



BSM 724 - Yapay Zekanın Prensipleri

Dr.Öğr.Üyesi Rafet Durgut

Python'da Benzetilmiş Tavlama Kullanarak Özellik Seçimi

(Feature Selection using Simulated Annealing in Python)

İbrahim Mete Çiçek – 2028130021

Veri Analizi ile Makine Öğrenmesi

Merhaba hocam, proje anlatımına geçmeden önce derste anlattığınız gibi giriş yapmak için ilk olarak veri analizinin nasıl yapıldığını genel olarak özetmek istedim. İlk önce veriyi tanırız. Verinin etiketlerini anlarız. Daha sonra eksik gözlemimiz var mı diye bakarız veri girişinde. Varsa bu eksik gözlemleri diğer gözlemlerden yararlanarak mod medyan ortalama gibi yöntemlerle doldururuz veya gözlemi tamamen veriden sileriz. Sonra verinin istatistiksel özelliklerine bakarız. Veriyi tanırız. Veriyi görselleştiririz. Varsa outlierları eliminate ederiz. Verimizi hazırladıktan sonra herhangi bir makine öğrenmesi modeli kullanmadan standardize ederiz veriyi. Daha sonra verimizi %70 train %30 test olmak üzere ikiye böleriz. %70'lik kısım ile kullanacağımız modeli eğitiriz. Daha sonra %30'luk daha önce modelin görmediği kısım ile tahmin yapmasını isteriz. Bu yapılan tahmin sonucunda confision matrix, ROC eğrisi gibi olgularla modelin performansını değerlendiririz. Modelin tahmin gücünü arttırmak için model -parametre tuning yaparız optimum modeli elde etmek için. Daha sonrasında modelimiz tam anlamıyla hazır olduğunda tahmin etmek istediğimiz yeni verileri modelimize verebiliriz.

İlk önce veriyi train ve test kısımları olmak üzere ikiye ayırırsın. Train verisini kullanarak model kendi kendini eğitecek ve sonra test verisi ile modelin tahmin gücünü ölçeceksin, en sonunda da model optimizasyonu yapacaksın mesela K en yakın komşu algoritmasıysa, modelde K değerini optimum olarak belirleyeceksin. Bu işlemler sonunda tahmin modeli hazır predict (modelimiz, yeni veri). Mesela örnek verecek olursak son 10 yılın stok verileri var elimizde. Biz bu yıl ne kadar satış yaparız tahmin edeceğiz, son 10 yılın datasıyla modeli eğiteceğiz sonra bu yılki taze verileri vereceğiz, artık modele söyle ne kadar satarız bu yıl rakam ver diyebileceğiz kabaca süreç bu şekilde olmakta. Sonuç olarak veri analiz kısmı, makine öğrenmesinde olmazsa olmazlarından bu

yüzden iki alanda birbiri ile doğrudan bağlantılı disiplinlerdir. Veri analizi kullanarak makineyi istediğimiz gibi eğitebilir ve mevcut sorunlara optimal çözüm bulabiliriz yani makine öğrenimi, analitik model oluşturmayı otomatikleştiren bir veri analizi yöntemi diyebiliriz. Sistemlerin verilerden öğrenebileceği, kalıpları belirleyebileceği ve minimum insan müdahalesi ile karar verebileceği fikrine dayanan bir yapay zeka dalıdır ve şu an günümüzde aktif şekilde kullanılan popüler olan aynı zamanda gelecek vadeden bir alan olmaktadır.



