds and repeating the entire exercise. Make sure that you use the consistent parameters in tuning process and in the final evaluation, or just include these parameters in the parameters grid if possible. In particular, check early_stopping_rounds and eval_metric. Are there other parameters that could influence or improve my model?

From your code it is unclear how many rounds you use. Either increase n_estimators or include it in the parameters grid. Given that you use AUCPR you might need to explicitly set the parameter maximize=True, otherwise in your final run you could minimize the AUCPR, which could explain poor results. Share Improve this answer Follow edited Sep 28, 2020 at 22:22 answered Sep 28, 2020 at 21:34 user avatar aivanov 1,43077 silver badges1414 bronze badges Add a comment

0

This question is a little wrong-worded. You cannot get worse after optimization, otherwise it wouldn't be optimization! (At worst you are at the same performance like before, getting the exact same parameters you already had)

As Grzegorz points out in a comment, first of all your parameter list isn't complete and doesn't contain the values you use later. For example the learning rate, but also max_depth. Secondly, a grid search where you don't really know where to look should contain a much larger variance for the parameters. You check [0.1, 0.01, 0.05] for the learning rate, but did you check [0.0001, 0.001, 1.]? The learning rate might be a bad example here but I hope it gets the point across, you might want to check magnitude/scale first, e.g. powers of ten, before checking small variations.

Depending on your dataset, the difference between runs with the same values might also come from different seeds! Check that you either always set the same seed, or try it enough times with different seeds to get a comparable answer (for example with KFold).

Is your model even converging for every training? Where do you make sure that you train long enough? You can plot the loss for the training and test sample and check if it's converging or not. This can be controlled with n_estimators in xgboost I believe.

There is nothing wrong in your code or process. Often times in machine learning performance on the test dataset is lower than than performance on the training data set. Your model is not generalizing perfectly to the data it has not seen before (i.e., the test dataset).

-2

When you do hyper-parameter tuning, you improve the regularization of the model. Before you did optimization, you could be overfitting. After optimization you regularized your model and now it performs just right.

enter image description here

Then your model score will be worse after optimization on training set. Your model score will also be bad for test set if your model heavily relies on one feature for classification.

You can use learning curve to see how the curve changes when you use optimization vs when you don't. And you can use df.corr() to see the correlation matrix for the correlation between feature values and target values. ++++++++++++++ Spis ilustracji{.unnumbered} ======== [^schemat_wzorowany]:Na podstwie materiałów opublikowanych na https://www.datacamp.com

Spis tabel

Bibliografia

[1] A. L. Fradkov, "Early history of machine learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, 2020, doi: https://doi.org/10.1016/j.ifacol.2020.12.1888.

[2] 2. Jesper E. van Engelen1 · Holger H. Hoos1, *A survey on semi-supervised learning.*

[3] W. Modrzejewski and W. J. Musiał, "Stare i nowe i czynniki ryzyka sercowo-naczyniowego - jak zahamować epidemię miażdżycy? Część i. Klasyczne czynniki ryzyka," *Forum Zaburzeń Metabolicznych*, vol. 1(2), pp. 106–114, 2010.

[4] "Międzynarodowa sieć firm audytorsko-doradczych ze szczególnym uwzględnieniem branży dóbr konsumpcyjnych, usług finansowych, nieruchomości i budownictwa, technologii informacyjnych, mediów i komunikacji (TMT), transportowej (TSL), produkcji przemysłowej, a także sektora publicznego."

[5] K. Karol, *Uczenie maszynowe w opiece zdrowotnej.* Roczniki Kolegium Analiz Ekonomicznych/Szkoła Główna Handlowa 56, 2019, pp. 305–316.

[6] V. Nasteski, "An overview of the supervised machine learning methods."

[7] J. Grus, *Data science from scratch:first principles with python.* pp. r11 pp140.

[8] T. Łysiak, *The use of machine learning methods in predicting stock prices on the stock exchange.*

[9] J. Laska, "An overview of machine learning methods used in sentiment analysis."

[10] H. van Engelen J. E., "A survey on semi-supervised learning," *Technologie informatyczne w administracji publicznej i służbie zdrowia*, pp. 373–440, 2020.

[11] G. ;. G. Pedregosa F.; Varoquaux, *Scikit-learn: Machine learning in PythonScikit-learn: Machine learning in python.* 2011, pp. 2825–2830.

[12] M. E. FENNER, *Machine learning with python for everyone.*

[13] D. Dua and C. Graff, "UCI machine learning repository [http://archive.ics.uci.edu/ml]," *Machine Learning Technical Reports*, vol. 1(1), pp. 1–6, 2019.

[14] dr n. med. Robert Detrano,.

[15] M. Andras Janosi,.

[16] W. S. M. M. P. MD,.

[17] R. H. F. Peshawa J. Muhammad Ali, "Data normalization and standardization: A technical report," *Machine Learning Technical Reports*, vol. 1(1), pp. 1–6, 2014.

[18] D. Kumar, "Introduction to data preprocessing in machine learning beginners guide for data preprocessing."

[19] A. G. D. Anguita L. Ghelardoni, "The 'k' in k-fold cross validation," *Journal of Machine Learning Research.*

[20] G. Bonaccorso, *Mastering machine learning algorithms: Expert techniques to implement popular machine learning algorithms and fine-tune your models.*

[21] *J. P. M. and J. S. Bojanowski, "Comparison of the novel probabilistic self-optimizing vectorized earth observation retrieval classifier with common machine learning algorithms."

[22] K. Matthew, *Thoughtful machine learning with python a test-driven approach.* pp. r.1 pp8.

[23] T. D. Breiman L., *Random forests, machine learning.* StatSoft Polska Sp. z o. o, 2001.

[24] S. HUANG, N. CAI, P. P. PACHECO, S. NARRANDES, Y. WANG, and W. XU, "Applications of support vector machine (SVM) learning in cancer genomics," *Cancer Genomics & Proteomics*, vol. 15, no. 1, pp. 41–51, 2018, Available: https://cgp.iiarjournals.org/content/15/1/41

[25] A. Pielowska, "Maszyna wektorów nośnych."

[26] H. S. R. Sudip Karki, "Comparison of a*, euclidean and manhattan distance using influence map in ms. Pac-man," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] C. B. Binbin Lua Martin Charltonb and P. Harrisc, "The minkowski approach for choosing the distance metric in geographically weighted regression."

[28] V. S. W. Samrudhi Rajendra Kaware, *Podejście porównawcze do algorytmów uczenia się maszynowego.* Reading, Massachusetts: Addison-Wesley, 1993.

[29] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[30] M. Grinberg, *Flask web development: Developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[31] H. J. Jung Yoonsuh, "A k-fold averaging cross-validation procedure," *Journal of nonparametric statistics*, vol. 27, 2015.