1. Projenin ilk aşamasında bir veri seti için supervised learning bağlamında sınıflandırma yapılması istenmiştir. Doğal olarak verinin herhangi bir sınıflandırma algoritmasına input olarak verilebilmesi amacıyla verinin uygun formata getirilmesi gerekiyordu. Verisetinde hem nümerik hem de alfa nümerik özelliler bulunmaktaydı. Bilindiği üzere bir algoritmaya input olarak text(string) verilerin verilmesi çok da uygulanabilir ve mümkün değildir. Dolayısıyla bu tarz veri içeren özelliklerin sayısallaştırılması ilk adım olarak gerçekleştirilmiştir. Bu bağlamda bir özellikteki her bir ayrı değeri ayrı bir kategori olarak değerlendirmemize olanak sağlayan Label Encoder metodundan yararlanarak 5.,6. ve 7. sütunlar için sırasıyla 22,289 ve 289 ayrı veri tespit edilmiştir. Takdir edersiniz ki bu verilerin sayısal olarak birbirleri arasında bir üstünlükleri bulunmamaktadır. Bu bağlamda bu verilerin sadece 0 ve 1’lerden oluşacak şekilde bir formata getirilmesi gerekmekteydi. Bu bağlamda One Hot Encoding’ten yararlanarak veriler 0 ve 1’lerden oluşan matrislere çevrildi. Örneğin 6 farklı veri (1,2,3,4,5,6) olsun. Eğer veri 1 ise 1’in belirtildiği sütun 1 diğerleri 0 olmaktadır. Bu mantıkla bütün özellikler ayrı ayrı değerlendirilmiş ve bu sayede başlangıçta (100000,9) olan veri seti (100000,605) formatına getirilmiştir. Modele etkileri hesaplanıp(ie. PCA ) bu özellik sayısı düşürülebilirdi ancak zaman darlığından bu işlem gerçekleştirilmemiştir. Veri seti’nin eğitilebilmesi için veri %80 training ve %20 testing olacak şekilde iki ayrı parçaya ayrılmış ve sonrasında sınıflandırma algoritmalarının tasarlanmalarına geçilmiştir. Seçilebilecek olan iki ayrı model olan lojistik Regresyon ve Naive Bayes algoritmalarının yanında LDA,kNN ve CART algoritmalarının performansları da değerlendirilmiştir. Gerçekleştirim python dili kullanılarak geliştirilmiştir. Geliştirilen kod paylaşılan drive linkinde mevcuttur. Elde edilen doğruluk oranları şu şekildedir :

Tablo.1. Sınıflandırıcıların Performansları

|  |  |  |
| --- | --- | --- |
| Sınıflandırıcı Algoritması | Accuracy %(Doğruluk) | Std |
| Lojistik Regresyon | 80.9 | 0.0011 |
| Naive Bayes | 38.5 | 0.0085 |
| LDA | 80.8 | 0.0014 |
| kNN | 78.8 | 0.0024 |
| CART | 75.2 | 0.0048 |

1. Makine öğrenmesi yöntemlerinde kullanılan algoritma kadar algoritmanın kendi parametrelerinin yani hiperparametrelerinin optimizasyonu da doğruluk oranını artırabilecek bir etmen olarak karşımıza çıkmaktadır. Bu bağlamda bu çalışmada epochs, batch\_size, optimizer, activation function, learning rate ve momentum parametreleri üzerinde durulmuştur. Gerçekleştirim yine python’da yapılmıştır. Sklearn kütüphanesinde bulunan GridSearchCV, RandomizedSearchCV ve Talos paketleri kullanılarak sonuçlar elde edilmiştir. Bununla birlikte sonuçların daha hızlı elde edilebilmesi amacıyla Google’ın kullanıcılara ücretsiz olarak sunduğu Coogle Colab’ta yine bu çalışmada efektif olarak kullanılmıştır. Bu bağlamda süre analizleri de yapılarak paylaşılan drive klasörüne konulmuştur. Bu algoritmalar, verilen parametrelerin hepsinin Kartezyen çarpımını arama uzayı olarak kabul ediyor diyebiliriz ve be bağlamda bütün uzayı arayarak en optimal parametre değerlerini vermektedir. Ancak veri setine olan bağlılığı ortadan kaldırabilmek için olaya random’lık katmak de gerekmektedir. Bu durum zaten aslında optimizasyonun temel mantığı olan doğayı taklidin bir sonucudur. GridSearchCV ve RandomizedSearchCV ile elde edilen sonuçlar şu şekildedir.

