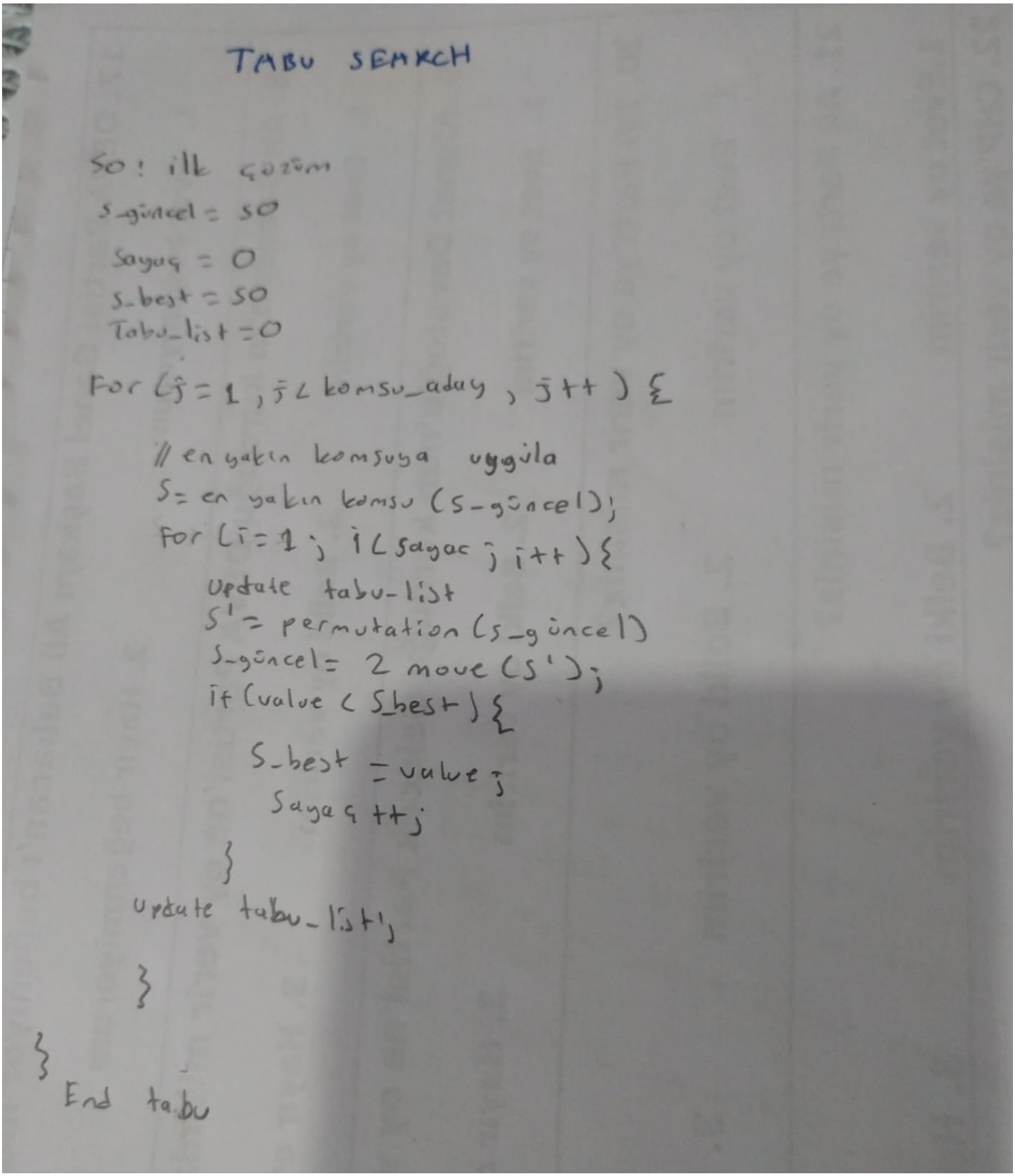
Yapay Zeka Yöntemleri

Proje - 1

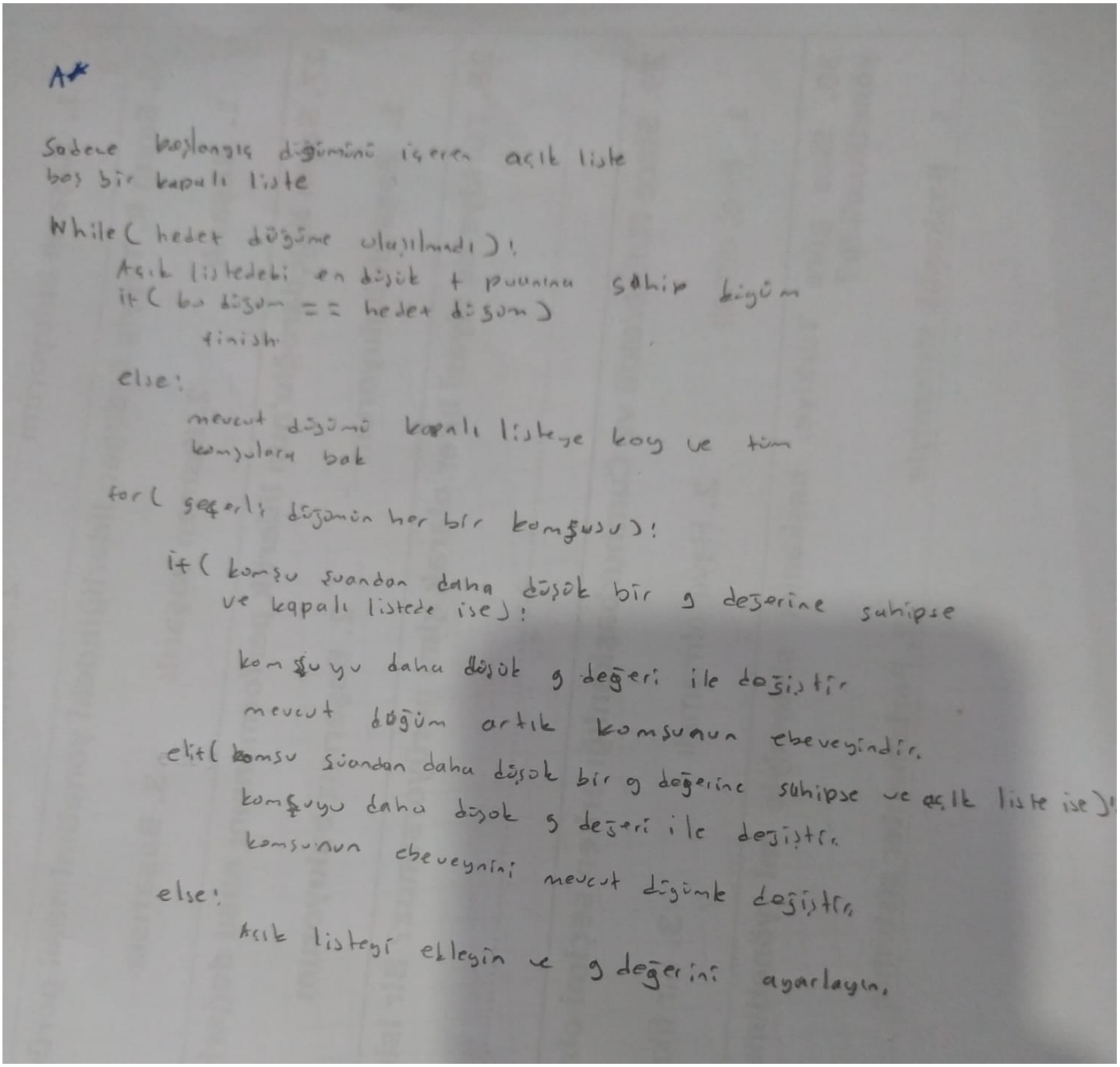
Rapor

Kemal Kağan Orhan 05160000785
Enes Yaşarbaş 05180000939
Güneş Hacıhalil 05170000770

1. Soru



```
TABU SEARCH

S0 : ilk çözüm
S_güncel = S0
Sayaç = 0
S_best = S0
Tabu_list = 0

For (j = 1 , j < komsu_aday , j++ ) {

    // en yakın komşuya uygula
    S = en yakın komsu (S_güncel);
    For (i = 1 ; i < sayac ; i++ ){
        Update tabu-list
        S' = permutation (S_güncel)
        S_güncel = 2 move (S');
        if (value < S_best ) {
            S_best = value ;
            Sayaç ++;
        }
    Update tabu_list ;

    }

}
End tabu
```



A Yıldız Arama Algoritması (A Star Search Algorithm, A*)

Bilgisayar bilimlerinde en kısa yol bulmak için kullanılan algoritmalardan birisidir. Örneğin seyyar tüccar problemi (travelling salesman problem, TSP)_ gibi bir problemin çözümünde kullanılabilir. Benzer şekilde oyun programlamada, oyunda bulunan oyuncuların en kısa yolu bularak hedefe gitmeleri için de sıklıkla kullanılan algoritmadır.

Kısaca bir düğümden (node) hedef bir düğüme (target node) en kısa hangi düğümler üzerinden gidileceğini bulmaya yarayan "en iyi yerleştirme (best fit)"_algoritmasıdır.

A* algoritması yapı olarak muteber sezgisel (admissable heuristic) bir algoritma olarak sınıflandırılabilir. Bunun sebebi algoritmasının mesafe hesaplamada kullandığı fonksiyondur:

$f(n) = g(n) + h(n)$

denklemindeki

$f(n)$ = hesaplama yapan sezgisel (heuristic) fonksiyon.

$g(n)$ = Başlangıç düğümünden mevcut düğüme kadar gelmenin maliyeti

$h(n)$ = Mevcut düğümden hedef düğüme varmak için tahmin edilen mesafe.

Dikkat edileceği üzere f(n) fonksiyonunun sezgisel olma sebebi, bu fonksiyon içerisinde bulunan ve tahmine dayalı olan h(n) sezgisel fonksiyonudur.

