Sprawozdanie - SI

Lab 4

Wiktor Sadowy

Cel zadania

W ramach zadania mieliśmy:

a) dokonać analizy danych
b) przygotować dane wykorzystując różne metody. W ramach zadania wykonaliśmy też zadanie bonusowe i przygotowaliśmy metody radzenia sobie z brakującymi danymi
c) przetestować klasyfikatory z użyciem różnych zestawów hiperparametrów. W ramach zadania wykonaliśmy też zadanie bonusowe i przetestowaliśmy bardziej zaawansowane algorytmy np. Las Losowy czy SVM
d) ocenić klasyfikację z pomocą różnych metryk oceny klasyfikacji i zinterpretować wyniki

Wszystkie zadania miały zostać wykonane z użyciem datasetu UCI – GLASS.

Opis teoretyczny

W zadaniu zajmujemy się uczeniem nadzorowanym. Jest to sposób uczenia w którym zbiór danych treningowych zawiera dołączone rozwiązanie problemu tzw. etykiety albo klasy. Dzięki temu możemy rozwiązywać problem regresji (przewidywanie wartości) i problem klasyfikacji (przewidywanie klas).

W ramach zadania podzieliśmy musieliśmy podzielić nasz zbiór na dwie części:

a) Zbiór uczący – zestaw danych używany do nauki algorytmu
b) Zbiór walidacyjny – zestaw danych używany do przeprowadzania testów modelu. Testy są przeprowadzone po to, aby jak najlepiej dopasować hiperparametry modelu. Dzięki użyciu zbioru walidacyjnego jesteśmy w stanie sprawdzić czy model się generalizuje i poradzi sobie na nowych danych. Ważne jest to, żeby dane w zbiorze walidacyjnym nie były używane w zbiorze uczącym. W przeciwnym przypadku nie jest możliwa obiektywna ocena działania modelu

W problemach regresji i klasyfikacji często się też dodaje kolejny zbiór danych czyli zbiór testowy. Dane w tym zbiorze nie są używane ani w zbiorze uczącym ani zbiorze walidacyjnym i ich celem jest przetestowanie działania modelu na nowych danych i ocena czy prawidłowo wybraliśmy model i dopasowaliśmy do niego hiperparametry

Mogliśmy też zastosować walidację krzyżową, która jest alternatywnym sposobem przygotowania zestawów treningowych i testowych dla modelu. Metoda ta polega na podzielenie zestawu danych na podzestawy i utworzeniu z nich zestawów danych walidacyjnych i treningowych.

Jest wiele rodzajów walidacji krzyżowej np. k-krotna walidacja krzyżowa, która polega na tym, że dzielimy próbki na k grup. Model jest trenowany na k-1 grupach i walidowany na jednej grupie. Po wykonaniu k treningów wyniki przeprowadzonych walidacji są uśredniane.

W ramach zadania wykorzystujemy następujące modele:

a) Naiwny klasyfikator Bayesa – prosty klasyfikator probabilistyczny, który naiwnie zakłada niezależność i tą samą istotność cech dla ustalonej etykiety klasy. Naiwny klasyfikator Bayesa opiera się na twierdzeniu Bayesa. Dużą zaletą algorytmu jest jego prostota.
b) Drzewo decyzyjne – jest to reprezentacja klasyfikatora w postaci drzewa wspomagającego proces decyzyjny. Drzewo jest budowane w oparciu o różne algorytmy i parametry (np. entropia lub przyrost wiedzy), które są wyliczane dla każdego nowego węzła drzewa decyzyjnego. Drzewa decyzyjne poprzez swoją strukturę są bardzo podatne na zjawisko nadmiernego dopasowania gdyż wraz ze wzrostem głębokości coraz lepiej dostosowują się do zbioru uczącego, ale równocześnie pogarsza się ich możliwość generalizacji przez co gorzej radzą sobie ze zbiorem testowym
c) Las losowy – jest to algorytm, który używa wiele drzew decyzyjnych. Każde drzewo jest trenowane na innej części danych. Wówczas mamy gwarancję, że będziemy mieli różne drzewa. Wynikiem predykcji lasu losowego jest najczęściej przewidywana klasa przez drzewa decyzyjne. Zaletą lasu losowego jest to, że wykorzystuje on silne strony drzew decyzyjnych, jednocześnie zmniejszając ryzyko nadmiernego dopasowania
d) Klasyfikator wektorów nośnych – jest to algorytm, który dzieli przestrzeń w którym są nasze dane za pomocą funkcji (może być to np. funkcja liniowa czy wielomianowa) i na podstawie podziału dokonuje predykcji klasy. W przypadku gdy mamy do czynienia z dwoma klasami wystarczy jedna funkcja do podziału przestrzeni. W przypadku gdy mamy do czynienia z większą liczbą klas trzeba utworzyć funkcję oddzielającą klasę od każdej innej klasy (czyli dla 3 klas będą to 3 funkcje, dla 4 będzie to 6 funkcji itd.). Zaletą algorytmu jest to, że działa on bardzo dobrze gdy możemy bezproblemowo odróżnić od siebie klasy w oparciu o atrybuty lub gdy liczba atrybutów jest większa od liczby próbek. Wadą modelu jest to, że jest on bardzo podatny na różnice w wielkościach atrybutów (np. jeżeli jeden atrybut mieści się w przedziale 1-3, a drugi w przedziale 100000-300000 to model może mieć problemy z dopasowaniem się do danych). Oznacza to, że trzeba przed trenowaniem przetworzyć dane używając np. standaryzacji.

Przykładowe zastosowanie

Uczenie maszynowe możemy użyć do rozwiązywania różnych problemów regresji lub klasyfikacji np. przewidzenie czy dany mail jest spamem czy nie. Dla danego przykładu stosowanie wytrenowanych modeli pozwala na oszczędzenie czasu, gdyż nie musimy wówczas ręcznie decydować czy dany mail jest spamem czy nie.

Implementacja

Podczas wykonywania zadania korzystaliśmy z Jupyter Notebook. Po kolei wykonywaliśmy kolejne zadania, które były na liście. Podczas implementacji naszym głównym celem było napisanie takiego kodu, który pozwalałby na łatwe testowanie kolejnych modeli.

Materiały dodatkowe

Do wykonania zadania została użyta instrukcja do zadania, dokumentacja scikit-learn - https://scikit-learn.org/stable/index.html oraz poniższe strony:

https://www.makeuseof.com/fill-missing-data-with-pandas/

https://medium.com/synthesio-engineering/precision-accuracy-and-f1-score-for-multi-label-classification-34ac6bdfb404

Opis wykorzystanych bibliotek

Do zadania zostały wykorzystane następujące wbudowane biblioteki:

a) copy – biblioteka pozwalająca na kopiowanie zmiennych

Oraz następujące dodatkowe biblioteki:

a) pandas – biblioteka pozwalająca na wczytanie pliku z danymi. Używając biblioteki jesteśmy w stanie przetwarzać dane zawarte w pliku np. wypełnić brakujące dane
b) matplotlib – biblioteka służąca do tworzenia wykresów
c) seaborn – biblioteka oparta o matplotlib pozwalają na tworzenie wykresów. W porównaniu do matplotlib tworzenie wykresów jest o wiele prostsze
d) sklearn – biblioteka pozwalająca na użycie metod związanych z uczeniem maszynowym

