1. Opis problemu.

Tematem poruszonym w projekcie jest problem klasyfikacji utworów muzycznych do odpowiadających im gatunków. Projekt dotyka zatem tematyki przetwarzania cyfrowych sygnałów dźwiękowych. Celem pracy jest sporządzenie kodu umożliwiającego przyporządkowanie dowolnemu utworowi najbliższego mu gatunku muzycznego (z ograniczeniem wynikającym z liczby przyjętych rodzajów).

2. Przyjęta metodyka.

Prace w projekcie zostały podzielone na 3 części:

- ekstrakcja cech ścieżki dźwiękowej,
- przygotowanie danych uczących klasyfikator,
- implementacja predyktora.

Zostaną one opisane w powyższej kolejności.

1) Wydobycie indywidualnych cech utworów.

Każdy dźwięk charakteryzuje się swoimi indywidualnymi cechami, takimi jak głośność, wysokość, tembr (barwa) i innymi, mniej naturalnymi (np. liczba przejść przez zero). Do narzędzi ułatwiających identyfikację dźwięku należy również interpretacja rozkładu energii w sygnale, co zakodowane jest w parametrach mel-cepstralnych (*ang. MFCC – mel frequency cepstral coefficients*).

Cechy cepstralne zawierają informacje o częstotliwości zmian w różnych pasmach widma sygnału. Ich główną zaletą jest wprowadzenie możliwości rozdzielenia dźwięku na dwie formujące go jednostki sygnałowe – źródło oraz filtr. W przypadku ludzkiej mowy źródłem są struny głosowe, a filtrem, nadającym dźwiękowi dodatkowe parametry jest tzw. kanał (trakt) głosowy.

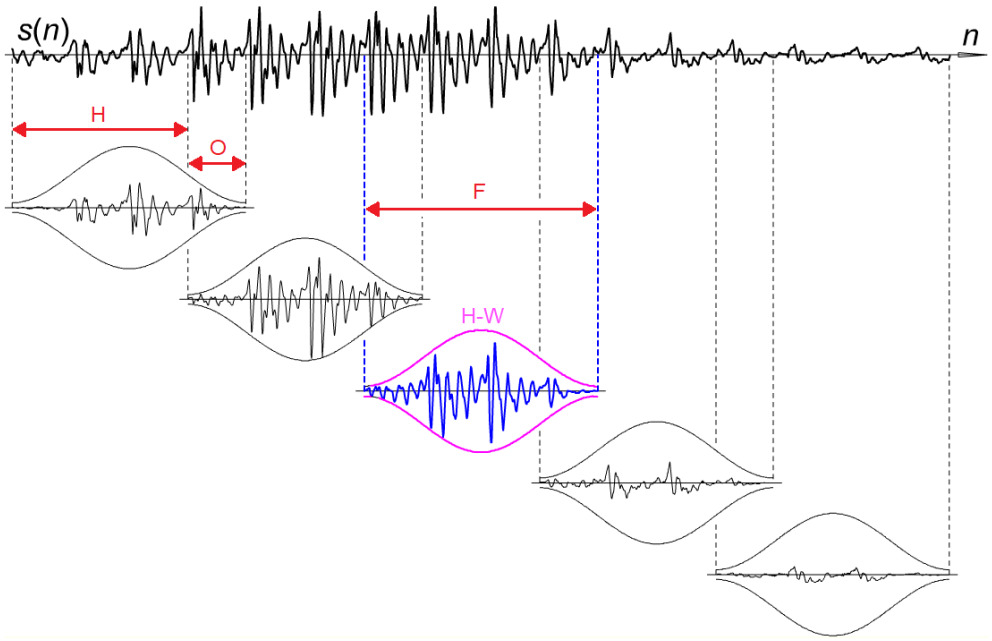
Współczynniki MFCC umożliwiają jednym słowem relatywnie łatwą separację źródła dźwięku od jego filtru. Dodatkowo, umożliwiają interpretację rodzaju dźwięku. Przykładowo, dodatnie wartości współczynników świadczą (w mowie) o obecności spółgłosek zwarto-otwartych, obecnych przy niskich częstotliwościach, natomiast ujemne o występowaniu spółgłosek szczelinowych przy częstotliwościach wysokich. Generalizując problem, za pomocą MFCC można określać trend występujący w ścieżce dźwiękowej. Przy zagadnieniu rozpoznawania gatunków muzycznych, różne style powinny charakteryzować się unikatowymi trendami. Klasyfikacji można zatem dokonywać na podstawie cech zawartych w dziedzinie cepstralnej.