Algoritmanın çalışması:

Algoritma yukarıdaki toplama işlemini kullanan oldukça basit bir yapıya sahiptir. Veri yapısı olarak bir öncelik sırası (priority queue) kullanan algoritmada en öncelikli olan düğüm f(n) değeri en düşük olan düğümdür.

1. Algoritma her adımda en düşük değeri (Ve dolayısıyla en önemli) düğümü alır (yani bu düğüme gider) ve düğümü sıradan (queue) çıkarır.
2. Gidilen bu düğüme göre komşu olan bütün düğümlerin değerleri güncellenir (artık bu düğüme gelmenin bir maliyeti vardır ve dikkat edilirse f(n) fonksiyonu içerisinde bu değer yer almaktadır.)
3. Algoritma yukarıdaki adımları hedefe varana kadar (yani hedef düğümü öncelik sırasında (priority queue) en öne gelene kadar) veya sırada (queue) düğüm kalmayana kadar tekrarlar.

Yapay Zekada Mini-Max Algoritması

- Mini-max algoritması, karar vermede ve oyun teorisinde kullanılan özyinelemeli veya geriye dönük bir algoritmadır. Rakibin de en iyi şekilde oynadığını varsayarak oyuncu için optimal bir hamle sağlar.
- Mini-Max algoritması, oyun ağacında arama yapmak için özyinelemeyi kullanır.
- Min-Max algoritması çoğunlukla AI'da oyun oynamak için kullanılır. Satranç, Dama, tic-tac-toe, go ve çeşitli çekici oyuncu oyunları gibi. Bu Algoritma, mevcut durum için minimum maksimum kararını hesaplar.

GPT-3 Nedir?

Generative Pre-trained Transformer 3") isminin kısaltılmış halini ifade eder.

OpenAI tarafından geliştirilen GPT-3, insan veya makine dili gibi bir dil yapısına sahip içerikler oluşturmada kendinden önce gelen her modelden daha iyi olan bir yapay zeka modelidir. , *OpenAI* ekibi, *Wikipedia* metninin tamamı dahil olmak üzere, "*CommonCrawl"* olarak bilinen halka açık veri kümelerini internette taramış ve toplanılan yaklaşık 570 GB boyutundaki metin verileri ile GPT-3'ü beslemişlerdir.

