Spis treści

Wstęp

W dzisiejszych czasach praktycznie wszędzie mamy do czynienia z coraz to większą ilością danych, wraz z rozwojem postępu rośnie też potrzeba na analizę przetwarzanych informacji. W odpowiedzi na to wyzwanie często stosuje się uczenie maszynowe, które jest gałęzią sztucznej inteligencji. Metody tej dziedziny pozwalają komputerom na analizę danych, wykrywanie wzorców i podejmowanie decyzji bez konieczności wyraźnego programowania. W swojej pracy podjąłem się tematu analizy spektrogramów różnych gatunków muzyki i wytrenowanie modelu, który będzie podejmował decyzję do jakiego katalogu przyporządkować dany utwór.
Celem tej pracy magisterskiej jest zgłębienie i zrozumienie technik uczenia maszynowego oraz ich potencjalnych zastosowań. Analizując różnorodne metody uczenia maszynowego, w tym uczenie nadzorowane, nienadzorowane i ze wzmocnieniem, oraz ich zastosowania w praktyce, praca ta ma na celu dostarczenie kompleksowego spojrzenia na aktualny stan wiedzy oraz perspektywy rozwoju tej dziedziny.

Rozdział 1

Uczenie maszynowe

1.1 Typy uczenia maszynowego

1.1.1 Uczenie nadzorowane

Uczenie maszynowe nadzorowane to rodzaj uczenia maszynowego, w którym model jest trenowany na zbiorze danych, który zawiera pary wejście-wyjście oraz odpowiadające im etykiety lub odpowiedzi. Celem jest nauczenie modelu mapowania z danego zbioru danych wejściowych na pożądane wyjście, co pozwala modelowi przewidywać poprawne etykiety dla nowych danych wejściowych. W uczeniu maszynowym nadzorowanym, każdy przykład w zbiorze danych treningowych składa się z cech (wejście) oraz odpowiadającej mu etykiety (wyjście). Model jest trenowany na tych danych poprzez dostosowywanie swoich parametrów w trakcie iteracji, aby minimalizować błąd między przewidywanymi etykietami a rzeczywistymi etykietami w zbiorze treningowym. Przykłady algorytmów uczenia maszynowego nadzorowanego obejmują:

1. **Regresję liniową:** Algorytm ten próbuje znaleźć liniową relację między cechami a docelowymi wartościami ciągłymi.

2. **Klasyfikację za pomocą maszyn wektorów nośnych (SVM):** SVM jest algorytmem klasyfikacji, który szuka hiperpłaszczyzny w wielowymiarowej przestrzeni, która najlepiej separuje punkty danych różnych klas.

3. **Algorytmy drzew decyzyjnych:** Te algorytmy budują drzewo decyzyjne na podstawie cech danych i ich etykiet, które mogą być wykorzystywane do przewidywania etykiet dla nowych danych.

4. **Sieci neuronowe:** Sieci neuronowe są modelami, które naśladują sposób działania ludzkiego mózgu poprzez składanie wielu neuronów w warstwy i przekazywanie informacji między nimi.

Uczenie maszynowe nadzorowane jest szeroko stosowane w różnych dziedzinach, takich jak rozpoznawanie obrazów, przetwarzanie języka naturalnego, analiza danych biomedycznych, predykcja rynków finansowych i wiele innych.

1.1.2 Uczenie nienadzorowane

Kolejnym rodzajem uczenia maszynowego jest uczenie nienadzorowane, które jak nazwa wskazuje wymaga minimalnej ingerencji człowieka. Ten Typ Machine learningu skupia się na wyszukiwaniu wzorców bez wcześniej zdefiniowanych etykiet. Dwie główne metody stosowane w uczeniu nienadzorowanym to analiza składowych głównych oraz analiza skupień.

Analiza składowych głównych to technika redukcji wymiarowości danych, która pomaga zidentyfikować i usunąć cechy, które przenoszą najmniej informacji. Dzięki temu możemy skupić się na istotnych aspektach danych, co ułatwia analizę i interpretację.

Z kolei **analiza skupień** to metoda grupowania zestawów danych na podstawie podobieństw między nimi. Poprzez identyfikację podobieństw, możemy grupować dane ze wspólnymi cechami lub atrybutami. Istnieje możliwość wykrycia nietypowych danych, które nie pasują do żadnej grupy, co może być przydatne w identyfikacji anomalii w zbiorze danych. Analiza skupień pozwala nam odkrywać ukryte struktury w danych i segmentować je w bardziej zrozumiały sposób.

1.1.3 Uczenie półnadzorowane

Uczenie półnadzorowane jest w pewnym sensie pomiędzy uczeniem nadzorowanym a nienadzorowanym. Maszyna dostaję zbiór: $< x, y, z >$ gdzie x definiujemy jako przykłady oznaczone, y zbiór etykiet a z zbiór przykładów nieposiadających etykiet.

1.2 Klasyfikacja w uczeniu maszynowym

W uczeniu maszynowym klasyfikacja jest problemem polegającym na określeniu do której z zestawów kategorii należy nowo obserwowana wartość. Zadaniem algorytmu klasyfikującego jest utworzenie modelu przy pomocy, którego możliwe będzie przyporządkowanie poszczególnych danych do konkretnej kategorii. Występują również wieloetykietowej, w której przykłady należeć mogą do wielu klas jednocześnie, jednak zagadnienie to nie jest tematem niniejszej pracy. Do zastosowań klasyfikatorów w praktyce zalicza się między innymi:

- filtrowanie spamu
- diagnostyka medyczna
- klasyfikacja biologiczna
- rozpoznawanie obrazu
- analiza nagrań z kamer

W zależności od tego z jakim obszarem zastosowania mamy do czynienia, będziemy wymagać od naszego algorytmu innej dokładności np. gdy diagnozujemy chorobę to chcemy aby model był bardzo dokładny.i. Z kolei w ekonomicznych często wymaga się interpretowalności modeli, co dyskwalifikuje pewne algorytmy charakteryzujące się wysoką dokładnością. Ogólny podział klasyfikacji przyjmuje się ze względu na ilość etykiet:

- binarne
- wieloklasowe

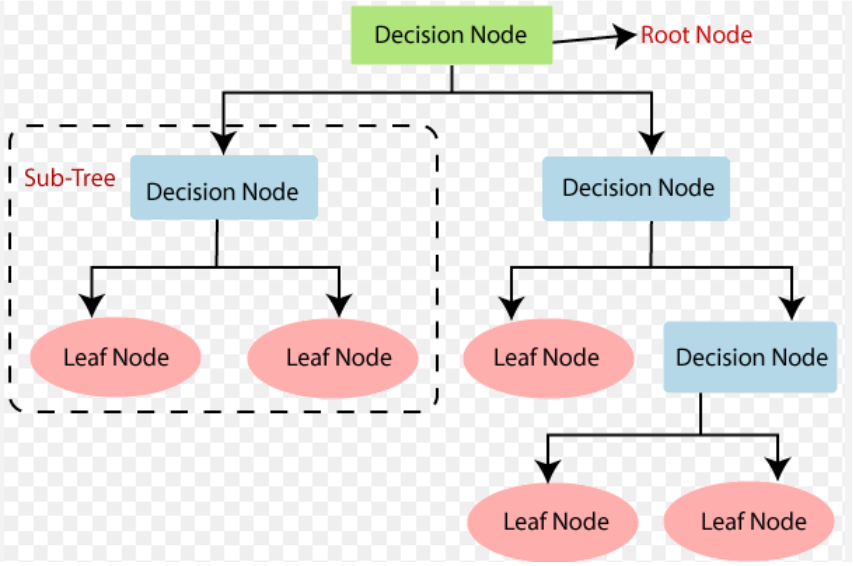
1.3 Przykłady algorytmów rozwiązujących problem klasyfikacji

1.3.1 Drzewa decyzyjne

Drzewa decyzyjne są to modele predykcyjne, w których tworzona jest struktura drzewa zawierająca 3 własności:

- **węzeł**: Każdy węzeł reprezentuje test na wartości atrybutu.
- **krawędź**: Każda krawędź odpowiada wynikowi tego testu.
- **liście**: Każdy liść zawiera etykietę klasy lub wartość numeryczną.

Drzewa decyzyjne dzielą zbiór danych na coraz mniejsze podzbiory w oparciu o wartości atrybutów, następnie algorytm przechodzi przez drzewo testując wartości atrybutów i poruszając się w dół drzewa na podstawie reguł podziału, aż do osiągnięcia liścia, który zawiera prognozę.



