Data obrony projektu: 16.04.2023

Systemy Teleinformatyczne

Dokumentacja projektu:

DETEKCJA PLAGIATU TEKSTU PDF / DOCX / DOC „ANTYPLAGIAT”

Wykonali:

EF/S - ZI - Grupa P01

Delikat Eryk
Haja Agnieszka
Kołecki Karol

Cel projektu:

Projekt **Detekcja plagiatu tekstu *"Antyplagiat"*** to aplikacja desktopowa stworzona w języku Python przy użyciu biblioteki *tkinter*. Celem projektu jest wykrywanie plagiatów w tekście napisanym w języku angielskim wprowadzonym przez użytkownika lub wczytanym z pliku w formacie PDF lub DOC/DOCX.

Model został wytrenowany na około 15 000 liniach tekstu, co daje 3,78 MB danych.

Opis projektu:

Biblioteki:

Aplikacja korzysta z bibliotek takich jak: *pandas, numpy, scikit-learn, PyPDF2* oraz *docx* do obsługi plików oraz do tworzenia wektorów *tf-idf* i trenowania klasyfikatora *SVM.*

Algorytmy i narzędzia:

Algorytmy uczenia maszynowego (ML) to zestaw technik i narzędzi, które pozwalają na przetwarzanie i analizę dużych ilości danych w celu generowania automatycznych prognoz lub decyzji. Działają one na zasadzie uczenia się z danych i próby generalizacji w celu przewidywania wyników dla nowych danych.

Zastosowany został algorytmy klasyfikacji - służą do przypisywania etykiet do danych wejściowych na podstawie zdefiniowanych kryteriów. Etykieta 1 oznacza plagiat, a 0 nieplagiat.

Algorytm tworzy wektory tekstu (ang. text vector) w kontekście uczenia maszynowego (ML) jest to sposób reprezentacji tekstu za pomocą liczb lub wektorów numerycznych, które mogą być przetwarzane przez modele ML.

Narzędzie *vectorizer* przekształca każde zdanie w zbiorze treningowym na wektor numeryczny o wymiarze zależnym od liczby słów w słowniku narzędzia.

Wektor ten reprezentuje każde zdanie ze zbioru treningowego w przestrzeni numerycznej, gdzie każda współrzędna odpowiada liczbie wystąpień danego słowa w zdaniu. Ostatecznie, przekształcony zbiór treningowy jest zwracany jako macierz rzadka (ang. sparse matrix), która zawiera wektory numeryczne dla każdego zdania w zbiorze treningowym.

Dzięki temu przekształceniu można wykorzystać te wektory jako dane wejściowe do modelu uczenia maszynowego, który będzie w stanie nauczyć się rozpoznawania wzorców i przewidywać etykiety dla nowych, nieznanych wcześniej zdań.

Działanie algorytmów ML opiera się na przetwarzaniu danych wejściowych, które są podawane na wejście modelu, a następnie są one analizowane i wykorzystywane do uczenia się modelu. Po zakończeniu procesu uczenia maszynowego, model jest gotowy do użycia do przewidywania wyników dla nowych danych.

Działanie programu:

Aby program działał należy zainstalować wszystkie niezbędne biblioteki zawarte w pierwszych 11 liniach main.py. Oraz wykonać komendę w konsoli: *pip install python-docx.*

Po uruchomieniu programu pojawia się okno z polami tekstowymi oraz przyciskami. Pierwsze pole tekstowe służy do wprowadzenia tekstu, który chcemy przetestować pod kątem plagiatu, drugie pole służy do wyświetlenia wyniku testu. Po wprowadzeniu tekstu należy kliknąć przycisk "Sprawdź", aby rozpocząć testowanie.

Czym dłuższy tekst wejściowy tym lepiej radzi sobie algorytm klasyfikacji, więc została wprowadzona minimalna granica długości tekstu wynosząca 20 znaków. Jeśli ciąg znaków wejściowych jest za krótki funkcja sprawdzająca nie ruszy.