Random Forest vs. Ensemble Learning

Rastgele orman, adından da anlaşılacağı gibi, bir topluluk olarak çalışan çok sayıda bireysel karar ağacından oluşur . Rastgele ormandaki her bir ağaç, bir sınıf tahmini verir ve en çok oyu alan sınıf, modelimizin tahmini haline gelir**.** Bireysel tahminlerin herhangi birinden daha doğru olan topluluk tahminleri üretebilir. Bunun nedeni, ağaçların birbirlerini bireysel hatalarından korumalarıdır Rastgele orman, hem sınıflandırmalar hem de regresyon problemleri için kullanılan denetimli bir toplu öğrenme algoritmasıdır Ensemble Learning ise . Genel performansı iyileştirmek için birden fazla modelden alınan kararları birleştirirler*.,* Tekli modellere kıyasla farklı modellerde daha iyi ve sağlıklı sonuçlar alırsınız. Yani kısaca random forest genelde sınıflandırılmada kullanılırken ensemble learning ise karar verme durumlarında kullanılır. Algoritmaları benzerdir fakat amaçları farklıdır.

1. https://towardsdatascience.com/understanding-random-forest-58381e0602d2

2. http://www.scholarpedia.org/article/Ensemble_learning#:~:text=Ensemble%20learning%20is%20the%20process,%2C%20function%20approximation%2C%20etc.)

Convolutional Neural Networks (CNN) vs. GAN

CNN görüntünün algılanmasına kullanılan algoritmadır.Yani biz aslında bir şeyleri görürüz fakat beynimiz onun ne olduğunu algılar.CNN ise tam olarak bunu yapıyor. GAN ise gerçek bir görüntüyü tıpkı bir ressam gibi kendi yorumunu katarak ya aynısını modelliyor yada benzerini ortaya çıkarıyor.Son yüzyılın en büyük ve ciddi tehlikelere yol açabilecek buluş olan deep fake teknolojisinin altında yatan algoritma da GAN dır.

https://medium.com/@muhammedbuyukkinaci/generative-adversarial-networks-gan-nedir-türkçe-5819fe9c1fa7

https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad

Strong AI vs. Weak AI

Terimler, güçlü AI'nın zayıf AI'dan daha iyi performans gösterdiğini veya "daha güçlü" olduğunu ima etmez. Popüler iPhone'un Siri ve Amazon'un Alexa'sı AI olarak adlandırılabilir,

ancak genellikle zayıf AI programlarıdır. Sesle etkinleştirilen yardım genellikle programlanmış bir yanıta sahip olduğundan, bu sınıflandırma denetimli ve denetimsiz programlama arasındaki farktan kaynaklanmaktadır.

Yaptıkları şey, zaten bildiklerine benzer şeyleri algılamak veya 'taramak' ve bunları buna göre sınıflandırmaktır.

Ancak, yalnızca yapmaya programlandığı şeye yanıt veriyor. Söylediklerinden anlamıyor ya da anlam çıkarmıyor. Pek çok filmde öne çıkan güçlü yapay zeka, daha çok insan beyni gibi işlev görüyor. Sınıflandırma yapmazlar ve verileri işlemek için kümeleme ve ilişkilendirme kullanırlar Güçlü AI, farklı durumlarda hareket etmesine yardımcı olan karmaşık bir algoritmaya sahipken, zayıf AI'lardaki tüm eylemler bir insan tarafından önceden programlanmıştır.

Makine Öğrenmesi alanına başlayanların kaçınması gereken hatalar

Yaygın Makine Öğrenimi Hatası # 1: Verilere Bakmama
Verilere dikkatlice bakmazsanız, yararlı içgörüleri gözden kaçırabilirsiniz.

Yaygın Makine Öğrenimi Hatası # 2: Veri Sızıntısı Aramamak
Veri sızıntısı , bir eğitim veri kümesi tahmin anında mevcut olmayacak bilgiler veya ipuçları içerdiğinde meydana gelir.
 genel bir kural olarak, sonuçlarınız son derece iyiyse (örneğin, çeşitli ölçümler için yaklaşık% 90'dan fazla) veri sızıntısı olduğundan şüphelenmek isteyebilirsiniz
Yaygın Makine Öğrenimi Hatası # 3: Test Setine Geliştirme
Tek bir test setinde (Kaggle "Herkese Açık") performansı artırmaya çalışırsanız, sonuçlarınızın yeni bir kör sette olduğundan daha iyi olduğunu düşünerek kendinizi kandırabilirsinizYeterli veriye sahip olduğunuzdan emin olun. Başlangıçta birden fazla test setini bir kenara koyun. ir test setini çok fazla kullanmaktan kaçının.

Yaygın Makine Öğrenimi Hatası 4: Modele Bakmama

Isı haritaları, vurgulama, kümeleme, t-dağıtılmış stokastik komşu gömme (t-SNE) , tek tip manifold yaklaşımı ve projeksiyonu (UMAP) ve gömme projektörü gibi görselleştirme tekniklerini kullanmayı düşünün .

Yaygın Makine Öğrenimi Hatası # 5: Basit Bir Temel Modelle Karşılaştırılmama

- Karmaşık bir modelle başlamayın. Önce basit bir temel modeli deneyin. Sınıflandırma problemleri için, çoğunluk sınıfı temelini hesaplayın. Sonra daha karmaşık modelleri deneyin.
- Mimarinizin her bölümünün katkısını ve mimarinin hangi bölümlerine gerçekten ihtiyaç duyulduğunu belirlemek için ablasyon çalışmaları yapın.

Yaygın Makine Öğrenimi Hatası # 6: Kullanım Senaryosuna Uygun Değil

Bir makine öğrenimi projesine başlamadan önce, projenin yapmaya değer olup olmadığını belirlemek ve sonuçlarını değerlendirmek önemlidir.

Yaygın Makine Öğrenimi Hatası # 7: Kullanıcıyı Anlamamak

Kullanıcınız (müşteri ve / veya işletme) gerçekten neye ihtiyaç duyuyor ve istiyor?

Kullanıcıların nasıl çalıştığını ve neye ihtiyaç duyduklarını ve istediklerini inceleyerek sisteminizi geliştirebilirsiniz. Bu, geleneksel yazılım için olduğu kadar makine öğreniminde de geçerlidir.