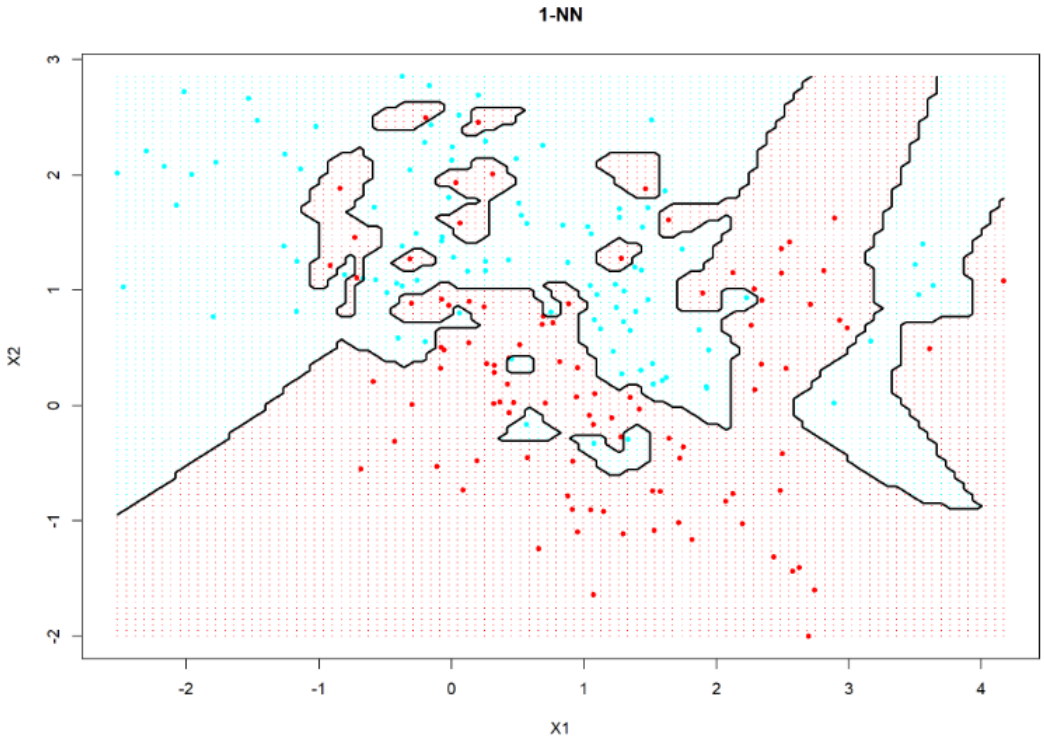
Rysunek 1.1: Przykład drzewa decyzyjnego

W praktyce drzewa decyzyjne często są używane pojedynczo lub jako elementy bardziej złożonych modeli zespołowych, takich jak lasy losowe (random forests) czy gradient boosting machines. Dzięki ich prostocie, interpretowalności i skuteczności są one powszechnie stosowane w różnych dziedzinach, od biznesu po nauki przyrodnicze

1.3.2 Metoda k-najbliższych sąsiadów

Technika k najbliższych sąsiadów przewiduje wartość zmiennej wynikowej na podstawie k najbliższych obserwacji zbioru uczącego. Może być wykorzystywana, zarówno do zadań klasyfikacyjnych, jak i regresyjnych. W obu przypadkach predykcja dla nowych wartości predyktorów przebiega podobnie. Niech x_0 będzie obserwacją, dla której poszukujemy wartości zmiennej wynikowej y_0. Na podstawie zbioru obserwacji $x \in T$ zbioru uczącego wyznacza się k najbliższych sąsiadów, gdzie k jest z góry ustaloną wartością. Następnie, jeśli zadanie ma charakter klasyfikacyjny, to y_0 przypisuje się modę zmiennej wynikowej obserwacji będących k najbliższymi sąsiadami. W przypadku zadań regresyjnych y_0 przypisuje się średnią lub medianę.

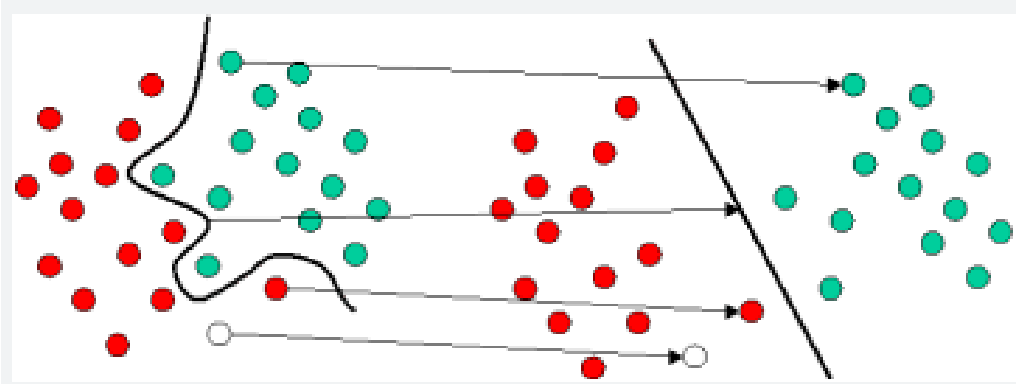
Olbrzymie znaczenie dla wyników predykcji na podstawie metody kNN ma dobór metryki. Nie istnieje obiektywna technika wyboru najlepszej metryki, dlatego jej doboru dokonujemy metodą prób i błędów. Należy dodatkowo pamiętać, że wielkości mierzone x mogą się różnić zakresami zmienności, a co za tym idzie, mogą znacząco wpłynąć na mierzone odległości pomiędzy punktami. Dlatego zaleca się standaryzację zmiennych przed zastosowaniem metody kNN.



Rysunek 1.2: Rozkład KNN

1.3.3 Maszyny wektorów nośnych

U podstaw metody wektorów nośnych (Support Vector Machines - SVM) leży koncepcja przestrzeni decyzyjnej, którą dzieli się budując granice separujące obiekty o różnej przynależności klasowej, czego przykład widzimy na poniższym rysunku. Mamy tu dwie klasy kółek: zielone i czerwone. Linia graniczna rozdziela je wyraźnie. Nowy, nieznany obiekt, jeżeli znajdzie się po prawej stronie granicy zostanie zaklasyfikowany jako zielony, a w przeciwnym wypadku, jako czerwony.



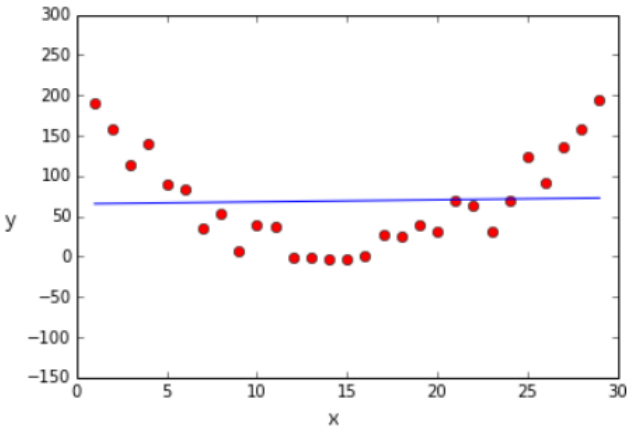
Rysunek 1.3: SVM

1.4 Problemy klasyfikacji

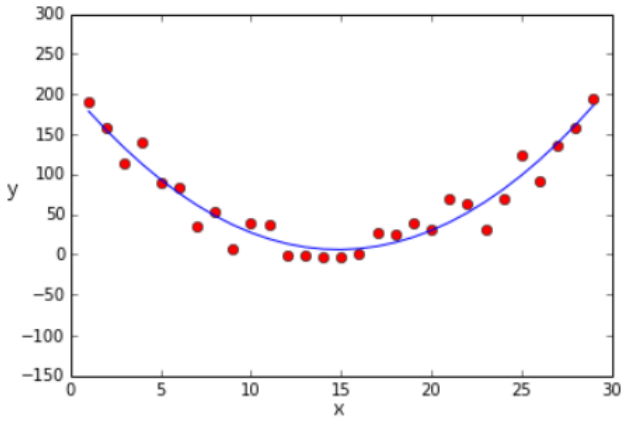
Główny problem jaki może wystąpić przy uczeniu modelu predykcyjnego jest nadmierne dopasowanie, oznacza to że model za bardzo odwzorowuje dane treningowe. Mamy tutaj do czynienia z bardzo małym błędem treningowym, co może powodować złudne wrażenie dobrej jakości modelu. To zjawisko wystepuje również w innych modelach predykcyjnych niż klasyfikatory. Takie zjawisko będzie dobrze zilustrować na przykładzie regresji liniowej. W problemie tym, etykiety przykładów są liczbami rzeczywistymi, a zadaniem algorytmu jest wyznaczenie funkcji h:

$$h(x) : x \mapsto y \in R$$

Gdy mówimy o nadmiernym lub niedostatecznym dopasowaniu warto wspomnieć o takich pojęciach jak obciążenie oraz wariancja modeli predykcyjnych. Pierwsze odnosi się do przypadku gdy model niedostatecznie odwzorowuje dane treningowe. Przeciwną sytuacją jest duża wariancja, która oznacza że wybór konkretnego zestawu danych treningowych bardzo mocno wpływa na postać modelu. Poniżej przedstawiam oba przypadki pokazane na rysunkach.



Rysunek 1.4: Niedostateczne dopasowanie



Rysunek 1.5: Prawidłowe dopasowanie

Rozdział 2

Sieci neuronowe

2.1 Historia

Historia sieci neuronowych sięga już lat 40. ubiegłego wieku, kiedy to Warren McCulloch, neurofizjolog, i Walter Pitts, matematyk, stworzyli pierwszy model matematyczny neuronu. Ich praca, opublikowana w 1943 roku, opisywała prosty model neuronu jako logiczną bramkę przełączającą, która przyjmowała wiele sygnałów wejściowych i generowała jedno wyjście. To był pierwszy krok w kierunku rozwoju sieci neuronowych.W latach 50. i 60. badania nad sieciami neuronowymi były aktywne, ale ograniczone przez brak odpowiednich zasobów obliczeniowych i danych. Jednakże, w 1969 roku, Bernard Widrow i Ted Hoff opracowali model Adaline (adaptive linear neuron), który wykorzystywał algorytm uczenia się adaptacyjnego do dostosowywania wag wejściowych w celu minimalizacji błędu wyjścia. To było istotnym krokiem w rozwoju uczenia maszynowego.W latach 80. i 90. sieci neuronowe zyskały popularność dzięki pojawieniu się algorytmu wstecznej propagacji błędu, który umożliwił skuteczne uczenie się wielowarstwowych sieci neuronowych. W 1986 roku David Rumelhart, Geoffrey Hinton i Ronald Williams opublikowali przełomową pracę, w której przedstawili skuteczny sposób trenowania sieci wielowarstwowych.Następnie, w latach 90., rozwój technologii komputerowych oraz dostępność większych zbiorów danych spowodowały nowy impuls w badaniach nad sieciami neuronowymi. Jednakże, mimo postępów, sieci neuronowe straciły na popularności na rzecz innych metod, takich jak Support Vector Machines (SVM) czy algorytmy drzew decyzyjnych.W ostatniej dekadzie, szczególnie od 2010 roku, sieci neuronowe, zwłaszcza modele głębokiego uczenia, ponownie zyskały na popularności dzięki wzrostowi mocy obliczeniowej, dostępności większych zbiorów danych oraz nowym technikom uczenia, takim jak wzmocnienie (reinforcement learning) czy generatywne modele adversarialne (GANs). Modele takie jak konwolucyjne sieci neuronowe (CNN) i rekurencyjne sieci neuronowe (RNN) znalazły zastosowanie w wielu dziedzinach, takich jak rozpoznawanie obrazów, przetwarzanie języka naturalnego, czy nawet w grach komputerowych.Dzięki postępowi technologicznemu oraz ciągłym badaniom naukowym, sieci neuronowe stały się kluczowym narzędziem w dziedzinie sztucznej inteligencji, otwierając drogę do rozwiązywania bardziej złożonych problemów i tworzenia inteligentnych systemów.

