Raport z analizy i modelowania zbioru danych "steel-plates-fault

Wstęp

Celem zadania było przeprowadzenie analizy i modelowania wybranego zbioru danych w celu poszerzenie swojej wiedzy związanej z eksploracją danych.

W repozytorium znajdują się nootebook jupyterowy zawierający kod oraz raport, analogiczny raport w formacie .pdf oraz pliki .txt i wykresy w formacie .png uzyskane w trakcie wykonywania programu.

Opis danych

Przy wykonywaniu zadania wykorzystano zbiór danych o nazwie "steel-plates-fault". Zbiór ten zawiera dane o defektach blach stalowych i składa się z 1941 wierszy oraz 34 kolumn. Spośród tych kolumn 27 zawiera zmienne opisujące parametry fizyczne płyty stalowej, 6 zawiera informacje o wystąpieniu częstych defektów (wartość binarna 1-defekt wystąpił, 0-defekt nie wystąpił, przy czym w jednym wierszu tylko jedna z tych kolumn może być niezerowa). Natomiast ostatnia kolumna to klasa występującego defektu, która może przyjąć wartość b'1' lub b'2'. Wartość pierwsza oznacza, że wystąpił jeden z sześciu najczęstszych defektów, natomiast wartość druga oznacza, że wystąpił inny rodzaj defektu. W zbiorze znajduje się 1268 rekordów pierwszego typu i 673 drugiego.

W udostępnionym do pobrania pliku .arff nazwy kolumn są skrótowe, a w związku z tym niejasne. Do celów przetwarzania zmieniono nazwy kolumn, zgodnie z informacjami podanymi na stronie źródłowej.

Źródło:

Dataset provided by Semeion, Research Center of Sciences of Communication, Via Sersale 117, 00128, Rome, Italy.

https://www.openml.org/search?type=data&sort=runs&status=active&id=1504

http://archive.ics.uci.edu/ml/datasets/steel+plates+faults

Opis procesu przygotowania danych do analizy i modelowania

Użyty zbiór danych jest zbiorem dedykowanym do wykorzystania przy uczeniu maszynowym. W związku z tym jest kompletny i poprawny czyli nie brakuje żadnych wartości oraz nie ma wartości nieprawidłowych.

Dane wykorzystano do proponowanego dla tego zbioru zadania czyli klasyfikacji na często występujące defekty oraz inne defekty.

Analiza danych

Najpierw podzielono cały zbiór danych ze względu na klasę odkształcenia. Następnie dla każdej z nich policzono współczynniki korelacji Pearsona oraz Spearmana. Na podstawie uzyskanych wyników zdecydowano się odrzucić kolumny dla których współczynnik korelacji w obu zbiorach był większy od 0,85. Odrzucono 10 kolumn:

- TypeOfSteel_A400
- Y_Maximum
- X_Maximum

- Sum_of_Luminosity
- SigmoidOfAreas
- Orientation_Index
- Luminosity_Index

- LogOfAreas
- Log_X_Index
- Y_Perimeter

W efekcie wymiar wektora wejściowego wykorzystywanego przy modelowania został zredukowany z 27 do 17 (28 do 18 wliczając wartość oczekiwaną).

Następnie na podstawie wartości otrzymanych z funkcji z-score z biblioteki SciPy odrzucono rekordy odstające. Za takowe uznano rekordy, dla których otrzymana wartość była większa od 3. Było ich 182 co stanowi około 9% całego zbioru danych. Po odrzuceniu danych pozostało 1181 rekordów z klasy b'1' oraz 578 z klasy b'2'.

Podjęto również próbę wykorzystania lasu izolacji, ale w takim przypadku odrzucana było około 14% zbioru. Wartość tą uznano za zbyt dużą, w związku z tym wykorzystano wcześniej wspomnianą metodę.

Modelowanie danych

Jako wektor wejściowy X do modelu wykorzystano kolumny, które pozostały po wcześniejszych krokach, przy czym wartości zostały poddane standaryzacji. Wektor wartości oczekiwanych przekształcono na wektor wartości binarnych 0 i 1.

