Metody sztucznej inteligencji – Projekt Implementacja algorytmu Support Vector Machines (SVM) z liniową funkcją bazową dla problemów wieloklasowych.

Mateusz Franków 259740, Bartosz Szostak 259750

Spis treści

1 Wstęp i przegląd literatury etapy 1 oraz 2

1.1 Wstęp

W projekcie zajmiemy się opisem i implementacją algorytmu Support Vector Machines (SVM) z liniową funkcją bazową dla problemów wieloklasowych. Algorytm Support Vector Machine (SVM) to jeden z najpopularniejszych algorytmów uczenia maszynowego wykorzystywany w dziedzinie klasyfikacji i regresji. Opiera się na koncepcji znalezienia hiperpłaszczyzny, która najlepiej rozdziela dane należące do różnych klas. Algorytm szczególnie dobrze radzi sobie z analizą przy bardzo dużej liczbie zmiennych predykcyjnych. W przypadku problemów z dwiema klasami SVM znajduje optymalną hiperpłaszczyznę, a w przypadku problemów z wieloma klasami stosowane są różne strategie, takie jak strategia one-vs-one (jeden vs jeden) lub one-vs-all (jeden vs reszta). Algorytm SVM wykorzystywany jest w wielu dziedzinach takich jak rozpoznawanie twarzy, tekstu, obrazów, ale także przewidywanie struktury białek czy rozpoznawaniu mowy.[6][2]

1.2 Powiązane prace

Algorytm SVM jest relatywnie starym algorytmem stworzonym jeszcze w 20 wieku przez Vladimira Vapnika wraz z współpracownikami w AT&T Bell Laboratiories. „*The SVM algorithm was originally proposed to construct a linear classifier in 1963 by Vapnik.*"[1] W sieci można znaleźć wiele prac związanych z problemem wieloklasowości jednak my odniesiemy się jedynie do kilku z najważniejszych prac. Na samym wstępie warto opisać sam model liniowy klasyfikacji binarnej:

$\hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b > 0$

Jeśli funkcja jest mniejsza od zera, przewidujemy klasę -1, jeśli większa, przewidujemy klasę +1. Wyjście $\hat{y}$, jest liniową funkcją cech: linią, płaszczyzną lub hiperpłaszczyzną. Algorytmy różnią się sposobem wyznaczania 'w' i 'b'. Jednym z najbardziej popularnych algorytmów jest regresja logistyczna oraz maszyny wektorów wsparcia SVM. My zajmiemy się tym drugim. SVM ma na celu stworzenie granicy decyzyjnej między dwiema klasami, nazywaną hiperpłaszczyzną. Umożliwia to przewidzenie etykiet na podstawie, po której części granicy(hiperpłaszczyzny) znajduje się badana próbka. Dąży się do tego, aby hiperpłaszczyzna znajdowała się jak najdalej od najbliższych punktów(próbek). „*The objective of training an SVM model is to find the w*

and b so that the hyperplane separates the data and maximizes the margin" [1] W przypadku większej ilości klas niż dwie, problem staje się wieloklasowy. Problem wieloklasowy może zostać rozbity na wiele przypadków klasyfikacji binarnej, one-vs-one i one-vs-rest. [2]

Dla LinearSVC parametrem kompromisowym (trade-off), który określa regularyzacje nazywany jest 'C', wyższe wartości 'C' odpowiadają zmniejszeniu regularyzacji, staramy się jak najlepiej dopasować C do zbioru treningowego.[2] W literaturze Blatz i Vapnik zalecili stosowanie C=5, jednak jest to w zupełności zależne od zbioru danych i problemu jaki rozwiązujemy. [4]