Analiza danych

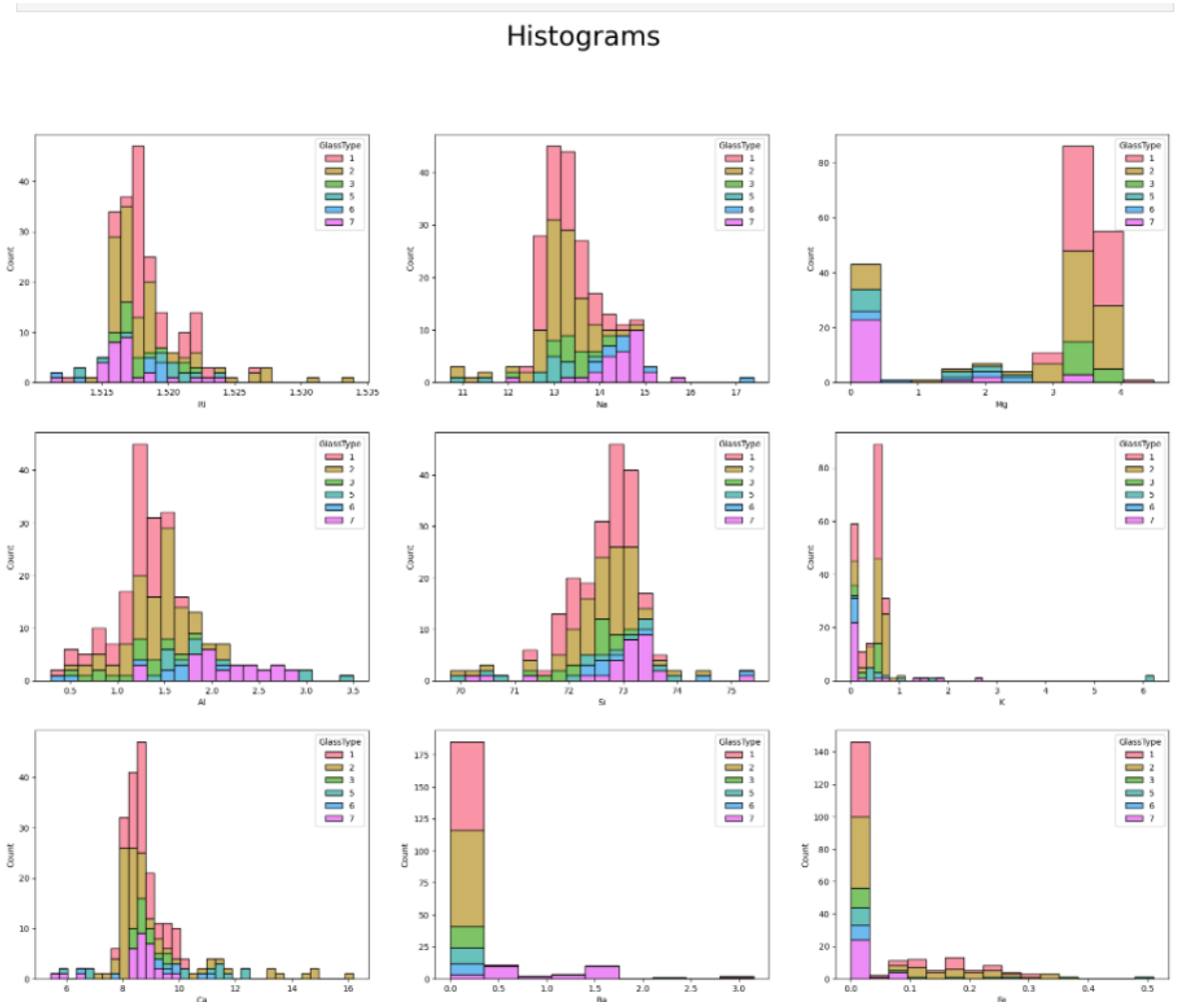
Wszystkie wykresy są dostępne w notebooku udostępnionym wraz z zadaniem.

Nasz zbiór danych składa się z 214 próbek i 10 atrybutów z czego naszym celem jest przewidzenie 10 atrybutu – typu szkła.

W danych nie ma żadnych brakujących wartości. Nie musimy więc zastanawiać się jak poradzić sobie z brakującymi danymi.

Dataset składa się z 6 typów szkła: 1, 2, 3, 5, 6, 7. W pliku zawierającym opis klas jest zawarty jeszcze typ szkła numer 4, ale nie jest obecny w datasecie.

