 Metody odkrywania wiedzy

Temat projektu:

Przewidywanie oceny filmu na podstawie grających aktorów

Klasyfikacja oceny jakości win

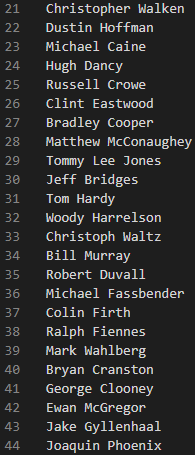
**Cel projektu** (Przewidywanie oceny filmu)

Celem projektu było zaimplementowanie programu, który za pomocą klasyfikatorów (KNN, SVM-SVC, Grid Search) pozwoli przewidywać średnie oceny filmów.  
Naszym zadaniem jest umożliwić za pomocą zaimplementowanego przez nas klasyfikatora tworzenia „Idealnego kina” uwzględniając oczekiwania widzów (osób oglądających) na temat pojawiających się aktorów w filmie.Program opiera się na metodzie klasyfikacji.

**Informacja o danych:**

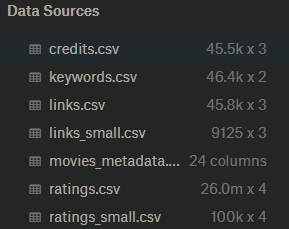
* Dane pochodzą ze serwisu internetowego MovieLens.org
* Ranking filmów to średnia z ocen zostawionych przez użytkowników serwisu (ponad 26 milionów ocen od 270 tysięcy użytkowników)
* 45573 próbek
* Naszym wymaganiom odpowiada 3930 (pierwotna liczba to 5345, ale przez brak informacji w pewnych próbkach odrzuciliśmy tę dane, które nie są kompletne)

**Selekcja danych**

Pierwszym etapem w naszym projekcie było wybranie 100 aktorów (subiektywnie). Przy czym ciekawym faktem jest to, że wszystkie wybrane aktory są umieszczone na liście Top-250 actors by IDMB.

Wśród wszystkich dostępnych dla nas danych w zbiorze danych dla przewidywania oceny filmu wybraliśmy takie parametry: aktor (**actor**), reżyser (**producer**) i gatunek (**genre**).

**Podjęte działania nad danymi:**

Przed przystąpieniem do etapu normalizacji danych, musieliśmy stworzyć jedno źródło danych (tzw. **Single Source of Truth**), ponieważ, potrzebne dla nas dane znajdowały się w różnych plikach **.csv**

Struktura danych przed ujednoliczeniem do jednego zbioru danych:

Dla realizacji powyżej opisanego procesu powształy skrypty:

*check\_actor\_availability\_in\_dataset\_2.py*

*create\_CSV\_header.py*

*create\_proper\_JSON.py*

*merge\_files\_3.py*

*SortingData.py*

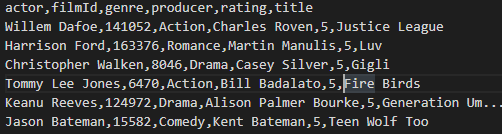
*updateFile.py*

**Filtracja danych**

Po umieszczeniu wszystkich potrzebnych danych do jednego pliku dostaliśmy plik o rozmiarze 3971 linii (ilość próbek):



Postać danych:



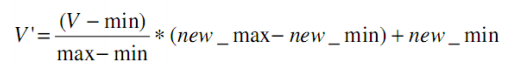
**Normalizacja**

Normalizacją jest nazywana procedura wstępnej obróbki danych surowych, po której jest możliwe i wygodniejsze porównanie danych i dalsza analiza tych danych. Normalizacja danych jest jednym z pod procesów eksploracji danych. (Prawie wszystkie dane muszą być poprzednio przygotowane przed ich analizą).