Gdy użytkownik kliknie przycisk "Sprawdź", program wykonuje następujące czynności:

1. Pobiera wprowadzony przez użytkownika tekst.
2. Ładuje dwa pliki CSV zawierające przykłady tekstów z plagiatami i bez plagiatów.
3. Usuwa z obu plików zbędne kolumny.
4. Łączy dane z obu plików w jeden DataFrame.
5. Dzieli dane na zbiór treningowy i testowy.
6. Tworzy wektory tf-idf na podstawie tekstu w zbiorze treningowym.
7. Trenuje klasyfikator SVM na podstawie wektorów tf-idf.
8. Testuje klasyfikator na zbiorze testowym i oblicza dokładność klasyfikacji.
9. Testuje klasyfikator na tekście wprowadzonym przez użytkownika i zwraca prawdopodobieństwo, że tekst jest plagiatem.
10. Na podstawie wyniku testu wyświetla informację o tym, czy tekst jest plagiatem czy nie. Jeżeli Prawdopodobieństwo plagiatu jest większe niż 60% program uznaje test za plagiat.

Program pozwala również na załadowanie pliku PDF lub Worda zawierającego tekst do pola tekstowego, który następnie będzie testowany. Po kliknięciu przycisku "Load File" program prosi użytkownika o wybranie pliku PDF lub Worda, wczytuje jego zawartość do pola tekstowego i rozpoczyna testowanie.

Ostatecznie, w zależności od wyniku testu, program wyświetla informację **"PLAGIAT"** lub **"NIEPLAGIAT"** w polu tekstowym oraz zmienia kolor tła pola tekstowego na czerwony lub zielony.

Przy każdym uruchomieniu funkcji sprawdzającej program uczy model od nowa, co przekłada się na nieznaczne zwiększenie czasu wyświetlania wyniku. Pozwala to na modyfikacje zbiorów danych, lub całkowitą ich zamianę przy minimalnych nakładzie pracy.

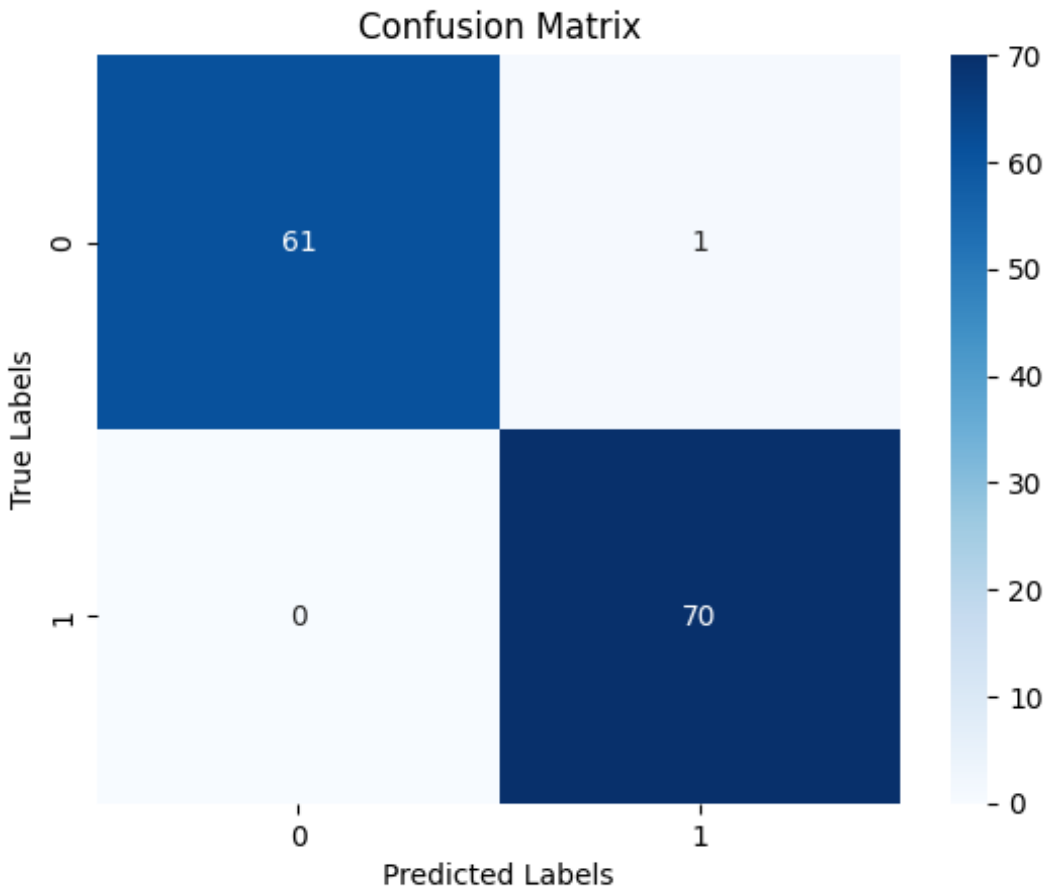
Wykresy:

1. Macierz pomyłek

Wiersze macierzy reprezentują prawdziwe etykiety klas, a kolumny reprezentują przewidywane etykiety klas. Na przykład, element w pierwszym wierszu i pierwszej kolumnie oznacza, że 61 próbek zostało sklasyfikowanych jako klasa 0 (Brak plagiatu), a 1 próbka została sklasyfikowana jako klasa 1 (Plagiat).

