

Sprawozdanie

Sztuczne Sieci Neuronowe - Porównanie metod Explainable AI

Krystian Śledź, Mirosław Kołodziej, Jakub Kraśniak, Aleksander Kluczka

1. Temat projektu

Tematem projektu było porównanie działania metod wyjaśniania istotności wektora cech (danych wejściowych) takich jak: SHAP, LIME oraz GSA dla danych numerycznych (dataset Iris) oraz obrazowych (dataset MNIST).

2. Czym są metody Explainable AI

Metody Explainable AI (w skrócie XAI) odnoszą się do technik mających na celu wyjaśnienie i interpretację decyzji podejmowanych przez algorytmy sztucznej inteligencji. Tradycyjne modele AI mogą być trudne do zrozumienia, ponieważ działają na zasadzie czarnej skrzynki (z ang. black-box) - czyli podawane są dane wejściowe i otrzymywane dane wyjściowe. Nie wiadomo natomiast, co z danymi dzieje się między wejściem, a wyjściem. W przypadku ważnych zastosowań, jak systemy medyczne albo finansowe, brak wyjaśnienia otrzymanych wyników może być problematyczny.

3. Historia analizy

Historia wyjaśnialnej sztucznej inteligencji (Explainable AI, XAI) jest ściśle związana z ewolucją i rozwojem dziedziny sztucznej inteligencji (AI) jako całości. Na początku, gdy badania nad AI koncentrowały się głównie na systemach ekspertowych i algorytmach opartych na regułach, wyjaśnialność była integralną częścią procesu. W tych wczesnych systemach, decyzje były podejmowane na podstawie wyraźnie zdefiniowanych reguł i procedur, co umożliwiało łatwe zrozumienie i wyjaśnienie procesu decyzyjnego. Jednak z upływem czasu i postępem technologicznym, szczególnie wraz pojawieniem się uczenia maszynowego i uczenia głębokiego, AI zaczęła stawać się coraz bardziej "czarną skrzynką". Mimo iż te nowe technologie i metody były znacznie skuteczniejsze, ich procesy decyzyjne były znacznie trudniejsze do zrozumienia i wyjaśnienia. Zrozumienie, jak sztuczna inteligencja podejmuje decyzje, stało się priorytetem w wielu dziedzinach, począwszy od medycyny, przez finanse, aż po prawo. To doprowadziło do powstania koncepcji wyjaśnialnej sztucznej inteligencji (XAI). Głównym celem XAI jest tworzenie modeli AI, które są nie tylko skuteczne, ale także transparentne i zrozumiałe dla ludzi. W 2017 roku, Agencja Zaawansowanych Projektów Badawczych Departamentu Obrony (DARPA) w Stanach Zjednoczonych uruchomiła program XAI, mający na celu rozwój nowych technologii AI, które byłyby jednocześnie skuteczne i "przezroczyste". Program ten skupił się na tworzeniu technologii AI, które umożliwiłyby użytkownikom zrozumienie, zaufanie i skuteczne zarządzanie systemami AI. Od tego czasu powstało wiele metod i technik XAI. Na przykład metoda LIME (Local Interpretable Model-Agnostic Explanations), która pozwala na

interpretację pojedynczych przewidywań modelu, lub SHAP (SHapley Additive exPlanations), która opiera się na teorii gier do wyjaśnienia wpływu poszczególnych cech na przewidywanie modelu. Te i wiele innych technik XAI pomagają w lepszym zrozumieniu, jak działają skomplikowane modele AI, co prowadzi do lepszego zaufania do tych systemów i ich efektywniejszego wykorzystania.

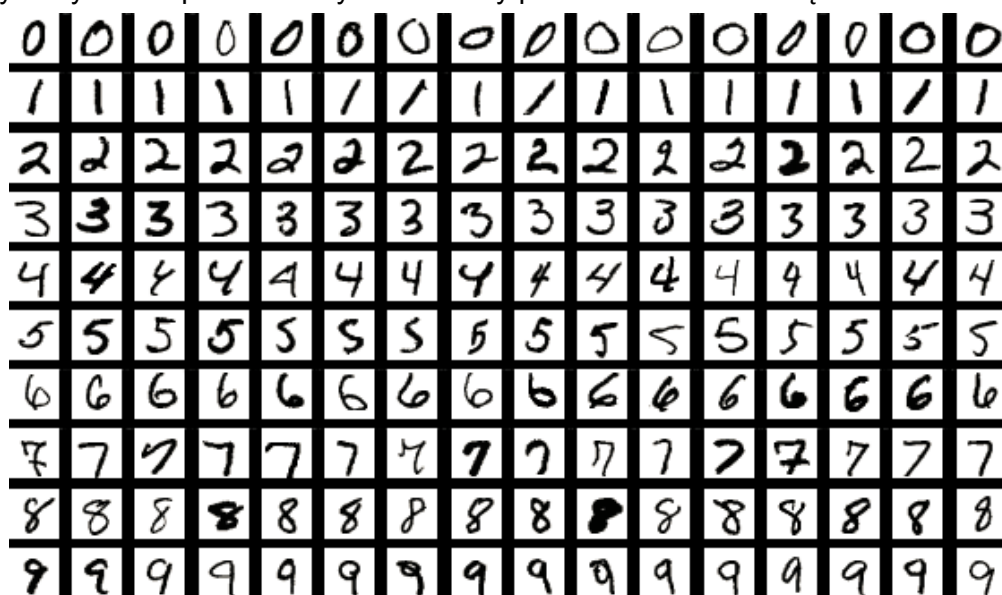
4. Wykorzystane zbiory danych

4.1 Iris

Iris to wielowymiarowy zbiór danych opracowany przez brytyjskiego statystyka i biologa Richarda Fishera w 1936 roku w referacie "The use of multiple measurements in taxonomic problems" jako przykład liniowej analizy dyskryminacyjnej. Zbiór zawiera po 50 próbek z trzech gatunków irysów: Iris setosa, Iris virginica oraz Iris versicolor. Dla każdej próbki zmierzono cztery parametry: długość płatków, szerokość płatków, długość działki kielicha oraz szerokość działki kielicha (wszystkie dane w centymetrach). Bazując na tych cechach, Fisher opracował liniowy model pozwalający na wyodrębnienie poszczególnych gatunków. Zbiór jest obecnie powszechnie używany, głównie w celach naukowych, jako przykład do klasyfikacji czy klasteryzacji.

4.2 MNIST

MNIST (Modified National Institute of Standards and Technology database) to baza napisanych odręcznie cyfr, opracowana przez amerykańską agencję federalną, będącą odpowiednikiem polskiego Głównego Urzędu Miar. Zbiór powstał w 1994 roku. Jest powszechnie używany do trenowania systemów przetwarzających obrazy oraz uczenia maszynowego. MNIST zawiera 60000 obrazów trenujących oraz 10000 testujących. Obrazy mają wymiary 28x28 pikseli. Wszystkie zostały przekształcone w skalę szarości.



Rysunek 1: Część obrazów zawartych w zbiorze MNIST

5. Stworzone modele SSN

5.1 Model dla datasetu Iris

```
iris_model = Sequential()
iris_model.add(Dense(512, activation='relu', input_shape=(4,)))
iris_model.add(Dense(3, activation='softmax'))

iris_model.compile(optimizer='rmsprop',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

Pierwsza warstwa Dense z 512 neuronami przyjmuje dane wejściowe o rozmiarze 4 (co odpowiada ilości cech w zbiorze danych Iris, tj. długość i szerokość płatków oraz długość i szerokość działek kielicha) i używa funkcji aktywacji “relu” (Rectified Linear Unit), która jest współcześnie najczęściej stosowaną funkcją aktywacji w dziedzinie sieci neuronowych.

Druga warstwa Dense z 3 neuronami używa funkcji aktywacji softmax, która jest szczególnie przydatna w klasyfikacji wieloklasowej, ponieważ zwraca prawdopodobieństwa dla każdej z klas, które sumują się do jedności. W tym przypadku mamy 3 klasy, które odpowiadają gatunkom irysów: setosa, versicolor i virginica.

Wreszcie, model jest kompilowany z optymalizatorem “rmsprop”, funkcją straty “categorical_crossentropy” (która jest standardową funkcją straty dla problemów klasyfikacji wieloklasowej) oraz metryką accuracy, która oblicza dokładność klasyfikacji.

5.2 Model dla datasetu MNIST

Ze względu na to, że zbiór MNIST zawiera dane dwuwymiarowe, sztuczna sieć neuronowa różni się od odpowiednika dla zbioru Iris. Dane wejściowe o wymiarach 28x28 przechowują informację o jasności pikseli w zakresie 0-255. Dla poprawy uczenia modelu wartości te znormalizowano do zakresu 0-1.

Sam model składa się z następujących warstw przedstawionych we fragmencie kodu poniżej:

```

model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

```

Na początku naprzemiennie użyte są dwuwymiarowe warstwy konwolucyjne oraz gęstego podpróbkowania. Ponieważ zwiększył się wymiar danych, w następnej kolejności stosowana jest warstwa spłaszczająca. Aby zapobiec przeuczeniu, wyłączana jest połowa neuronów w warstwie odrzucającej. Na koniec stosowana jest warstwa w pełni połączona, aby na wyjściu uzyskać tyle neuronów, ile klas znajduje się w zbiorze danych MNIST (czyli 10). Ostatecznie funkcja aktywacji przekształca wyjście sieci na prawdopodobieństwa przynależności do poszczególnych klas.

Jako funkcję straty przyjęto entropię krzyżową (z ang. "categorical_crossentropy"). Wybrano też optymalizator Adam ze względu na jego powszechnie dobre działanie.

6. Metody użyte w projekcie

6.1 SHAP

6.1.1 Opis metody

SHAP (SHapley Additive exPlanations) to metoda wyjaśniania modeli oparta na teorii gier. Jej celem jest określenie, jak dużo każda cecha przyczynia się do przewidywanej decyzji modelu.

Podstawowym elementem SHAP jest koncepcja wartości Shapleya. W kontekście teorii gier wartość Shapleya jest metodą sprawiedliwego podziału zysków wynikających z kooperacji między graczami. Przeniesiona do kontekstu modeli uczenia maszynowego, wartość Shapleya oblicza, jaki wpływ miała każda cecha na wynik danego przewidywania.

Ogólny proces, w jaki sposób SHAP jest stosowany do interpretowania modeli uczenia maszynowego:

1. Trenowanie dowolnego modelu uczenia maszynowego.
2. Dla danej instancji do wyjaśnienia, obliczenie wszystkich możliwych kombinacji cech (podzbiorów).
3. Dla każdej kombinacji cech, tworzenie dwóch wersji oryginalnej instancji: jednej z cechami z danej kombinacji i drugiej bez nich. Następnie obliczenie różnicy w wynikach przewidywanych przez model dla tych dwóch wersji.
4. Wyliczenie wartości Shapleya dla każdej cechy, będącej średnią ze wszystkich różnic obliczonych w punkcie 3, ważoną przez liczbę możliwych kombinacji cech zawierających daną cechę.
5. Powtórzenie kroków 2-4 dla wielu instancji w celu zrozumienia ogólnych trendów.

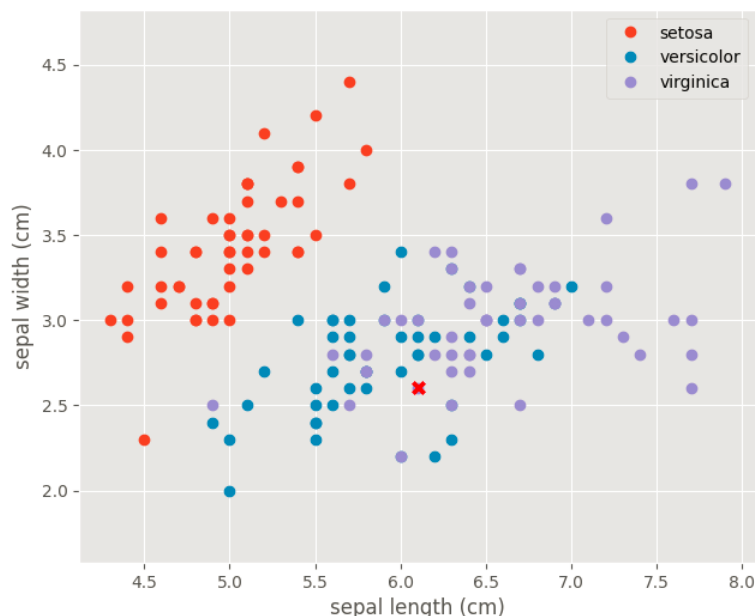
Metoda SHAP ma wiele zalet, w tym to, że pozwala na dokonywanie uporządkowanych dekompozycji predykcji na sumy składników. Dzięki temu można lepiej zrozumieć, jakie cechy mają największy wpływ na model jako całość. SHAP jest szczególnie użyteczny w dziedzinach, gdzie interpretowalność i przejrzystość są kluczowe.

6.1.2 Omówienie wyników

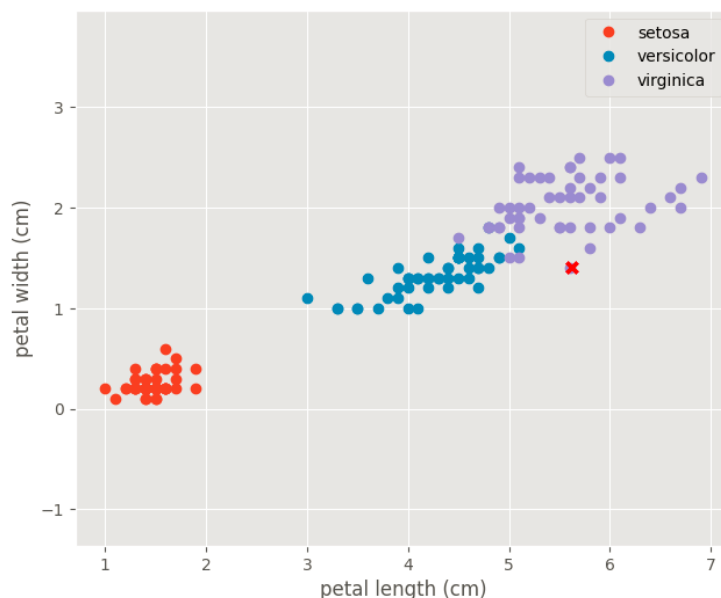
6.1.2.1 Dataset Iris

W przypadku datasetu Iris, wybrano do analizy wyników jeden z irysów gatunku virginica o parametrach:

- petal length (długość płatk) - 5.60 cm
- petal width (szerokość płatk) - 1.40 cm
- sepal width (szerokość działki kielicha) - 2.60 cm
- sepal length (długość działki kielicha) - 6.10 cm



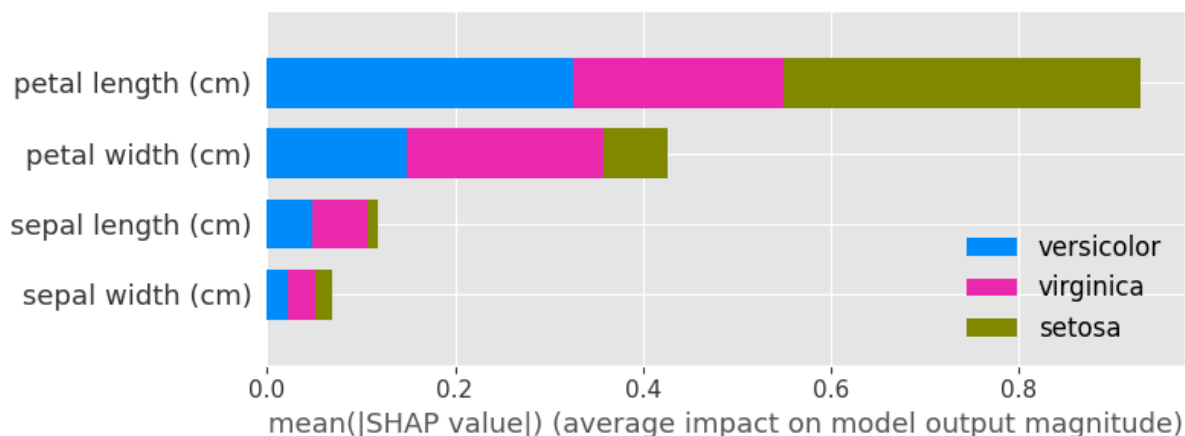
Rysunek 2: Położenie sprawdzanego irysa na wizualizacji parametrów działki kielicha



Rysunek 3: Położenie sprawdzanego irysa na wizualizacji parametrów płatk

Za pomocą techniki SHAP i dostępnych w niej metod otrzymano następujące wyniki:

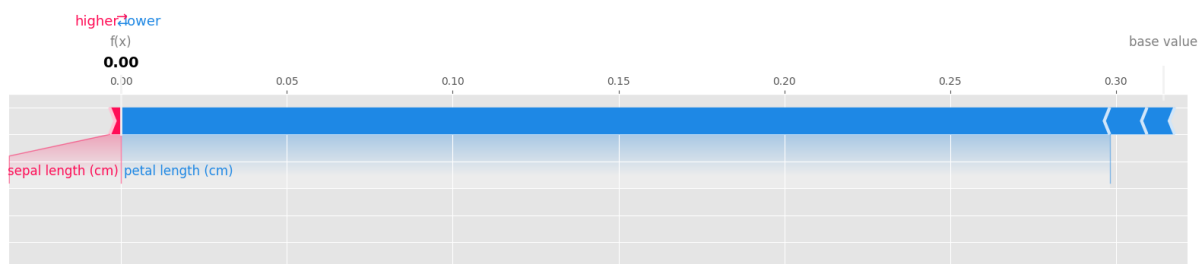
summary_plot - Wyświetla wykres podsumowujący wartości Shapleya dla wszystkich cech. Pozwala to zobaczyć, jakie cechy mają największy wpływ na przewidywania modelu.



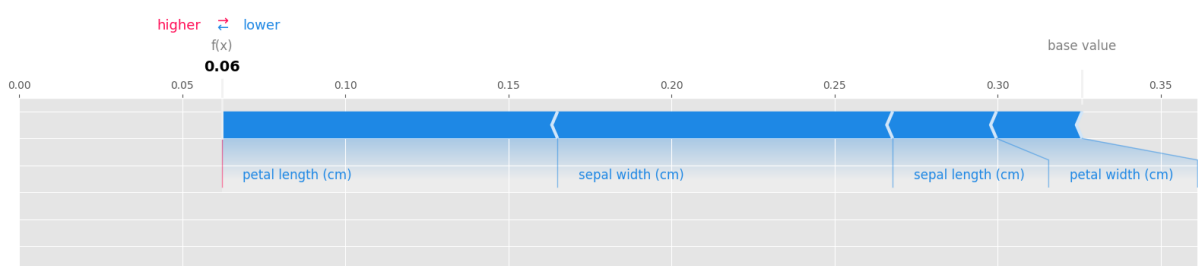
Rysunek 4: Wartości Shapleya dla wszystkich cech

Jak można zauważyć najważniejszą cechą jest długość i szerokość płatk.

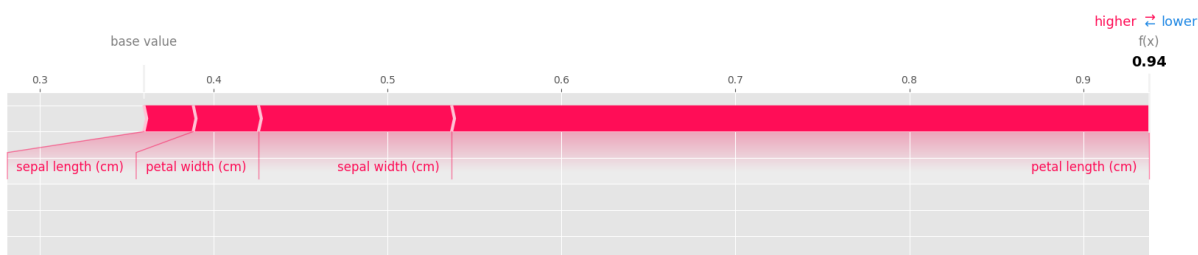
force_plot - Wyświetla wykres sił, który pokazuje wpływ poszczególnych cech na daną przewidywaną wartość. Pozwala to na dogłębne zrozumienie, jakie cechy przyczyniły się do konkretnego przewidywania. Czerwone paski wskazują, że dana cecha zwiększała prawdopodobieństwo danej klasy, a niebieskie paski wskazują, że cecha zmniejszała prawdopodobieństwo.



Rysunek 5: Wykres sił dla klasy setosa



Rysunek 6: Wykres sił dla klasy versicolor

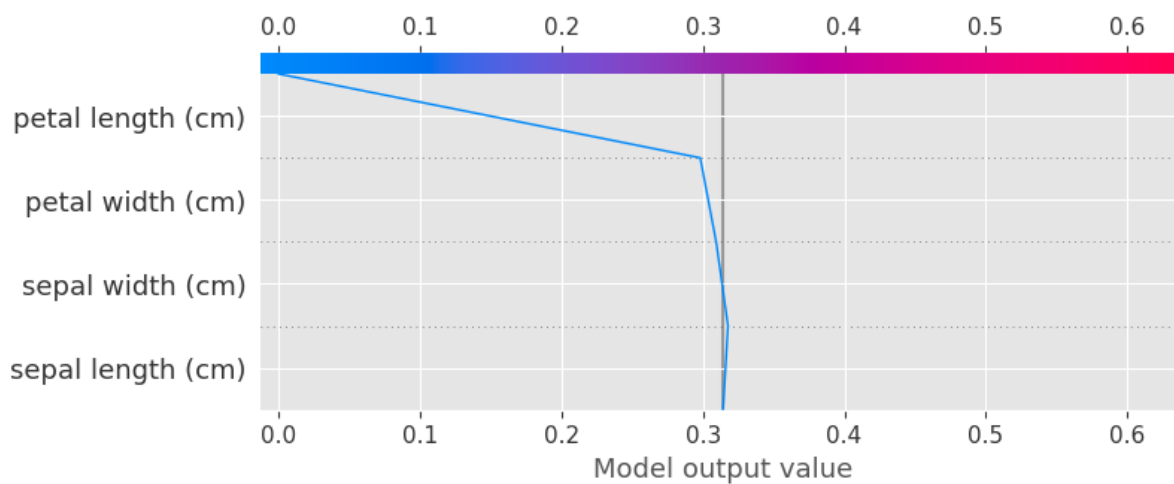


Rysunek 7: Wykres sił dla klasy virginica

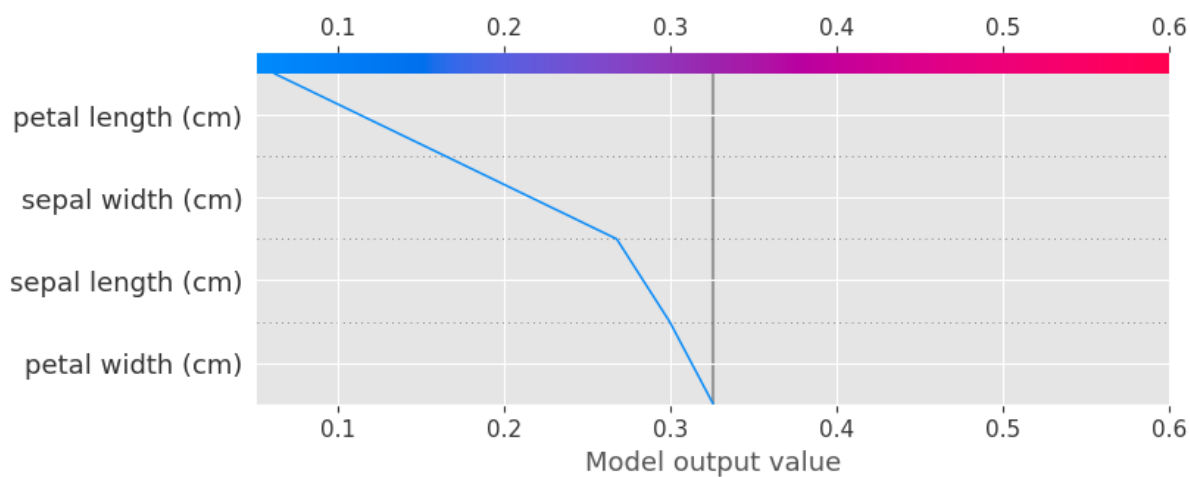
Jak można zauważyć, opracowany model poradził sobie z rozpoznaniem gatunku kwiatu - oszacował, że na 94% będzie to virginica. Każda cecha w znaczący sposób zwiększyła prawdopodobieństwo danej klasy.