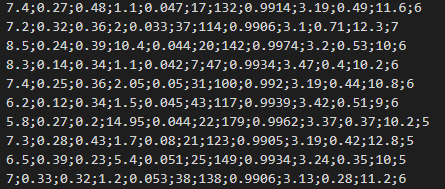
* W procesie eksploracji danych wyrózniają takie typy:  
  Smoothing – czesto jest stosowana pod czas problemów klasterezacji oraz regresji.
* Generalization (Uogólnianie danych) – technika mająca na celu przetworzenie danych do jednej postaci (np. Boolean -> integer, String -> Integer)
* Normalization (Normalizacja) – z koleji jest bezstratnym procesem organizowania danych. (np. skalowanie pierwotnych danych do małego,specyficznego przedziału). Najcześciej używanymi rodzajami normalizacji są:
  + Standaryzacja
  + Usuniecie elementów odstających.
* Stworzenie nowych cech lub tworzenia nowych cech na podzstawie już istniejących

Pod czas realizacji naszego projektu wykorzystaliśmy normalizacje typy **min-max** do przedziału **[0,1]. (Jakość wina)**

Ten typ normalizacji jest określony wzorem:

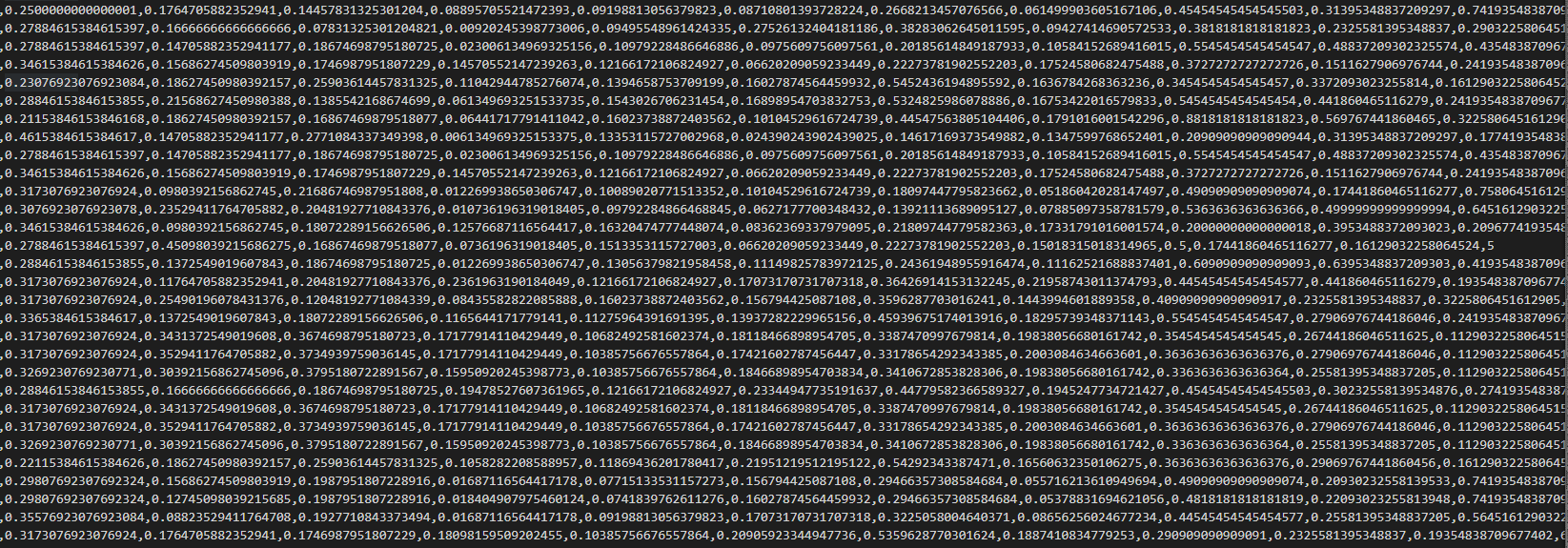


gdzie **min**, **max** jest przedziałem, w którym mieszczą się dane wejściowe, natomiast **new\_min**, **new\_max** jest nowym przedziałem danych.

Po tej normalizacji nowe przeskalowane dane zostały zapisane do pliku nad którym były wykonane dalsze prace.