Veri Bilimi



Veri Analizi



Makine Öğrenmesi

Şekil 1. Disiplinler arasındaki yakın ilişkiler

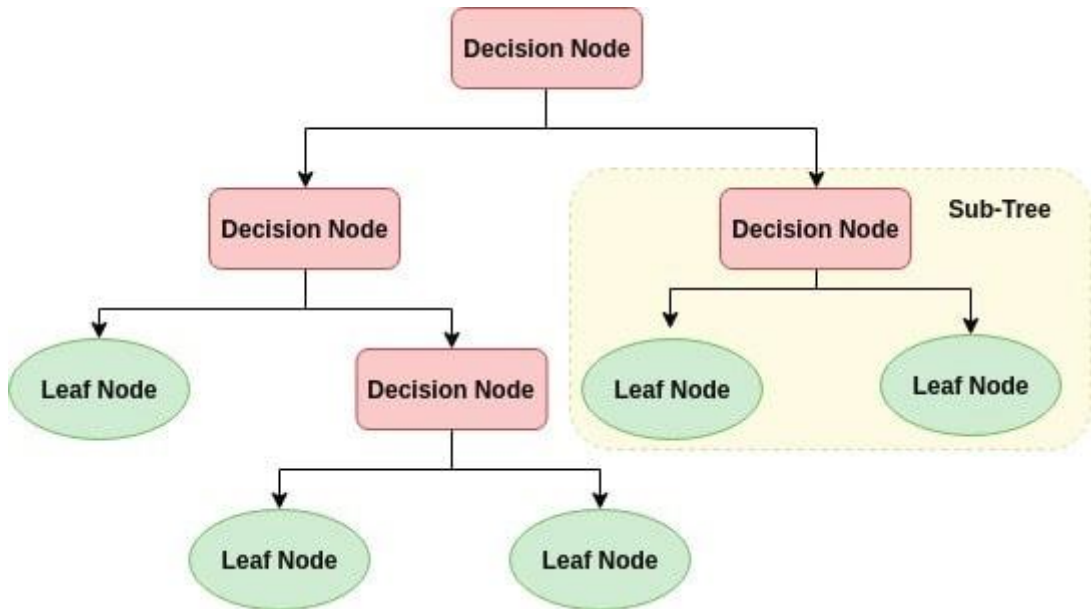
Benzetilmiş Tavlama Kullanarak Özellik Seçimi

Tavlama simülasyonu kontrollü rastgele bir aramadır; yeni aday özellik alt kümesi, mevcut duruma göre tamamen rastgele seçilir. Yeterli sayıda yinelemeden sonra, her bir öngörücü ile ve her bir tahminci olmadan performans farkını ölçmek için bir veri seti oluşturulabilir. Temel özellik alt küme seçim yöntemlerine genel bir bakış verilmiştir. Büyük ticari veri tabanları kullanarak, özellik alt kümesi kalitesinin bir dizi ölçümünü araştırıyoruz. Veri tabanlarımıza göre en iyi performansı verdiği gösterilen ağaçları oluşturmak için bilgi kazanma yaklaşımına dayalı bir entropik ölçü geliştiriyoruz. Bu ölçü, basit bir özellik alt küme seçim algoritmasında kullanılır ve teknik olarak veri tabanlarından yüksek kaliteli özelliklerin alt kümelerini oluşturmak için kullanılır.

Karar Ağacı Sınıflandırıcıları (Decision Tree Classifier)

Karar ağacı algoritması makine öğrenmesinde çok kullanılan bir yaklaşımdır. Bu yaklaşım sınıflandırma ağaçları ve regresyon ağaçları olmak üzere 2 ana kısma ayrılır. Biz bu projede sınıflandırma ağaçlarını kullanacağız. Karar ağacı sınıflandırıcılar birçok farklı alanda başarıyla kullanılmaktadır. En önemli özellikleri, sağlanan verilerden tanımlayıcı karar verme bilgilerini yakalama becerisidir. Karar ağacı eğitim setlerinden oluşturulabilir.

Karar algoritmasını kullanarak, ağaç kökünden başlıyoruz ve en büyük bilgi kazanımıyla sonuçlanan özellik üzerindeki verileri bölüyoruz. Yinelemeli bir süreçte, yapraklar saf hale gelene kadar bu bölme prosedürünü her çocuk düğümde tekrar edebiliriz. Bu, her bir yaprak düğümündeki örneklerin hepsinin aynı sınıfa ait olduğu anlamına gelir. Uygulamada, aşırı oturmaya önlemek için ağacın derinliğine bir sınır koyabiliriz. Son yapraklarda hala bir miktar kirlilik olabileceğinden, burada bir şekilde saflıktan ödün veriyoruz.