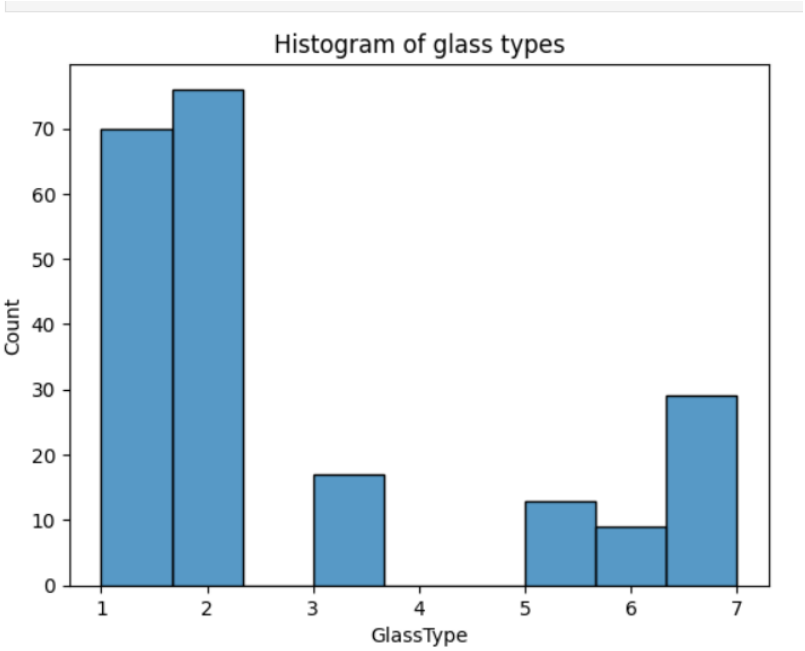
Poniżej znajduje się dystrybucja danych (histogramy)



Patrząc na histogramy możemy zauważyć kilka rzeczy:

a) Atrybuty mają różne przedziały wartości np. wartość Mg mieści się w przedziale 0-5, a wartość Na mieście się w przedziale 11-17
b) Widzimy, że część atrybutów jest zróżnicowana np. Na. Są też atrybuty niezróżnicowane np. Ba czy Fe gdzie w większości przypadków wartość tych atrybutów jest równa 0