2.2 Definicja

Sztuczne sieci neuronowe (ang. Artificial Neural Networks, ANNs) zalicza się do tzw. Uczenia głębokiego, jak sama nazwa wskazuje swoim działaniem mają odzwierciedlać budowę ludzkiego mózgu dokładnie działanie neuronów. Główna ideą stojącą za naturą sieci neuronowych polega na ich zdolności do skutecznego przeprowadzania skomplikowanych obliczeń oraz tworzenia hierarchicznej struktury przetwarzanych danych. W przyrodzie neurony, które są połączone aksonami i dendrytami, tworzą złożone sieci neuronowe, umożliwiające przesyłanie i wymianę informacji, przechowywanie pośrednich wyników obliczeń, generowanie abstrakcyjnych reprezentacji oraz dzielenie procesu uczenia na wiele etapów. W efektywnie zbudowanym modelu obliczeniowym tego rodzaju system powinien być również zdolny do przeprowadzania procesów uczenia w sposób równie efektywny, jak ma to miejsce w procesach biologicznych. Do najpopularniejszych zastosowań algorytmów uczenia głębokiego należą: tworzenie modeli predykcyjnych dla rynków finansowych, przetwarzanie ludzkiej mowy na język komputerów, interpretacja zaszumionych i trudnych do zrozumienia dla oka ludzkiego obrazów medycznych, a także analiza obrazów z kamer stosowanych w pojazdach autonomicznych.

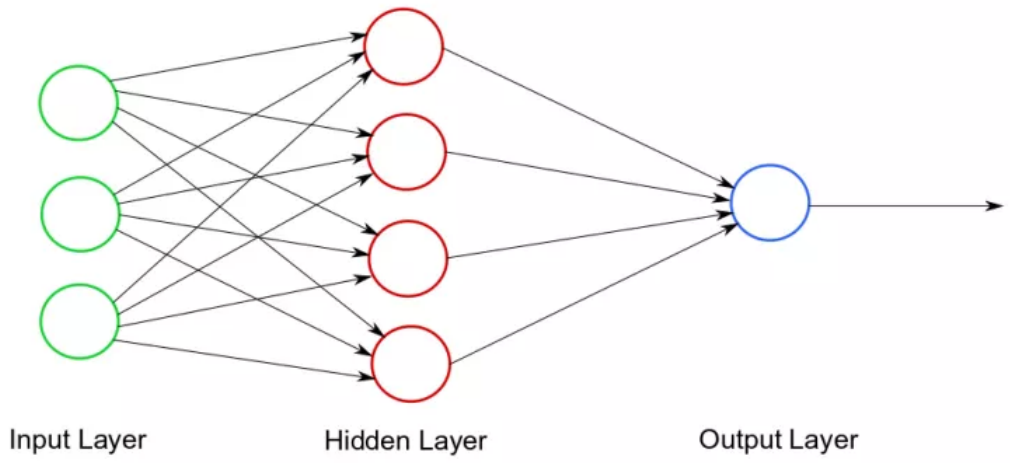
2.3 Uczenie sieci neuronowych

Można zobrazować proces uczenia się sieci neuronowej w mikroskali, porównując ją do biologicznego odpowiednika - komórki nerwowej. W tej analogii, neuron jest podstawową jednostką obliczeniową, która przyjmuje pojedyncze lub wiele sygnałów na wejściu (przez dendryty w biologii), sumuje je i przetwarza, porównując z funkcją aktywacji danego neuronu, a następnie przekazuje do kolejnego elementu sieci. Najprostsza możliwa sieć neuronowa składa się z pojedynczej jednostki, której wyjście odpowiada również za wyjście całej sieci. Oprócz cech wejściowych, neuron może mieć bias i wagę przypisaną do każdego wejścia. Wagi modyfikują wartość sygnału wejściowego i określają, jak ważna jest dana cecha. Wagi są dostosowywane przez sieć w procesie propagacji wstecznej. Bias to dodatkowa stała, która przesuwa funkcję aktywacji, co pozwala neuronowi na naukę wektorów, które normalnie byłyby odrzucane. Zwiększa to zdolność uczenia się neuronu kosztem ilości obliczeń. Funkcja aktywacji jest kluczowym elementem w przetwarzaniu danych przez sieć neuronową, ponieważ decyduje, czy sygnał zostanie przekazany do kolejnej warstwy sieci. W związku z tym każdy węzeł sieci neuronowej może być postrzegany jako prosty model regresyjny przetwarzający dane w mikroskali. Działanie pojedynczego bloku można opisać ogólnym wzorem:

$$\text{output} = \text{f(x)} = \begin{cases} 1, & \sum_{i=1}^{m} w_i x_i + \text{bias} \geqslant 0 \\ 0 & \sum_{i=1}^{m} w_i x_i + \text{bias} < 0 \end{cases}$$

gdzie wi to wagi kolejnych wejść, a xi to kolejne cechy sygnału. W zależności od potrzeb, blok aktywacyjny $f(x)$ może być opisany różnymi funkcjami. Do najbardziej

popularnych funkcji aktywacji można zaliczyć: liniową, skoku jednostkowego, a także funkcje nieliniowe. Wśród nich można wymienić: sigmoidalną, tangens hiperboliczny, a także aktualnie najczęściej wykorzystywaną w praktyce – funkcję ReLU (ang. Rectified Liniear Unit) oraz jej różne odmiany. Swoją popularność może zawdzięczać bardzo ważnej własności, wedle której nie istnieje możliwość aktywowania wszystkich neuronów warstwy w tym samym czasie. Jest to związane z faktem, iż tylko sygnały wejściowe i wartości powyżej 0 pozwolą na aktywację. W związku z tą cechą, ReLU jest znacznie bardziej wydajna niż np. funkcja sigmoidalna lub tangens hiperboliczny, lecz czasami prowadzi do tworzenia się „martwych" neuronów, które nie reagują na żadne wejścia. W przypadku, gdy sieć posiada wiele warstw, każda z nich może charakteryzować się osobną funkcją aktywacji. Wśród aktualnie najpopularniejszych architektur, najczęściej wykorzystuje się funkcję ReLU jako aktywację warstw ukrytych i liniową, bądź Softmax dla warstw wyjściowych.



Rysunek 2.1: Przykład prostej wielowarstwowej sieci neuronowej

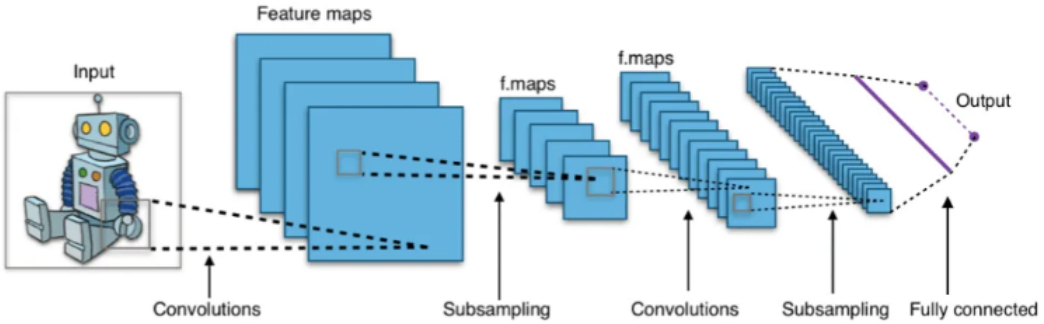
2.4 Konwulsyjne sieci neuronowe

2.4.1 deinicja

Sieć konwolucyjna, znana również jako Convolutional Neural Network (CNN), to rodzaj sztucznej sieci neuronowej, która specjalizuje się w przetwarzaniu obrazów. Jej główną cechą jest wykorzystanie operacji matematycznej konwolucji zamiast standardowej operacji mnożenia macierzy przynajmniej w jednej z jej warstw. To, co wyróżnia sieci CNN spośród innych typów sieci neuronowych, to ich zdolność do wykrywania lokalnych cech - sieć konwolucyjna skupia się na konkretnym obszarze podczas analizy, co skutkuje znacznym zwiększeniem efektywności przetwarzania obrazów, wideo i sygnałów audio. Kluczowym elementem sieci CNN są również filtry, które służą do przekształcania obrazów w taki sposób, który ułatwia wydobycie istotnych informacji.