Şekil 2. Karar Ağacı Sınıflandırma Şeması

Genetik Algoritma (GA)

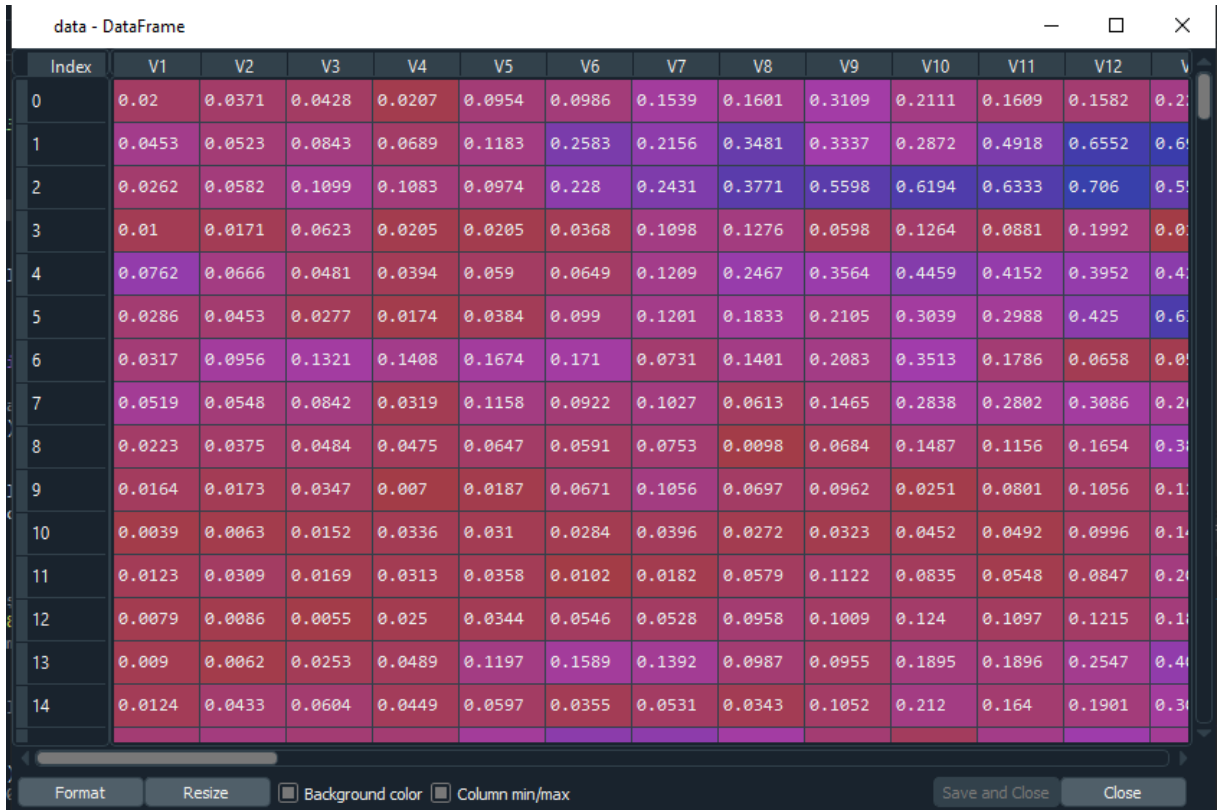
Genetik Algoritmalar arama ve optimizasyon problemlerini çözer. Bir Genetik Algoritmanın temel fikri en iyi hipotezi bulmak için bir hipotez alanını araştırmaktır. Bir havuz popülasyon olarak adlandırılan ilk hipotezlerin oranı rastgele oluşturulur ve her hipotez bir uygunluk işlevi ile değerlendirilir. Daha fazla uygunluğa sahip hipotezlerin olma olasılığı daha yüksektir gelecek nesli yaratmak için seçilir. En iyinin bir kısmı hipotezler sonraki nesle yeniden eğitilebilir, geri kalanı çaprazlama ve mutasyon gibi genetik işlemlere tabi tutulur yeni hipotezler üretir. Bu süreç önceden tanımlanmış bir uygunluk kriteri karşılanıncaya veya önceden ayarlanmış maksimum nesil sayısına ulaşana kadar tekrarlanır.