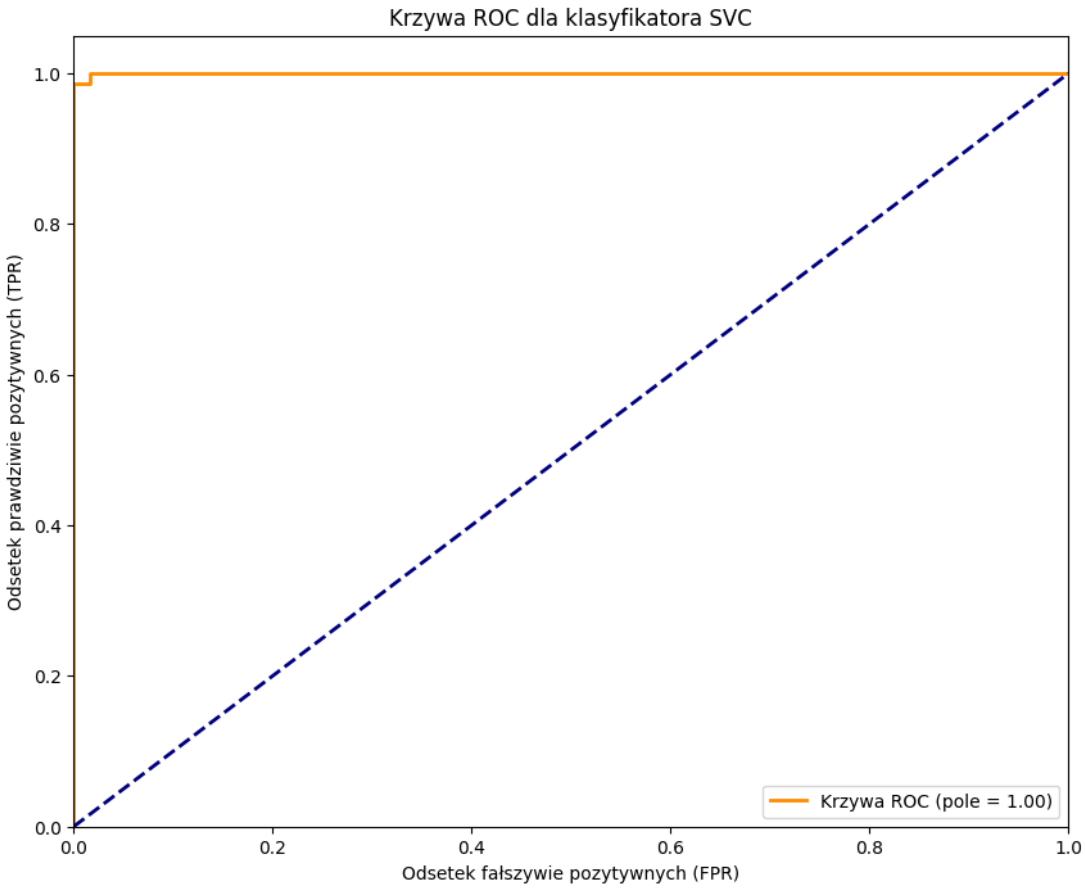
Podobnie, element w drugim wierszu i drugiej kolumnie (70) oznacza, że 70 próbek zostało poprawnie sklasyfikowanych jako klasa 1, a żadna próbka nie została błędnie sklasyfikowana jako klasa 0.

Można zauważyć, że klasyfikator osiągnął bardzo dobry wynik, ponieważ tylko jedna próbka została błędnie sklasyfikowana.