Zrobiliśmy też histogram typów szkła

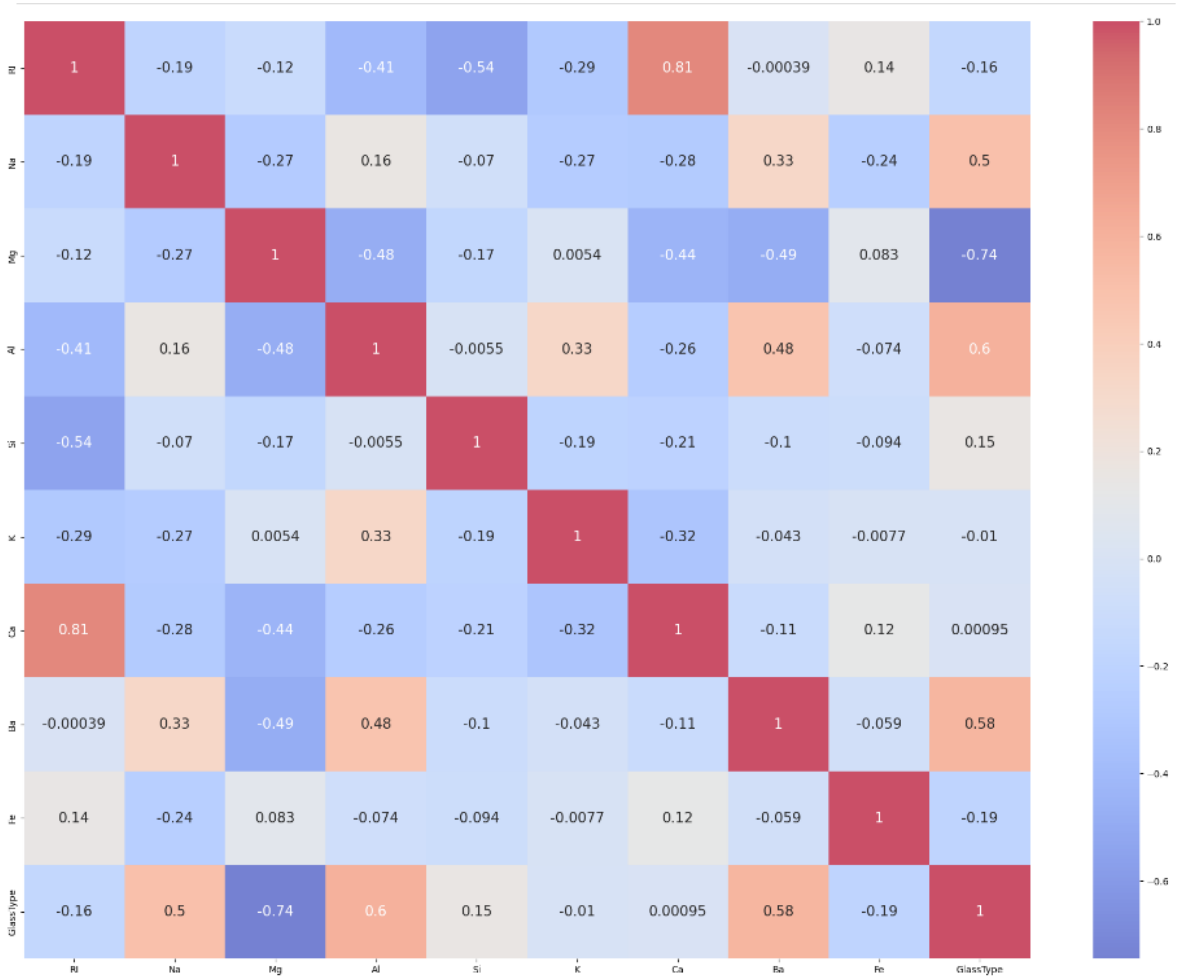


Widzimy po histogramie, że nasze klasy są niezbalansowane. Mamy więcej szkieł typu 1 i 2 niż pozostałych. Takie niezbalansowanie może wpłynąć na efektywność modelu gdyż model ma więcej danych, żeby nauczyć się rozpoznawać szkło typu 1 niż np. szkło typu 3

W ramach analizy stworzyliśmy tzw. regploty (czyli wykresy przedstawiającą dwa atrybuty i linię najlepszego dopasowania). Patrząc na wykresy jesteśmy w stanie zauważyć, że np. atrybuty Ca i RI są mocno skorelowane ze sobą. Gdy rośnie wartość atrybutu Ca rośnie też wartość atrybutu RI. Możemy więc zastanawiać się czy jest sens używać obu atrybutów do predykcji typu szkła skoro są one tak mocno zależne od siebie.

Kolejnym wykresem jaki wykonaliśmy był tzw. pairplot. Wykres przedstawia relację między danymi parami atrybutów. Widzimy, że na podstawie wartości dwóch atrybutów trudno jest ustalić typ szkła. To oznacza, że problem predykcji typu szkła jest bardziej złożony i wymaga wielu atrybutów.

Ostatnim wykresem, który zrobiliśmy był tzw. heatmap, który pozwala prześledzić w jaki sposób dane są skorelowane ze sobą



Analizując heatmap widzimy, że dla części atrybutów mamy pozytywną korelację np. Na. Oznacza to, że gdy rośnie wartość Na to rośnie też wartość typu szkła. Są też wartości z ujemną korelacją np. Mg. To oznacza, że tym wyższa wartość Mg tym niższy typ szkła. Analizując heatmap jesteśmy w stanie zauważyć, że najwięcej danych o typie szkła przekażą nam Na, Mg, Al i Be. Są też atrybuty, które mają korelację równą 0. Oznacza to, że dany atrybut nie jest powiązany liniowo z żadnym innym atrybutem.