1.3 Najważniejsze pojęcia dla projektu

- SVM - algorytm uczenia maszynowego stosowany do klasyfikacji binarnej i regresji, jego zadaniem jest znalezienie hiperpłaszczyzny. [2]
- Hiperpłaszczyzna - wykorzystywana do oddzielenia danych wejściowych na poszczególne klasy. [2]
- Margines - odległość między hiperpłaszczyzną, a najbliższym punktem ze zbiory uczącego [2]
- Wektor wag - znaczenie poszczególnych cech wejściowych dla klasyfikacji. [2]
- Bias - wartość dodawana do wyniku iloczynu skalarnego wag wektora i cech, ma za zadanie dopasowanie hiperpłaszczyzny do danych treningowych. [5]
- Parametr C - parametr kompromisowy dla LinearSVC, określa reguły regularyzacji. [2]
- Wektor wsparcia - punkty danych które znajdują się bliżej hiperpłaszczyzny i wpływają na jej położenie i orientacje, wpływają na margines klasyfikatora [7]

1.4 Omówienie istniejących metod, które rozwiązują opisany problem

- One-vs-All (jeden przeciwko wszystkim) - polega na szkoleniu oddzielnego klasyfikatora SVM dla każdej klasy w zbiorze danych. Każdy klasyfikator jest uczony, aby rozróżniać tę klasę od innych, traktując wszystkie pozostałe klasy jako unikalną kategorię. Ostatecznie, dla każdej nowej próbki, każdy klasyfikator SVM ocenia, czy próbka należy do jego klasy, a klasa z najwyższym wynikiem jest wybrana jako wynik. Metoda One-vs-All jest stosowana, gdy liczba klas jest większa od 2. Szybsza i łatwiejsza do implementacji, ale może zawierać niezrównoważoną klasyfikacje (względem one-vs-one).

Istnieje możliwość skonstruowania n hiperpłaszczyzn, dla metody one-vs-all, gdzie n to ilość klas. Funkcja decyzyjna dla wieloklasowego SVM przedstawia się wzorem:

$f(x) = sign(\ w_n * x + b_n)$ [8]
$w_n - waga\ hiperpaszczyzny$,
$b_n - przesunicie(bias)$

- One-vs-One (jeden na jednego) - klasyfikator SVM jest szkolony na każdej parze klas w zbiorze danych. Klasyfikator jest uczony, aby rozróżniać jedną klasę od drugiej. Po nauczeniu każdy klasyfikator określa, do jakiej klasy należy próbka. Na podstawie wyników wszystkich klasyfikatorów wybierana jest klasa z największą liczbą głosów do której przydzielana jest próbka. Dokładniejsza, wymagająca złożonych obliczeń – przez co wolniejsza i cięższa w implementacji (względem one-vs-all).

Istnieje możliwość skonstruowania $\frac{n(n-1)}{2}$ hiperpłaszczyzn, dla metody one-vs-one, gdzie n to ilość klas. Funkcja decyzyjna dla wieloklasowego SVM przedstawia się wzorem:

$f(x) = arg\max_n [(w_n * x) + b_n]\ ,\quad n = 1, \ldots, k$ [5]

$$w_n - waga\ hiperpaszczyzny,$$

$$b_n - przesunicie(bias)$$

- Metoda uśredniania - uśrednianie predykcji klasyfikatorów, każdy trenowany jest na innych podzbiorach danych i podzbiorach atrybutów

1.5 Wskazanie metod referencyjnych

Biblioteki sklearn:

- SVC - one vs one
- SVC - one vs all
- KNN
- GNB

2 Metoda etap - 4

2.1 One vs one

One vs one dzieli problem wieloklasowy na problem binarny. Biorąc pod uwagę nasz problem Irisów gdzie występują trzy klasy, one vs one podzieli problem wieloklasowy na trzy problemy binarne. W każdym z problemów binarnych porówna ze sobą dwa zbiory danych o różnych etykietach. Każdy klasyfikator przewiduje jedną etykietę, metoda kończy się głosowaniem, w rezultacie której wygrywa klasyfikator z największą liczbą głosów.
Klasa dziedziczy po BaseEstimator z biblioteki scikit-learn.
Metoda init inicjalizuje dwie listy dla klas i klasyfikatorów, dziedziczy po self.
Metoda fit dziedziczy po self, X, y. Inicjalizuje tablice self.classes która zawiera unikalne klasy w zbiorze etykiet, dzieli klasy na wszystkie możliwe kombinacje par klas, maska tworzy maskę logiczną która wskazuje do której klasy (a lub) należy etykieta. Para_X i para_y jest zbiorem danych treningowych X i etykiet y, dla których maska jest prawdziwa. Następnie tworzona jest binarna reprezentacja etykiet i przypisywana jest wartość 1 dla class_a i -1 dla class_b. Podzbiór danych treningowych para_X i odpowiadających jej etykiet para_y jest ćwiczony. W ostatnim etapie informacje o wszystkich trenowanych klasyfikatorach dla par klas przekazywana jest do tablicy, która użyta zostanie w predict.
Metoda predict dziedziczy po self i X_test. W pierwszym punkcie inicjalizowana jest lista przewidywanie_etykiet. Następnie realizowana jest metoda głosowania większościowego. Iteracja po wszystkich danych testowych, inicjalizacja listy glosy_class, iteracja po wszystkich wytrenowanych klasyfikatorach, użycie metody predict – gdzie zwrócona wartość pokazuje przewidywaną klasę dla próbki, class_a_idx sprawdza elementy w tablicy self.classess

równe class_a – zwraca tablice wartości, gdzie True(1) oznaczają pasujące elementy, analogicznie działa class_b_idx dla elementów równych class_b. Dzięki temu zwiększane są liczby głosów w każdej z klas, używając funkcji np.argmax(glosy_class) wybierana jest klasa z największą liczbą głosów – zwracany jest jej index. Następnie przewidziana etykieta jest przypisywana do danej próbki, przewidziane etykiety dodawane są do listy przewidywane_etykiet. Return zwraca przewidywane_etykiet.

2.2 One vs all

One vs all podobnie jak one vs one dzieli problem wieloklasowy na problemy binarne. Biorąc pod uwagę nasz problem Irisów, gdzie występują trzy klasy, one vs all podzieli problem wieloklasowy na trzy problemy binarne. Każda z klas (z wcześniejszym zapamiętaniem ich) podzielona zostanie na klasę a i klasę b. W pierwszej iteracji porównywane są etykiety 0 z 1 i 2, gdzie 0 będzie klasą a (wybór jej da 1), natomiast 1 i 2 klasą B (wybór jej da -1). Czyli klasyfikator binarny trenowany jest na danych, gdzie pozytywną klasą jest jedna z klas, a negatywną pozostałe klasy. Następnie tworzona jest macierz wag, gdzie znajdują się wybory danych klas, na postawie macierzy wag tworzona jest predykcja etykiety.
Klasa one_vs_all dziedziczy po BaseEstimator z biblioteki scikit-learn. Metoda init dziedziczy po self i inicjalizuje listę klasyfikatorów.
Metoda fit dziedziczy po self, X, y. Inicjalizuje tablice self.classes która zawiera unikalne klasy w zbiorze etykiet, funkcja flatten spłaszcza tablice. Obliczana jest ilość klas na podstawie unikalnych etykiet. Następuje iteracja po unikalnych etykietach, binarna_etykieta = jeżeli etykieta należy do etykieta_klasy to 1 jeżeli należy do każdej innej to -1.Następnie w fit następuje trenowanie klasyfikatora na podstawie danych X i etykiet binarnych. Para etykieta, klasyfikator dodawana jest do listy classifiers.
Metoda predict dziedziczy po self i X_test. Metoda wybiera tą predykcje, która dostała największe wsparcie. Inicjalizowana macierz o wymiarach ilość próbek na ilość klas wypełniona zerami – przechowywane w niej będzie wsparcie dla klas. Iteracja po wszystkich klasach, przypisanie wartości z listy classifiers[i] do zmiennych etykieta_klasy i clf, następnie obliczenie wsparcia dla danych testowych, macierz_wsparc[:,i] przypisuje wartość dla danej klasy do odpowiedniej kolumny macierzy wsprcia. Przewidywanie_etykiety za pomocą metody argmax zwraca index(etykiete) elementu o największej wartości. Następnie return zwraca przewidywanie_etykiet.

