2. Тражење оптималних хиперпараметара методом унакрсне валидације

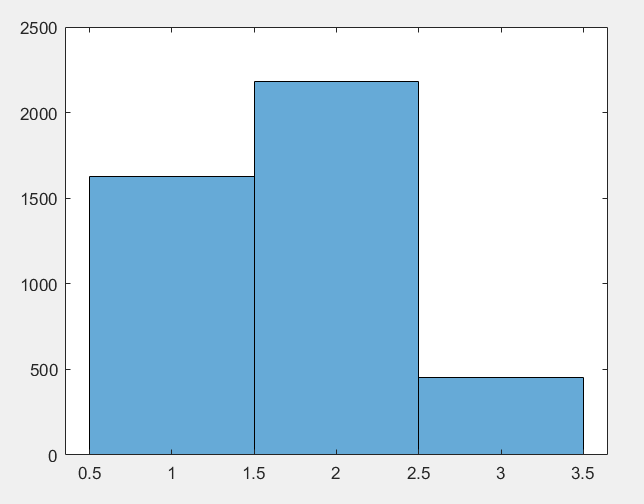
Наш задатак биће класификација пиринча. Улазна датотека у којој нам се налазе подаци је „Rice.csv“. Улазни подаци које ћемо користити за одређивање врсте пиринча су:

1. Area – површина пиринча
2. Perimeter – обим пиринча
3. MajorAxis – дужа оса
4. MinorAxis –краћа оса
5. Eccentricity – ексцентричност
6. Convex-Area – конвексна поршина
7. Extent

Ове податке ћемо сместити у улазну матрицу, коју ћемо транспоновати и добити матрицу 7xN (где је N укупан број мерења). Пиринач се разврстава у 3 класе. Видимо да је заступљеност по класама следећа:

* Cammeo – 1629 узорака
* Osmancik – 2181 узорака
* Kecimen – 449 узорака

Излазе ћемо сместити у један низ (izlazBrojevi у нашем коду). У том низу елементи ће имати вредности 1, 2 и 3 у зависности од тога којој класи припадају улазни подаци на одговарајућим местима. Бројне вредности класама додељујемо по горе наведеном редоследу класа. Примећујемо да су улазна мерења неравномерно распоређена, те да нам се класа 2 издваја у односу на друге две класе, а да имамо много мање примерака треће класе у поређењу са прве две. Због тога ће бити добро да при одабиру тежина обратимо пажњу на трећу класу.

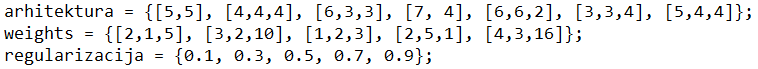


Податке ћемо даље одвојити на валидациони, тест и тренинг скуп. План нам је да одвојимо 70% свих улазних подата за тренирање, а 15% за тестирање и 15% за валидацију. Међутим, пошто у овом задатку немамо униформу расподелу по класама тј. неке класе се јављају чешће у улазним подацима, биће велика вероватноћа да ће наши скупови бити нерепрезентативни (нпр. имаћемо само 5% укупног броја узорака треће класе у тест и валидационом скупу, а 90% у тренинг). Зато ћемо узети по 70% узорака из сваке класе и од њих направити тренинг сет. Исто то ћемо урадити и за валидациони сет и за тренинг сет, само што ћемо за њих узети по 15% узорака сваке класе. Иако смо извршили ову поделу, мрежа ће опет најбоље класификовати пиринче друге класе јер имамо највише улазних података за ту класу.

Користићемо 3 хиперпараметра код метода отпимизације. Изабрали смо да ти хиперпараметри буду:.