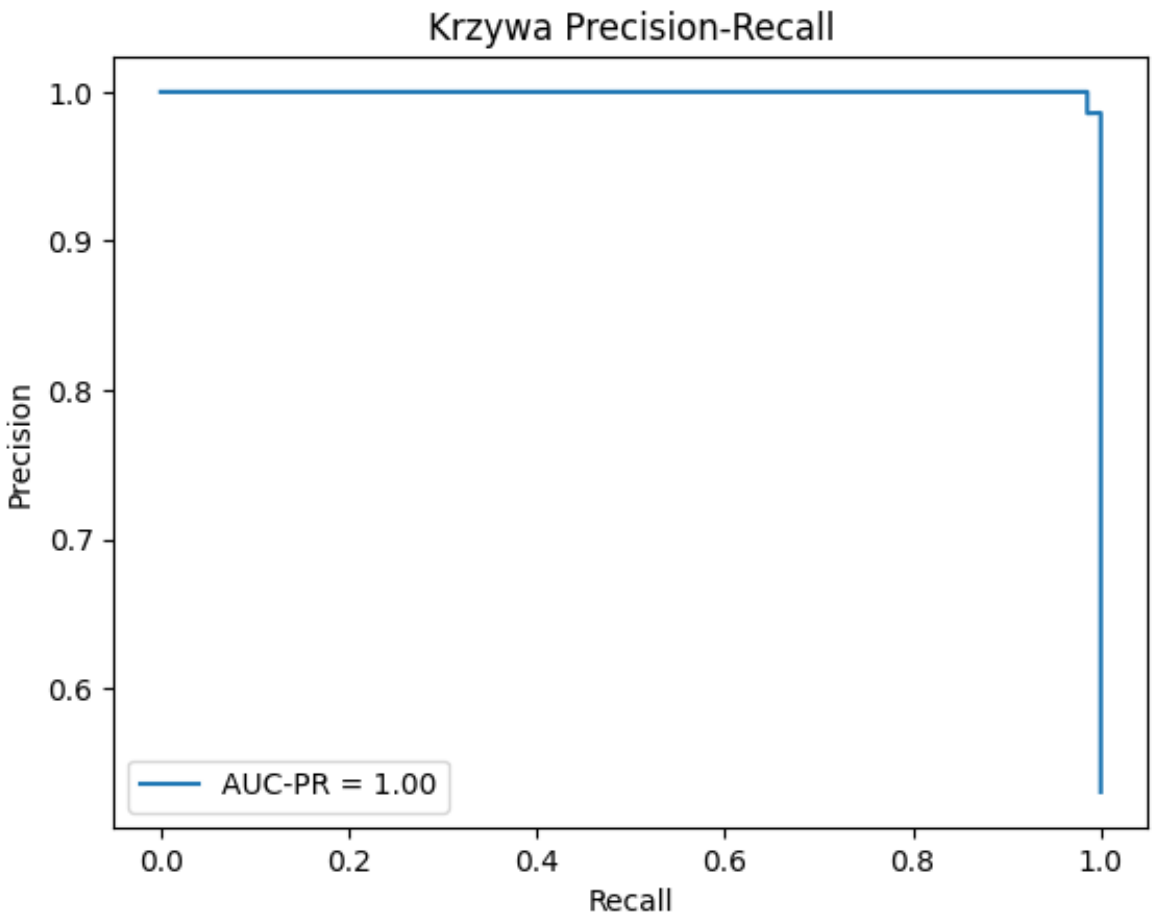
2. Krzywa ROC

Krzywa ROC to wykres, który przedstawia wydajność modelu klasyfikacyjnego w zależności od wartości progu. Na osi X wyświetla się odsetek fałszywie pozytywnych (FPR), a na osi Y odsetek prawdziwie pozytywnych (TPR). Im wyżej znajduje się krzywa na wykresie, tym lepsza wydajność modelu. Krzywa optymalna to taka, która znajduje się w lewym górnym rogu wykresu. Pole pod krzywą ROC (AUC) można interpretować jako miarę jakości klasyfikatora - im wyższe pole, tym lepszy klasyfikator.



3. Krzywa Precision-Recall

Wykres Precision-Recall jest użytecznym narzędziem do oceny klasyfikatora w przypadku niezrównoważonych klas, gdyż bierze pod uwagę zarówno precyzję (jak wiele wykrytych pozytywnych przypadków było faktycznie poprawnie sklasyfikowanych) oraz recall (jak wiele pozytywnych przypadków zostało wykrytych). Im większa powierzchnia pod krzywą Precision-Recall, tym lepsza jakość klasyfikacji.