*Dane przed normalizacją:*

*Dane po normalizacji:*

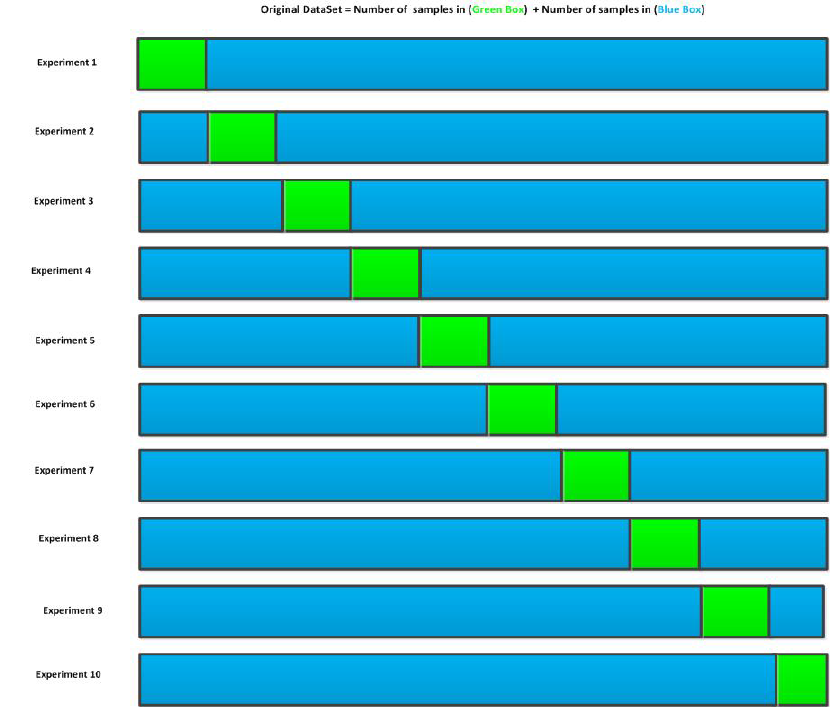


**k-Fold Cross-Validation**

Kolejnym krokiem w wykonywanych zadaniach było wykonanie zadań dotyczących uniknięcia(lub maksymalnego możliwego zminimalizowania) zjawiska pt ‘’Overfitting”, czyli mówiąc prościej przetrenowania modelu.W tym celu stosuje się metodę walidacji krzyżowej (**Cross-Validation)**.

**Walidacja krzyżowa** jest metodą estymacji jakości algorytmów uczących na małych zbiorach danych. Pozwala ona także na odpowiednie dobranie hiperparametrów danego algorytmu uczącego

W naszym projekcie używamy odmiany z wersją **k-Fold**. Działa ona w taki sposób, że na samym początku dzieli zbiór na **k** części. Zakładając, że zbiór ma x wierszy, ilość wierszy naszego pojedynczego zbioru wynosi **x/k**.  
Na potrzeby projektu, wartość parametru **k** ustawiamy na 10. W takim przypadku, nasz ogólny zestaw danych stworzy 10 zbiorów. W każdym z dziesięciu zbiorów dochodzi do podziału na podzbiory uczące i testowe. W takim przypadku w pojedynczym zbiorze, ilość wierszy naszego zbioru testowego to x/k. Natomiast ilość wierzy zbioru uczącego wynosi (x/k) \* (k-1).  
W każdym kolejnym podzbiorze przesuwamy się indeksami o liczbę x/k, więc w każdym kroku posiadamy inny podzbiór testowy.



Przed rozpoczęciem Cross-Validation, postanowiliśmy zadbać o równomierne rozłożenie danych. Stwierdziliśmy, że losowy rozkład może nie dać pozytywnego rezultatu ze względu na niewielką ilość filmów o wysokich i niskich ocenach. Przypisanie ustawionej kolejności – równomiernej względem ilości ocen w poszczególnym zbiorze.