decision_plot - umożliwia lepsze zrozumienie, jak poszczególne cechy wpływają na końcowe przewidywanie modelu dla danej obserwacji. Wykres decyzyjny prezentuje ścieżkę od wartości oczekiwanej modelu do końcowego wyniku przewidywania. Każdy krok na tej ścieżce to dodanie do wartości oczekiwanej wpływu jednej cechy. Kolejność kroków jest ważna, ponieważ wpływ cechy może zależeć od wartości wprowadzonych wcześniej cech. Wykres decyzyjny pozwala na zobrazowanie tej zależności.

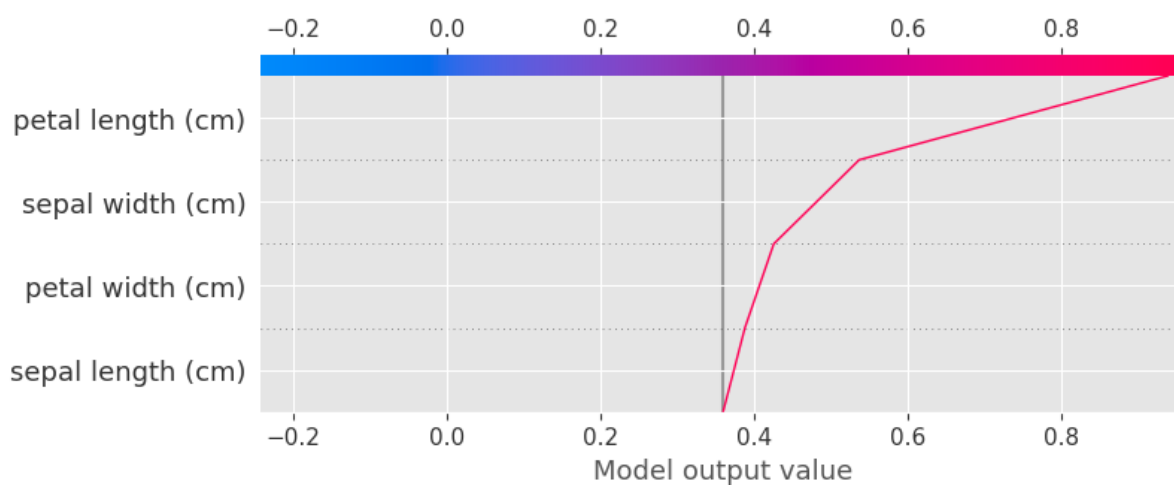
Na wykresie, oś pionowa pokazuje przewidywaną wartość modelu, a oś pozioma prezentuje kolejne cechy. Kolor linii wskazuje na wpływ dodanej cechy: czerwony oznacza zwiększenie przewidywanej wartości, a niebieski oznacza jej zmniejszenie.



Rysunek 8: Wykres decyzyjny dla klasy setosa



Rysunek 9: Wykres decyzyjny dla klasy versicolor



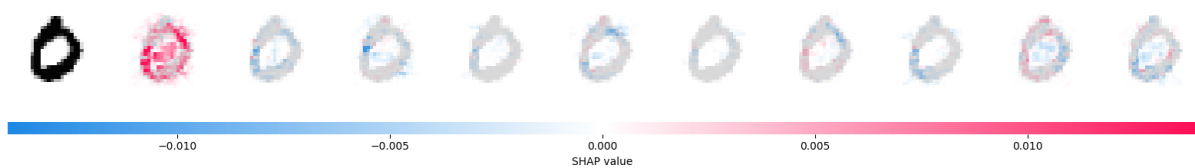
Rysunek 10: Wykres decyzyjny dla klasy virginica

Jak można zauważyć na rysunkach powyżej najważnią cechą, która wpłynęła na przypisanie tego kwiatka do klasy virginica jest długość płatków oraz szerokość działki. Możemy też zauważyć, że na odrzucenie klasy setosa najbardziej wpłynęła długość płatków, a na odrzucenie klasy versicolor po równo długość płatków oraz szerokość działki.

6.1.2.2 Dataset MNIST

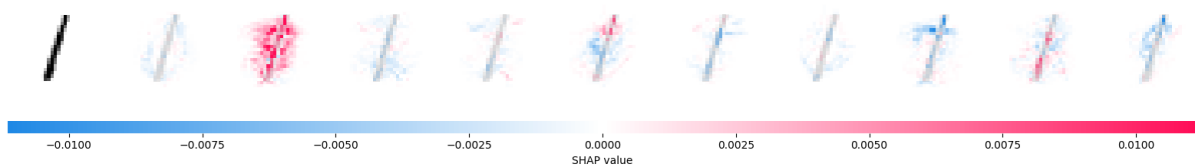
Do analizy przy użyciu techniki SHAP wybrano jeden rekord każdej klasy z datasetu. Za pomocą techniki SHAP i dostępnej w niej metody analizy obrazów (image_plot) otrzymano następujące wyniki:

Cyfra 0 - pewność modelu (99.99%)



Rysunek 11: Oceny obszarów dla każdej klasy

Cyfra 1 - pewność modelu (99.98%)



Rysunek 12: Oceny obszarów dla każdej klasy

Cyfra 2 - pewność modelu (100%)



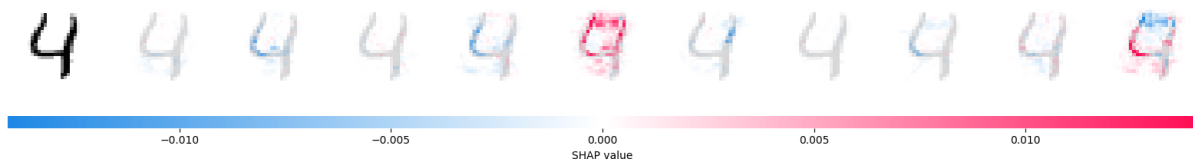
Rysunek 13: Oceny obszarów dla każdej klasy

Cyfra 3 - pewność modelu (41.64%)



Rysunek 14: Oceny obszarów dla każdej klasy

Cyfra 4 - pewność modelu (100%)



Rysunek 15: Oceny obszarów dla każdej klasy

Cyfra 5 - pewność modelu (94.93%)



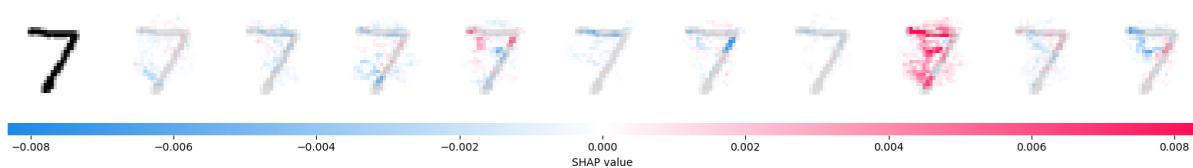
Rysunek 16: Oceny obszarów dla każdej klasy

Cyfra 6 - pewność modelu (100%)



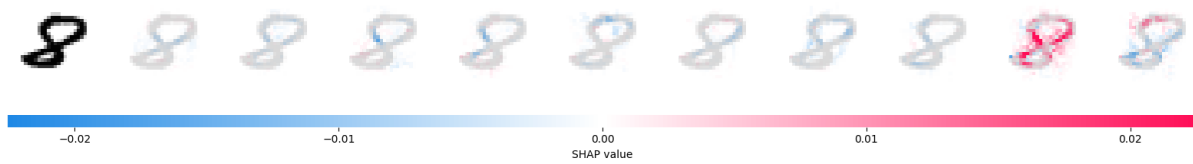
Rysunek 17: Oceny obszarów dla każdej klasy

Cyfra 7 - pewność modelu (100%)



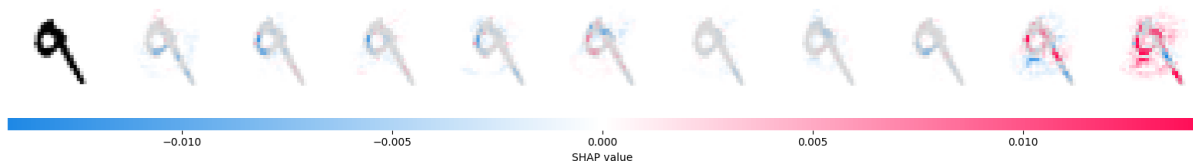
Rysunek 18: Oceny obszarów dla każdej klasy

Cyfra 8 - pewność modelu (100%)



Rysunek 19: Oceny obszarów dla każdej klasy

Cyfra 9 - pewność modelu (99.97%)



Rysunek 20: Oceny obszarów dla każdej klasy

Każda klasa wyjściowa ma swój własny wykres, a czerwone obszary na wykresie pokazują obszary obrazu, które zwiększyły prawdopodobieństwo danej klasy, podczas gdy niebieskie obszary pokazują obszary, które je zmniejszyły.

6.1.2.3 Wnioski

Wykorzystanie SHAP do analizy danych IRIS i MNIST pozwala na lepsze zrozumienie, jak modele uczenia maszynowego, takie jak sieci neuronowe, podejmują decyzje. SHAP pomaga wyjaśnić, które cechy mają największy wpływ na predykcje modelu i jakie są ich konkretne wpływy.

- Analiza danych IRIS: SHAP pomaga zrozumieć, które cechy (długość i szerokość płatków i działek kielicha) mają największy wpływ na klasyfikację różnych gatunków Irysów. Możemy to zobaczyć na przykład na wykresie ``summary_plot`` - pokazuje on, które cechy są najważniejsze i jak wpływają na wynik (pozytywnie czy negatywnie). Możemy także zobaczyć wpływ poszczególnych cech na predykcje konkretnej obserwacji (np. przy użyciu ``force_plot``).
- Analiza danych MNIST: SHAP pozwala zrozumieć, które piksele obrazu najbardziej wpłynęły na klasyfikację danego obrazu do konkretnej klasy (np. cyfry od 0 do 9) (np. przy użyciu ``image_plot``). Jednym z ważnych wniosków jest to, że nie tylko obszary obrazu z konkretnymi cechami (np. liniami i kształtami cyfr) mają wpływ na decyzje modelu. Równie ważne mogą być puste obszary, czyli obszary bez żadnych wyraźnych cech. Na przykład, puste obszary mogą pomóc modelowi odrzucić niektóre klasy (np. jeśli w pewnym miejscu na obrazie nie ma żadnej linii, to może sugerować, że obraz nie reprezentuje cyfry, która ma linię w tym miejscu). Z drugiej strony, puste obszary mogą również wpłynąć na przypisanie obrazu do konkretnej klasy (np. jeśli pewien obszar jest pusty, może to sugerować, że obraz reprezentuje