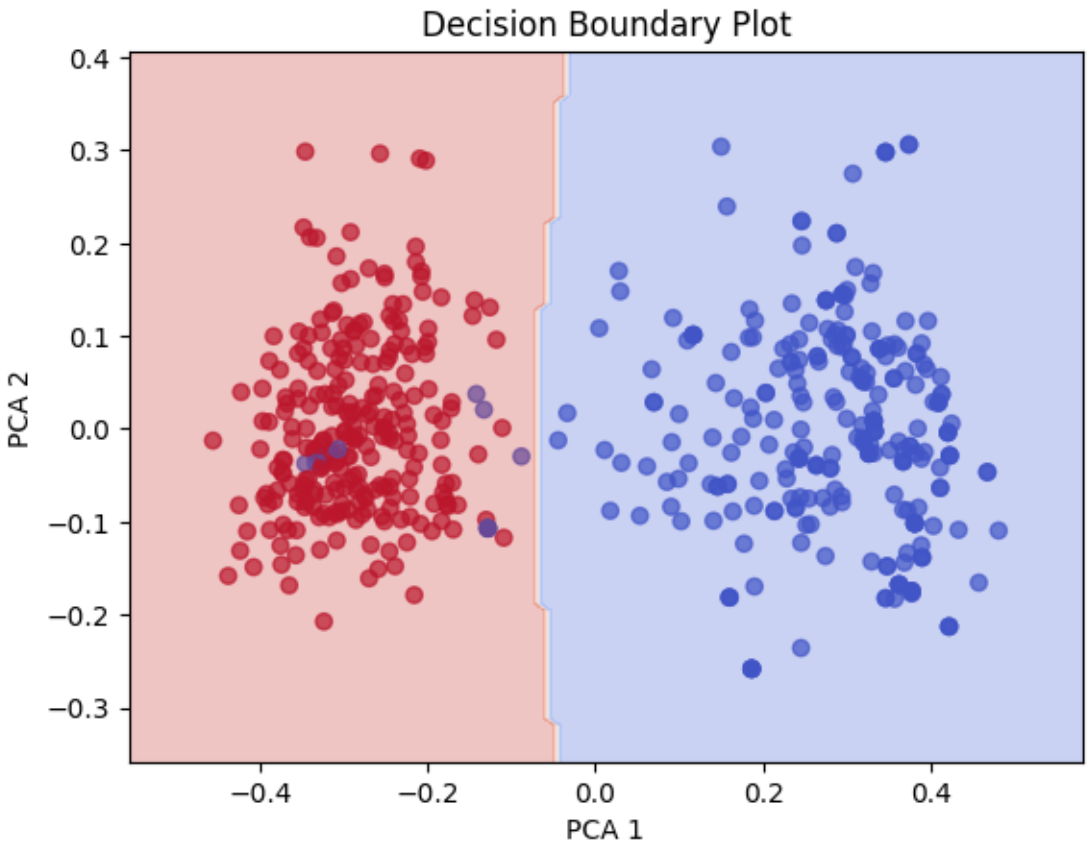


4. Decision boundary plot

Decision boundary plot to wykres, na którym przedstawione są granice decyzyjne klasyfikatora dla dwóch zmiennych objaśniających (cech). Wykres ten pozwala zobaczyć, w jaki sposób klasyfikator rozdziela różne klasy obiektów na podstawie wartości cech.

Na wykresie każda próbka z danych treningowych jest oznaczona kolorem w zależności od przypisanej jej klasy. Granica decyzyjna jest wyznaczona jako linia lub krzywa, która dzieli przestrzeń cech na dwie części, odpowiadające dwóm klasom. W przypadku klasyfikacji binarnej granica decyzyjna oddziela obserwacje przypisane do klasy pozytywnej od tych, które należą do klasy negatywnej.

Decision boundary plot umożliwia wizualizację działania klasyfikatora oraz ocenę jego jakości, np. poprzez sprawdzenie, czy granice decyzyjne są poprawne i czy klasyfikacja jest dokładna. Wykres ten może także pomóc w wykryciu problemów związanych z przeuczeniem lub niedouczeniem klasyfikatora.



Wkład zespołu w projekt:

- znalezienie i odpowiednie zaimplementowanie biblioteki pozwalającej na ML,

- znalezienie zbiorów danych na których będziemy uczyć model (musiały być możliwie jak największe, ponieważ ML na tekście wymaga wielu danych, aby uzyskać zadowalający efekt),

- opracowanie odpowiedniego etykietowania danych użytych do trenowania modelu,

- testowanie algorytmów i dobieranie parametrów w celu uzyskania jak największej trafności, która teraz sięga 98%,

- stworzenie GUI, podglądu danych wejściowych oraz możliwości importowania tekstu do sprawdzania, z plików w formacie PDF lub DOC/DOCX.