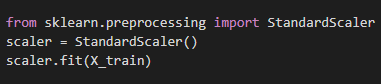
Wyniki walidacji krzyżowej:

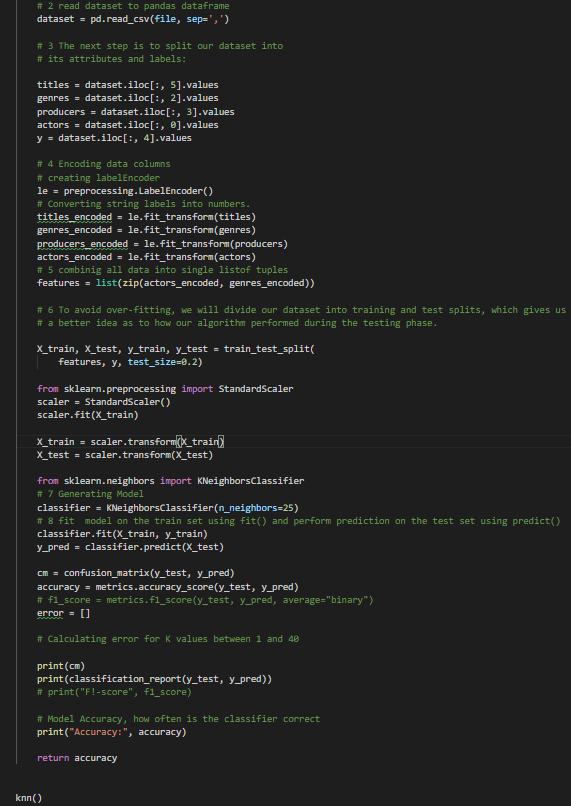
* Najniższy: ***0.439***
* Najwyższy: ***0.465***
* Średnie: ***0.45***

**kNN**

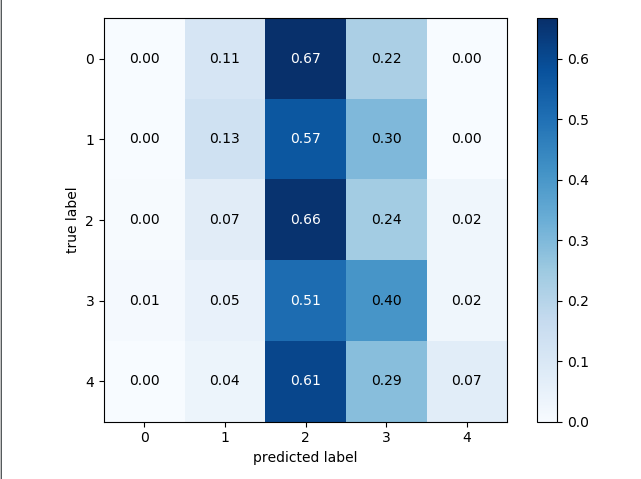
Algorytm K-najbliższego sąsiada (KNN) to rodzaj nadzorowanych algorytmów uczenia maszynowego. KNN jest niezwykle łatwy w implementacji w swojej najbardziej podstawowej formie, a jednocześnie wykonuje dość złożone zadania klasyfikacyjne. Jest to leniwy algorytm uczenia się, ponieważ nie ma specjalistycznej fazy treningu. Zamiast tego wykorzystuje wszystkie dane do szkolenia podczas klasyfikowania nowego punktu danych lub instancji. KNN jest nieparametrycznym algorytmem uczenia, co oznacza, że ​​nie zakłada niczego na temat podstawowych danych.

Przed rozpoczęciem klasyfikacji, musieliśmy użyć enkodowania wartości kategorycznych. Jest to podstawowy i niezbędny krok przez wypełnieniem modelu danych. Zakodowane w ten sposób dane, pozwalają stworzyć macierz cech odpowiedniego dla modelu typu.