cyfrę, która ma pusty obszar w tym miejscu). Dzięki temu możemy zrozumieć, jak model interpretuje zarówno puste, jak i niepuste obszary obrazu, co daje nam głębsze zrozumienie, jak model "widzi" obrazy.

W obu przypadkach, SHAP daje nam lepsze zrozumienie, jak model podejmuje decyzje, co jest szczególnie ważne w kontekście interpretowalności modeli AI. Zrozumienie, dlaczego model podejmuje określone decyzje, jest kluczowe dla zaufania do modelu i może pomóc w identyfikacji problemów.

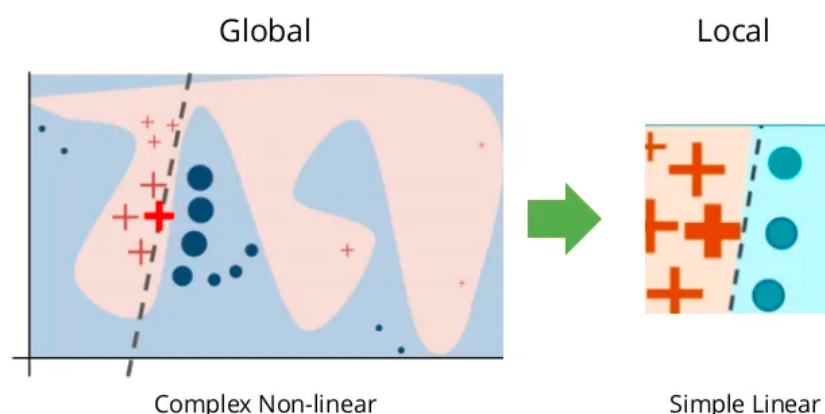
6.2 LIME

6.2.1 Opis metody

LIME, czyli Local Interpretable Model-Agnostic Explanations to technika wyjaśniania modeli opartych na uczeniu maszynowym bez wiedzy o nich, opracowana przez Marco Ribeiro w 2016 roku. Ma ona na celu interpretację oraz zrozumienie decyzji podejmowanych przez dany model. Człony nazwy techniki oznaczają:

- Local – wyjaśnia tylko pewną część modelu, używa lokalnie ważonej regresji liniowej
- Model-Agnostic – bez wiedzy o modelu (czarna skrzynka), działa dla dowolnego rodzaju modelu

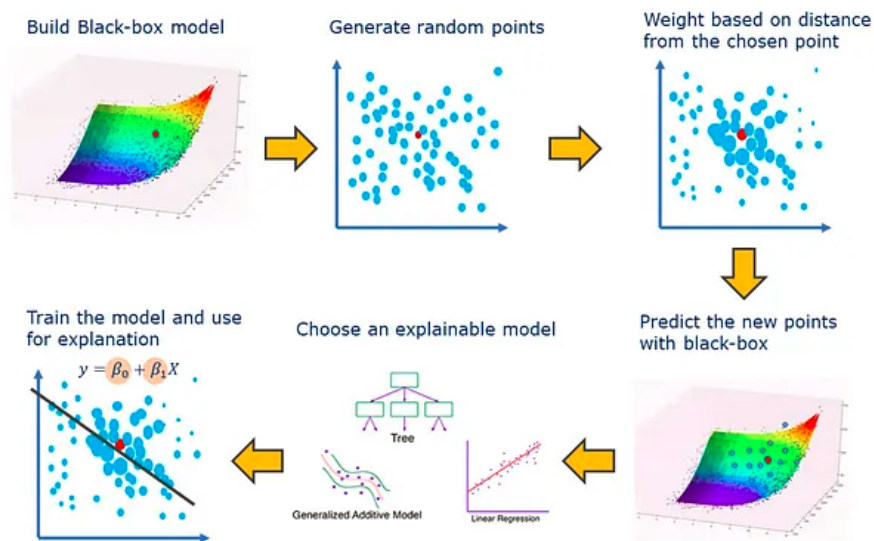
Metoda LIME buduje lokalny, łatwiejszy do zinterpretowania model poprzez analizę tzw. czarnej skrzynki, pozwalający na wytłumaczenie poszczególnych predykcji. Aby lepiej zrozumieć kształt funkcji użytej w trakcie uczenia, LIME generuje punkty wokół analizowanej instancji (na obrazku niżej oznaczonej poprzez czerwony krzyż). Rekordy poddane próbkowaniu są predykowane przy pomocy funkcji, a następnie ważone odległością do badanej instancji. Przerywana linia na rysunku poniżej odpowiada przybliżeniu krawędzi funkcji. Jest ona granicą decyzji oraz definiuje wyjaśnienie wiarygodne tylko lokalnie.



Rysunek 21: Idea działania LIME

Schemat działania LIME:

- Wybór modelu uczenia maszynowego oraz punktu odniesienia, który ma być wyjaśniony.
- Generowanie punktów na przestrzeni R^p . (próbkowanie wartości X z rozkładu normalnego na podstawie zestawu treningowego.)
- Przewidywanie współrzędnej Y dla wygenerowanych punktów za pomocą modelu uczenia maszynowego (wygenerowane punkty leżą na powierzchni modelu).
- Przypisywanie wag na podstawie bliskości do wybranego punktu. (przy użyciu jądra RBF, które przypisuje wyższe wagi punktom bliższym odniesieniu.)
- Trenowanie przy użyciu liniowej regresji Ridge'a na wygenerowanym zbiorze danych ważonych: $E(Y) = \beta_0 + \sum \beta_j X_j$. Współczynniki β są traktowane jako wyjaśnienia LIME.



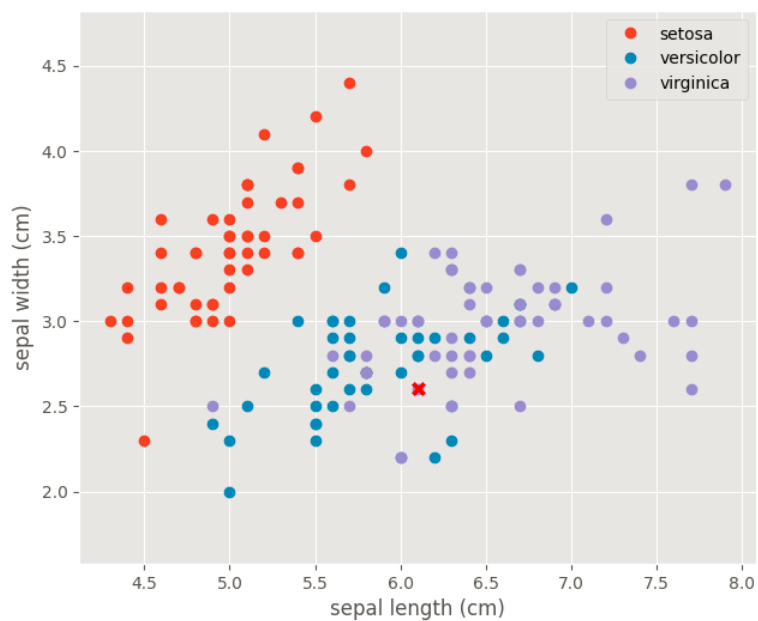
Rysunek 22: Schemat działania LIME

6.2.2 Omówienie wyników

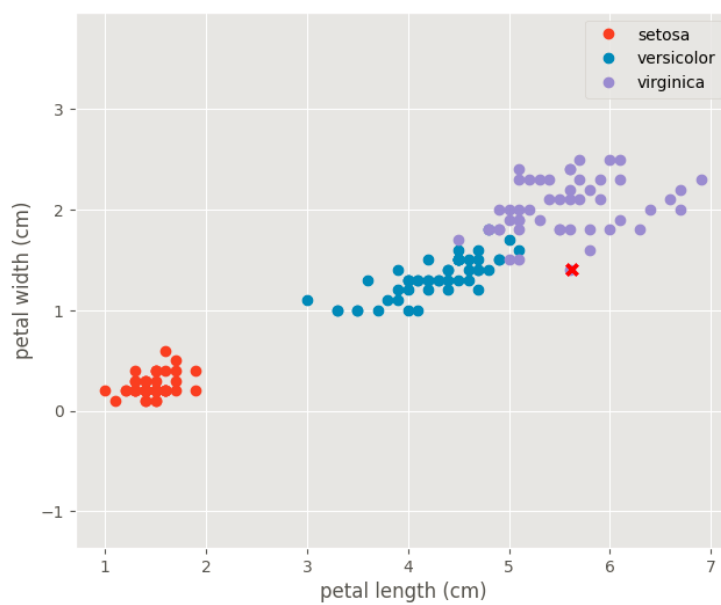
6.2.2.1 Dataset Iris

W przypadku datasetu Iris, wybrano do analizy wyników jeden z irysów gatunku virginica o parametrach:

- petal length (długość płatk) - 5.60 cm
- petal width (szerokość płatk) - 1.40 cm
- sepal width (szerokość działki kielicha) - 2.60 cm
- sepal length (długość działki kielicha) - 6.10 cm

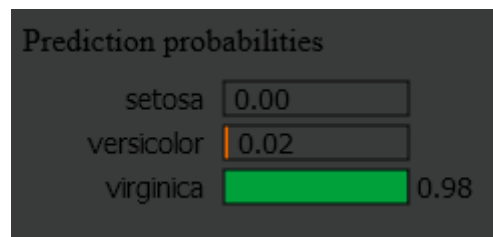


Rysunek 23: Położenie sprawdzanego irysa na wizualizacji parametrów działki kielicha



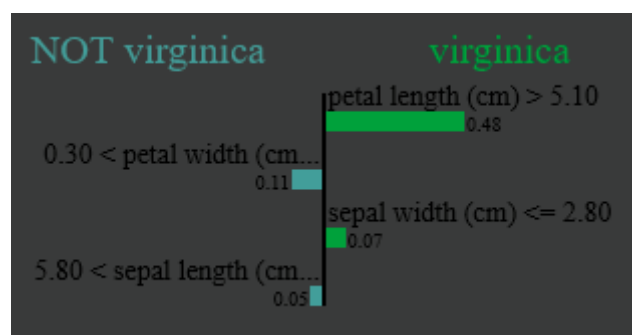
Rysunek 24: Położenie sprawdzanego irysa na wizualizacji parametrów płatk

Za pomocą techniki LIME otrzymano następujące wyniki:



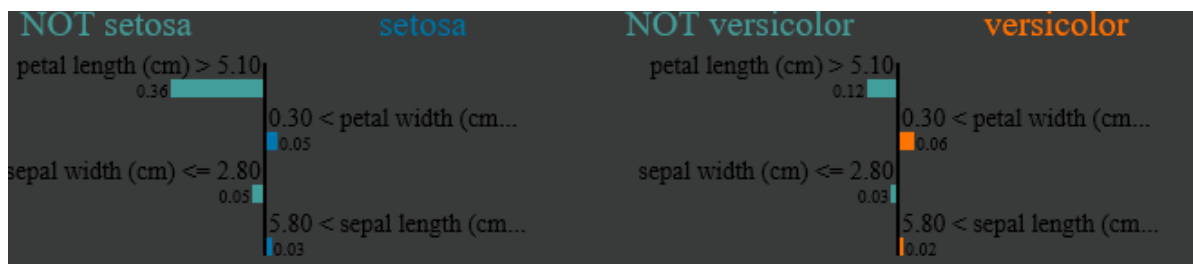
Rysunek 25: Wyniki predykcji dla wybranego irysa

Jak można zauważyć, opracowany model poradził sobie z rozpoznaniem gatunku kwiatu - oszacował, że na 98% będzie to virginica. Co na to jednak wpłynęło?



Rysunek 26: Uzasadnienie LIME wyboru gatunku virginica

Według LIME'a, model dokonał wyboru na podstawie dwóch parametrów: długości płatk (dłuższy niż 5.1 cm) oraz szerokości działki kielicha (szerokość co najmniej 2.8 cm). Pozostałe parametry wskazywały, że nie będzie to ten gatunek, jednak wartości nie odbiegały na tyle mocno od wskazanych przez model.

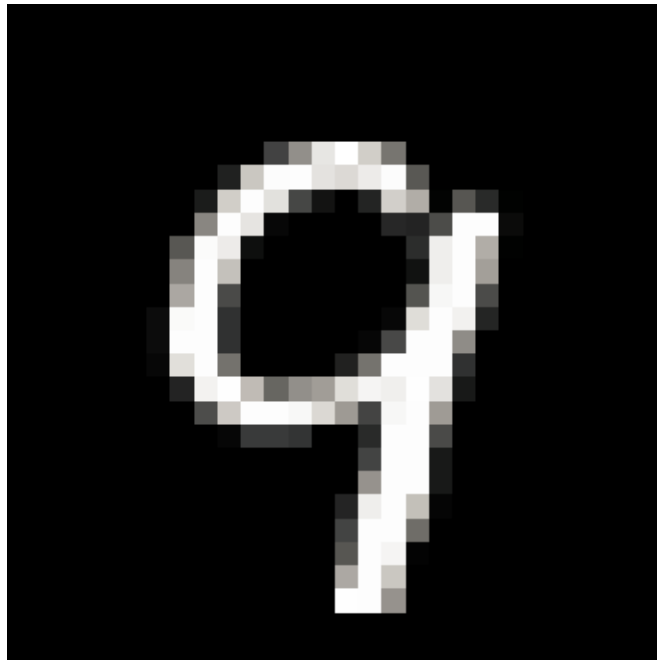


Rysunek 27: Wyniki LIME dla pozostałych gatunków

Jak można zauważyć na rysunku powyżej, LIME wykazał, że model w przypadku pozostałych gatunków był przekonany w większym stopniu, że badany irys nie należy do żadnego z nich, pomimo tego, że mogły na to lekko wskazywać parametry szerokości płatk oraz długości działki kielicha.