Proces ekstrakcji MFCC składa się z kilku kroków. W projekcie odpowiedzialny za tę funkcjonalność jest plik *feature_extractor.py.*

Pierwszym krokiem jest podzielenie sygnału dźwiękowego na nachodzące na siebie ramki. Za długość ramki (*frame size*) arbitralnie przyjęto 800 próbek, co w oparciu o częstotliwość próbkowania utworów znalezionych na potrzeby projektu równą 20.05kHz, daje około 35ms na ramkę. Wprowadzony również został współczynnik nakładania się na siebie ramek (*overlap)* równy 0.2. Parametr skoku (*hop size)* obliczany jest na podstawie rozmiaru ramki i overlap'u. Określa on liczbę próbek, pomiędzy kolejnymi ramkami.

Mechanizm dzielenia utworu na nachodzące na siebie ramki wprowadzony został ze względu na możliwość pojawiania się nagłych skoków częstotliwości na granicach ramek, co mogłoby skutkować pojawieniem się zakłóceń przekładających się na przekłamanie widma. Dodatkowo, każda ramka splatana jest z oknem czasowym o długości równej długości ramki. Za funkcję okna przyjęto funkcję Hann'a. Skutkiem takiego zabiegu jest wytłumienie próbek sygnały na krańcach ramki, co przekłada się na uzyskanie gładkich, niezakłóconych przejść pomiędzy

ramkami. W celu zabezpieczenia początku i końca całego utworu przed wyzerowaniem przez okno czasowe, zastosowano padding o długości połowy ramki, który wydłuża utwór o lustrzane odbicie próbek na jego początku i końcu. W związku z tym, na środek okna Hann'a przypada pierwsza oraz ostatnia próbka oryginalnego utworu. Dobór odpowiednich parametrów tworzących ramki stanowi fundament poprawności działania całego projektu. Na rysunku 1 przedstawiony został proces dzielenia utworu na ramki. W kodzie programu odpowiedzialna jest za to funkcja *framing()*.



Rys.1. Proces dzielenia utworu na ramki (H-*hop size*, O-*overlap*, F-*frame size*, H-W-*okno Hann'a)*

Kolejnym krokiem jest obliczenie dla każdej ramki dyskretnej transformaty Fouriera jej próbek, a następnie obliczenie mocy każdej przekształconej próbki przeskalowanej do postaci logarytmicznej. W ten sposób otrzymywany jest rozkład mocy sygnału w dziedzinie częstotliwości. Krok ten ma na celu imitację ludzkiego słuchu, który określa, jakie przedziały częstotliwości niosą ze sobą więcej energii.

Z uwagi na charakterystykę ludzkiego słuchu, który jest bardziej czuły dla niższych częstotliwości (różnica pomiędzy 300Hz a 400Hz jest dużo bardziej zauważalna niż różnica pomiędzy 2600 a 2700Hz), wprowadzana jest skala melowa. Jest to skala wysokości dźwięku określająca subiektywny odbiór poziomu dźwięku przez ucho ludzkie względem obiektywnej skali pomiaru częstotliwości dźwięku w hercach. Przekształcenie dane jest następującym wzorem:

$$f_{mel} = 2595 \cdot log_{10}\left(1 + \frac{f}{700}\right) \quad (1)$$

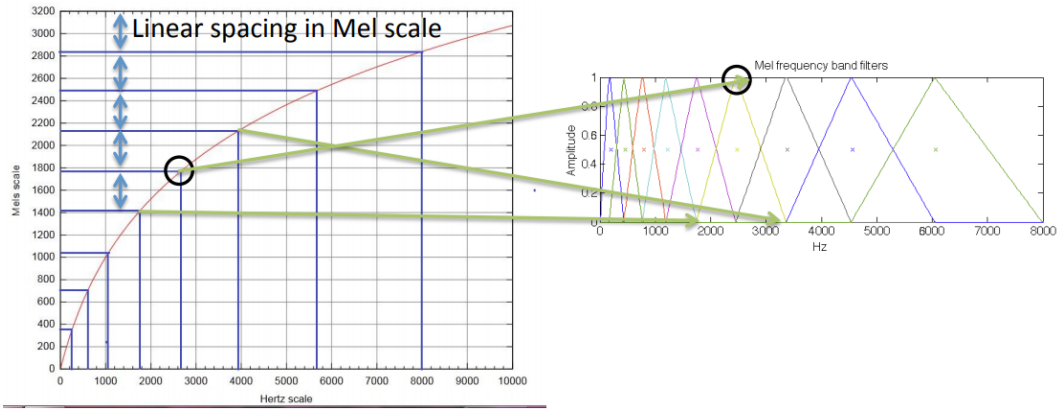
gdzie f_{mel} oznacza częstotliwość w skali melowej, a f częstotliwość w skali liniowej.

W celu odwzorowania tego zjawiska, w następnym kroku wprowadza się tzw. zespoły filtrów melowych. Tworzone są one na podstawie dwóch częstotliwości granicznych. W projekcie jako dolną częstotliwość przyjęto 0Hz, a za górną – połowę częstotliwości próbkowania, jako że zgodnie z twierdzeniem o próbkowaniu, jest to częstotliwość Nyquista. Punkty te zgodnie ze wzorem (1) przekształca się do przestrzeni melowej, po czym przestrzeń pomiędzy nimi wypełnia się równomiernie rozłożonymi punktami, których liczbę określa n zespołów filtrów melowych. W projekcie przyjęto, że będzie to 10 filtrów. Następnie powraca się do wartości w skali liniowej i konstruuje zespoły filtrów zgodnie ze wzorem:

$$H_n(k) = \begin{cases} 0, & k<f(n-1) \\ \frac{2(k-f(n-1))}{f(n)-f(n-1)}, & f(n-1)\leq k\leq f(n) \\ \frac{2(f(n+1)-k)}{f(n+1)-f(n)}, & f(n)<k\leq f(n+1) \\ 0, & k>f(n+1) \end{cases} \quad (2)$$

gdzie $H_n(k)$ oznacza wagę przypisaną k-temu przedziałowi widma mocy odnoszącego się do n-tego filtru.

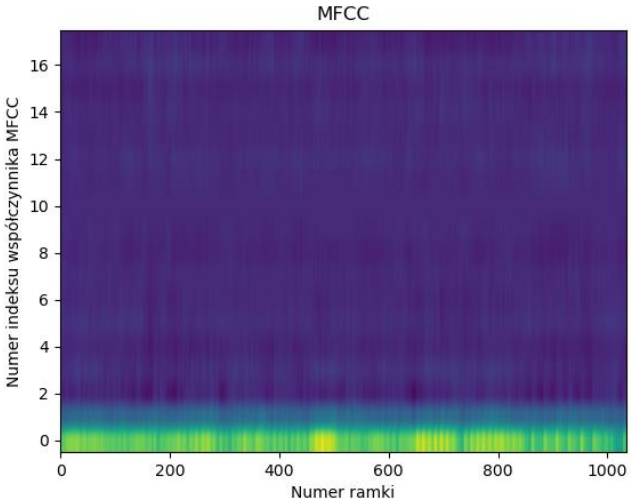
Proces ten przedstawiony został na rysunku 2. W kodzie odpowiedzialne za niego są funkcje *mel_filter_point_coords()* oraz *mel_filters_construct()*. Po wygenerowaniu zespołu filtrów dokonuje się ich normalizacji poprzez podzielenie wag przez długości odpowiadających ich pasm. Następnie z powstałym bankiem filtrów mnoży się macierz zawierającą widmo mocy spróbkowanego pliku audio.