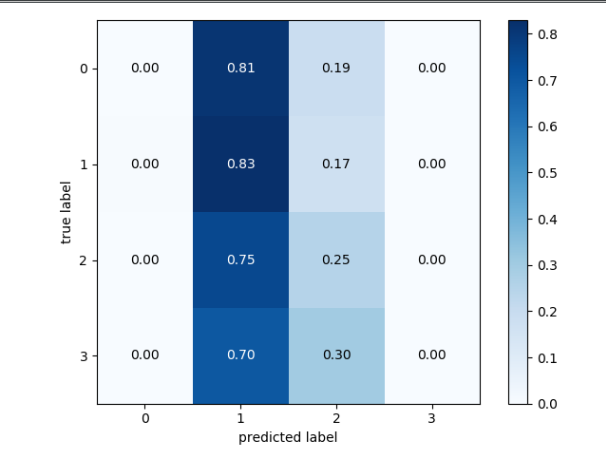
Dodatkowo tuż przed rozpoczęciem i zaraz po endkodowaniu macierzy cech – macierzy niezależnych zmiennych, używamy Standard Scalera, w celu standaryzacji wielkości zmiennych, co pomaga w osiągnięciu lepszej skuteczności – wyższej precyzji modelu.

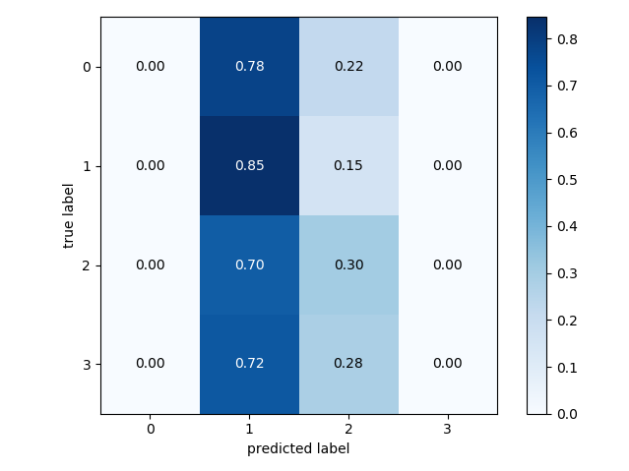
**Cały kod klasyfikatora kNN:**

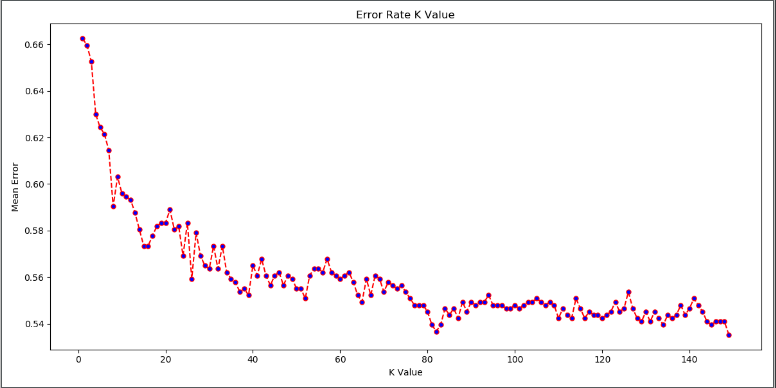
Dla realizacji klasyfikatora KNN wykorzystaliśmy biblioteke skleran.

**Wyniki kNN:**

*K=5, 5 Classes, Accuracy=****0.39***

*K=25, 4 Classes, Accuracy=****0.45***

*K=95, 4 Classes, Best Accuracy=****0.51****, avarage=****0.48***

**

*Błąd klasyfikatora można zobaczyć na wykresie*:

**SVM**

Działanie algorytmu SVM polega na mapowaniu danych na wielowymiarową przestrzeń właściwości w sposób umożliwiający kategoryzację punktów danych, nawet jeśli danych tych nie można w inny sposób liniowo oddzielić. Najpierw odszukiwany jest separator między kategoriami. Następnie dane są przekształcane w sposób umożliwiający wyrysowanie separatora jako hiperpłaszczyzny. Po wykonaniu tych czynności charakterystyki nowych danych mogą służyć do przewidywania grupy, do której powinien należeć nowy rekord.