Genetik Algoritma Tabanlı Özellik Seçimi ve Karar Ağaçları

Önerilen Genetik Algoritma tabanlı özellik seçimi algoritmamız şunlara dayanmaktadır:

Arama bileşeni bir Genetik Algoritmadır ve değerlendirme bileşeni bir karar ağacıdır. İlk popülasyon rastgele oluşturulur. Nüfusun her bir bireyinde, giriş verilerinin bir özelliğini temsil eden genler 1 veya 0 'a atanabilir. Yani 1, temsil edilen özelliğin kullanıldığı anlamına gelir karar ağaçları oluşturulurken; 0 kullanılmadığı anlamına gelir. Sonuç olarak, mevcut popülasyondaki her bir birey bir seçimini temsil eder bu özellikler. Mevcut popülasyondaki her birey için bir karar ağacı kullanılarak oluşturulur. Bu sonuç karar ağacı daha sonra test edilir; sınıflandırma hatası oranı oluşturulur. Bunun birey uygunluğu, bu sınıflandırma hata oranlarının toplamıdır. Bireysel olarak sınıflandırma hata oranı ne kadar düşükse, uygunluk o kadar iyi demektir. Mevcut popülasyondaki tüm bireylerin uygunluk değerleri hesaplanır ve Genetik Algoritma gelecek nesil oluşturmaya başlar.

Öncelikle projemizde 60 özellik olan sınıflandırma için hazırlanmış bir veri seti kullanılmıştır. Şimdi projemizde yapılan bu işlemleri bölümlere ayırıp detaylı olarak inceleyecek olursak, ilk olarak deneklerden alınan sample'larımızın bulunduğu dataframe (Python'da yazılmış olan pandas kütüphanesinde bulunan çok boyutlu veri yapıları) bu şekildedir. (Burada v1, v2, v3, v4, v5 ... bireyin kan tahlilin elde edilen özelliklere yani kan PH değeri gibi her bir parametrenin sütun etiket ataması yapılmıştır.)



Index	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0.02	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	0.1609	0.1582	0.2111
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	0.4918	0.6552	0.6552
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.228	0.2431	0.3771	0.5598	0.6194	0.6333	0.706	0.5598
3	0.01	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	0.0881	0.1992	0.0881
4	0.0762	0.0666	0.0481	0.0394	0.059	0.0649	0.1209	0.2467	0.3564	0.4459	0.4152	0.3952	0.4152
5	0.0286	0.0453	0.0277	0.0174	0.0384	0.099	0.1201	0.1833	0.2105	0.3039	0.2988	0.425	0.6194
6	0.0317	0.0956	0.1321	0.1408	0.1674	0.171	0.0731	0.1401	0.2083	0.3513	0.1786	0.0658	0.0658
7	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	0.1465	0.2838	0.2802	0.3086	0.2802
8	0.0223	0.0375	0.0484	0.0475	0.0647	0.0591	0.0753	0.0098	0.0684	0.1487	0.1156	0.1654	0.3039
9	0.0164	0.0173	0.0347	0.007	0.0187	0.0671	0.1056	0.0697	0.0962	0.0251	0.0801	0.1056	0.1056
10	0.0039	0.0063	0.0152	0.0336	0.031	0.0284	0.0396	0.0272	0.0323	0.0452	0.0492	0.0996	0.1056
11	0.0123	0.0309	0.0169	0.0313	0.0358	0.0102	0.0182	0.0579	0.1122	0.0835	0.0548	0.0847	0.2111
12	0.0079	0.0086	0.0055	0.025	0.0344	0.0546	0.0528	0.0958	0.1009	0.124	0.1097	0.1215	0.1097
13	0.009	0.0062	0.0253	0.0489	0.1197	0.1589	0.1392	0.0987	0.0955	0.1895	0.1896	0.2547	0.4152
14	0.0124	0.0433	0.0604	0.0449	0.0597	0.0355	0.0531	0.0343	0.1052	0.212	0.164	0.1901	0.3039