Rysunek 2. Proces tworzenia zespołu filtrów melowych

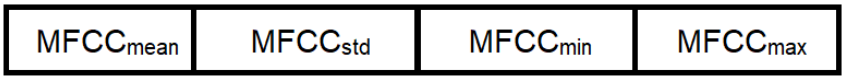
Ostatnim krokiem prowadzącym do otrzymania MFCC jest zastosowanie dyskretnej transformacji cosinusowej typu III na macierzy wynikowej z poprzedniego mnożenia. W ten sposób dla każdej ramki otrzymuje się zespół współczynników zawierających tylko część rzeczywistą. Transformacji cosinusowej używa się z powodu bardziej wydajnego działania – w tym kroku można by skorzystać również z przekształcenia Fouriera, lecz związane by to było z generowaniem większej liczby współczynników opisujących widmo, co byłoby nieefektywne.

Wynikiem działania transformacji cosinusowej jest macierz zawierająca wszystkie ramki. Każda ramka natomiast przechowuje współczynniki MFCC określające rozmieszczenie widma mocy w danej części utworu określonej indeksem próbek zawartych w ramce. Oznacza to powrót do dziedziny czasu dyskretnego. Do przeprowadzania dalszych obliczeń zwykle używa się pierwszych kilkunastu współczynników MFCC, przy czym pierwszy z nich określa średnią energię w skali logarytmicznej sygnału wejściowego, stąd jest często pomijany. Graficzna reprezentacja pierwszych 18 współczynników MFCC w przykładowym utworze analizowanym w projekcie została pokazana na rysunku 3.



Rysunek 3. Graficzna reprezentacja współczynników MFCC

Dla obliczonych MFCC można następnie zebrać opisujące je dane statystyczne, takie jak: wartość średnia, odchylenie standardowe, minimalna oraz maksymalna wartość. Dokonując tych czterech operacji dla każdej ramki, a następnie łącząc wyniki w jedną całość, otrzymuje się wektor o wymiarze [1x4*N*], gdzie *N* to liczba branych pod uwagę MFCC. Wektor ten przedstawiony został na rysunku 4.



Rysunek 4. Wektor cech utworu

Wektor ten jest wynikiem zwracanym przez skrypt *feature_extractor.py.*

2) Przygotowanie danych uczących.

Jako zbiór danych w projekcie został użyty standardowy dataset GITZAN zawierający tysiąc 30-sekundowych nagrań podzielonych na 10 gatunków muzycznych.

W projekcie zdecydowano o podziale danych uczących na maksymalnie 7 gatunków (country, *classical*, *hiphop*, *jazz*, *metal*, pop, *rock*). Skrypt *feature_generator()* odpowiedzialny jest za wygenerowanie macierzy cech i odpowiadających im gatunków. W procesie tworzenia zbioru uczącego dla każdej piosenki pobieranej z folderu z danymi uczącymi generowany jest jej wektor cech, a w osobnej macierzy na odpowiadającym indeksie przyznawany jest numer reprezentujący gatunek. Zapisywane są one również do osobnych plików .npy.

Jako liczbę piosenek przeznaczonych na uczenie klasyfikatora przeznaczono 30 dla danego gatunku, liczba ta jednak może być dobrowolnie zmieniana w zakresie ograniczonym przez zdrowy rozsądek użytkownika oraz liczbę dostępnych utworów (dla sensowności rozwiązań ograniczono zbiór do 50 plików).

3) Właściwy klasyfikator.