3 Projekt eksperymentów – etapy 3 oraz 4

3.1 Protokół eksperymentalny

W celu stabilizowania informacji o uzyskanych rezultatach zostanie wykorzystana walidacja krzyżowa. Walidacja krzyżowa dzieli dane na k zbiorów. Następnie z tych podzbiorów zostaje wybrany zbiór testowy, a pozostałe podzbiory tworzą zbiór treningowy. Operacja zostaje powtórzona k-krotnie aż każdy podzbiór będzie pełnił rolę zbioru testowego. Dzięki wykorzystaniu walidacji wiemy, że poprawność uczenia nie będzie testowana na tych samych danych co następowało uczenie. Walidacja pomaga w jak najlepszym oszacowaniu predykcji modelu.

3.2 Metryki oceny wyników

- dokładność (accuracy_score) -umożliwia zmierzenie dokładności, pokazuje, ile procent wyników zostało poprawnie przewidzianych, ukazuje stosunek poprawnie przewidzianych prognoz do wszystkich
- precyzja (precision_score) – bazuje na macierzy pomyłek sprawdza, pozwala określić, ile dobrze przewidzianych prognoz jest przewidzianych z dobrych powodów, jest to stosunek przypadków dobrze przewidzianych z dobrych powodów do wszystkich dobrze przewidzianych prognoz
- czułość (recall_score) – określa na podstawie macierzy pomyłek, ile przypadków zostało faktycznie dobrze przewidzianych, jest to stosunek przypadków dobrze przewidzianych z dobrego powodu do sumy przewidzianych dobrze i źle z dobrego powodu
- f1 (f1_score) – jest to zależność pomnożonego przez dwa iloczynu precyzji i czułości do sumy precyzji i czułości, pozwala ona na uśrednienie wartości i obserwacje wyników bliskich zeru

3.3 Wykorzystywane testy statystyczne

- Test T-studenta -prosty test służący do określenia istotnej zależności statystycznej pomiędzy metodami. Posiada paramerty p i t. Parametr p większy od 0.05 określa brak istotnej różnicy statystycznej pomiędzy wynikami, a w przeciwnym przypadku występuje istotna różnica statystyczna.

3.4 Opis poszczególnych eksperymentów wraz z ich celami

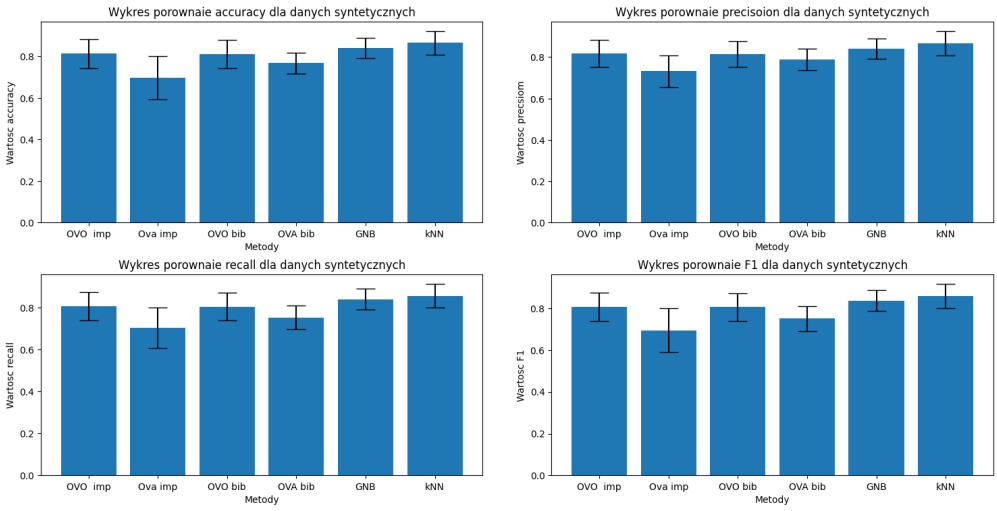
Wyznaczenie metryk oceny takich jak acurracy, precision, recall i F1 - porównanie ich z wynikami metryk klasyfikatorów z biblioteki sklearn.