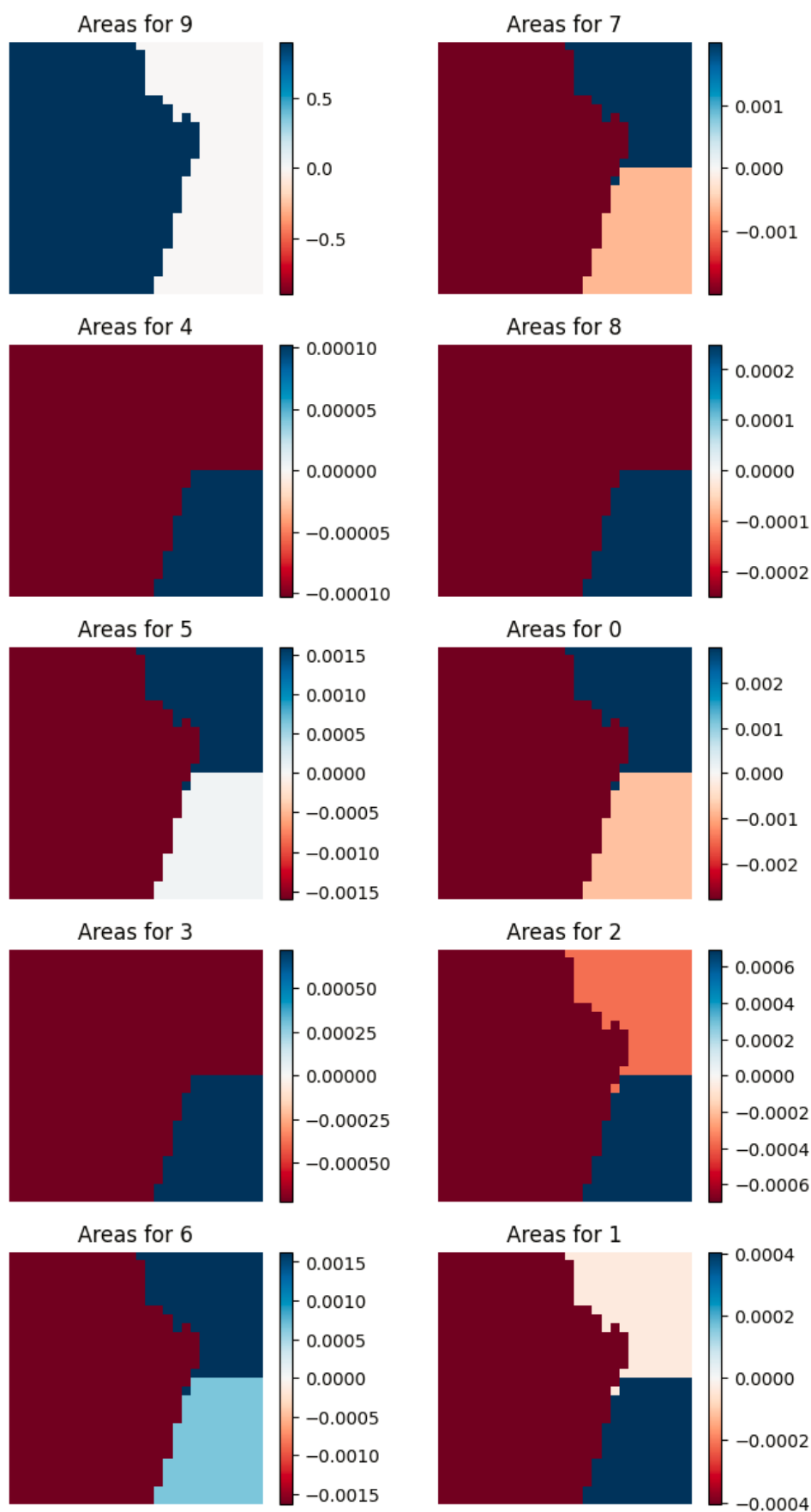
6.2.2.2 Dataset MNIST

Podobnie jak w przypadku datasetu Iris, do analizy przy użyciu techniki LIME wybrano jeden rekord z datasetu. Był nim poniższy obrazek:

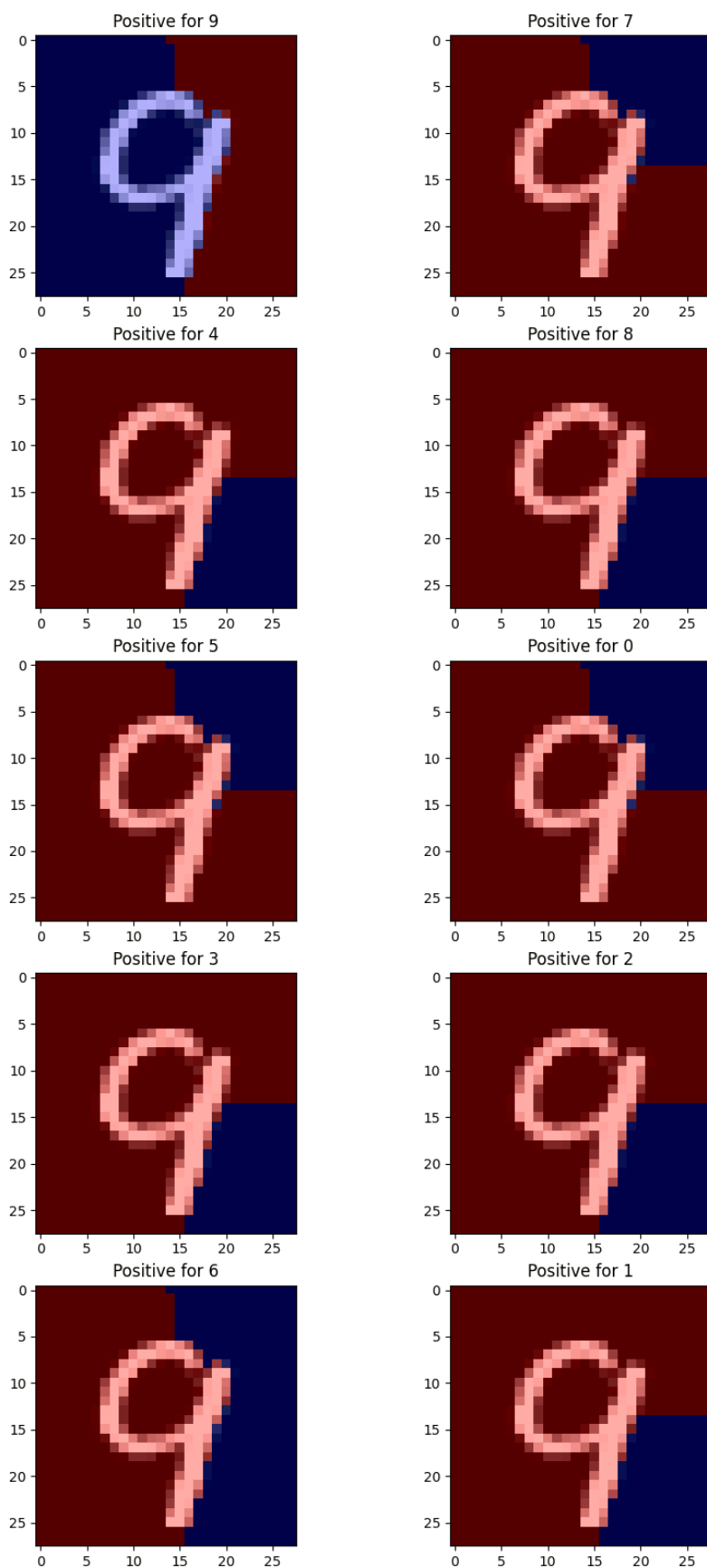


Rysunek 28: Wybrany obrazek

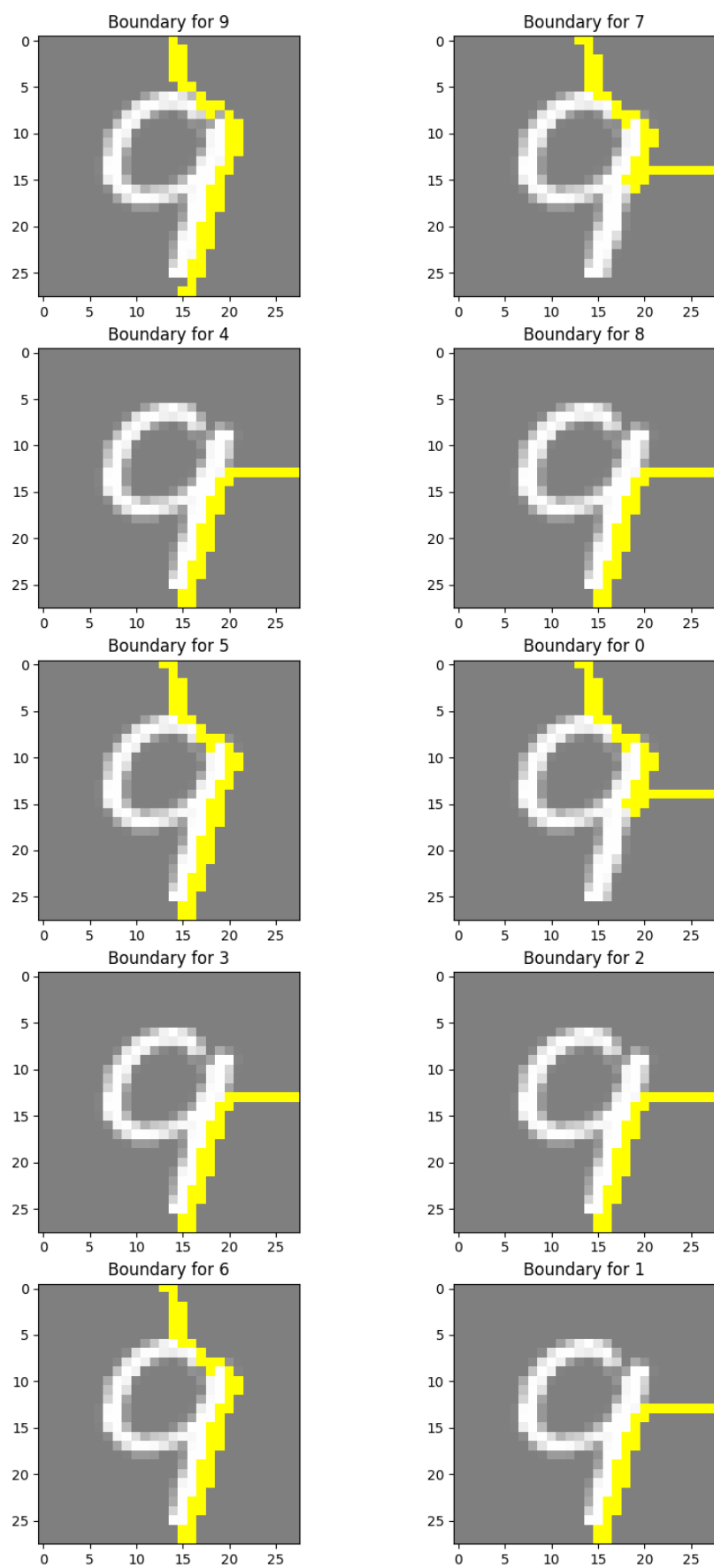
Model poprawnie rozpoznał cyfrę, był praktycznie pewny, że jest to cyfra dziewięć. Według LIME wpłynęły na to następujące czynniki:



Rysunek 29: Oceny obszarów dla każdej z cyfr



Rysunek 30: Na niebiesko: obszary, które wpłynęły pozytywnie na przypisanie do danej klasy, na czerwono negatywnie



Rysunek 31: Granice najbardziej znaczących obszarów

Jak wskazują powyższe obrazki, model wysoko ocenił tylko przynależność do cyfry 9. Pozostałe klasy uzyskały co najwyżej wyniki dwa rzędy wielkości niższe. Ponadto, tylko w przypadku przypisania do klasy 9, sam napis znalazł się w obszarze pozytywnie wpływającym na przypisanie do danej klasy.

6.2.2.3 Wnioski

Technika LIME pomaga wytłumaczyć, w jaki sposób modele sieci neuronowych podejmują decyzje. Dzięki temu, można lepiej zrozumieć dlaczego sieć dokonała danego wyboru. LIME, dla dobrze opracowanej sieci, zadziała dobrze zarówno na danych tekstowych jak i obrazowych.

6.3 SALib/GSA

6.3.1 Opis metody

GSA, czyli Global Sensitivity Analysis to metoda analizy wrażliwości globalnej, która ma na celu zrozumienie wpływu różnych danych wejściowych na wynik modelu lub systemu.

Metoda GSA bada, jak zmienność w wartościach wejściowych przekłada się na zmienność w wynikach modelu, co umożliwi identyfikację najważniejszych czynników wpływających na wynik.

Wpływ badany jest poprzez rozkładanie wariancji na składowe przypisane poszczególnym zmiennym. Najpopularniejszą techniką jest analiza Sobola, która oparta jest na dekompozycji wariancji na czynniki pierwszoplanowe i interakcje między parametrami wejściowymi.

Pierwotna wersja GSA powstała w 1982 roku w opublikowanej pracy "A distribution-free approach to inducing rank correlation among input variables" autorstwa Imana i Conovera. Zaprezentowano tam podstawowe założenia GSA oraz metodologię będącą podstawą analizy Sobola.

Według Wikipedii, analiza wrażliwości to "badanie, jak niepewność wyjścia modelu matematycznego lub systemu (numerycznego lub innego) może być przypisana różnym źródłom niepewności w jego wejściach". Wrażliwość każdego wejścia jest często reprezentowana przez wartość numeryczną, zwaną wskaźnikiem wrażliwości. Wyróżniamy kilka form wskaźników wrażliwości:

Wskaźniki pierwszego rzędu (first order indices): mierzą przyczynek pojedynczego wejścia modelu do wariancji wyjścia.

Wskaźniki drugiego rzędu (second order indices): mierzą przyczynek do wariancji wyjścia spowodowany interakcją dwóch wejść modelu.

Wskaźnik całkowitego rzędu (total order index): mierzy przyczynek do wariancji wyjścia spowodowany przez wejście modelu, uwzględniając zarówno efekty pierwszego rzędu (zmienność wejścia samodzielnie), jak i wszystkie interakcje wyższego rzędu.

6.3.2 SALib

SALib to otwarta biblioteka napisana w Pythonie służąca do przeprowadzania analizy wrażliwości. SALib zapewnia odseparowany przepływ pracy, co oznacza, że nie łączy się bezpośrednio z modelem matematycznym lub obliczeniowym. Zamiast tego, SALib jest odpowiedzialny za generowanie wejść modelu, przy użyciu jednej z funkcji próbkowania (sampling), oraz obliczanie wskaźników wrażliwości na podstawie wyjść modelu, przy użyciu jednej z funkcji analizy. Typowa analiza wrażliwości przy użyciu SALib obejmuje cztery kroki:

- Określenie wejść modelu (parametrów) oraz ich zakresu próbkowania.
- Uruchomienie funkcji próbkowania w celu wygenerowania wejść modelu.
- Ocena modelu przy użyciu wygenerowanych wejść, z zapisaniem wyjść modelu.
- Uruchomienie funkcji analizy na wyjściach, aby obliczyć wskaźniki wrażliwości.

SALib oferuje kilka metod analizy wrażliwości, takich jak Sobol, Morris i FAST. Wiele czynników determinuje, która metoda jest odpowiednia dla konkretnego zastosowania, co omówimy później. Niezależnie od wybranej metody, korzystamy tylko z dwóch funkcji: próbkowania i analizy.

6.3.3 Omówienie wyników

6.3.3.1 Dataset Iris

Na potrzeby analizy wrażliwości definiujemy nasz problem jako:

```
# Define the problem
problem = {
    'num_vars': 4,
    'names': ['sepal_length', 'sepal_width', 'petal_length',
'petal_width'],
    'bounds': [[4.3, 7.9], [2.0, 4.4], [1.0, 6.9], [0.1, 2.5]]
}
```

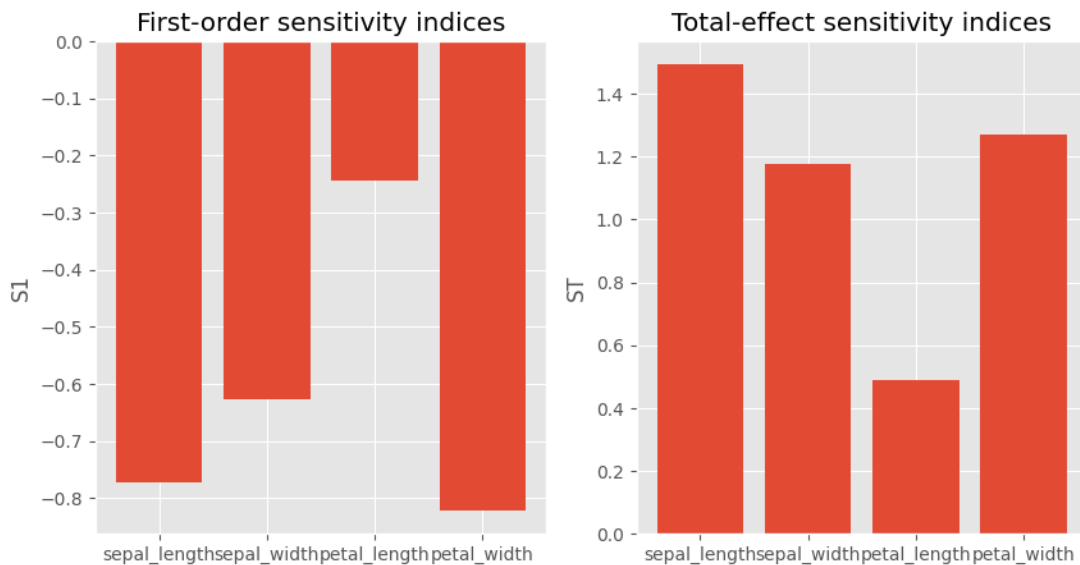
Następnie definiujemy liczbę generowanych próbek oraz określamy, że chcemy liczyć również wpływ zmian poszczególnych zmiennych na siebie nawzajem.