Zalety SVM:

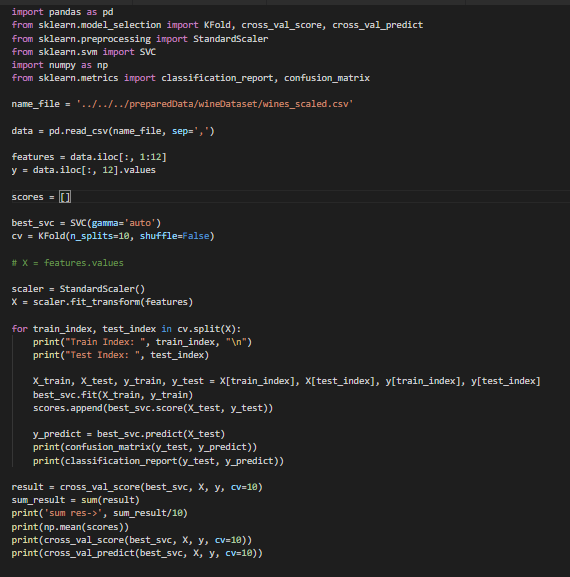
* Znajduje maksymalne odległości (marginesy) pomiędzy grupami punktów
* Efektywne obliczeniowo - złożoność rośnie tylko liniowo wraz z liczbą wymiarów
* Rozwiązuje problemy liniowe jak i nieliniowe

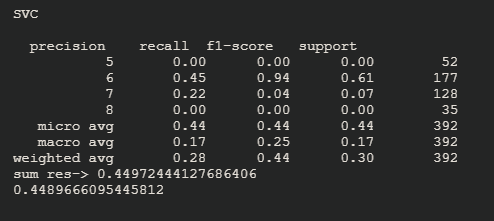
Tak samo, jak i w KNN dane zostały najpierw użyliśmy encodowania wartości kategorystycznych dla lepszej skuteczności – wyższej precyzji modelu.

W ten sposób wyselekcjonowane oraz przetworzone dane używamy do modelu:

**C-Support Vector Classification** z rodziny **Support Vector Machine**. Jest to popularny model pozwalający na dobra klasyfikacje z możliwością manipulacji parametrów w celu uzyskania lepszych wyników. W wykonywanym projekcie jako kernel używamy metody: ‘**’rbf’’**. Jest to optymalny wybór w rozpatrywanym problemie klasyfikacji. Inne metody np. **‘’linear’’** mają inną zasadę działania, która nie sprawdza się najlepiej w celu wyszukiwania rozwiązań oraz nie przynosi zadowalających rezultatów.

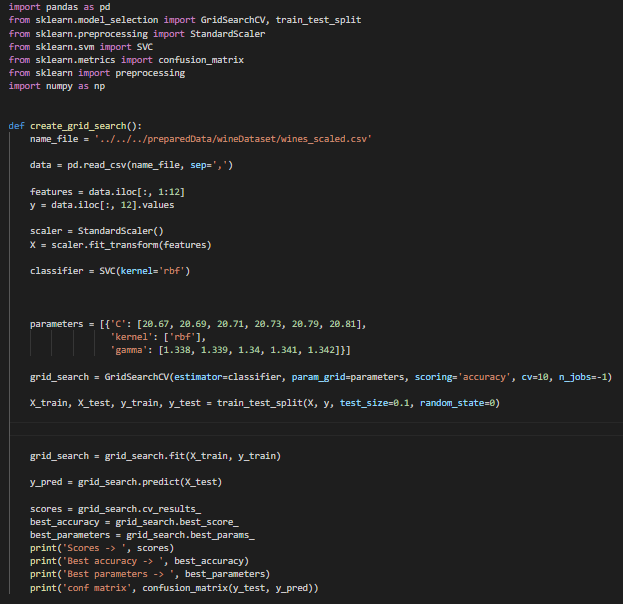
*Kod SVM-SVC:*



**Wyniki dla SVM-SVC:**

Dodatkowo używamy metodyki **Grid Search CV**, która opiera się na modelu **SVC**, lecz posiada rozszerzoną możliwość manipalacji parametrami, które przekładają się na skuteczność. Oprócz możliwości wyboru kernela mamy zmienne: **C** oraz **gamma**. **Grid Search** także działa przy użyciu **Cross Validation**. Sama metodyka Grid Searcha opiera się na testowaniu modelu z różnymi wymienionymi parametrami oraz zaprezentowaniu je z poszczególnych splitach(podzbiorach).