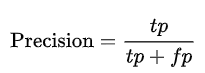
1. Архитектура мреже – од саме архитектуре зависи колико ће се мрежа добро обучити. Уколико имамо једноставнију мрежу (са премало неурона и скривених слојева) може доћи до тога да се мрежа лоше обучи, док у супротном, ако имамо веома комплексну структуру мреже доћи ће до преобучавања и мрежа ће лоше да класификује податке над којима се није обучавала. Зато је битно наћи оптималну архитектуру мреже.
2. Коефицијент регуларизације – Вредност може бити између 0 и 1. Представља параметар који нам говори колико регуларизација као метода заштите од преобучавања учествује у рачунању грешке.
3. Тежина класа – параметар који придодаје значај одређеним класама. У принципу, треба да буде већи за оне класе које имају мањи број узорака (да би им се више дало на вредности).

Узећемо следеће вредности за претходно наведене хиперпараметре:



Код архитектуре, сваки елемент представља мрежу, где дужина низа представља број скривених слојева, док конкретан број представља број неурона у слоју који одговара његовој позицији у низу. Што се тиче тежине класа (weights), сваки елемент представља низ од три елемента који говори о тежини за одређену класу тј. “колико јој се даје на значају”. За коефицијенте регуларизације смо узели вредности 0.1, 0.3, 0.5, 0.7, 0.9. Проћи ћемо кроз све комбинације, не би ли утврдили која комбинација ће дати најбоље резултате за нашу класификацију.

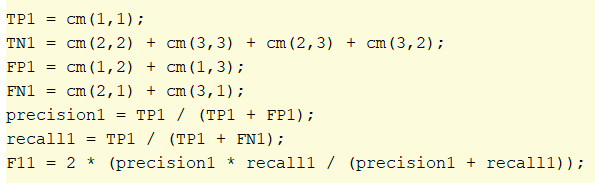
Приликом обучавања сваке од комбинација, рачуна се конфузиона матрица над валидационим скупом и перформансе по формули

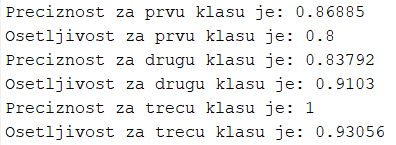


Расподела података по класама ће делимично утицати на решење у смислу да ће се слабије класификовати кандидати који имају веће шансе да напусте посао, јер их има много мање, а тежина класа делимично спречава ту појаву. Осетљивост можемо израчунати формулом:



Програмски су ове две вредности рачунате на следећи начин (аналогно иде израчунавање за друге две класе):