3.5 Opis rzeczywistych zbiorów danych wraz z metodami ich pozyskania lub sposobów generowania danych syntetycznych

Dane rzeczywiste pozyskane z Iris.csv, przedstawiają one zależności pomiędzy kwiatami. Dane wykorzystywane do badań zostaną wygenerowane w sposób syntetyczny z wykorzystaniem biblioteki scikit-learn za pomocą funkcji make_classification.

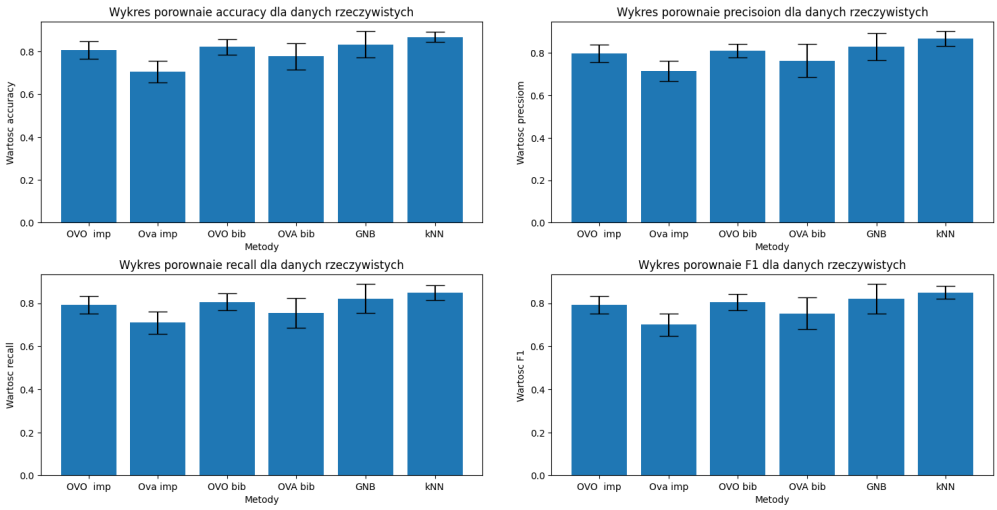
4 Interpretacja uzyskanych wyników

4.1 Wyniki poszczególnych metryk przedstawione na wykresach



Rysunek 1: Porównanie metryk danych syntetycznych

Na rysunku 1 przedstawiono porównanie metryk: accurancy, precision, recall i F1 dla danych syntetycznych dla OVO, OVA, OVO z biblioteki sklearn, OVA z biblioteki sklearn, GNB i KNN.



Rysunek 2: Porównanie metryk danych rzeczywistych

Na rysunku 2 przedstawiono porównanie metryk: accurancy, precision, recall i F1 dla danych rzeczywistych dla OVO, OVA, OVO z biblioteki sklearn, OVA z biblioteki sklearn, GNB i KNN.

Analizują rysunek numer 1 i 2 można zaobserwować, że wśród algorytmów zaimplementowanych w ramach projektu najwyższe accuracy posiada one vs one dla danych rzeczywistych, co oznacza, że właśnie dla tego klasyfikatora najwięcej próbek zostało poprawnie zaklasyfikowanych. Ten klasyfikator również przyjmuje najwyższe wartości metryk takich jak recall, f1 oraz precision. Największe odchylenia standardowe przyjmują wartości dla one vs all. Biorąc pod uwagę wszystkie analizowane klasyfikatory najlepiej sprawdza się KNN, który posiada największe wartości metryk.