2.4.2 Mechanizm działania sieci konwolucyjnej w procesie uczenia

To rodzaj sieci neuronowej, która specjalizuje się w analizie danych przestrzennych, takich jak obrazy. Jej działanie opiera się na przetwarzaniu wejściowego obrazu poprzez serię warstw konwolucyjnych, związanych z określonymi funkcjami aktywacji. Podstawą działania jest operacja konwolucji, która polega na nakładaniu filtrów - małych macierzy obliczeniowych - na obraz. W ten sposób identyfikowane są istotne cechy, takie jak krawędzie czy kolory, które są następnie wykorzystywane do klasyfikacji obrazów. W trakcie procesu uczenia sieć konwolucyjna dostosowuje wagi filtrów na podstawie otrzymywanych danych wejściowych oraz dzięki mechanizmowi propagacji wstecznej, co prowadzi do coraz lepszych wyników w rozpoznawaniu i klasyfikacji obrazów.



Rysunek 2.2: Przykład konwulsyjnej sieci neuronowej

2.4.3 Zastosowania

Sieci konwolucyjne znalazły szerokie zastosowanie w różnych dziedzinach ze względu na swoją potężną moc obliczeniową. Na przykład w przemyśle samochodowym są one wykorzystywane do rozwijania pojazdów autonomicznych, które analizują otoczenie w czasie rzeczywistym, rozpoznając obiekty, sygnalizację drogową oraz pieszych. Dzięki transparentności wyników, jaką zapewniają sieci konwolucyjne, są one nieocenione w medycynie, gdzie lekarze mogą diagnozować choroby na podstawie obrazów medycznych w sposób bardziej precyzyjny i mniej inwazyjny dla pacjenta. Ponadto, umożliwiają one realistyczne generowanie obrazów w grafice komputerowej, co jest kluczowe dla tworzenia wysokiej jakości symulacji i gier. Oczywiście, to tylko kilka przykładów zastosowań, które ilustrują praktyczne możliwości sieci konwolucyjnych w rozwiązywaniu rzeczywistych problemów.

2.5 Przygotowanie modelu sieci neuronowej

2.5.1 Pojęcia

W tej sekcji chciałbym opisać pojęcia, których będę używał w przypadku opisu tworzenia sieci neuronowej:

1. **Optymalizator (optimizer):** Optymalizator jest algorytmem używanym podczas trenowania modelu do aktualizacji wag sieci neuronowej w celu minimalizacji funkcji straty. Jego celem jest dostrojenie wag w modelu w taki sposób, aby osiągnąć jak najlepsze dopasowanie do danych treningowych. Popularnymi optymalizatorami są na przykład 'adam', 'SGD' (stochastic gradient descent), 'RMSprop', które różnią się w swoich strategiach aktualizacji wag.

2. **Metryki (metrics):** Metryki są miarami używanymi do oceny skuteczności modelu podczas treningu i testowania. Są one używane, aby zrozumieć, jak dobrze model radzi sobie z problemem, dla którego został zbudowany. Dla problemów klasyfikacji, popularne metryki to 'accuracy' (dokładność), 'precision', 'recall', 'F1-score'. Dla problemów regresji, metryki mogą obejmować 'mean squared error' (błąd średniokwadratowy), 'mean absolute error' (błąd średniowy).

3. **Hiperparametry:** Hiperparametry to parametry modelu, które nie są uczone w trakcie treningu, ale muszą być ręcznie ustawione przed rozpoczęciem treningu. Przykłady hiperparametrów to liczba warstw i neuronów w modelu, tempo uczenia, rozmiar batcha, funkcja aktywacji itp. Wybór odpowiednich hiperparametrów może mieć duży wpływ na skuteczność modelu.

4. **Epoka (epoch):** Epoka to jedno przejście przez cały zbiór treningowy przez model podczas treningu. Podczas jednej epoki model przetwarza wszystkie dane treningowe raz. Zazwyczaj trening modelu obejmuje wiele epok, aby umożliwić modelowi dostatecznie dużo iteracji w celu nauczenia się cech danych.

2.5.2 Przygotowanie modelu

Teraz opiszę jak wygląda przygotowanie modelu takiej sieci w poszczególnych krokach.

1. **Zrozumienie problemu:**

   [label=--]Pierwszym krokiem jest zrozumienie problemu, który chcemy rozwiązać za pomocą sieci neuronowej. Należy określić rodzaj problemu (klasyfikacja, regresja, segmentacja itp.) oraz jego cele.

2. **Zbieranie danych:**

   - Następnie zbieramy odpowiednie dane treningowe, które będą wykorzystywane do uczenia modelu.
   - Dane te powinny być reprezentatywne dla problemu i dobrze przygotowane do analizy.

3. **Przygotowanie danych:**

   - Dane mogą wymagać różnych przekształceń, takich jak normalizacja, standaryzacja, kodowanie kategorii, usuwanie brakujących wartości itp.
   - Dobrze przygotowane dane są kluczowe dla skuteczności modelu.

4. **Definiowanie architektury modelu:**

   - Następnie definiujemy architekturę sieci neuronowej, czyli strukturę warstw i połączeń między nimi.
   - Architektura może różnić się w zależności od rodzaju problemu i danych.

5. **Kompilacja modelu:**

   - Po zdefiniowaniu architektury modelu, kompilujemy go, określając funkcję straty, optymalizator i metryki.
   - Funkcja straty określa, jak skutecznie model osiąga zamierzone cele podczas uczenia.
   - Optymalizator jest używany do aktualizacji wag modelu w trakcie treningu.
   - Metryki są używane do oceny skuteczności modelu podczas treningu i testowania.

6. **Trenowanie modelu:**

   - Po skompilowaniu modelu, trenujemy go na danych treningowych.
   - Podczas treningu model dostosowuje swoje wagi w celu minimalizacji funkcji straty, wykorzystując optymalizator i funkcję straty zdefiniowane podczas kompilacji.

7. **Ocena modelu:**

   - Po zakończeniu treningu modelu oceniamy jego skuteczność na danych testowych, używając funkcji straty i metryk zdefiniowanych podczas kompilacji.

8. **Dostosowywanie modelu:**

   - W zależności od wyników oceny modelu, możemy dostosować jego architekturę, hiperparametry, funkcję straty itp., aby poprawić jego skuteczność.

Rozdział 3

Analiza dźwięku

Analiza dźwięku to proces badania cech akustycznych nagrań dźwiękowych w celu uzyskania informacji na temat ich właściwości i charakterystyk. W kontekście klasyfikacji gatunków muzyki na podstawie analizy dźwięku, istotne są różnorodne techniki i metody analizy, które pozwalają ekstrahować istotne cechy z dźwięku i wykorzystywać je do klasyfikacji. Poniżej przedstawiam kilka kluczowych aspektów analizy dźwięku:

1. **Spektrogram:** jest graficzną reprezentacją dźwięku w dziedzinie częstotliwości i czasu. Pozwala on wizualizować zmiany w amplitudzie dźwięku w zależności od czasu i częstotliwości. Spektrogramy są często wykorzystywane do ekstrakcji cech dźwiękowych, takich jak pasma częstotliwościowe i ich zmiany w czasie.

2. **Mel-Frequency Cepstral Coefficients (MFCC):** Jest to popularna metoda ekstrakcji cech dźwiękowych, które naśladują sposób, w jaki ludzkie ucho percepcyjnie analizuje dźwięk. MFCC reprezentuje dźwięk jako zestaw współczynników opisujących jego charakterystyki częstotliwościowe i temporalne.

3. **Energetyczne charakterystyki dźwięku (Energy-Based Features):** W analizie dźwięku istotne są również energetyczne cechy dźwiękowe, takie jak energia dźwięku w różnych pasmach częstotliwościowych, które mogą być używane do identyfikacji cech charakterystycznych różnych gatunków muzycznych.

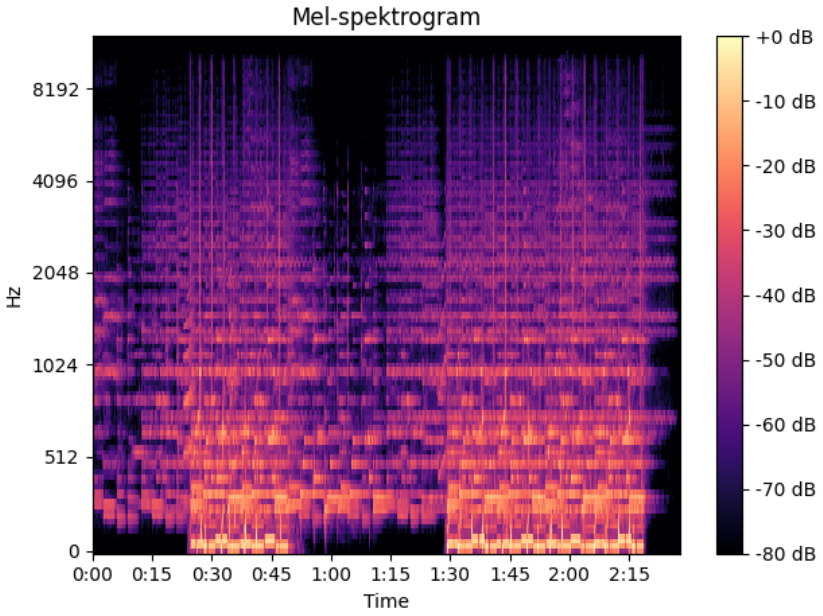
4. **Eksploracyjna analiza danych (Exploratory Data Analysis, EDA):** Przed przystąpieniem do treningu modeli uczenia maszynowego, istotne jest przeprowadzenie eksploracyjnej analizy danych dźwiękowych, aby zrozumieć ich rozkład, zmienność i relacje między cechami.

5. **Normalizacja i skalowanie:** W celu poprawy wydajności modeli klasyfikacji, często stosuje się normalizację i skalowanie danych dźwiękowych, aby zapewnić, że różne cechy mają podobne zakresy wartości.