Dokonując analizy danych możemy wyciągnąć następujące wnioski:

a) Nie ma brakujących danych, więc nie musimy martwić się o ich wypełnienie
b) Klasy są niezbalansowane. Może to wpłynąć na efektywność modelu
c) Niektóre atrybuty są niezróżnicowane. Atrybuty, które głównie przyjmują jedną wartość mogą być nieprzydatne do predykcji typu szkła
d) Niektóre atrybuty są liniowo powiązane z wartością typu szkła, gdy inne nie. Możemy się więc spodziewać, że atrybuty z wysoką dodatnią lub ujemną korelacją mogą być w pierwszej kolejności używane do ustalenia z jakim typem szkła możemy mieć do czynienia.
e) Niektóre atrybuty są powiązane ze sobą liniowo. Mając np. wartość Ra może nie być nam potrzebna wartość Ca ze względu na wysoką korelację obu atrybutów

Jak przygotowane dane wpływają na predykcję

W ramach zadania przygotowaliśmy 5 typów danych:

a) Nieprzeprocesowane dane
b) Znormalizowane dane
c) Standaryzowane dane
d) Dane, które zostały przetworzone przy użyciu PCA
e) Dane z których zostało usunięte 5% wartości danych. Dane zostały uzupełnione poprzez użycie mediany wartości atrybutów.

Do porównywania danych użyliśmy domyślnych modeli (modeli z domyślnymi wartościami hiperparametrów). Jako metrykę oceny wybraliśmy F1-score.

Poniżej znajdują się wyniki.

	NaiveBayes	DecisionTree	RandomForest	SVM
NotProcessedData	0.548325	0.750921	0.753895	0.083276
NormalizedData	0.507553	0.579623	0.701961	0.083276
StandardizedData	0.548325	0.750921	0.753895	0.545659
PCATransformData	0.228148	0.529808	0.442094	0.298568
FilledMissingData	0.444242	0.671892	0.753895	0.085043

Z rzeczy wartych zauważenia należy wymienić:

a) Duży wpływ przetworzenia danych na efektywność modelu w przypadku SVM. Tak jak było wcześniej wspomniane przy dużych różnicach w wielkościach atrybutów model ten nie najlepiej sobie radzi. Przy danych standaryzowanych nie ma żadnych różnic w wielkości, co też przyczynia się do poprawy efektywności modelu
b) Nawet przy brakujących danych modele są w stanie sobie dobrze poradzić z danymi, ale widać spadek w efektywności. Jest to spowodowane tym, że mediana nie jest najlepszą metodą na wypełnienie brakujących danych. Dodatkowo niektóre modele np. drzewo decyzyjne są bardzo podatne na zmiany w danych
c) Nie każde przetworzenie danych przełoży się na zwiększoną efektywność. W przypadku źle przetworzonych danych możemy nawet odnotować spadek efektywności modelu. Przykładem tego jest użycie PCA i próba zmniejszenia wymiaru danych do 3.
d) Warto zauważyć, że tylko w przypadku SVM przetworzenie danych wpłynęło na poprawę efektywności modelu. W przypadku innych modeli nie zauważyliśmy różnic. Wynika to z tego, że tylko SVM jest podatne na różnice w wielkościach atrybutów. Oznacza to, że jeżeli np. stosujemy drzewo decyzyjne to nie ma potrzeby przetwarzania danych

Jak wybór hiperparametrów wpływa na jakość predykcji

W ramach zadania przygotowaliśmy 16 różnych modeli:

a) Naiwny Bayes z domyślnymi hiperparametrami
b) Naiwny Bayes z ustawionymi prawdopodobieństwami dla klas
c) Naiwny Bayes z ustawioną wartością części największej wariancji wszystkich atrybutów, która jest dodawana do wariancji dla stabilności obliczeń
d) Naiwny Bayes z ustawionymi prawdopodobieństwami dla klas i wartością części największej wariancji wszystkich atrybutów, która jest dodawana do wariancji dla stabilności obliczeń
e) Drzewo decyzyjne z domyślnymi hiperparametrami
f) Drzewo decyzyjne ze zmienionym parametrem służącym do mierzenia jakości podziału
g) Drzewo decyzyjne z ustawioną maksymalną głębokością drzewa oraz zwiększoną liczbą próbek potrzebnych do ustalenia podziału / ustalenie czy mamy do czynienia z liściem drzewa
h) Drzewo decyzyjne ze zmienionym parametrem służącym do mierzenia jakości podziału, ustawioną maksymalną głębokością drzewa oraz zwiększoną liczbą próbek potrzebnych do ustalenia podziału / ustalenie czy mamy do czynienia z liściem drzewa
i) Las losowy z domyślnymi hiperparametrami
j) Las losowy ze zmienionym parametrem służącym do mierzenia jakości podziału oraz zwiększoną liczbą drzew decyzyjnych
k) Las losowy z ustawioną maksymalną głębokością drzew decyzyjnych oraz zwiększoną liczbą próbek potrzebnych do ustalenia podziału / ustalenie czy mamy do czynienia z liściem drzewa
l) Las losowy ze zmienionym parametrem służącym do mierzenia jakości podziału, zwiększoną liczbą drzew decyzyjnych, ustawioną maksymalną głębokością drzew oraz zwiększoną liczbą próbek potrzebnych do ustalenia podziału / ustalenie czy mamy do czynienia z liściem drzewa
m) SVM z domyślnymi hiperparametrami
n) SVM ze zmienionym typem kernela (z kernela opartego na radialnej funkcji bazowej na kernela opartego na wielomianie 4-tego stopnia)
o) SVM ze zmienionym typem kernela (z kernela opartego na radialnej funkcji bazowej na kernela opartego na wielomianie 4-tego stopnia) oraz ze zmienioną wartością parametru regularyzacji oraz zmienioną wartością współczynnika kernela
p) SVM ze zmienionym typem kernela (z kernela opartego na radialnej funkcji bazowej na kernela opartego funkcji liniowej)

Do porównania danych użyliśmy danych standaryzowanych oraz wszystkich metryk porównawczych. Dodatkowo porównaliśmy działanie modeli dla danych treningowych i testowych. Poniżej znajdują rezultaty testów.

	Accuracy	Recall	Precision	F1
NaiveBayes Default	0.388889	0.510210	0.607655	0.444242
NaiveBayes Ver1	0.388889	0.510210	0.599802	0.437949
NaiveBayes Ver2	0.444444	0.558418	0.621673	0.532896
NaiveBayes Ver3	0.425926	0.539899	0.575258	0.507367
DecisionTree Default	0.648148	0.615824	0.813341	0.671892
DecisionTree Ver1	0.555556	0.580910	0.686688	0.578067
DecisionTree Ver2	0.481481	0.422354	0.480708	0.416667
DecisionTree Ver3	0.425926	0.414401	0.484575	0.428254
RandomForest Default	0.740741	0.729114	0.866429	0.753895
RandomForest Ver1	0.759259	0.737886	0.870833	0.762919
RandomForest Ver2	0.666667	0.608744	0.858521	0.630556
RandomForest Ver3	0.703704	0.692077	0.885417	0.699878
SVM Default	0.259259	0.175439	0.069909	0.085043
SVM Ver1	0.259259	0.175439	0.085000	0.083276
SVM Ver2	0.555556	0.550835	0.615123	0.552549
SVM Ver3	0.537037	0.499422	0.526111	0.490292

	Train	Test
NaiveBayes Default	0.355147	0.444242
NaiveBayes Ver1	0.348517	0.437949
NaiveBayes Ver2	0.405882	0.532896
NaiveBayes Ver3	0.393144	0.507367
DecisionTree Default	1.000000	0.671892
DecisionTree Ver1	1.000000	0.578067
DecisionTree Ver2	0.664960	0.416667
DecisionTree Ver3	0.610923	0.428254
RandomForest Default	1.000000	0.753895
RandomForest Ver1	1.000000	0.762919
RandomForest Ver2	0.852782	0.630556
RandomForest Ver3	0.855534	0.699878
SVM Default	0.137230	0.085043
SVM Ver1	0.135544	0.083276
SVM Ver2	0.858115	0.552549
SVM Ver3	0.659054	0.490292