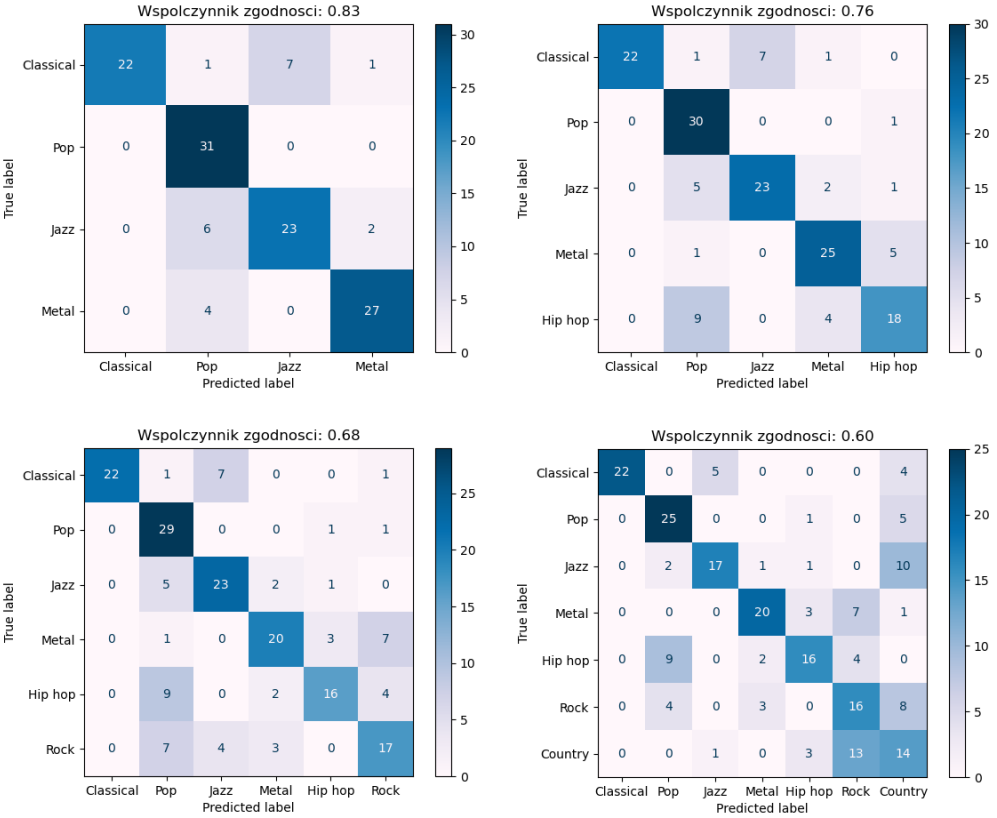
Jako klasyfikator utworów muzycznych na podstawie ich cech związanych z MFCC wybrany został naiwny klasyfikator bayesowski. W celu osiągnięcia najlepszych wyników zdecydowano o skorzystaniu z gotowego rozwiązania dostępnego w bibliotece *scikit-learn*. Decyzja została poparata faktem, iż opiera się on na założeniu wzajemnej niezależności danych i kategorycznej klasyfikacji danych, co zgodne jest z problemem projektowym.

Na wejście klasyfikatora w skrypcie *classify.py* podaje się wartości otrzymywane w wyniku działania omówionego wyżej modułu *feature_generator()*. Następnie, w celu dokonania

klasyfikacji, jako argument metody *predict()* modelu klasyfikatora podaje się wektor cech testowego utworu, otrzymany przy pomocy działania modułu ekstrakcyjnego. Klasyfikator następnie oblicza przynależność ścieżki dźwiękowej do danego gatunku bazując na algorytmie gaussowskiego naiwnego Bayesa.

3. Prezentacja wyników.

W celu przetestowania działania programu klasyfikacyjnego zdecydowano o wprowadzeniu czterech zbiorów testowych, zawierających 31 utworów muzycznych odpowiednio dla 4, 5, 6 i 7 gatunków muzycznych. Jako graficzna reprezentacja wyników została wybrana tablica pomyłek, oferująca najbardziej miarodajny i naturalny wgląd w rezultaty. Dodatkowo, wprowadzony został współczynnik zgodności, określający liczbę poprawnych dopasowań w stosunku do liczby wszystkich klasyfikacji. Zestawienie wyników zostało przedstawione na rysunku 5.



Rysunek 5. Wyniki klasyfikacji dla różnej liczby gatunków

Testy programu wykazały, iż jest on w stanie poprawnie zaklasyfikować większość utworów, nawet w przypadku 7 różnych gatunków. Na podstawie wyników można zauważyć, iż niektóre gatunki (*pop*, *classical*) wykazują się zdecydowanie bardziej unikatowym rozkładem energii w widmie od pozostałych gatunków. Zgodnie z oczekiwaniami, poprawność klasyfikacji maleje odwrotnie proporcjonalnie do liczby branych pod uwagę gatunków muzycznych. Jest to związane z zacieraniem się granic zawartych danych statystycznych, przez co klasyfikator bazujący na najprostszych metodach probabilistycznych przestaje być wydajny. W celu ulepszenia pracy klasyfikatora, należałoby go wyposażyć w mechanizm wektorów nośnych (SVM), lub zaprojektować klasyfikującą sieć neuronową.