```
# Generate parameter samples using the Sobol' sequence
n_samples = 8
param_values = saltelli.sample(problem, n_samples,
calc_second_order=True)

# Run model and calculate outputs
Y = np.empty([80, 1, 3])
for i, params in enumerate(param_values):
    Y[i] = iris_model.predict(params.reshape(1, -1))
```

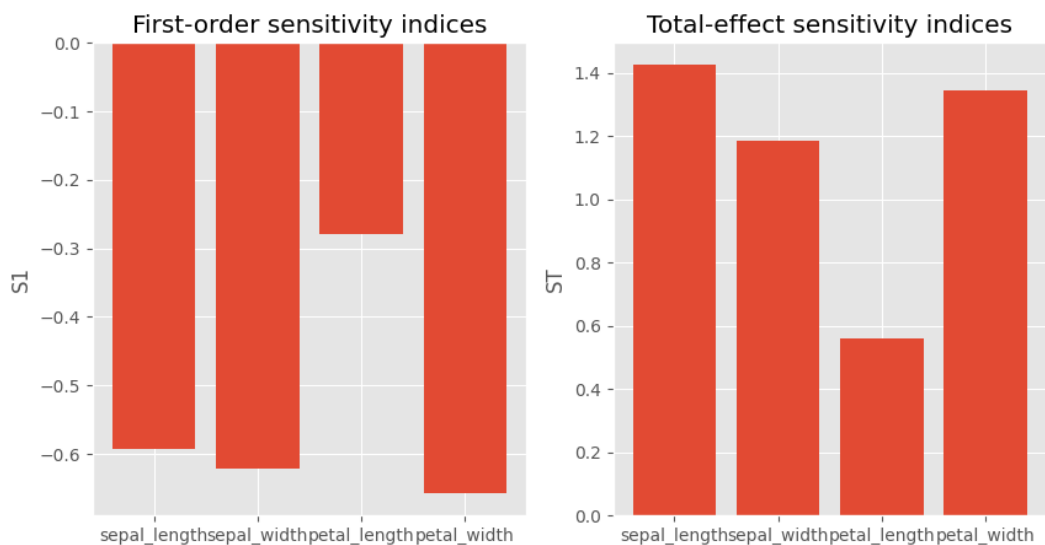
```
Y = Y.reshape(240)