4.2 Wyniki metryk przedstawione w tabeli dla każdego analizowanego klasyfikatora

Tabela 1: Prezentacja jakości metryk w formie tabeli po przeprowadzeniu walidacji krzyżowej

Metoda	Accuracy	Std Accuracy	Precision	Std Precision	Recall	Std Recall	F1	Std F1
One vs one dane syntetyczne	0.813	0.070	0.817	0.065	0.807	0.067	0.805	0.068
One vs one dane rzeczywiste	0.807	0.041	0.796	0.042	0.792	0.042	0.792	0.041
One vs all dane syntetyczne	0.697	0.104	0.732	0.077	0.704	0.096	0.694	0.106
One vs all dane rzeczywiste	0.705	0.051	0.714	0.048	0.710	0.052	0.700	0.052
One vs one z biblio dane syntetyczne	0.811	0.068	0.815	0.062	0.805	0.067	0.804	0.067
One vs one z biblio dane rzeczywiste	0.821	0.036	0.810	0.032	0.806	0.040	0.805	0.037
One vs all z biblio dane syntetyczne	0.768	0.051	0.787	0.052	0.754	0.055	0.750	0.061
One vs all z biblio dane rzeczywiste	0.777	0.061	0.764	0.077	0.756	0.069	0.753	0.074
GNB z biblio dane syntetyczne	0.840	0.048	0.841	0.048	0.841	0.050	0.835	0.049
GNB z biblio dane rzeczywiste	0.832	0.061	0.829	0.064	0.822	0.066	0.821	0.068
KNN z biblio dane syntetyczne	0.865	0.056	0.867	0.057	0.856	0.056	0.857	0.057

W tabeli 1 zostały przedstawione wyniki metryk dla wszystkich klasyfikatorów z zaokrągleniem do 3 miejsca po przecinku. Najlepsza skuteczność w przewidywaniu klas należy do algorytmu KNN, posiada on też największą wartość precyzji, recall i f1. Porównując zdefiniowane alorytmy ovo i ova z algorytmami z bilbioteki sklearn można zauważyć podobne wartości wyników. Dla one vs one zarówno dla danych rzeczywistych jak i syntetycznych wyniki są do siebie podobne.W przypadku algorytmu one vs all występuję różnica wartościach wynosząca około 7 punktów procentowych. Posiadają one jednak spore odchylenie standardowe, co oznacza znaczące zróżnicowanie wyników. Różnice mogą wynikać z odmiennego sposobu implementacji algorytmów w trakcie projektu w porównaniu z algorytmami z biblioteki sklearn.

4.3 Wynki testów T studenta dla danych syntetycznych w formie tabeli

Tabela 2: Test-T studenta wartość parametru p dla metryki accuracy dane syntetyczne zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.00041	0.681	0.035	0.104	0.021
Ova imp	0.00041	x	0.00021	0.00052	0.00038	0.00092
OVO bib	0.681	0.00021	x	0.022	0.177	0.036
OVA bib	0.035	0.00052	0.022	x	0.0092	0.0075
GNB	0.104	0.00038	0.177	0.0092	x	0.214
kNN	0.021	0.00092	0.036	0.0075	0.214	x

Tabela 3: Test-T studenta wartość parametru p dla metryki precision dane syntetyczne zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.00015	0.522	0.140	0.075	0.018
Ova imp	0.00015	x	0.00006	0.00137	0.00003	0.00064
OVO bib	0.522	0.00006	x	0.094	0.140	0.038
OVA bib	0.140	0.00137	0.094	x	0.017	0.023
GNB	0.075	0.00003	0.140	0.017	x	0.221
kNN	0.018	0.00064	0.038	0.023	0.221	x

Tabela 4: Test-T studenta wartość parametru p dla metryki recall dane syntetyczne zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.00045	0.779	0.029	0.039	0.019
Ova imp	0.00045	x	0.00024	0.0014	0.00033	0.00098
OVO bib	0.779	0.00024	x	0.020	0.067	0.030
OVA bib	0.029	0.0014	0.020	x	0.0033	0.0052
GNB	0.039	0.00033	0.067	0.0033	x	0.339
kNN	0.019	0.00098	0.030	0.0052	0.339	x