Yaygın Makine Öğrenimi Hatası # 8: Teknoloji Aşkına Teknoloji Geliştirme
Yaygın Makine Öğrenimi Hatası 9: Mevcut Çözümleri Kullanmamak

- Orada ne olduğunun farkında olun.
- Geçmiş çalışmaları arayın.
- Düzenli olarak yeni bir arama yapın. Teknolojinin durumu sürekli değişiyor. Birçok araştırma makalesi arXiv.org e-print arşivinde yayınlanır ve arXiv e-posta uyarı hizmetine abone olabilirsiniz .

Yaygın Makine Öğrenimi Hatası # 10: Hata Analizi Gerçekleştirmeme

Sisteminizin farklı arıza kategorilerinin sıklığının bir analizi olan bir arıza analizi yapmazsanız, çok az sonuç için çok fazla çaba harcıyor olabilirsiniz.

- Arızaları analiz edin. Hataları problem kategorilerine ayırın ve her problem kategorisinin sıklığını hesaplayın.

2. soru

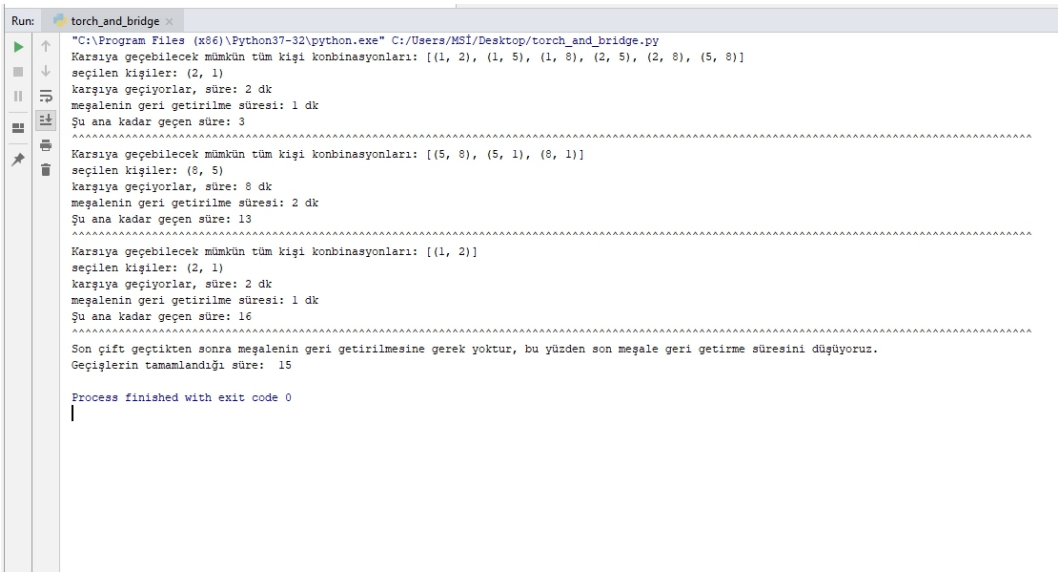
Problemin tanımı:

Her bir kişinin köprüden geçme ve meşaleyi geri getirme sürelerini farklı mesafeli yol seçenekleri gibi düşünüp, bu durumu algılayan ve en kısa yolu, yani en kısa süreyi hesaplayan bir A* mekanizması geliştirmek. Bu sistemi bilgisayar ortamı üzerinde göstermek ve istenilen A* mekanizmasını fonksiyon olarak tasarlamak. Bu A* mekanizmasının F(n)=G(n)+H(n) formülündeki H(n) tahmini değeri öyle bir şekilde tasarlanmalı ki, H(n) değeri hem meşalenin geri döndürüleceği en kısa süre için köprünün karşısındaki bireyleri, hem de köprüden henüz geçmemiş bireyleri göz önüne almalı.

Programın özellikleri:

Bulmacada köprüden geçmesi gereken kişiler, geçiş süreleri ile temsil edilerek 'buradakiler' adlı bir liste olarak yazıldı. (Bu listedeki bu kişiler bulmacanın orjinalinde verildiği değerler ile yazılmıştır, ancak değiştirilebilir, bu listeye başka değer ekleyip grubu büyütmek yada listedeki insanların geçiş sürelerini değiştirmek mümkündür.) Karşıya geçmesini tamamlamış köprünün öbür tarafındaki insanlar için de boş bir liste açıldı. Henüz karşıya geçmemiş kişiler arasından mümkün tüm çift kombinasyonlarını belirlemek üzere itertools kütüphanesine ait komutlar kullanıldı. Köprünün karşısından meşaleyi geri getirecek kişi, çift değil tek olduğu için **dönen_sec** fonksiyonu bu iş için tanımlandı. Köprünün karşısına geçecek olası çiftlerden belirli bir tanesini değerlendirmesi için **çift_degerlendir** fonksiyonu tanımlandı, tasarlanan F(n)=G(n)+H(n) formülü burada implemente edildi. **cift_degerlendir** fonksiyonunu tüm çiftler üzerinde uygulayarak en uygun çifti seçmesi için **cift_sec** metotu tanımlandı. Tüm bu fonksiyonlar, gidiş-geliş işlemleri herkes karşıya geçene kadar tekrar tekrar yapılsın diye while döngüsü içerisinde çağırıldı. Sistem git-gel dinamiği üzerine kurulu olduğu için son çiftin karşıya geçme işleminden sonra bir kişi tekrar geri dönüyordu, bu işlem while döngüsünden hemen sonra matematiksel olarak geri alındı. Gerekli görülen işlemler ve değerler ekrana print aracılığı ile yazdırıldı. Bu şekilde liste ile geçiş süreleri verilen bireyler için en kısa karşıya geçme süresini (bulmacaya uygun olarak) hesaplayan kod yazılmış oldu.