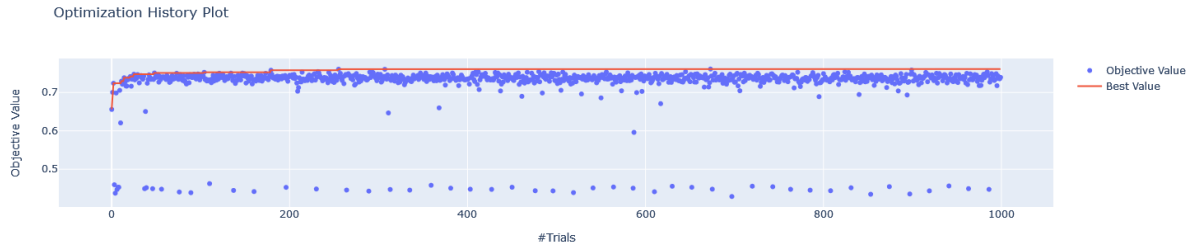
Do modelowania zbioru danych wybrano maszynę wektorów nośnych, a dokładnie SVC z biblioteki scikit-learn. W celu optymalizacji hiperparametrów modelu wykorzystano pakiet optuna. Optymalizowano wartość parametru regularyzacji - C, rodzaj jądra maszyny wektorów nośnych oraz stopień wielomianu. Przy czym stopień wielomianu ma wpływ tylko na jądro typu "poly" czyli wielomianowego, a w pozostałych przypadkach jest ignorowany. Wartością maksymalizowaną jest suma trafności na zbiorze treningowym, precyzji na zbiorze treningowym, miary f1 na zbiorze treningowym, trafności na zbiorze testowym, precyzji na zbiorze testowym oraz miary f1 na zbiorze testowym. Wartości te są sumowane z wagami odpowiednio 0.025, 0.025, 0.2, 0.075, 0.075 i 0.6. Wartości średnie poszczególnych metryk dla każdej z prób zostały zapisane do pliku tekstowego. Przy czym wartości te, to tak naprawdę średnie otrzymane z walidacji krzyżowej ("Cross-validation") z podziałem na 5 części.

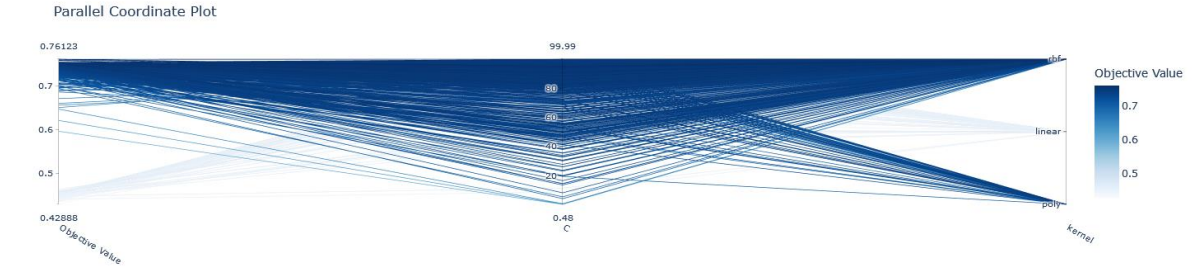
Optunę wykorzystano w sumie trzykrotnie, po 1000 prób za każdym razem.

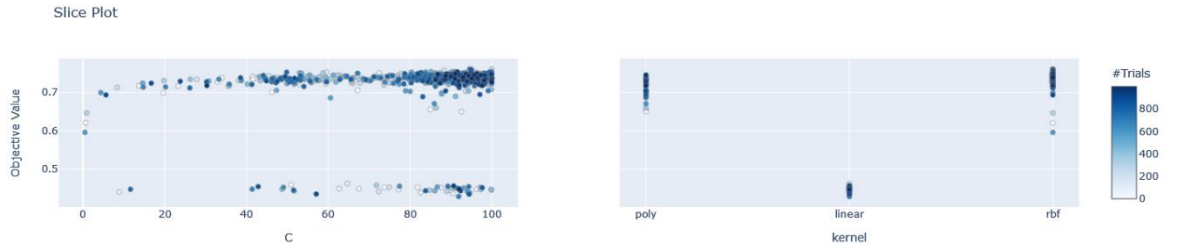
Najpierw sprawdzano wartości dla C z zakresu 0.1 do 100.0, jądra typu "linear", "rbf" lub "poly" oraz stopnia wielomianu z zakresu od 2 do 5. Najlepszą wartość uzyskano dla C = 94.54320781753651, jądra 'rbf' i stopnia = 5, przy czym stopień nie ma znaczenia bo nie jest wykorzystane jądro typu "poly". Uzyskane wartości metryk dla tego przykładu przedstawiono w poniższej tabeli.