Si = sobol.analyze(problem, Y, calc_second_order=True,
print_to_console=True)
```

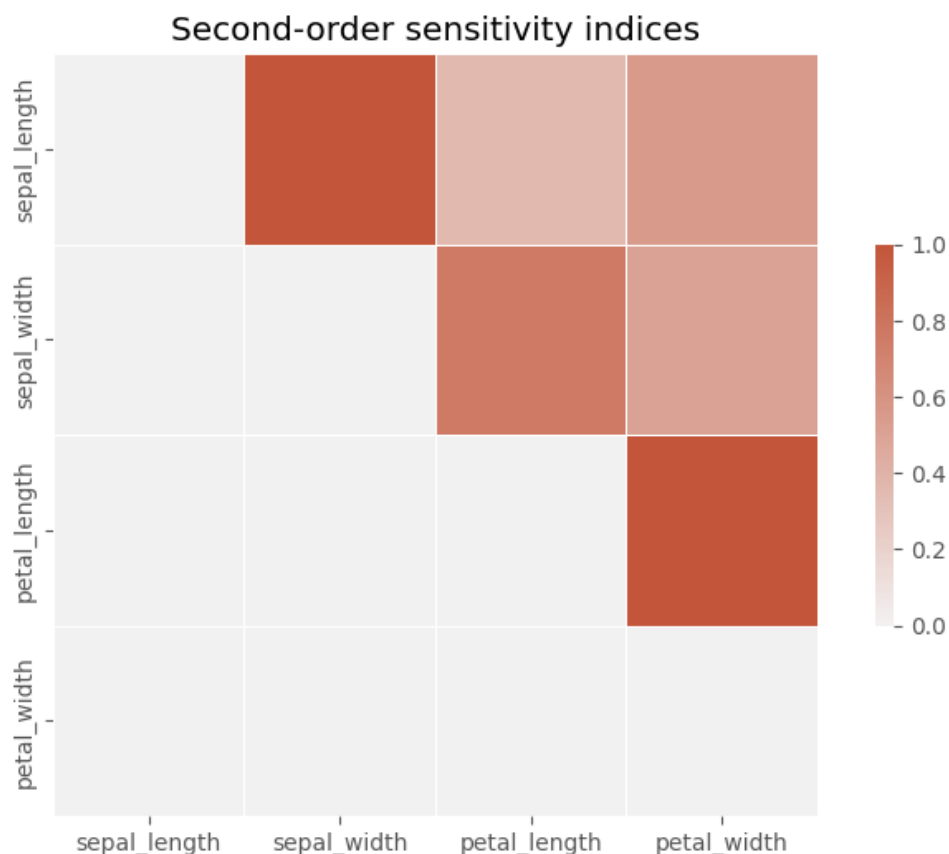


Rysunek 32: Diagramy wskaźników pierwszego i całkowitego rzędu dla $n_samples=8$.

Z nieznanых przyczyn wskaźniki pierwszego rzędu dla wszystkich zmiennych przyjmują ujemne wartości. Zazwyczaj wartości te powinny być dodatnie, ale nie zauważamy tu żadnej anomalii, gdzie wszystkie wskaźniki poza na przykład jednym są dodatnie, tylko wszystkie kolektywnie przyjmują wartości ujemne. Podjęto próby analizy na większej ilości sampli, mianowicie 64 oraz 1024, ale nie zauważono zmiany znaku wskaźników, a znacznie zwiększył się jedynie wskaźnik dla zmiennej sepal width:



Rysunek 33: Diagramy wskaźników pierwszego i całkowitego rzędu dla $n_samples=1024$.



Rysunek 34: Heatmap pokazujący zależności między zmianami poszczególnych zmiennych wejściowych (wskaźniki drugiego rzędu)

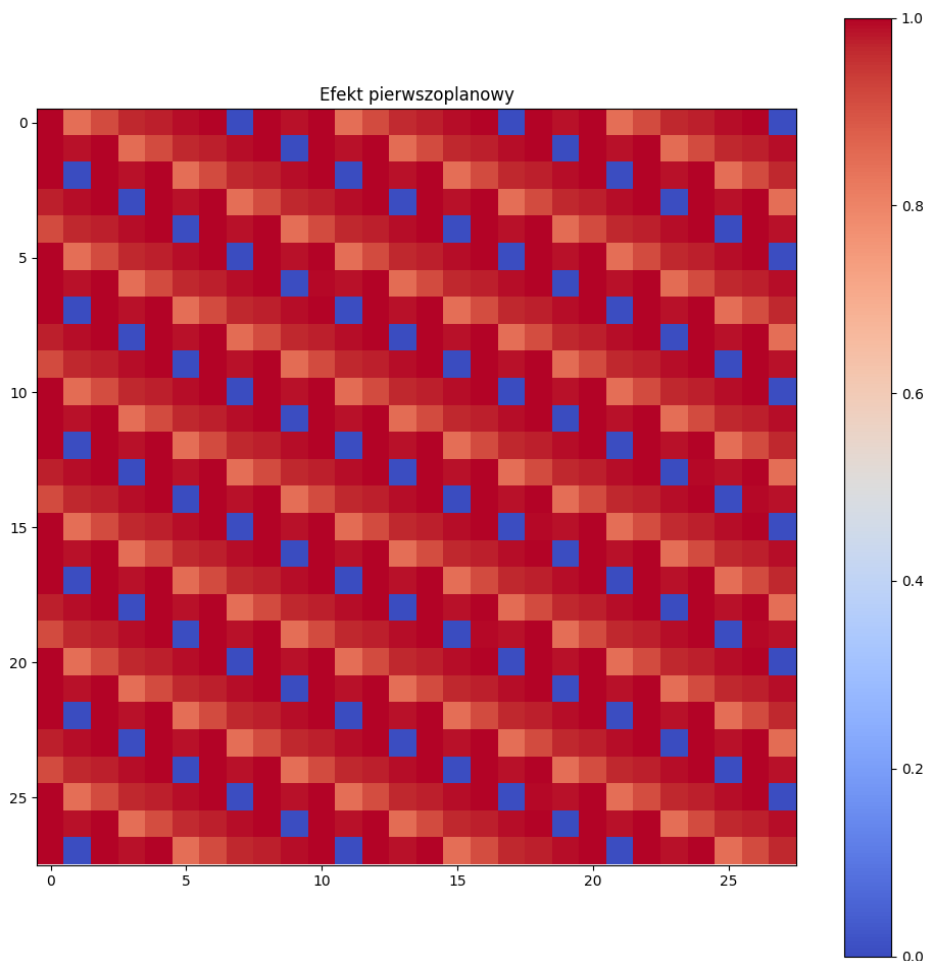
Z heatmapy zauważamy, że największy wskaźnik wrażliwości zachodzi dla par szerokości i długości płatków oraz szerokości i długości działek kielicha. Wynik ten wydaje się rozsądny.

6.3.3.2 Dataset MNIST

Analiza wrażliwości metodą Sobol (ale też innymi sposobami) nie jest poprawnym sposobem na szacowanie wpływu parametrów wejściowych na predykcje modelu z danych MNIST. O ile dla zbioru Iris ma to większy sens ze względu na małą liczbę parametrów wejściowych, tak dla danych dwuwymiarowych - czyli obrazów - nie daje ona żadnych miarodajnych wyników.

Wynika to z faktu, że analiza wrażliwości modyfikuje pojedyncze parametry wejściowe i szacuje zmiany w danych wyjściowych. Przy obrazach 28x28, czyli przy 784 pikselach (parametrach), zmiana pojedynczego piksela nie ma w zasadzie żadnego wpływu na predykcję modelu. Dlatego wyniki GSA dla obrazów nie są miarodajne.

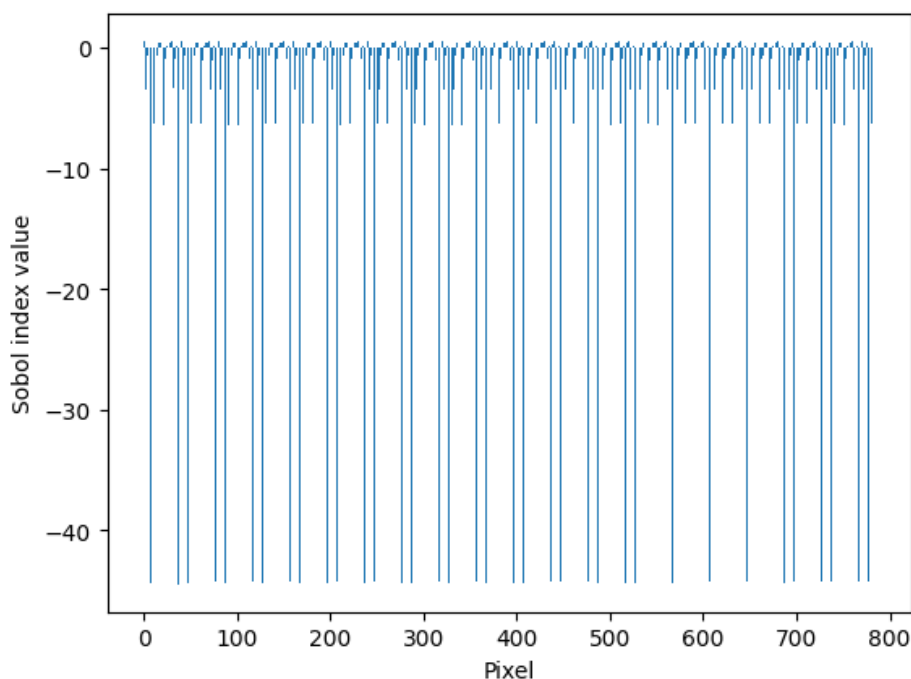
Na dowód zaprezentować można efekt pierwszoplanowy wygenerowany dla 16 próbek Sobolowskich (czyli w przypadku tego zbioru danych - 25120 zmodyfikowanych obrazów):



Rysunek 35: Wartości efektu pierwszoplanowego dla wszystkich pikseli obrazów MNIST

Na przedstawionym powyżej rysunku wyraźnie widać pewien wzór, który powtarza się przez wszystkie 784 piksele. Nie do końca wiadomym jest, co on przedstawia, natomiast są dwa podejrzenia. Pierwsze wytłumaczenie może być takie, że jest to przebieg algorytmu próbkowania Sobolowskiego. Druga teoria jest taka, że to przebieg algorytmu analizującego otrzymane próbki.

Na powyższym wykresie indeksy Sobolowskie zostały znormalizowane, natomiast w naturalnej postaci wyglądają one następująco:



Rysunek 36: Wykres słupkowy wskaźnika pierwszego rzędu dla wszystkich pikseli obrazu

Na powyższym diagramie słupkowym również widać wzór, w którym w równych odstępach niektóre indeksy osiągają ujemne wartości w innym rzędzie wielkości.

Każdy piksel w teorii przedstawia wpływ na końcową predykcję modelu. Oczekiwany wynik to przykładowo zerowy wpływ dla pierwszych i ostatnich 56 pikseli, a większy dla tych umieszczonych na środku obrazu. Wywnioskować więc można, że wskaźnik pierwszego rzędu analizy Sobolowskiej nie dał w tym przypadku żadnych miarodajnych wyników.

Pokrótkę należy rozważyć również wskaźnik drugiego rzędu, zwany inaczej efektem drugoplanowym. Mierzy on wpływ interakcji między dwoma wejściami, czyli w tym przypadku pikselami. W tym przypadku w teorii jeden piksel nie ma wpływu na drugi, ale zauważyć można pewne statystyczne zależności między nimi. Przykładowo, każdy piksel stanowiący tło (czyli czarny piksel) ma znacznie większą szansę na sąsiedztwo z innym czarnym pikselem. Z kolei im piksel jest jaśniejszy, tym większa szansa, że znajduje się w otoczeniu innych jasnych pikseli (wynika to z "rozmycia" narysowanych cyferek - obrazy nie są perfekcyjnie ostre) i w związku z tym stanowi część cyferki.

Jest to jednak cecha wspólna dla wszystkich klas, w związku z tym nie jest miarodajna w kontekście wrażliwości na predykcje wykonywane przez model.

W jaki sposób w takim razie uzyskać sensowne rezultaty? Teoria jest taka, że parametrów wejściowych jest po prostu zbyt wiele, dlatego trzeba by zacząć od zmniejszenia ich liczby. Wymaga to wyciągnięcia z obrazów pewnych zagregowanych danych statystycznych, np.:

- liczby pikseli jaśniejszych, niż 0,5;
- średnią liczbę pikseli-sąsiadów zawierających potrzebne informacje (piksele, które nie są tłem) dla danych pikseli (czyli piksel w rogu miałby maks 2 sąsiadów, ale jako

że oba są zupełnie czarne, to de facto ma 0 przydatnych sąsiadów, a piksel stanowiący część cyfry miał tych sąsiadów np. 4);

- jasność pewnych wybranych pikseli o największym zróżnicowaniu wartości między obrazami.

Posiadając takie dane, należałoby nauczyć na ich podstawie nowy model i dopiero wtedy zastosować na wynikach GSA. Wtedy analiza wrażliwości miałaby większe szanse powodzenia, ponieważ każdy parametr stanowiłby większy wpływ na predykcje.

Pomysł ten jednak nie został zrealizowany, ponieważ nie udało się nauczyć modelu do otrzymywania sensownych predykcji. Skuteczność poprawnego oszacowania wynosiła około 10%, czyli de facto równowartość rzutu kostką o 10 ścianach (liczbie klas w zbiorze MNIST).

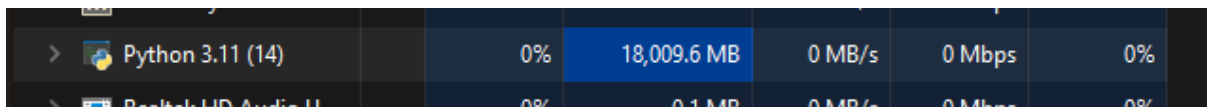
Drugie potencjalne rozwiązanie, to forsowne podniesienie wymiaru komputacji i modlitwa o uzyskanie sensownych wyników. Przykładowo, można by zwiększyć liczbę próbek Sobolowskich z 16 na 2048 (przynajmniej dwukrotność liczby parametrów wejściowych, jak w przypadku zbioru Iris), co przekłada się na 3215360 zmodyfikowanych obrazów podanych do predykcji na wejście modelu.

Pomysł ten wymaga ogromnych zasobów obliczeniowych oraz pamięciowych. Dla powyższych danych można w prosty sposób policzyć potrzebną przestrzeń pamięci RAM do komputacji. Jeden piksel ma zakres 0-255, czyli $0-2^8$, zakres ten mieści się w jednym bajcie. W związku z tym obraz 28x28 zajmuje 784 bajty. Mnożąc tę wartość przez 3215360 wychodzi ostatecznie 2520842240 bajtów, czyli 2,52 gigabajta.

Może to wcale nie brzmi jak dużo danych, ale należy wziąć pod uwagę, że jest to idealny przypadek. Do tego wyniku należy wziąć poprawkę na następujące aspekty:

- ile biblioteka numpy faktycznie rezerwuje miejsca dla jednego obrazu;
- ile miejsca python potrzebuje na przechowywanie tego typu danych;
- jak surowe dane układają się w blokach i rejestrach pamięci;
- ile tymczasowych zmiennych jest tworzonych na potrzeby obliczeniowe;
- ilekrotnie kopiowane są dane na potrzeby obliczeniowe.

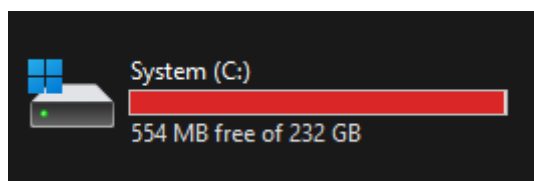
Aby zobrazować skalę problemu, na maszynie o 32GB pamięci RAM proces Python po dwóch minutach wygląda następująco:



>	Python 3.11 (14)	0%	18,009.6 MB	0 MB/s	0 Mbps	0%
>	Realtek HD Audio U	0%	0.1 MB	0 MB/s	0 Mbps	0%

Rysunek 37: Proces Python po dwóch minutach kalkulacji analizy wrażliwości dla 2048 próbek

Oczywiście nie jest to całkowita zarezerwowana pamięć. Kiedy całe zasoby sprzętowe RAM zostaną wykorzystane, system (żeby nie umrzeć) robi tzw. swapy albo dumpy pamięci na dysk twardy, które potem wczytuje do pamięci podręcznej, kiedy są potrzebne przechowywane tam dane. Po dwóch minutach obliczeń, z 40 GB wolnego miejsca na dysku zostało tyle:



Rysunek 38: Miejsce na dysku podczas działania analizy wrażliwości dla 2048 próbek

Jest to chyba wystarczający dowód na to, że na zwykłym komputerze wykonanie tego typu obliczeń jest po prostu nierealne.

6.3.3.3 Wnioski

Dla zbioru danych Iris, pomimo ujemnych wartości S1 wyniki wydają się być prawidłowe i dobrze określać wrażliwość modelu na poszczególne zmienne wejściowe oraz ich zależności między sobą.

Z kolei dla zbioru MNIST nie udało się osiągnąć miarodajnych rezultatów, wynika to potencjalnie ze złego zastosowania analizy wrażliwości dla danych dwuwymiarowych.

7. Podsumowanie

Przedmiotem naszego projektu było zastosowanie i porównanie różnych metod Explainable AI (XAI), a dokładnie metod SHAP, LIME oraz GSA. Te metody, mające na celu wyjaśnienie i interpretację procesów decyzyjnych podejmowanych przez algorytmy sztucznej inteligencji, są szczególnie istotne w kontekście rosnącej popularności i wszechstronności modeli sztucznej inteligencji, które coraz częściej mają status "czarnej skrzynki", o której procesach decyzyjnych niewiele wiemy. Projekt był motywowany ewolucją sztucznej inteligencji i coraz większym naciskiem na zrozumienie procesów, które kierują decyzjami AI. W ramach naszej pracy, skoncentrowaliśmy się na dwóch zestawach danych: Iris, które jest zbiorem numerycznym składającym się z pomiarów cech trzech gatunków irysów, oraz MNIST, zawierającym obrazy napisanych odręcznie cyfr. Te zestawy danych, dostarczyły nam platformy do oceny i porównania metod SHAP, LIME oraz GSA.

W ramach projektu stworzyliśmy dwa modele sztucznych sieci neuronowych (SSN) dla różnych zestawów danych Iris i MNIST. Wykorzystane modele uwzględniającą strukturę i naturę danych wejściowych. Wykorzystaliśmy szereg różnych technik przetwarzania, takich jak warstwy konwolucyjne, podpróbkiowanie i odrzucenie, aby dostosować się do złożoności danych wejściowych. Obydwa modele były kompilowane z wykorzystaniem różnych optymalizatorów, funkcji straty i metryk, w celu osiągnięcia najlepszych wyników dla danego zestawu danych.

Następnie, zastosowaliśmy metodę SHAP (SHapley Additive exPlanations) do interpretacji wyników modelu. Dzięki zastosowaniu koncepcji wartości Shapleya, byliśmy w stanie określić, jak dużo każda cecha przyczynia się do przewidywanej decyzji modelu. Przykładowe analizy dla zestawu danych Iris pokazują, że metoda SHAP jest bardzo skuteczna w identyfikowaniu kluczowych cech wpływających na decyzje modelu. Na przykład, dla wybranego irysa gatunku virginica, metoda SHAP pozwoliła nam zrozumieć, że

długość i szerokość płatka były najważniejszymi cechami wpływającymi na klasyfikację modelu. Dla każdej klasy w zestawie danych MNIST wybrano jeden rekord do analizy za pomocą techniki SHAP. Wyniki pokazały pewność modelu dla każdej cyfry, a także jak różne obszary obrazu wpływały na prawdopodobieństwo klasyfikacji do danej klasy. Czerwone obszary na wykresie zwiększały prawdopodobieństwo danej klasy, a niebieskie je zmniejszały. Użycie SHAP do analizy zestawów danych IRIS i MNIST pozwala na lepsze zrozumienie, jak modele uczenia maszynowego podejmują decyzje. SHAP pomaga w zrozumieniu, które cechy są najważniejsze dla predykcji modelu i jakie mają one konkretne wpływy. W przypadku zestawu danych IRIS, SHAP pomaga zrozumieć, które cechy (długość i szerokość płatków i działek kielicha) mają największy wpływ na klasyfikację różnych gatunków Irysów.

W przypadku zestawu danych Iris, LIME był używany do analizy jednego irysa gatunku virginica. Wyniki pokazały, że model z 98% pewnością zidentyfikował kwiat jako virginica, na podstawie długości płatka i szerokości działki kielicha. Podobnie jak w przypadku zestawu danych Iris, LIME był używany do analizy jednego obrazu z zestawu danych MNIST. Model poprawnie zidentyfikował obraz jako cyfrę dziewięć, a LIME pokazał, które obszary obrazu najbardziej wpłynęły na tę klasyfikację. Technika LIME pomaga wytłumaczyć, w jaki sposób modele sieci neuronowych podejmują decyzje, co pozwala na lepsze zrozumienie ich działania.

Analiza wrażliwości globalnej (GSA) za pomocą biblioteki SALib umożliwiła przeprowadzenie badania wpływu różnych danych wejściowych na wynik modelu dla zbiorów danych Iris i MNIST. W ramach GSA, skorzystaliśmy z metody Sobola, która dekomponuje wariancję na składowe przypisane poszczególnym zmiennym. Przeprowadziliśmy analizę wrażliwości dla danych Iris, definiując problem i generując próbki parametrów. W przypadku danych Iris, ujawniliśmy największy wpływ na wyniki dla par szerokości i długości płatków oraz szerokości i długości działek kielicha. Próbowaliśmy zastosować podobną procedurę dla danych MNIST, jednak okazało się, że metoda Sobola nie daje miarodajnych wyników dla obrazów złożonych z wielu pikseli. Teoretyczne podejście do poprawy wyników mogłoby polegać na zmniejszeniu liczby parametrów wejściowych poprzez wyciągnięcie z obrazów pewnych zagregowanych danych statystycznych. Inne potencjalne rozwiązanie, choć wymagające znacznych zasobów obliczeniowych i pamięciowych, to zwiększenie liczby próbek Sobolowskich. Wszystko to ukazuje złożoność i wyzwanie jakie niesie za sobą przeprowadzanie analizy wrażliwości dla różnych typów danych.