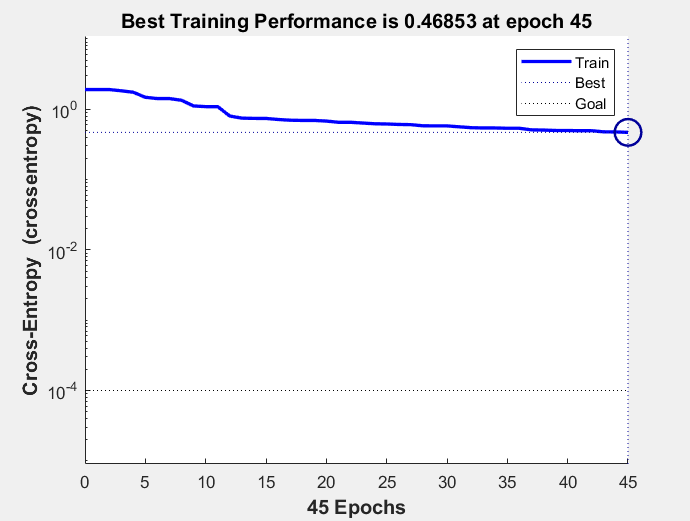


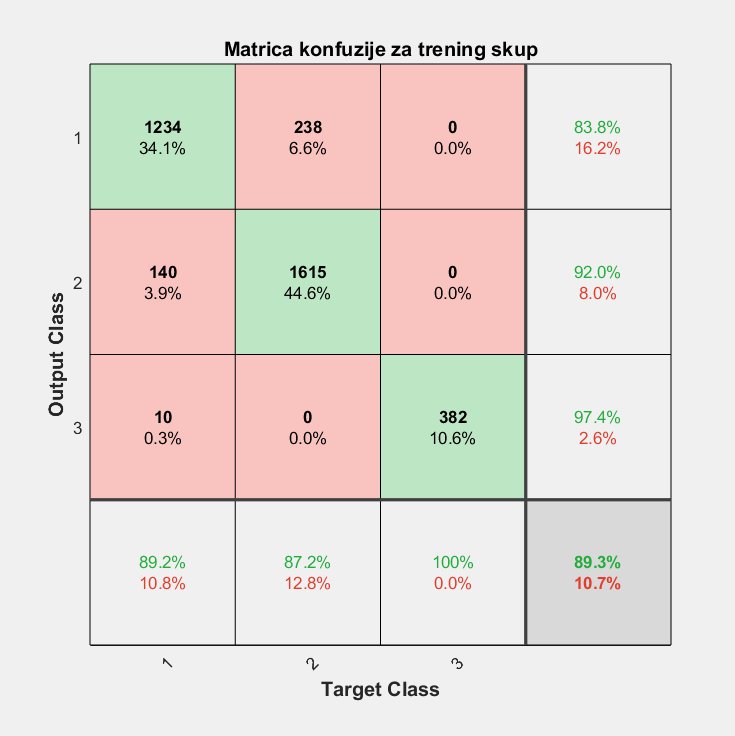
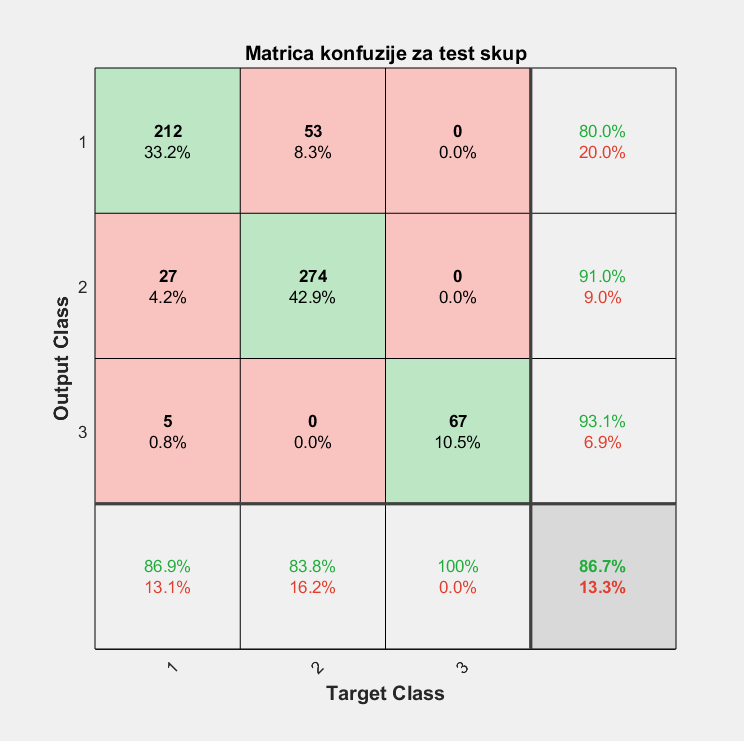


**Вредности оптималних хиперпараметара су:**

* **Архитектура = [6, 6, 2]**
* **Коефицијент регуларизације = 0.1**
* **Тежина класа = [3, 4, 16]**

Као што видимо са слике, мрежа се зауставља након 45 епоха, проналазећи оптимум.



Ако погледамо добијене резултате са матрица конфузије, приметићемо да је мрежа добро обучена (с обзиром на то да су у питању реални подаци), и да је разлика између успеха на тест и тренинг скупу у дозвољеним границама.