Şekil 5. Projemizde kullandığımız veri seti

Bu elimizdeki verideki etiketlerden (v1, v2, v3, v4 ...) hangilerinin modelimize bağımsız değişken olarak alacağımızı benzetilmiş tavlama kullanarak özellik seçimi yaparak belirliyoruz. Verideki özellik sayısı arttıkça problem daha çok karmaşıklaşıyor ve başarı oranı düşüyor (Curse of dimensionality). Buradaki amacımız kötü olan özelliklerin çıkartılması ve en iyi olan özelliklerin seçilmesi, o yüzden özellik seçimini metasezgisel optimizasyon algoritmalarından biri olan benzetilmiş tavlama kullanarak yapıyoruz. Binary sınıflandırma yaparak çözümün 1 olduğu yerleri alıyoruz (0 değeri özelliğin kullanılmadığını, 1 ise kullanıldığını ifade ediyor, biz veri kümesini ayırırken sadece 1 olan sütunları alıyoruz). Amacımız maximum başarıyı elde etmek.

```
# Feature selection kodlaması, simulated annealing buradan itibaren başlıyor
def gen_kodlaması(): # Gen Kodlaması
    while True:
        temp = []
        kromozom_degeri_1 = False # Bu kromozomun değerinin 1 olmasının sorgulanması
        for j in range(kromozom_uzunlugu):
            rand = random.randint(0,1)
            if rand == 1:
                kromozom_degeri_1 = True
            temp.append(rand)
        if kromozom_degeri_1: # Kromozomların tümü 0 olamaz
            return temp

def model_tutarlığı(x):
    X_test = X

    kromozom_degeri_1 = False
    for j in range(kromozom_uzunlugu):
        if x[j] == 0:
            X_test = X_test.drop(columns = j)
        else:
            kromozom_degeri_1 = True
    X_test = X_test.values

    if kromozom_degeri_1:
        gtf = tree.DecisionTreeClassifier() # Karar ağacı model yapılandırması
        model_tutarlığı = cross_val_score(gtf, X_test, y, cv=5).mean() # 5 kez çapraz doğrulama optimizasyonu

    return model_tutarlığı
else:
    model_tutarlığı = 0 # Tüm modellerin uygunluğu 0'dır
    return model_tutarlığı
```

Şekil 6. Benzetilmiş tavlama kod bloğu

Bu yöntem sonunda seçilen özellikleri karar ağacı sınıflandırma algoritmasına bağımsız değişken olarak input vererek modelimizi kuruyoruz.

```
def model_tutarlılığı(x):
    X_test = X

    kromozom_degeri_1 = False
    for j in range(kromozom_uzunlugu):
        if x[j] == 0:
            X_test = X_test.drop(columns = j)
        else:
            kromozom_degeri_1 = True
    X_test = X_test.values

    if kromozom_degeri_1:
        gtf = tree.DecisionTreeClassifier() # Karar ağacı model yapılandırması
        model_tutarlılığı = cross_val_score(gtf, X_test, y, cv=5).mean() # 5 kez çapraz doğrulama optimizasyonu

    return model_tutarlılığı
else:
    model_tutarlılığı = 0 # Tüm modellerin uygunluğu 0'dır
    return model_tutarlılığı

# Eski çözümden yeni çözüm üretilmesi
def yeni_kromozom(x):
    mutasyon_noktası = random.randint(0, kromozom_uzunlugu-1) # Rastgele mutasyon noktaları seçimi
    if x[mutasyon_noktası] == 1:
        x[mutasyon_noktası] = 0
    else:
        x[mutasyon_noktası] = 1
    return x
```