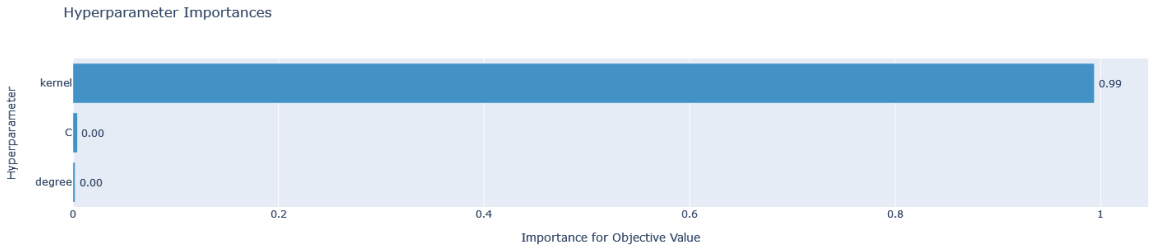
trafność treningowy	**trafność testowy**	**precyzja treningowy**	**precyzja testowy**	**miara f1 treningowy**	**miara f1 testowy**
0.974986	0.778288	0.938107	0.613579	0.960972	0.644733

Dodatkowo wizualizacja otrzymanych wyników:





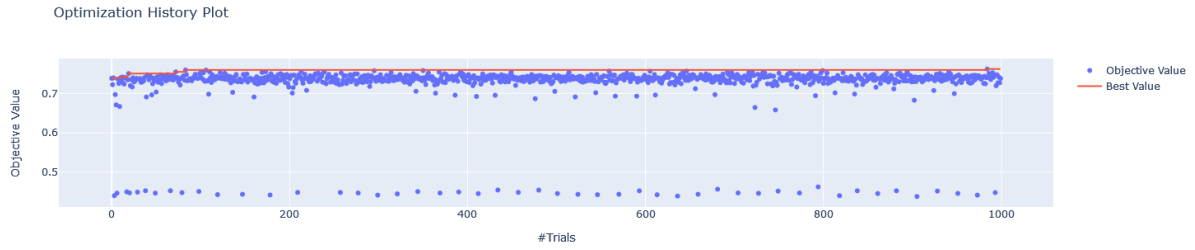


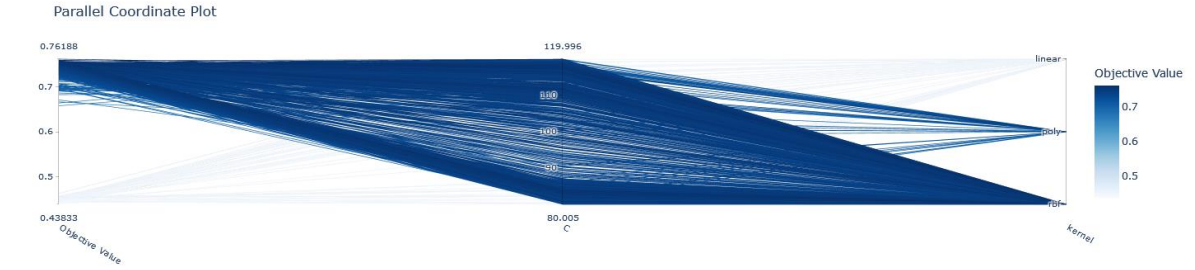


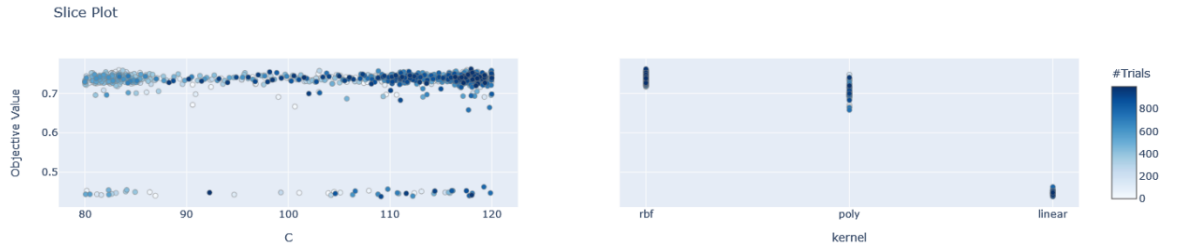
Przy drugiej próbie na podstawie wcześniejszych wyników (Slice Plot) zmieniono zakres przeszukiwania C na od 80.0 do 120.0, a reszta pozostała bez zmian. Najlepszą wartość uzyskano dla C = 117.95849387058576, jądra 'rbf' i stopnia = 4, przy czym stopień nie ma znaczenia bo nie jest wykorzystane jądro typu "poly". Uzyskane wartości metryk dla tego przykładu przedstawiono w poniższej tabeli.