6. **Wizualizacja cech dźwiękowych:** Wizualizacja cech dźwiękowych może być pomocna w zrozumieniu różnic między różnymi nagraniami muzycznymi oraz identyfikacji potencjalnych wzorców i korelacji między nimi.

3.1 Spektrogram

Przy analizie danych dźwiękowych jest bardzo przydatną formą reprezentacji danych. Pozwala zobaczyć jak częstotliwość dźwięku rozkłada się w różnych fragmentach badanej próbki.



Rysunek 3.1: Przykład spektrogramu

Najczęściej pozioma oś reprezentuje czas, pionowa częstotliwość, a intensywność każdego punktu odzwierciedla amplitudę określonej częstotliwości w danym punkcie czasu. Niekiedy wykres jest redukowany do dwóch wymiarów i wtedy intensywność jest oznaczana poprzez grubość kresek, kolor czy odcień szarości.

3.2 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) są to współczynniki cepstralne obliczane z sygnałów dźwiękowych. Są one używane w dziedzinie przetwarzania mowy i rozpoznawania mowy, zwłaszcza w zastosowaniach związanych z przetwarzaniem sygnałów dźwiękowych. Pierwszym krokiem w procesie uzyskiwania MFCCs jest przekształcenie sygnału dźwiękowego do dziedziny częstotliwości za pomocą transformaty Fouriera. Następnie wykorzystuje się skalę Mel do przekształcenia widma częstotliwościowego na melowe widmo częstotliwościowe. Skala Mel jest skalą, która pozwala na reprezentację częstotliwości dźwięku w sposób, który jest bardziej zgodny z percepcją ludzkiego słuchu.Po przekształceniu widma na skalę Mel, stosuje się logarytm do uzyskania cepstrum, czyli logarytmicznej reprezentacji widma melowego. Następnie stosuje się dyskretną transformatę kosinusową (DCT) do uzyskania współczynników cepstralnych.Ostatecznie, zbiór współczynników cepstralnych jest poddawany dalszej analizie, redukcji wymiarów lub klasyfikacji w celu wykorzystania ich do rozpoznawania

mowy lub innych zadań przetwarzania sygnałów dźwiękowych.MFCCs są popularnymi funkcjami do analizy i reprezentacji cech sygnałów dźwiękowych ze względu na ich zdolność do uwzględnienia cech percepcyjnych ludzkiego słuchu oraz ich stosunkowo niskiego wymiaru, co ułatwia przetwarzanie i analizę danych dźwiękowych.

Rozdział 4

Założenia Pracy

Obiektem mojej pracy naukowej było wytrenowanie modelu klasyfikatora rodzajów muzyki, konkretnie 4 różnych gatunków: jazz, rock, latino i muzyka klasyczna. Model na wejściu otrzymywał dane w postaci spektrogramów i z wykorzystaniem najnowszych technik uczenia maszynowego z nadzorem był trenowany aby osiągnąć zadowalający rezultat. W trakcie prac nad owym zagadnieniem natknąłem się na kilka problemów związanych zarówno z przygotowaniem danych jak i samym treningiem modelu, które postaram się szerzej opisać w kolejnych rozdziałach. Jak już wcześniej napisałem celem pracy jest poszerzenie i zgłębienie współczesnych technik uczenia maszynowego, które jest szeroko wykorzystywane w dzisiejszym świecie, także praca nad takim klasyfikatorem dostarczyła mi sporo wiedzy o aktualnych trendach panujących w świecie analizy danych. Chociaż stworzony przeze mnie model wykorzystałem tylko do celów naukowych to uważam że spokojnie mógłby posłużyć w komercyjnej aplikacji do odtwarzania muzyki jakich dzisiaj na rynku jest wiele.

Rozdział 5

Dane

W tym rozdziale dokonam charakterystyki danych, które posłużyły mi przy trenowaniu modelu.

5.1 Źródło

Dane użyte w modelu zostały w całości pobrane z internetu głównie z darmowych, otwartych stron umożliwiających dostęp do muzyki różnego rodzaju bez ograniczeń. Wybór internetu jako głównego źródła danych do trenowania modeli uczenia maszynowego może być uzasadniony z kilku powodów:

- **Dostępność i powszechność**: Internet jest powszechnie dostępnym źródłem danych, które jest łatwo dostępne praktycznie dla każdego użytkownika. Dzięki temu można znaleźć różnorodne dane dotyczące praktycznie każdej dziedziny, co sprawia, że jest to bogate źródło informacji do trenowania modeli.
- **Różnorodność danych:**: Internet oferuje ogromną różnorodność danych, obejmującą teksty, obrazy, multimedia, dane społecznościowe, informacje geograficzne i wiele innych. Ta różnorodność pozwala na trenowanie modeli, które są w stanie radzić sobie z wieloma różnymi typami danych.
- **Aktualność**: Internet dostarcza danych w czasie rzeczywistym, co oznacza, że modele uczenia maszynowego mogą być trenowane na aktualnych informacjach. To jest istotne w wielu dziedzinach, takich jak analiza sentymentu w mediach społecznościowych, przewidywanie trendów rynkowych, czy też monitorowanie wydarzeń na świecie.

5.2 Format

Wszystkie dane, które użyłem w późniejszym przetwarzaniu były w formacie mp3, nagrania podzieliłem na próbki trwające 1 sekundę. Następnie po wczytaniu ich za pomocą API librosa uzyskałem spektrogramy, które ostatecznie zapisałem do formatu csv. Uzyskane przez mnie dane prezentowały się w następującej postaci:

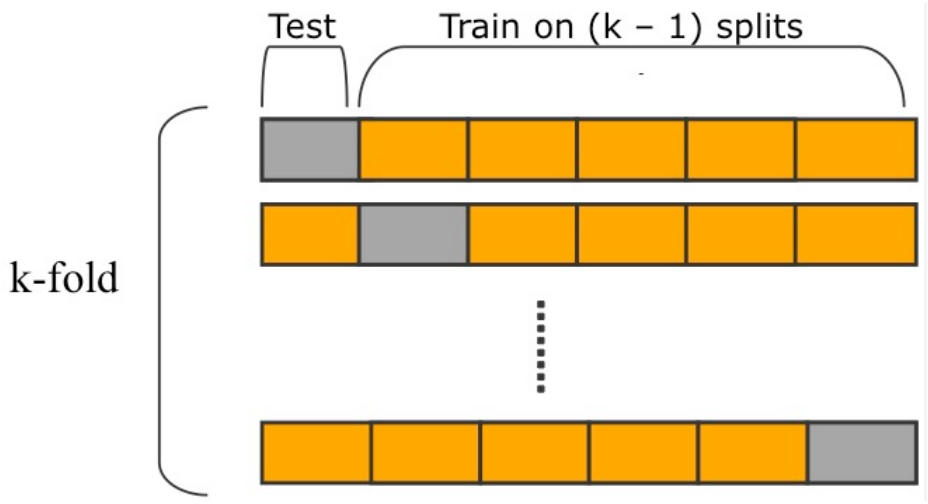
```
-2.35869293212890625e+01,-1.10922622680664062e+01,-8.46299743652343750e+00,
-2.35869293212890625e+01,-1.51551361083984375e+01,-1.35888977050781250e+01,
-2.35869293212890625e+01,-1.37939529418945312e+01,-6.32476806640625000e+00,
-2.35869293212890625e+01,-1.00634078979492187e+01,-5.00769042968750000e+00,
-2.07130584716796875e+01,-1.20847702026367187e+01,-1.27049331665039062e+01,
-2.27225952148437500e+01,-1.02959594726562500e+01,-9.47045135498046875e+00,
-1.87396774291992187e+01,-6.00949859619140625e+00,-5.89671325683593750e+00,
-2.10047607421875000e+01,-7.58222198486328125e+00,-6.40071868896484375e+00,
-2.35869293212890625e+01,-1.52949829101562500e+01,-1.11758193969726562e+01,
-2.35869293212890625e+01,-1.57349014282226562e+01,-1.01124649047851562e+01,
-2.35869293212890625e+01,-1.21265029907226562e+01,-1.03147125244140625e+01,
-2.35869293212890625e+01,-1.03802490234375000e+01,-4.31954193115234375e+00,
-2.35869293212890625e+01,-1.53547668457031250e+01,-6.32678985595703125e+00,
-2.35869293212890625e+01,-1.41033630371093750e+01,-1.23892745971679687e+01,
-2.35869293212890625e+01,-1.24262313842773437e+01,-8.12390136718750000e+00,
-2.35869293212890625e+01,-1.62906036376953125e+01,-6.91078186035156250e+00,
-2.35869293212890625e+01,-1.44772491455078125e+01,-1.00017089843750000e+01,
-2.35869293212890625e+01,-1.05398330688476562e+01,-7.29473876953125000e+00,
-2.35869293212890625e+01,-1.54281234741210937e+01,-1.34570465087890625e+01,
-2.35869293212890625e+01,-2.17528839111328125e+01,-1.43276824951171875e+01,
-2.35869293212890625e+01,-2.24654159545898437e+01,-1.41288909912109375e+01,
-2.35869293212890625e+01,-1.36021957397460937e+01,-8.69273376464843750e+00,
-2.35869293212890625e+01,-1.54941711425781250e+01,-7.54667663574218750e+00,
```

Rysunek 5.1: Wizualizacaja danych

gdzie każda wartość odpowiada częstotliwości w skali mel. Dane w powyższym formacie użyłem do treningu klasyfikatora.