Şekil 7. Karar ağacı sınıflandırıcı kod bloğu

```
C:\Users\metec\Desktop\Mete

Source Console Object

Variable explorer Files Help Plots

Console 1/A x

[1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1]
0.6068524970963995

Sonuç: Doğrudan Çocuk sahibi olabilir
17.792619848968343
[1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1]
0.6452961672473867

Sonuç: Doğrudan Çocuk sahibi olabilir
17.436767451988977
[1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1]
0.6452961672473867

0.678443075397166>=0.38449109174404017, Sonuç: Çocuk sahibi
olamaz(1)
17.436767451988977
[1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1]
0.6452961672473867

0.7988398005854387<0.875274809408801, Sonuç: Çocuk sahibi olma
ihtimali var

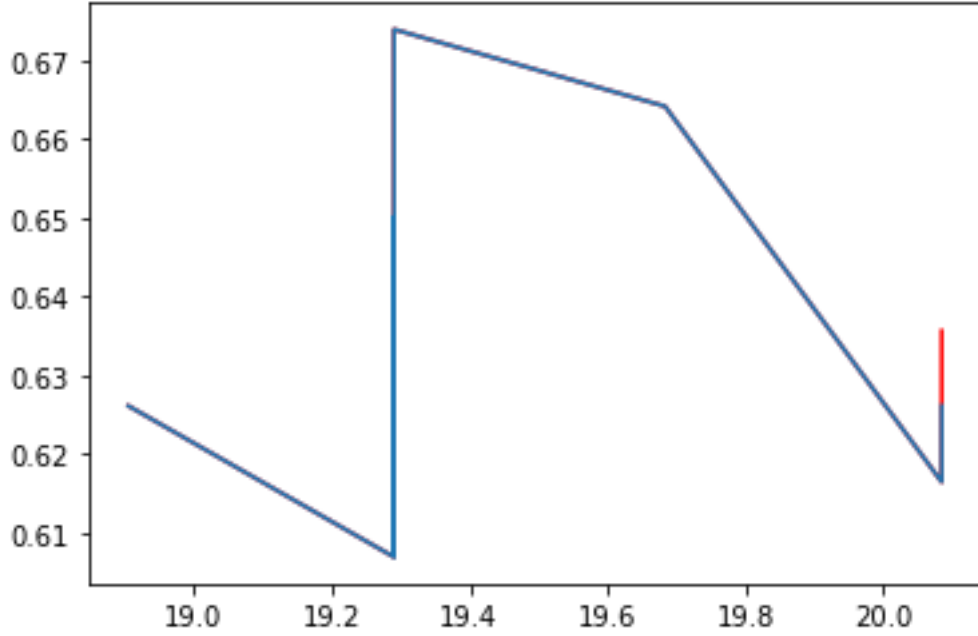
In [9]:

IPython console History

conda: base (Python 3.8.3) Line 7, Col 1 UTF-8 LF RW Mem 79%
```

Şekil 8. Programın olasılıksal çıktısı

Bir üstteki şekilde görüldüğü gibi karar ağacı algoritmasının yaptığı sınıflandırma sonucunda değerlendirilen gözlemin çocuk sahibi olacağı sonucu çıkıyor.



Şekil 9. Programın grafiksel çıktısı

Modelimizin sıcaklık (temperature) parametresine bağlı tahmin gücü yukarıdaki grafikte mevcuttur. Eşik değeri 19.35'tir. Optimum çalışma koşulu ise 19.35 – 19.65 C derece arasındadır. Diğer koşullarda alınan sample'larda bozulma olduğu gözlemlendiği için tutarlılık farklılık gösteriyor. Sonuç olarak iterasyon sayısı arttıkça meydana gelen iyileşmeyi (improving) grafikler üzerinden anlık olarak gözlemleyebiliyoruz.

Referanslar:

- Stein, G., Chen B., Wu, A.S., Hua, K. A. (2008). *Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection*. University of Central Florida. Retrieved from https://www.researchgate.net/publication/224377436_Network_intrusion_detection_using_feature_selection_and_Decision_tree_classifier
- Alfarisy, G. A. F., Sihananto, A. N., Fatyanosa, T. N., Burhan, M. S., Mahmudy, W. F. (2016). *Hybrid Genetic Algorithm and Simulated Annealing for Function Optimization*. Universitas Brawijaya, Faculty of Computer Science. Retrieved from <https://www.inf.unioeste.br/~adair/MOA/Artigos/Hybrid%20Genetic%20Algorithm%20Simulated%20Annealing%20Function%20Optimization.pdf>
- Varma, P., Bhattacharjee, B. (2015). *Evaluating Performance of Simulated Annealing and Genetic Algorithm Based Approach in Building Envelope Optimisation*. Retrieved from <http://www.ibpsa.org/proceedings/BS2015/p2514.pdf>
- Anilkumar, A. K. (2017). *Simulated Annealing Conditions to Generate a New Population in Genetic Algorithm*. Global Journal of Pure and Applied Mathematics. Retrieved from https://www.ripublication.com/gjpam17/gjpamv13n10_44.pdf
- Leung, T. W., Chan, C. K., Troutt, M. D. (2008). *A Mixed Simulated Annealing-Genetic Algorithm Approach to the Multi-Buyer, Multi-Item Joint Replenishment Problem: Advantages of Meta-Heuristics*. Retrieved from <https://www.aims sciences.org/article/exportPdf?id=54b8c141-d670-4186-8c20-b22a68c97c50>