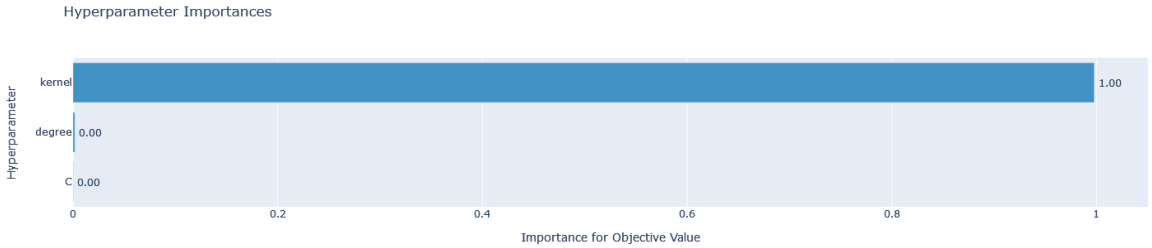
trafność treningowy	**trafność testowy**	**precyzja treningowy**	**precyzja testowy**	**miara f1 treningowy**	**miara f1 testowy**
0.982377	0.793637	0.955512	0.67482	0.972732	0.68126

Dodatkowo wizualizacja otrzymanych wyników:





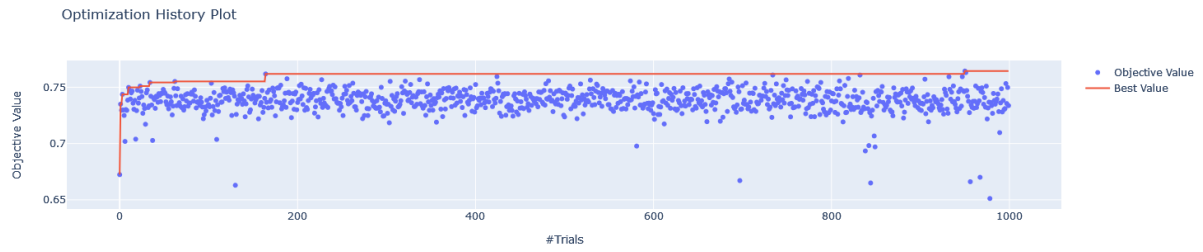


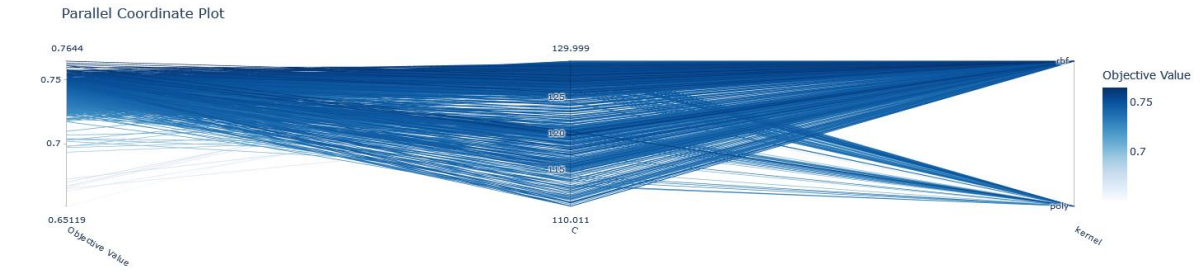


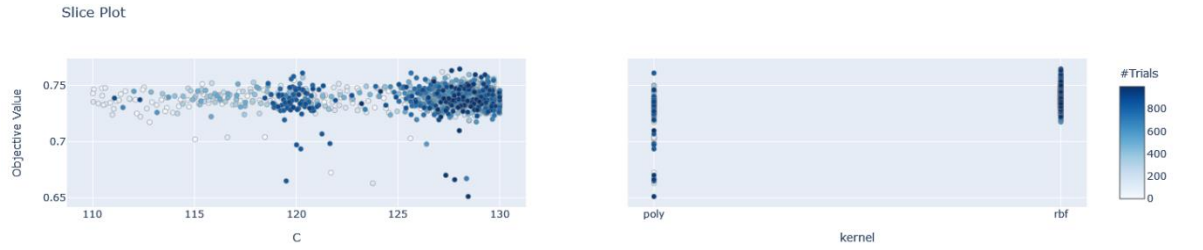
W przypadku trzeciej próby zmieniono zakres przeszukiwania C na od 110.0 do 130.0 oraz przestano brać pod uwagę jądro typu "linear", ze względu na to, że dawało ono wyniki znacznie słabsze niż dwa pozostałe. Reszta ustawień pozostała taka jak poprzednio. Najlepszą wartość uzyskano dla C = 128.0280732226804, jądra 'rbf' i stopnia = 3, przy czym stopień nie ma znaczenia bo nie jest wykorzystane jądro typu "poly". Uzyskane wartości metryk dla tego przykładu przedstawiono w poniższej tabeli.