Tabela 5: Test-T studenta wartość parametru p dla metryki F1 dane syntetyczne zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.00064	0.714	0.028	0.048	0.018
Ova imp	0.00064	x	0.00036	0.00083	0.00036	0.00106
OVO bib	0.714	0.00036	x	0.018	0.089	0.030
OVA bib	0.028	0.00083	0.018	x	0.00403	0.00538
GNB	0.048	0.00036	0.089	0.00403	x	0.262
kNN	0.018	0.00106	0.030	0.00538	0.262	x

W tabelach 2,3,4,5 przedstawiono wyniki testów T studenta dla metod porównywanej każda z każdą dane symtetyczne. Tabele przedstawiają wartość parametru p dla wszystkich wyznaczanych metryk. Kolorem niebieskim w tabelach zaznaczono wartości większe od przyjętego poziomu istotności $\alpha = 0.05$. Jeśli parametr$p > \alpha$ oznacza to, że nie można stwierdzić, że pomiędzy grupami istnieją istotne różnice statystyczne. W przeciwnej sytuacji $p < \alpha$ oznacza to, że istnieje istotna różnica statystyczna pomiędzy wynikami. Analizując tabele można zaobserwować pary pomiędzy, którymi nie występują istotne różnice statystyczne. Dla wszystkich metryk są to one vs one z biblioteki z one vs one zaimplementowanym, one vs one z biblioteki z GaussianNB oraz GaussianNB z KNeighborsClassifier. Dla accuracy dodatkowo: one vs one implementowany z GaussianNB, dla precission: one vs one implementowany z GaussianNB, one vs all z bibilioteki z one vs one implementowanym, one vs all z bibilioteki z one vs one z biblioteki.

4.4 Wynki testów T studenta dla danych rzeczywistych w formie tabeli

Tabela 6: Test-T studenta wartość parametru p dla metryki accuracy dane rzeczywiste zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.000000729	0.464	0.002	0.454	0.037
Ova imp	0.000000729	x	0.00000206	0.00007	0.000000106	0.000000563
OVO bib	0.464	0.00000206	x	0.005	0.287	0.033
OVA bib	0.002	0.00007	0.005	x	0.001	0.001
GNB	0.454	0.000000106	0.287	0.001	x	0.107
kNN	0.037	0.000000563	0.033	0.001	0.107	x

Tabela 7: Test-T studenta wartość parametru p dla metryki precision dane rzeczywiste zaokrąglone

Metoda	**OVO imp**	**Ova imp**	**OVO bib**	**OVA bib**	**GNB**	**kNN**
OVO imp	x	0.000001658	0.378	0.005	0.256	0.027
Ova imp	0.000001658	x	0.000001694	0.0002	0.00000079	0.00000971
OVO bib	0.378	0.000001694	x	0.00347	0.400	0.070
OVA bib	0.005	0.0002	0.00347	x	0.00212	0.00104
GNB	0.256	0.00000079	0.400	0.00212	x	0.127
kNN	0.027	0.00000971	0.070	0.00104	0.127	x

Tabela 8: Test-T studenta wartość parametru p dla metryki recall dane rzeczywiste zaokrąglone

Metoda	OVO imp	Ova imp	OVO bib	OVA bib	GNB	kNN
OVO imp	x	0.00000533	0.476	0.000146	0.502	0.0671
Ova imp	0.00000533	x	0.00000863	0.0000405	0.000000942	0.0000102
OVO bib	0.476	0.00000863	x	0.000201	0.390	0.0592
OVA bib	0.000146	0.000405	0.000201	x	0.000288	0.000577
GNB	0.502	0.000000942	0.390	0.000288	x	0.125
kNN	0.0671	0.0000102	0.0592	0.00577	0.125	x

Tabela 9: Test-T studenta wartość parametru p dla metryki F1 dane rzeczywiste zaokrąglone