5.3 Walidacja krzyżowa i stratyfikacja

Walidacja krzyżowa (cross-validation) to technika oceny wydajności modelu, która polega na podziale dostępnych danych na zestawy treningowe i testowe, co pozwala na wielokrotne trenowanie i testowanie modelu na różnych podzbiorach danych. Podstawowym celem walidacji krzyżowej jest ocena zdolności generalizacji modelu na nieznanych danych. Jedną z najczęściej stosowanych form walidacji krzyżowej jest k-krotna walidacja krzyżowa (k-fold cross-validation). Proces ten polega na podziale danych na k równych podzbiorów (foldów), a następnie wielokrotnym trenowaniu i testowaniu modelu na tych podzbiorach, z wykorzystaniem każdego z nich jako zestawu testowego i reszty jako zestawu treningowego. Wyniki z wszystkich k iteracji są następnie łączone, aby uzyskać końcową ocenę wydajności modelu. Walidacja krzyżowa jest ważnym narzędziem w ocenie modeli uczenia maszynowego, ponieważ umożliwia bardziej obiektywne oszacowanie skuteczności modelu niż pojedynczy podział danych na zbiór treningowy i testowy. Dzięki temu możemy uzyskać lepsze zrozumienie zachowania modelu na różnych danych oraz uniknąć problemów związanych z obciążeniem danych.



Rysunek 5.2: Przykład walidacji krzyżowej

Stratyfikacja danych odnosi się do procesu podziału zbioru danych na podzbiory w taki sposób, aby zachować równowagę między klasami lub wartościami cech. Jest to szczególnie istotne w zadaniach klasyfikacji, gdzie mamy zróżnicowane klasy docelowe lub etykiety. Głównym celem stratyfikacji danych jest zapewnienie, że wszystkie klasy lub wartości cech występujące w danych treningowych są odpowiednio reprezentowane w zbiorze treningowym i zbiorze walidacyjnym lub testowym. Dzięki temu unikamy sytuacji, w której któryś z podzbiorów danych jest zdominowany przez jedną klasę, co może prowadzić do błędnych wniosków podczas procesu uczenia maszynowego. W swojej pracy zastosowałem połączenie tych dwóch technik aby proces oceny modelu był bardziej uczciwy i reprezentatywny. W trakcie przygotowania danych do modelu użyłem odpowiedniego narzędzia, które implementującego walidację krzyżową z zachowaniem stratyfikacji. Liczba foldów które użyłem wynosi 3.

Rozdział 6

Trening modelu

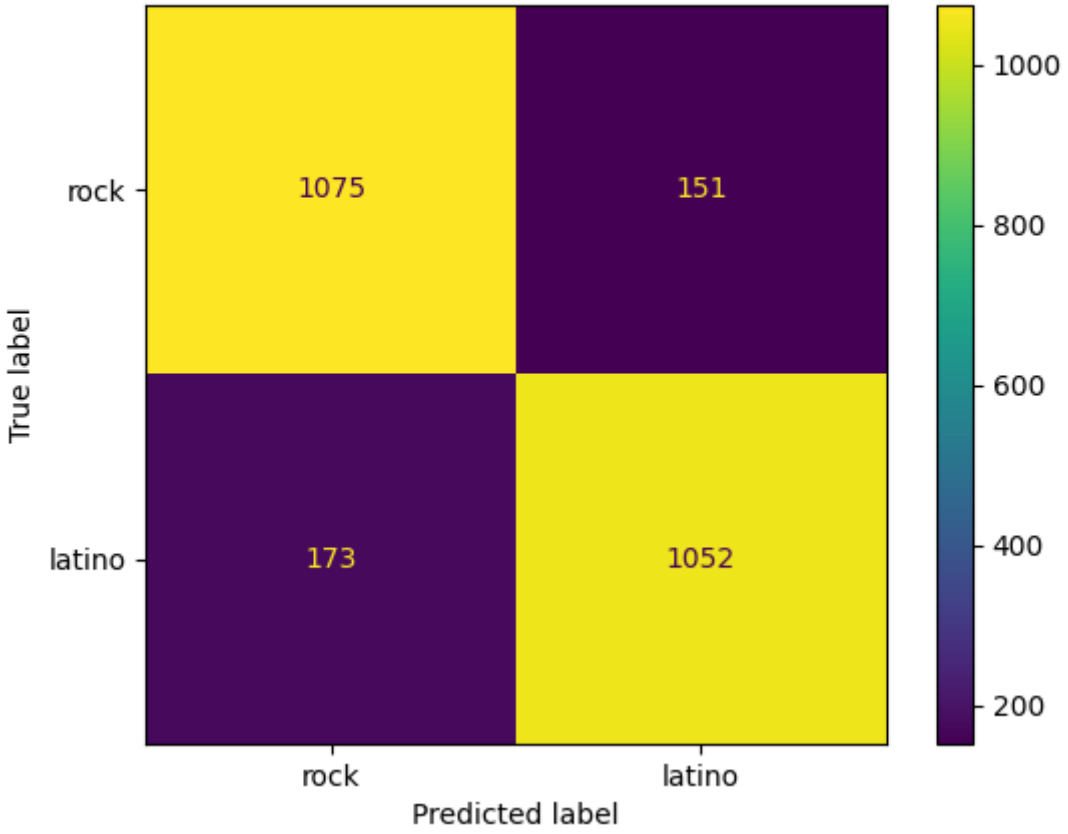
Po etapie przygotowania danych przystąpiłem do trenowania modelu. W tym, rozdziale postaram się szczegółowo omówić przebieg treningu modelu, wykorzystane metody, opiszę po krótce pracę jaką wykonałem do uporządkowania danych wczytywanych do modelu i postaram się opowiedzieć o problemach jakie spotkałem w trakcie swojej pracy.

6.1 Eksperyment 1

Na początek do 1 eksperymentu wybrałem 2 etykiety mianowicie: 'rock' i 'latino'. Swoją pracę rozpocząłęm od przygotowania danych. Do treningu użyłem 3054 próbek z kategorii 'latino' i 3073 próbek z kategorii 'rock'. Każda próbkę wczytałem jako macierz o wymiarze $127x20$, potem każdej próbce nadałem etykietę, następnie zmieniłem wymiar tablicy przechowujacej wczytywane dane na $6256x2540$ aby znormalizować dane do modelu. Po przygotowaniu danych podzieliłem próbki na zbiór trenujący i testowy w proporcji 0.4 dla danych testowych.

6.1.1 GradientBoostingClassifier

Pierwszym klasyfikatorem z jakiego skorzystałem był GradientBoostingClassifier uzyskałem wynik 0.86 co jest wynikiem dość satysfakcjonujacym. Do lepszej prezentacji wyników teraz i w dalszej części pracy posłużyłem się macierzą korelacji, która ładnie ilustruję dokładność wytrenowanego modelu i w taki sposób przedstawiała się owa macierz:

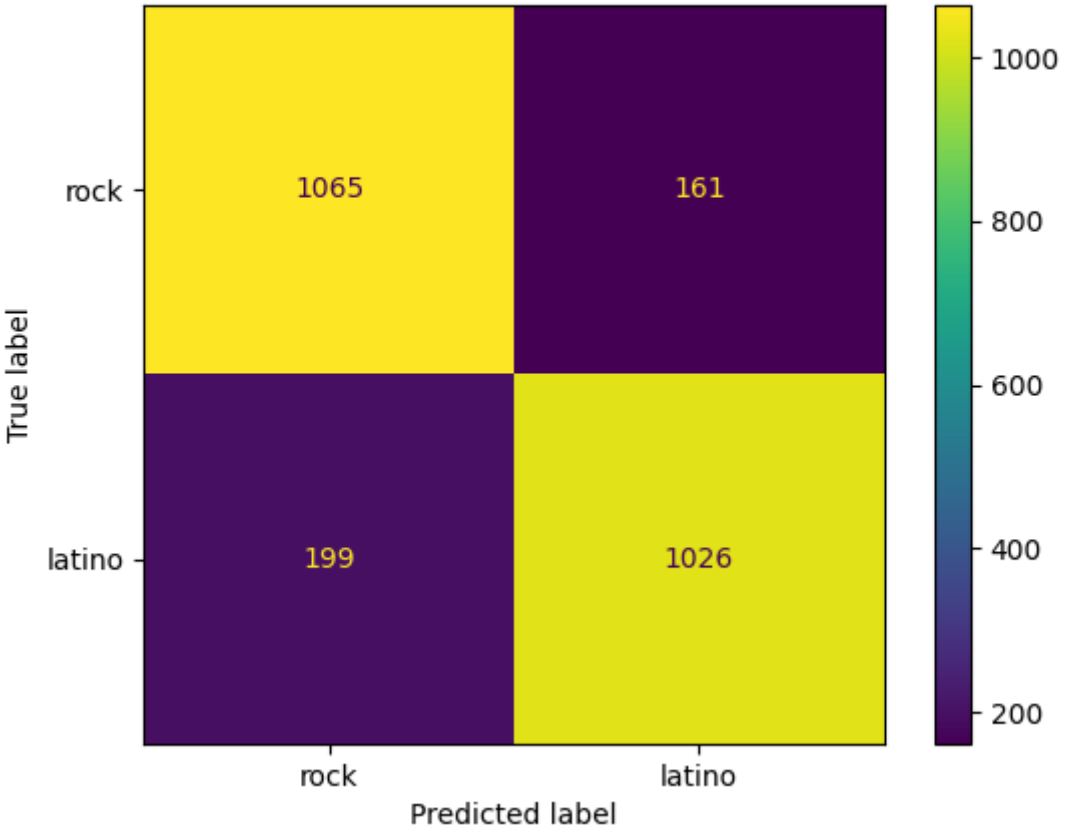


Rysunek 6.1: Macierz dla GradientBoostingClassifier

Widzimy że dla 1075 próbek testowych klasyfikator dobrze przyporządkował kategorię dla etykiety 'rock' a pomulił się 173 razy, natomiast dla kategorii latino pomylił się 151 razy a dobrze przyporządkował 1052 razy.