Çıktı Ekranı:



3. soru

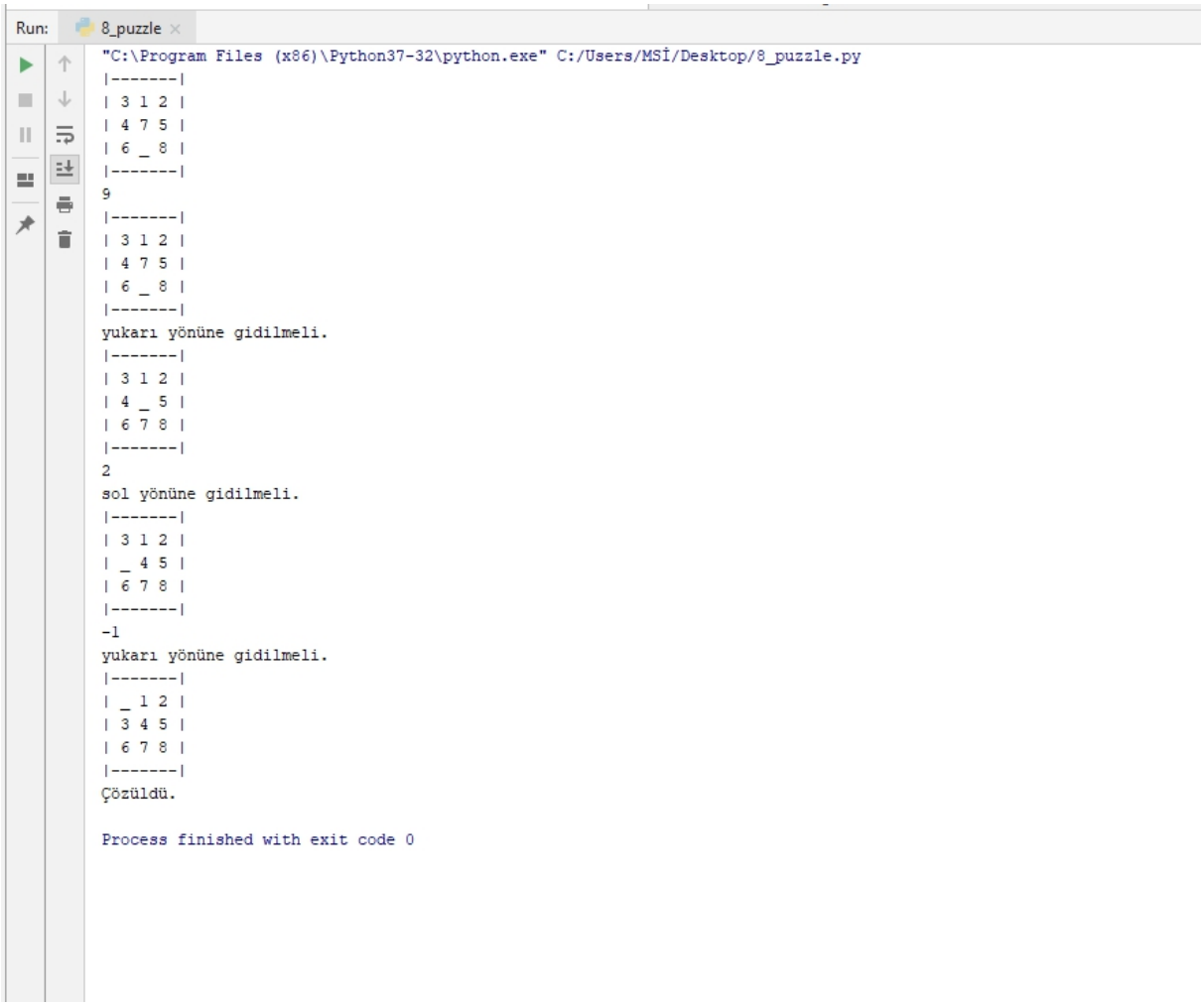
Problemin tanımı :

8-puzzle oyunu sürekli olarak mümkün hamleler arasından en mantıklı gözükenini şeçip uygulayarak çözmeye çalışan ve bu sayede genetik AI ı simüle etmekte olan bir program yazılmalı. Her seferinde o anki tahta ile istenen/hedeflenen tahta kıyaslanmalı, hangi hamlede hedef tahtaya daha çok yaklaşılacağı bu şekilde bulunmalı ve uygulanmalı. Bu işlemi çözüme ulaşana kadar tekrar eden döngü yazılmalı. Ancak bu bulmacada (8-puzzle'da) çözülmesi mümkün olmayan durumlar var olduğu ve gayet sıkça görüldüğü için uygulama böyle bir durumda yada benzeri bir durumda sağ-sol yada aşağı-yukarı döngüsünde sonsuza kadar takılacağı için sonsuz döngüye girmesi önlenmeli.

Programın özellikleri:

Her seferinde oyundaki tahtayı ve hedeflenen tahtayı temsil ve simüle etmesi için 2 adet liste oluşturuldu. Tahtaları ekrana yazdıran ve bir hamle yapıldığında gerekli değişikleri uygulayan metodlar kabaca tasarlandı. Boş karenin konumuna göre yapılabilecek hamleler sınırlı oluduğu için yapılabilecek hamleleri belirlemesi için, **gidilebilecek_yönleri_belirle** fonksiyonu tanımlandı. Önceki bahsedilen hamle gibi metodlar, **gidilebilecek_yönleri_belirle** fonksiyonunu çağıracak şekilde güncellendi. O anki tahtayı hedefle kıyaslaması için kontrol metodu geliştirildi, bu metod her bir karedeki sayıyı aynı sayının hedeflenen tahtadaki konumu ile karşılaştırıp, iki konum arasındaki uzaklığı kriter adı verilen bir sayıya ekliyordu, bu sayede tüm sayıların olması gereken yere uzaklıkları toplamı kriter adı ile hesaplanıp döndürülüyordu. Bir kademe daha etkin hale getirilmesi için hedeflenen konumda bulunan her sayı için kriterin 1 düşürülmesi düşünüldü, kontrol bu şekilde güncellendi. Verilen bir hamle yönünün tam tersini hesaplayan **yön_tersini_bul** fonksiyonu, geri alma ve döngü önleme gibi işlemler için oluşturuldu. O anki tahta da mümkün tüm hamleleri deneyip en uygununu seçmesi için **jenerasyondan_sec** fonksiyonu yazıldı. Bu fonksiyon her seferinde tüm mümkün hamleler için sırası ile; o hamleyi yapıp, kontrol 'ü çağırıp kriteri hesapladıktan sonra bu hamleyi geri alıyordu. Tüm bu fonksiyonlar kullanılarak bulmacayı çözmeye çalışan döngü tasarlandı, git-gel döngüsünde takılması önlendi. Boş karenin ne yöne ilerlediğini ve tahtanın o anki durumunu ekrana yazdıracak print ve **tahta_yazdır** 'lar eklendi.