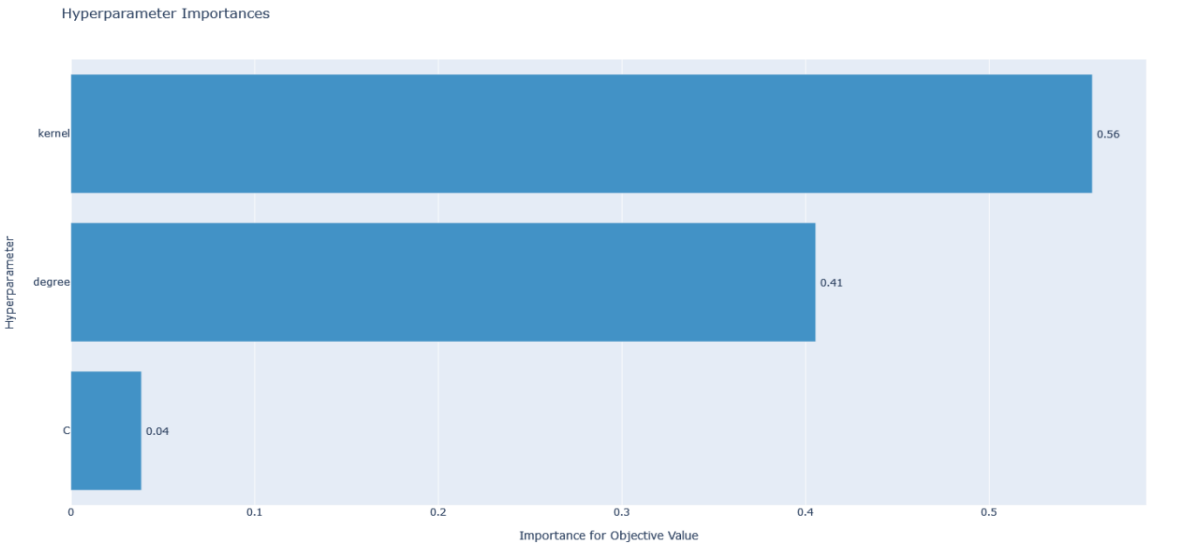
trafność treningowy	**trafność testowy**	**precyzja treningowy**	**precyzja testowy**	**miara f1 treningowy**	**miara f1 testowy**
0.98266	0.79363	0.95844	0.683672	0.973181	0.68407

Dodatkowo wizualizacja otrzymanych wyników:









Ze względu na małą poprawę parametrów, uznano otrzymany wynik za wynik końcowy.

Wnioski

Ostatecznie najlepszy model uzyskano dla C = 128.0280732226804, jądra 'rbf' i stopnia = 3, przy czym stopień nie ma znaczenia bo nie jest wykorzystane jądro typu "poly". Uzyskane wartości metryk są następujące:

trafność treningowy	trafność testowy	precyzja treningowy	precyzja testowy	miara f1 treningowy	miara f1 testowy
0.98266	0.79363	0.95844	0.683672	0.973181	0.68407

Ze względu na to że projekt ten ma w zasadzie charakter samorozwojowy, ciężko określić czy uzyskany wynik jest zadowalający i czy odrzucenie kolumn o wartościach skorelowanych było dobrym posunięciem. Poza uzyskanym wynikiem, projekt pokazuje jak bardzo hiperparametry modelu mogą mieć wpływ na uzyskany wynik. Dodatkowo w trakcie implementacji projektu wystąpił nietypowy błąd. Mianowicie, jeśli projekt został uruchomiony z użyciem jupyter-lab w przeglądarce Firefox to wywołanie "optimize" na obiekcie "study" z pakietu optuna powodowało zajmowanie przez przeglądarkę coraz większej ilości pamięci RAM. Przed ukończeniem 1000 prób przeglądarka przekraczała dostępną na maszynie wielkość RAM ‘u co powodowało jej zamknięcie przez system operacyjny. Ostatecznie ominięto problem poprzez wykorzystanie przeglądarki Microsoft Edge.