Metoda	OVO imp	Ova imp	OVO bib	OVA bib	GNB	kNN
OVO imp	x	0.00000101	0.512	0.00165	0.487	0.0474
Ova imp	0.00000101	x	0.00000246	0.000356	0.000000199	0.000000907
OVO bib	0.512	0.00000246	x	0.00111	0.351	0.045
OVA bib	0.00165	0.000356	0.00111	x	0.000805	0.000689
GNB	0.487	0.000000199	0.351	0.000805	x	0.115
kNN	0.0474	0.000000907	0.045	0.000689	0.115	x

W tabelach 6,7,8,9 przedstawiono wyniki testów T studenta dla metod porównywanej każda z każdą dane rzeczywiste. Tabele przedstawiają wartość parametru p dla wszystkich wyznaczanych metryk. Kolorem niebieskim podobnie jak w przypadku danych syntetycznych oznaczono wartości p, które osiągają większe wartości niż zakładany poziom istotności.

Analizując wyniki testów t-studenta można zaobserwować pary pomiędzy, którymi nie występują istotne różnice statystyczne. Dla wszystkich metryk są to one vs one z biblioteki z one vs one zaimplementowanym, one vs one z biblioteki z GaussianNB, GaussianNB z KNeighborsClassifier, oraz one vs one zaimplementowane oraz GaussianNB. Dla metryki precision jest dodatkowo para KNeighborsClassifier z one vs one z bilbioteki, dla recall: KNeighborsClassifier z one vs one implementowany oraz KNeighborsClassifier z one vs one z bilbioteki.

5 Wnioski-etap 8

Celem projektu było zrealizowanie implementacji i opisu algorytmu Support Vector Machines (SVM) z liniową funkcją bazową dla problemów wieloklasowych. Dokładnie przeanalizowaliśmy i zastosowaliśmy implementację metody one-vs-one i one-vs-all z zastosowaniem zdefiniowanego liniowego SVMa.

Z zastosowanych eksperymentów wynika, że zdefiniowany przez nas algorytm one-vs-one ma accuracy na poziomie 0,811 dla danych syntetycznych (150 próbek) i 0,807 dla danych rzeczywistych (150 próbek), w porównaniu z one-vs-one z biblioteki sklearn.multiclass.OneVsOneClassifier, który accuracy ma na poziomie 0,813 dla danych syntetycznych i 0,821 dla danych rzeczywistych. Różnica w wartościach wynosi około 3 procent. Odchylenia standardowe i resztę metryk przedstawiliśmy w tabeli 1.

Natomiast zdefiniowany one-vs-all osiąga wyniki dla accuracy w wartościach 0,697 dla danych syntetycznych i 0,705 dla danych rzeczywistych, porównując go z one-vs-all z biblioteki sklearn.multiclass.OneVsRestClassifier który osiąga wyniki 0,768 dla danych syntetycznych i 0,777 dla danych rzeczywistych. Różnica w wartościach wynosi około 10 procent. Odchylenia standardowe i resztę metryk przedstawiliśmy w tabeli 1.
Porównując metody referencyjne najlepiej wypada algorytm KNN osiągając accurancy na poziomie 0,863 dla danych syntetycznych i 0,868 dla danych rzeczywistych.

6 Repozytorium

Postępy projektu były na bieżąco aktualizowane w repozytorium na platformie github.
Link do repozytorium: https://github.com/MitFar/Project_SVM

Literatura

[1] Applications of Support Vector Machine (SVM) Learning in Cancer Genomics SHUJUN HUANG1,2, NIANGUANG CAI 2, PEDRO PENZUTI PACHECO2, SHAVIRA NARRANDES2,3, YANG WANG4 and WAYNE XU1,2,3

[2] Introduction to Machine Learning with Python By Andreas C. Müller and Sarah Guido

[3] Multiclass Classification with Support Vector Machines (SVM), Dual Problem and Kernel Functions by Hucker Marius

[4] A comparison of numerical optimizers for logistic regression, Thomas P. Minka

[5] Support Vector Machines for Multi-Class Pattern Recognition, J. Weston and C. Watkins

[6] https://www.ibm.com/docs/pl/spss-modeler/saas?topic=models-about-svm

[7] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[8] In Defense of One-Vs-All Classification Ryan Rifkin, Aldebaro Klautau