Çıktı Ekranı:



Program kodu:

```
sample_random=[3,1,2,4,7,5,6,"_",8]
perfect=["_",1,2,3,4,5,6,7,8]



def mutlak_deger(x):
    if x>=0:
        return x
    else:
        return -1*(x)

def tahta_yazdır(list):
    print("|-------|")
    print("|",list[0], list[1], list[2],"|")
    print("|",list[3], list[4], list[5],"|")
    print("|",list[6], list[7], list[8],"|")
    print("|-------|")

tahta_yazdır(sample_random)
```

```
##tahta_yazdır(perfect)

def hamle(yön):
   pozisyon=sample_random.index("_")

   L=gidilebilecek_yönleri_belirle(sample_random)
   if yön in L:
      if yön == "sağ":
         sample_random[pozisyon] = sample_random[pozisyon + 1]
         sample_random[pozisyon + 1] = "_"
      elif yön == "sol":
         sample_random[pozisyon] = sample_random[pozisyon - 1]
         sample_random[pozisyon - 1] = "_"
      elif yön == "aşağı":
         sample_random[pozisyon] = sample_random[pozisyon + 3]
         sample_random[pozisyon + 3] = "_"
      elif yön == "yukarı":

         sample_random[pozisyon] = sample_random[pozisyon - 3]
         sample_random[pozisyon - 3] = "_"

   else :
      print("bu yöne şu an gidemez.")


def kontrol(sample_random,perfect):
  kriter=0
  for i in range (0,9):
     if sample_random[i]==perfect[i]:
        kriter=kriter-1
     else:
        kriter=kriter+mutlak_deger(i-perfect.index(sample_random[i]))

  return kriter


def gidilebilecek_yönleri_belirle(list):
   yönler=["sol","sağ","yukarı","aşağı"]

   if list.index("_")==0 or list.index("_")==1 or list.index("_")==2 :
      yönler.remove("yukarı")

   if list.index("_")==6 or list.index("_")==7 or list.index("_")==8 :
      yönler.remove("aşağı")

   if list.index("_")==0 or list.index("_")==3 or list.index("_")==6 :
      yönler.remove("sol")

   if list.index("_")==2 or list.index("_")==5 or list.index("_")==8 :
      yönler.remove("sağ")

   return yönler


def jenerasyondan_sec():
   yönler=gidilebilecek_yönleri_belirle(sample_random)
   degerler=[]
   for i in yönler:
      hamle(i)
      degerler.append(kontrol(sample_random,perfect))
      hamle_geri_al(i)

   return yönler[degerler.index(min(degerler))]

def hamle_geri_al(move):
   hamle(yön_tersini_bul(move))

def yön_tersini_bul(yön):
   if yön == "sağ":
      return ("sol")
```

```
    elif yön == "sol":
        return ("sağ")

    elif yön == "yukarı":
        return ("aşağı")

    elif yön == "aşağı":
        return ("yukarı")


print(kontrol(sample_random,perfect))

##hamle("yukarı")
##print(kontrol(sample_random,perfect))
tahta_yazdır(sample_random)

önceki_hamle="1"
while kontrol(sample_random,perfect)>-9 and önceki_hamle!=yön_tersini_bul(jenerasyondan_sec()):
    print(jenerasyondan_sec(), "yönüne gidilmeli.")
    önceki_hamle = jenerasyondan_sec()
    hamle(jenerasyondan_sec())
    tahta_yazdır(sample_random)

    if (kontrol(sample_random, perfect))!=-9:
        print(kontrol(sample_random, perfect))
    elif (kontrol(sample_random, perfect))==-9:
        print("Çözüldü.")
```

4. Soru

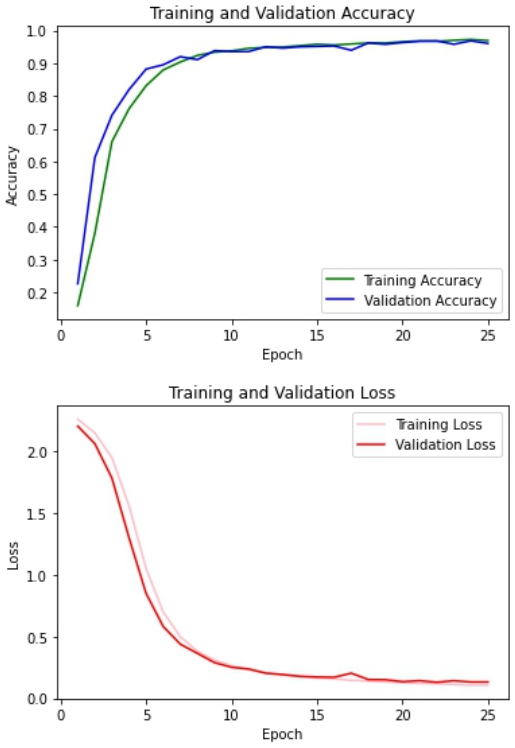
Verisetinin ve Problemin tanımı

Verisetinin ismi: Optical Recognition of Handwritten Digits
Öznitelik sayısı: 64
Kategori sınıfları: 0,1,2...9
Toplam kategorisinif sayısı: 10
Veri sayısı: 3823 Training verisi, 1797 Test seti verisi

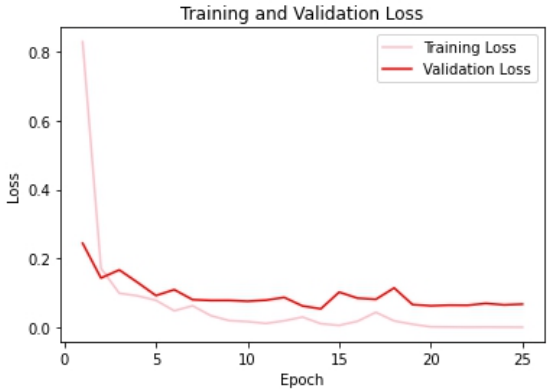
Tanımı:

NIST tarafından kullanıma sunulan ön işlemeli programlarını, önceden yazılmış bir formdan normalleştirilmiş bit haritalarını çıkarmak için kullanılmıştır. Bu dataset'in toplam 43 kişiden 30'u training verisi, 13'ü de test seti verisine katkıda bulundu. 32x32 bit haritaları, birbirleri üzerine denk gelmeyen 4x4 bloklara bölünür ve bu oluşan bloklardaki pixek sayıları elde edilir. Bu işlemler bize 8x8lik bir girdi matrisi oluşturur ve her elemanı 0 ile 16 sayıları arasındadır. Bu boyutsallığı azaltır ve küçük bozulmalara değişmez izni vermez.