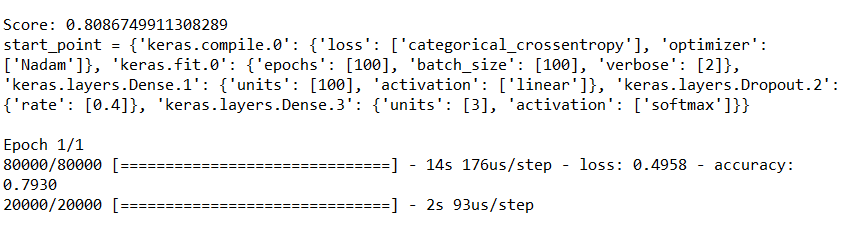
Tablo.2. GridSearchCV Sonuçları

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Teknik | Hiperparametre 1 | Hiperparametre 2 | En İyi Doğruluk | Std |
| GridSearchCV | Epoch : 50 | Batch\_size :40 | 81.0 | 0.0018 |
| GridSearchCV | Optimizer:RMSProp | - | 81.2 | 0.0012 |
| GridSearchCV | Learning Rate : 0.001 | - | 81.1 | 0.0008 |
| GridSearchCV | Activation F. : softplus | - | 81.1 | 0.0015 |
| GridSearchCV | Dropout Rate: 0.0 | Weight :2 | 81.07 | 0.0015 |

Tablo.3. RandomizedSearchCV Sonuçları

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Teknik | Hiperparametre 1 | Hiperparametre 2 | En İyi Doğruluk | Std |
| RandomizedSearchCV | Epoch : 100 | Batch\_size :20 | 81.16 | 0.0008 |
| RandomizedSearchCV | Optimizer:Adamax | - | 81.22 | 0.0011 |
| RandomizedSearchCV | Learning Rate : 0.01 | - | 81.15 | 0.0001 |
| RandomizedSearchCV | Activation F. : relu | - | 81.24 | 0.0010 |
| RandomizedSearchCV | Dropout Rate: | Weight : |  |  |

3-4 – Bütün arama uzayına bakıp en optimal parametre değerlerini elde etmek her ne kadar teoride doğru gözükse de pratikte çok da doğru bir yaklaşım olmayabilir. Eğer veriseti çok büyükse ve parametreler için verilen aralıklar veyahutta çeşitler fazla ise çok ciddi derecede zaman alan zahmetli bir iş haline dönebilir. Bu sebeplerden dolayı heuristic(sezgisel) yaklaşımlar cankurtaran vazifesi görebilir. Örneği bu çalışmada Talos kütüphanesi kullanılarak gerçekleştirilen bir deney yaklaşık olarak 70 saat yani 3 gün sürmüştür. Bu da günümüzün hızına ve akan verinin hakimiyetinin olduğu bu zamanda kabul edilemez bir senaryodur. Bunun için bu çalışmada python’da bulunan hyperactive kütüphanesi kullanılmıştır. Bir sezgisel yaklaşım olan simulated annealing yöntemi kullanılmıştır. Sebebi ise lokal minima ve maksima’larda başarılı sonuçlar üretebilme kabiliyetinden dolayıdır. Paylaşılan scriptte de görüldüğü üzere hiperparametrelerin arama uzayının belirtildiği search\_config bloğunda ayarlamalar yapılmıştır. Yine aynı şekilde veri %80 ve %20 olarak ikiye ayrılmış ve sonuç üretilmiştir. Sonuç çıktısının ekran görüntüsü şu şekildedir.

  
Şekil 1. Simulated Annealing Sonucu

5- Son olarak bu yöntemlerden elde edilen sonuçlar ve parametrelerin benzerlikleri değerlendirilecektir. Ayrıca sonuç üretmek için geçen süre de yine bu karşılaştırmaya dahil edilecektir.

Doğruluk Oranı(Accuracy %): GridSearchCV : 81.2

RandomizedSearchCV : 81.24

Meta-Heuristic Tekniği : 79.3

Epochs : GridSearchCV : 50

RandomizedSearchCV : 100

Meta-Heuristic Tekniği : 100

Batch\_Size: GridSearchCV : 40

RandomizedSearchCV : 20

Meta-Heuristic Tekniği : 100

Optimizer: GridSearchCV : RMSProp

RandomizedSearchCV : Adamax

Meta-Heuristic Tekniği : Nadam

Learning Rate: GridSearchCV : 0.001

RandomizedSearchCV : 0.01

Meta-Heuristic Tekniği : 0.01

Activation Function: GridSearchCV : Sigmoid

RandomizedSearchCV : Relu

Meta-Heuristic Tekniği : Linear

Süre: GridSearchCV : ~ 70 saat

RandomizedSearchCV : ~ 22 saat

Meta-Heuristic Tekniği : ~ 1,5 saat

Bütün bu değerlere bakıldığında şu sonuç çıkarılabilir. Eğer gerçekleştirilmesi gereken işlem çok yüksek hassasiyetlerde doğru çalışması gerekiyorsa yani hataya olan tolerans azsa ve süre olarak bir sıkıntı bulunmamaktaysa exhaustive search bağlamında Grid Search veyahutta Randomized Search tercih edilebilir. Ancak gerçekleştirilmesi gereken işteki hata payı yüksek olarak tolore edilebilecekse süre parametresi göz önüne alınarak meta-heuristic yöntemler kullanılabilinir.

Gerçekleştirilen scriptler, hazır hale getirilen data, zaman analizleri, elde edilen sonuçlar ve Colab’ta kullanılan ipynb notebook paylaşılan drive linkindedir.