6.1.2 RandomForestClassifier

Następnie do klasyfikacji skorzystałem z lasu decyzyjnego, tutaj wynik prezentował się bardzo podobnie - był na poziomie 0.85 i tak zaprezentowała się macierz:

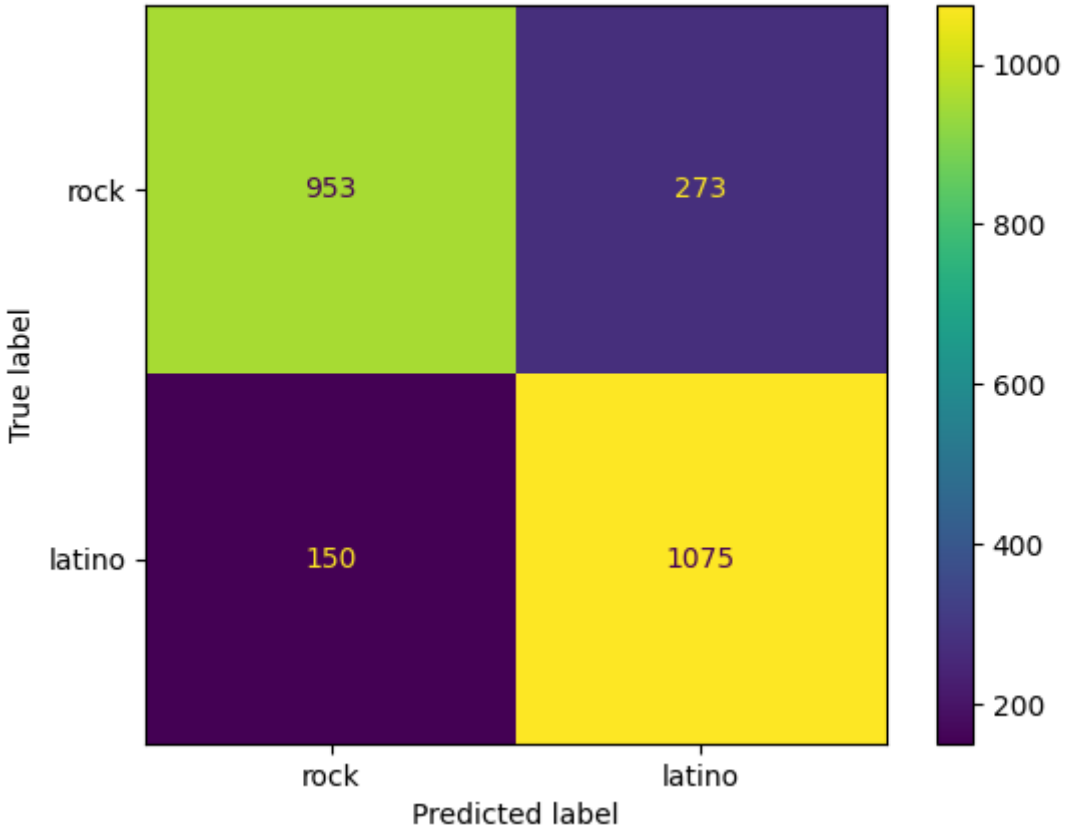


Rysunek 6.2: Macierz dla lasu losowego

Tutaj jak widzimy wyniki prezentują się bardzo podobnie jak w poprzednim przykładzie z nieco lepszym wynikiem dla kategorii 'latino' jednak nie jest to jakaś znacząca różnica.

6.1.3 Maszyna wektorów nośnych

Trzecim modelem z jakiego skorzystałem jest maszyna wektorów nośnych (SVM) , tutaj otrzymałem wynik nieco gorszy - 0.82 jednak dalej jest to dość dobry rezulatat.



Rysunek 6.3: Macierz dla maszyny wektorów nośnych

Tutaj jak widać wynik jest trochę gorszy dla kategorii 'rock' natomiast utrzymuje się na dość dobrym poziomie jeśli chodzi o 'latino'.

6.1.4 Podsumowanie pierwszych kroków

Po pierwszych treningach modelu wyniki prezentują się następująco:

	rock	latino	wynik
gradientBoostingClassifier	1075	1052	0.86
maszyna wektorów nośnych	1065	1026	0.85
las losowy	953	1075	0.82

gdzie wynik jest dokładnością modelu a liczba w poszczególnej kategorii oznacza ile prawidłowych przykładów zaklasyfikował model na zbiorze testowym.

6.1.5 Dodanie walidacji krzyżowej i stratyfikacji

Następnie aby poprawić jeszcze jakość modelu spróbowałem dodać walidację krzyżową i stratyfikację do mojego modelu

6.2 Eksperyment 2

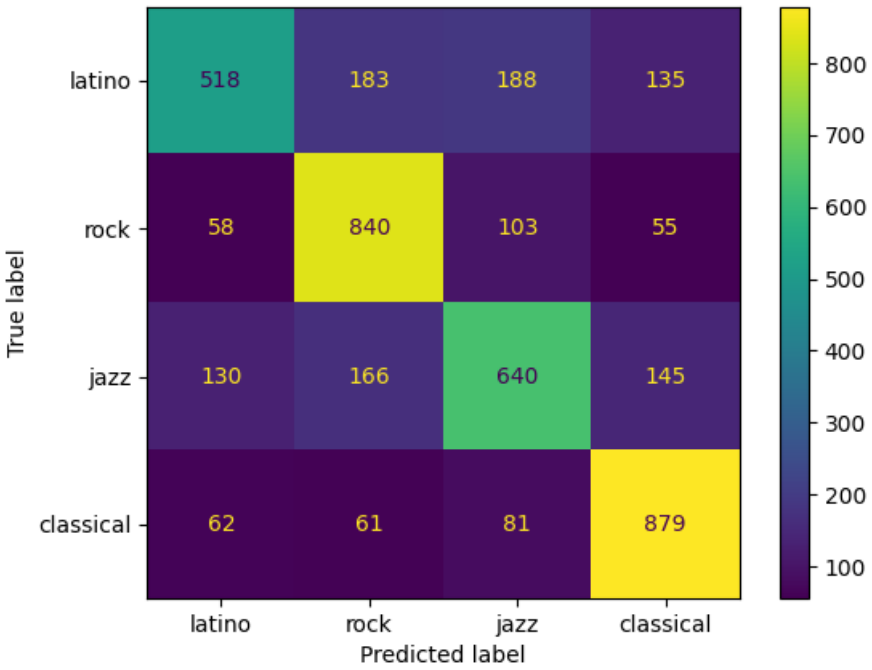
W 2 eksperymencie posłużyłem się maksymalną liczbą etykiet: 'rock', 'latino', 'classical' i 'jazz'. Liczba próbek użytych na poszczególną kategorię prezentuję się następująco:

- **latino** - 2610
- **classical** - 2652
- **jazz** - 2662
- **rock** - 2684

Proces wczytywania i przygotowywania danych do modelu przebiegał bardzo podobnie jak w przypadku 1 eksperymentu, zacząłem od pogrupowania próbek i przyporządkowania do poszczególnych etykiet. Wymiary poszczególnych macierzy również były identyczne jak w 1 przypadku.

6.2.1 Maszyna wektorów nośnych

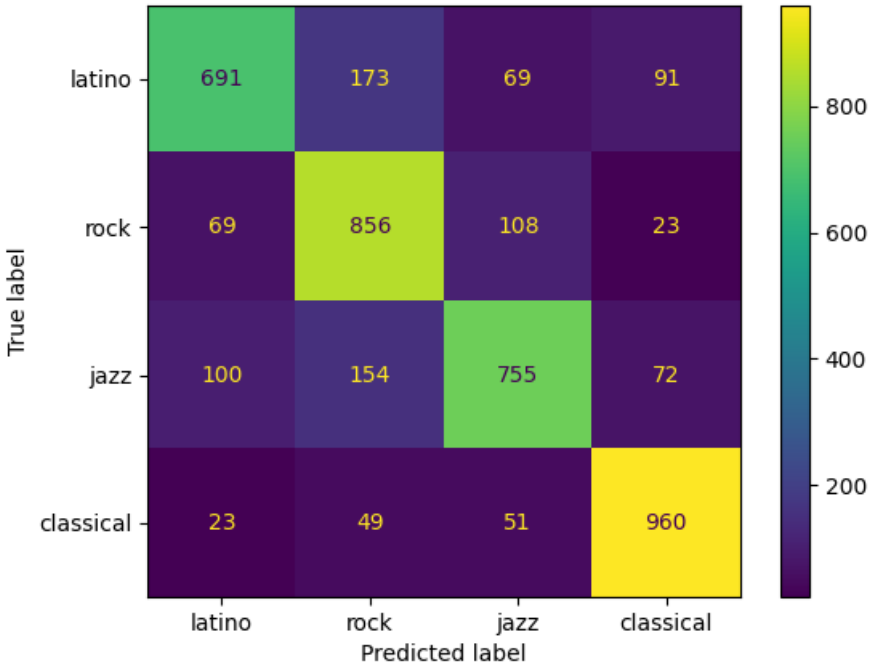
Kolejność klasyfikatorów użytych w modelu była inna niż w 1 eksperymencie dla 2 kategorii, najpierw użyłem Maszyny wektorów nośnych. Dokładność wynosiła 0.67 a macierz korelacji dla tego klasyfikatora przedstawiała się następująco:



Rysunek 6.4: Macierz dla maszyny wektorów nośnych

6.2.2 RandomForestClassifier

Dokładność dla tego klasyfikatora okazała się sporo lepsza - na poziomie 0.76, a tak przedstawiała się macierz korelacji:



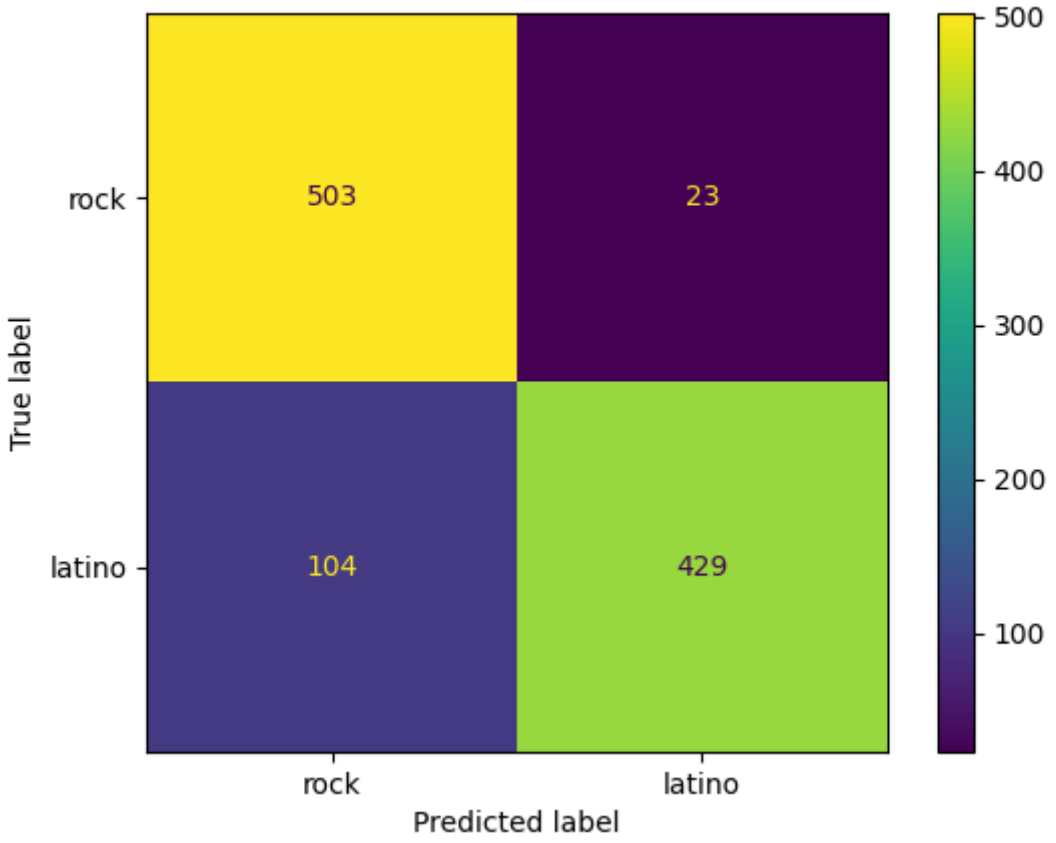
Rysunek 6.5: Macierz dla RandomForestClassifier

6.3 Eksperyment 3

Do 3 eksperymentu wykorzystałem Konwulsyjne sieci neuronowe (CNN). Pierwsze kroki wykonałem identycznie jak w eksperymentach powyżej; przygotowałem dane, w tym eksperymencie użyłem takich samych danych do treningu jak w przypadku eksperymentu 1 i dokładnie tyle samo etykiet - 2. Przygotowanie danych przebiegało identycznie dla spektrogramów jednak różniło się jeśli chodzi o etykiety. Tutaj kategorie przekształciłem na liczby całkowite gdyż takie były wymagane przez model CNN. Kolejnym krokiem było przygotowanie modelu sieci. Początkowo moja sieć składała się z 7 warstw, jednak ostatecznie zrezygnowałem z 2 co znacząco wpłynęło na jakość wyników. Tak przedstawiały się warstwy, których użyłem do treningu modelu:

1. **Warstwa konwolucyjna (Conv2D):**
   - Ta warstwa Conv2D zawiera 32 filtry konwolucyjne o rozmiarze (3, 3).
   - Funkcja aktywacji tej warstwy to ReLU (Rectified Linear Unit), co oznacza, że wartości ujemne są zamieniane na zero, a wartości dodatnie pozostają niezmienione.
   - Kształt wejścia = (127, 20, 1). Dane wejściowe to obrazy o wymiarach 127x20 pikseli (szerokość x wysokość) i jednym kanale (w skali szarości).
2. **Warstwa MaxPooling (MaxPooling2D):**
   - Ta warstwa MaxPooling2D przeprowadza operację maksymalnego łączenia (max pooling) na danych wyjściowych z poprzedniej warstwy.
3. **Warstwa Flatten:**
   - Warstwa Flatten służy do przekształcenia dwuwymiarowej mapy cech (wyniku konwolucji) na jednowymiarowy wektor, aby można go było przekazać do warstw gęstych (fully connected).
4. **Warstwa gęsta (Dense):**
   - Funkcja aktywacji tej warstwy to ReLU, co pozwala na modelowanie nieliniowych zależności w danych.
5. **Warstwa wyjściowa (Dense):**
   - Ostatnia warstwa Dense zawiera 2 neurony co świadczy że model ma przewidywać 2 klasy.
   - Funkcja aktywacji tej warstwy to softmax, która przekształca wyniki na rozkład prawdopodobieństwa dla każdej klasy, umożliwiając modelowi wybór najbardziej prawdopodobnej klasy.

Następnie przystąpiłem do kompilacji modelu, podczas kompilacji modelu określamy różne parametry, które będą używane podczas treningu, takie jak funkcja straty (loss function), optymalizator (optimizer) oraz metryki do oceny modelu. W tym przypadku funkcja straty została ustawiona na 'categorical crossentropy', co jest odpowiednie dla problemów klasyfikacji wieloklasowej, gdzie dane wyjściowe są kodowane w postaci one-hot. Optymalizator został wybrany jako 'adam', co jest popularnym wyborem dla wielu problemów uczenia maszynowego ze względu na jego skuteczność i efektywność. Po pierwszym dopasowaniu otrzymałem wynik równy około 0.89 co jest rezultatem dość dobrym, a tak prezentowała się macierz korelacji:

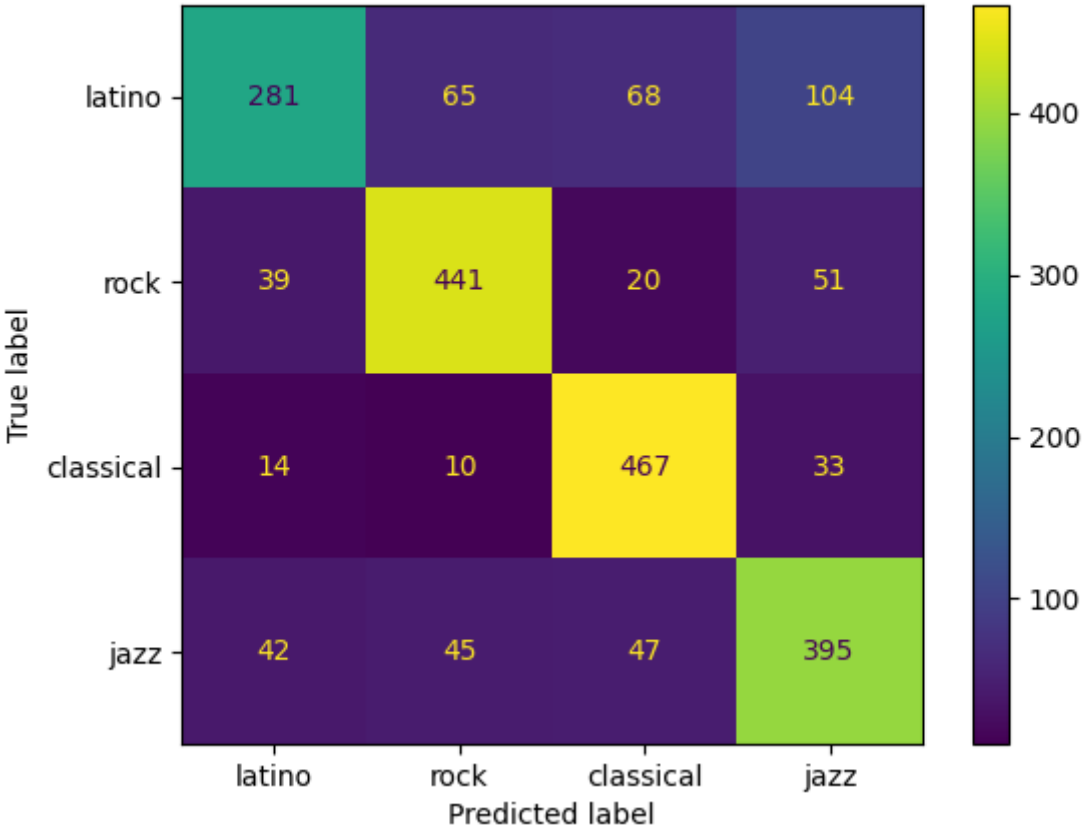


Rysunek 6.6: Macierz dla CNN

Jak widać wynik jest znacznie lepszy dla muzyki 'latino' gdyż jest to około 80 próbek dopasowanych poprawnie więcej, aby zniwelować tą dysproporcję psotanowiłem dodać walidację krzyżową.

6.4 Eksperyment 4

W tym eksperymencie również posłużyłem się konwulsyjnymi sieciami neuronowymi, tym razem dla 4 etykiet i uzyskałem dokładność na poziomie 0.80 a tak przedstawiała się macierz korelacji:



Rysunek 6.7: Macierz dla CNN