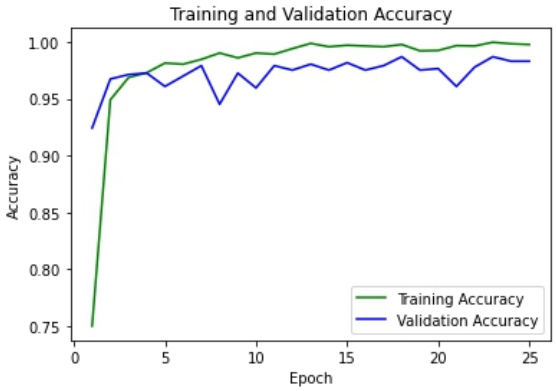
MLP Sınıflandırıcı Çıktıları

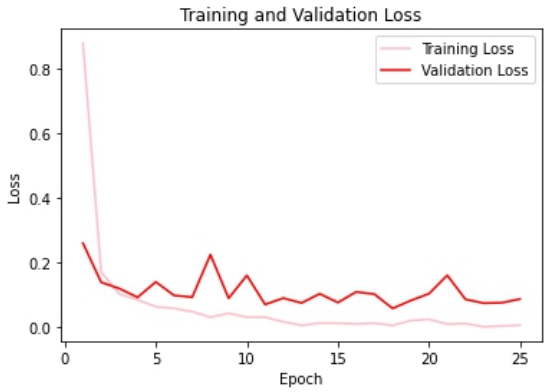


Model 1

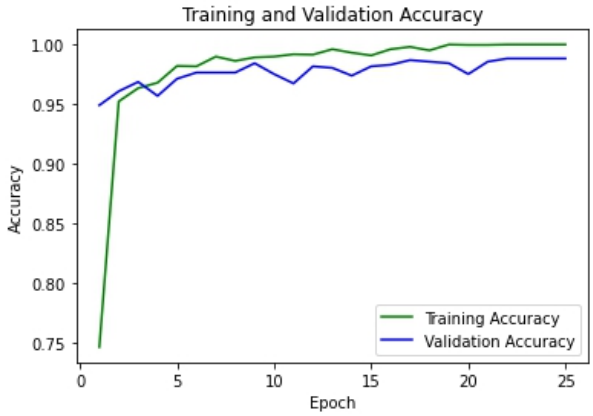


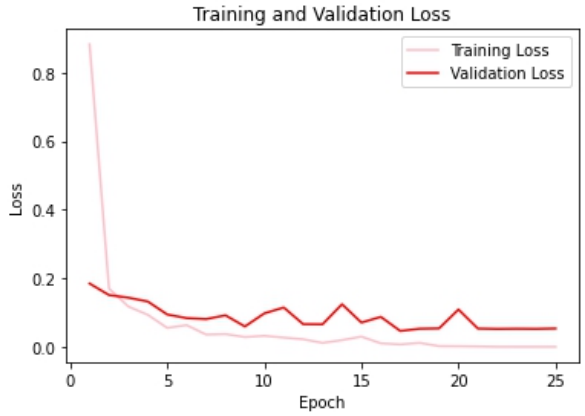
Model 2



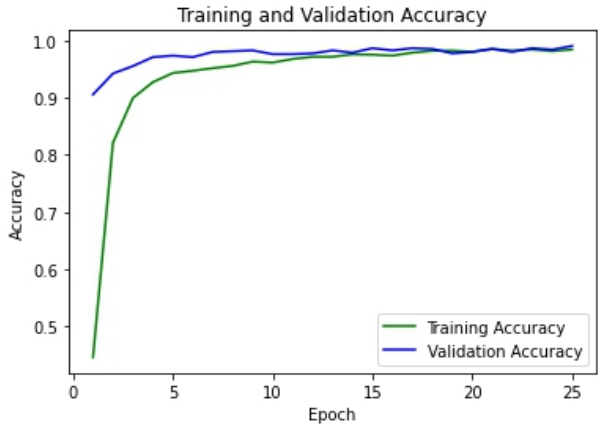


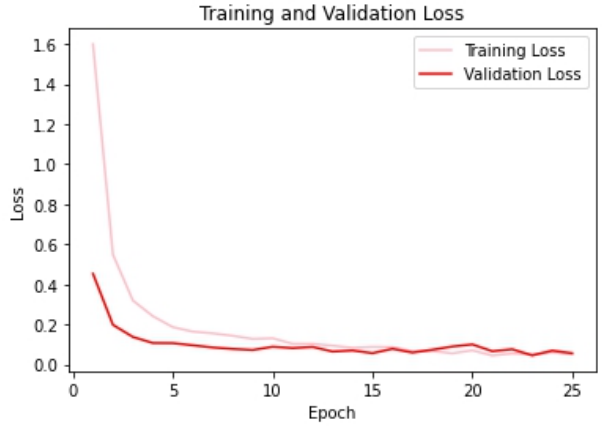
Model 3



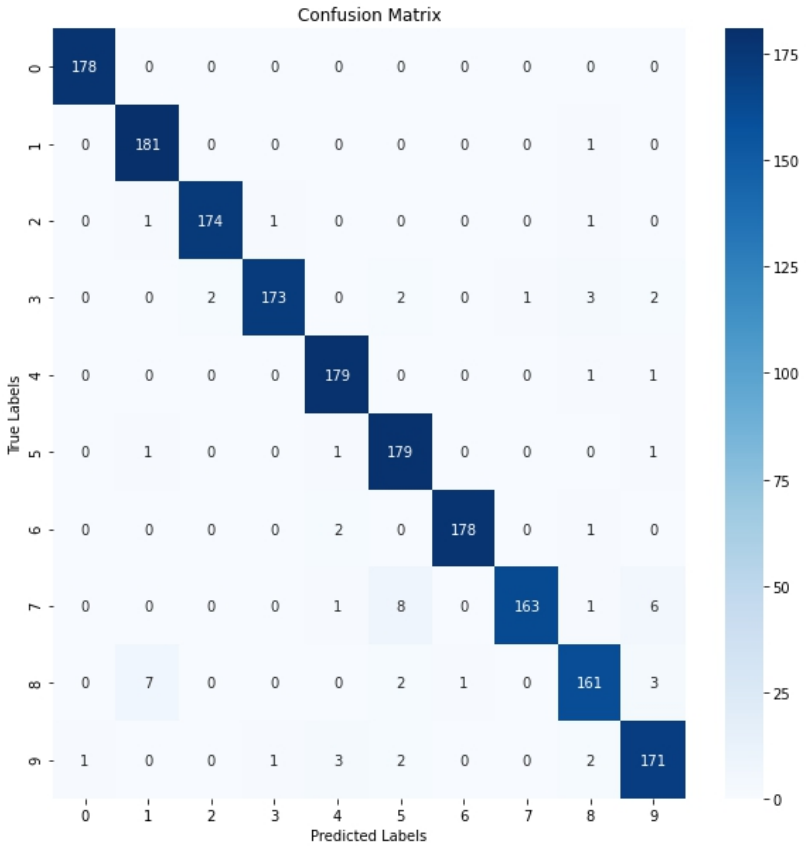


Model 4



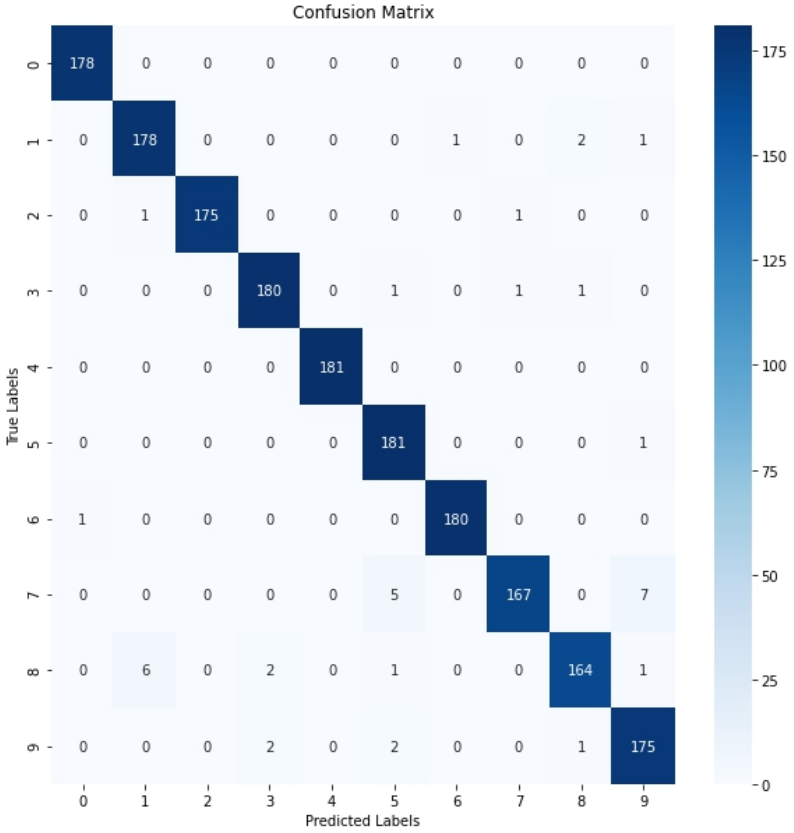


Model 5



SVM Sınıflandırıcı Çıktıları

	precision	recall	f1-score	support
0	0.99	1.00	1.00	178
1	0.96	0.98	0.97	182
2	1.00	0.99	0.99	177
3	0.98	0.98	0.98	183
4	1.00	1.00	1.00	181
5	0.95	0.99	0.97	182
6	0.99	0.99	0.99	181
7	0.99	0.93	0.96	179
8	0.98	0.94	0.96	174
9	0.95	0.97	0.96	180
accuracy			0.98	1797
macro avg	0.98	0.98	0.98	1797
weighted avg	0.98	0.98	0.98	1797

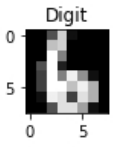


Konsol Çıktıları

MLP Sınıflandırıcı:

```
# Plotting entering sample matrix form

digit = new_sample_feature_values.reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(digit, cmap='gray')
plt.title("Digit")
```

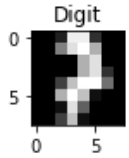
```
Text(0.5, 1.0, 'Digit')
```



SVM Sınıflandırıcı:

```
# Plotting entering sample matrix form

digit = new_sample_feature_values.reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(digit, cmap='gray')