Z analizy powyższych tabel możemy wyciągnąć następujące wnioski:

a) Niektóre wersje modeli działają lepiej od domyślnych modeli. Oznacza to że warto modyfikować hiperparametry, żeby utworzyć jak najlepszy model
b) Większość naszych modeli charakteryzują się wysoką precyzją w porównaniu do innych metryk. W niektórych sytuacjach wyższa wartość tej metryki może być przydatna np. budujemy model do predykcji spamu – bardziej nam wówczas zależy na tym, żeby do spamu trafiał tylko spam nawet jeżeli czasami dostaniemy na skrzynkę główną spam niż na tym, żeby do spamu trafiały wszystkie spamy oraz wiadomości, które spamem nie są.
c) Niektóre modele zbytnio dopasowały się do danych np. domyślne drzewo decyzyjne czy losowy las. F1-score równe 1 dla danych treningowych oznacza, że model nauczył się struktury zbioru uczącego. Dobry model będzie w stanie zgeneralizować się tzn. będzie miał podobne rezultaty dla danych treningowych i walidacyjnych. Mając to na uwadze warto się zastanowić czy nie lepiej wybrać czasami model, który osiągnął gorszy wynik dla danych walidacyjnych, ale za to lepiej się zgeneralizował (przykładowo można wybrać losowy las w wersji trzeciej zamiast domyślnego lasu losowego). Wówczas mamy gwarancję, że z nowymi danymi model dobrze sobie poradzi.
d) Warto zauważyć, że niektóre modele lepiej poradziły sobie z problemem niż inne modele np. las losowy lepiej sobie poradził niż drzewo decyzyjne. Nie jest to zaskoczeniem, bo las losowy jest bardziej zaawansowanym algorytmem niż drzewo decyzyjne i dobrze skonfigurowane powinno zwracać lepsze rezultaty. Podczas wyboru optymalnego modelu warto wziąć pod uwagę nie tylko efektywność modelu, ale też jego zaawansowanie. Jeżeli możemy osiągnąć podobne rezultaty mniej zaawansowanym algorytmem to lepiej jest go wybrać niż bardziej zaawansowany algorytm.

Podsumowanie

W ramach zadania zapoznaliśmy się z podstawowymi krokami realizacji projektu opartego o uczenie maszynowe. Z tego ćwiczenia wyciągnęliśmy następujące wnioski:

a) Przed wykonaniem zadania należy podzielić zbiór danych na treningowy i walidacyjny. Inaczej nie będziemy w stanie obiektywnie ocenić efektywności modelu
b) Przetworzenie danych przed trenowaniem wpływa na efektywność modelu. Niektóre modele wymagają przetworzenia danych (np. SVM dobrze poradził sobie tylko dla danych standaryzowanych). Warto jednak zauważyć, że nie każde przetworzenie danych polepszy działanie modelu. Czasami wręcz może ono pogorszyć działanie modelu.
c) Brak danych wpływa na działanie modelu. Najbardziej podatne są modele, które są wrażliwe na wszelakie zmiany w zbiorze treningowym
d) Nie ma optymalnego modelu do każdego problemu. Warto testować kilka modeli, żeby wybrać najbardziej optymalny. Warto uwzględnić podczas wyboru modelu nie tylko to czy osiąga on dobre rezultaty, ale też jego zawiłość. Jeżeli jesteśmy w stanie

mniej zaawansowanym modelem osiągnąć ten sam rezultat co zaawansowanym modelem to lepiej wybrać mniej zaawansowany model. Warto też pamiętać o tym, że celem naszego modelu jest przede wszystkim generalizacja.

e) Warto sprawdzać czy model nie nauczył się struktury danych treningowych (tzn. nie doszło do nadmiernego dopasowania). Takie modele słabo się generalizują i będą miały problemy z nowymi danymi.
f) Warto dostosowywać hiperparametry modelu, bo robiąc to będziemy w stanie uzyskać lepiej działający model niż używając domyślnych hiperparametrów
g) Do oceny modelu możemy używać różnych metryk. W zależności od problemu możemy się koncentrować na określonych metrykach np. precyzji. Wówczas podczas budowy modelu koncentrujemy się na maksymalizacji tej jednej metryki, nawet kosztem obniżenia wartości pozostałych metryk.