**Grid Search CV** jest użyteczną metodą pozwalająca na dogłębną analizę modelu przy różnych wariancjach splitów oraz wartości modelu, co powoduje szeroki wachlarz możliwości.

Testowanie modeli na zbiorach danych odbywało metod to obliczania precyzji modeli.

*Kod* ***Grid Search CV****:*

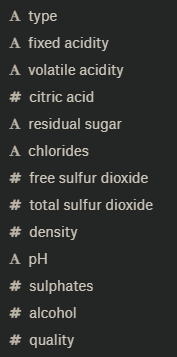
Dodatkowo używane były też funkcje miedzy innymi do pomiaru: wyników Cross Validation, predykcji Cross Validation oraz macieczy błedów i raportu klasyfikacji, czyli odpowiednio: **confusion\_matrix** oraz **classification\_report**. Użycie tych opcji, funkcji pozwala na dokładną analizę wyników oraz przegląd rozwiązań. W przypadku użycia metody Grid Search CV używaliśmy list wartości parametrów **C** oraz **gamma**. Były to wartości z przedziałów:

**[0.001, 0.01, 0.1, 1.0, 10, 100, 1000]**. Po wstępnym teście, dalsze analizy odbywały się w ściślej określonych przedziałach, np: **[1, 10, 100, 500]** dla parametru **C** oraz **[1.0, 0.8, 0.5, 0.3, 0.1, 0.05, 0.01, 0.001]** dla parametru gamma. Z każdym krokiem – testem, wartości te były zawężane do zakresów, w których model sprawdzał się najlepiej. Finalnie, najlepszymi wartościami parametramów okazały się wartości: 0.7 dla parametru **‘C’** oraz 7.8 dla parametru ‘**gamma’**.

Pozwoliły one na uzyskanie precyzji na poziomie **46,7 %,** co nie należy do najlepszych wyników.

**Klasyfikacja oceny jakości win**

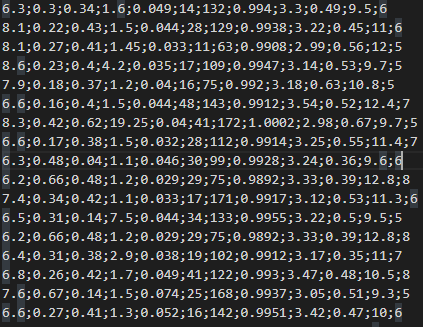
Lekko zawiedzeni oraz zniesmaczeni precyzją modelu, po konsultacji z Panem Doktorem , postanowiliśmy podjąć się próby testu naszych stworzonych modeli na innych danych. Wybór padł na dane dotyczące jakości win.

Zbiór danych z winami składa się z 4898 próbek. Gdzie każda próbka posiada 12 wymiarów takich jak:

Postępowania były podobne co wyżej opisane kroki, czyli:

* Selekcja
* filtracja danych
* KNN
* Cross Validation
* SVC
* Grid Search CV
* Іtandaryzacja wartości itd.

*Dane przed normalizacją:*



*Postać danych po normalizacji:*



Finalnie, najlepszymi wartościami parametramów okazały się wartości: **20.71** dla parametru ‘**C’** oraz **1.34** dla parametru ‘**gamma’**. Pozwoliły one na uzyskanie precyzji na poziomie **67,22** %, co powoduje znacznie lepszy wynik w porównaniu do precyzji modelu z danymi filmowymi.

Na jednym ze splitów – podzbiorów, precyzja modelu była na poziomie **72,17 %,** co czyni to wynikiem powodującym zadowolenie.