plt.title("Digit")
```

```
Text(0.5, 1.0, 'Digit')
```



Kaynak kod:

MLP Sınıflandırıcı

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report

import keras
from keras.models import Model, Sequential
from keras.layers import *
from keras import optimizers
from keras.wrappers.scikit_learn import KerasClassifier

#Training Dataset
dataset_train = pd.read_csv('dataset/optdigits.tra', header=None)

dataset_feautes = dataset_train.iloc[:,:-1].values
dataset_labels = dataset_train.iloc[:,-1].values
dataset_feautes
dataset_feautes.shape
dataset_labels
dataset_labels.shape

#Test Dataset
dataset_test = pd.read_csv('dataset/optdigits.tes', header=None)

dataset_feautes_test = dataset_test.iloc[:,:-1].values
dataset_labels_test = dataset_test.iloc[:,-1].values
dataset_feautes_test
dataset_labels_test
dataset_labels_test.shape

X_train, X_val, y_train, y_val = train_test_split(dataset_feautes, dataset_labels,
                                  test_size = 0.2,
                                  random_state = 42)
print(X_train.shape, X_val.shape)
X_test = dataset_feautes_test.reshape(1797, 64)
X_test.shape
X_train[1]
print((min(X_train[1]), max(X_train[1])))
X_train

# Feature Normalization
X_train = X_train.astype('float32')
X_val= X_val.astype('float32')
X_test = X_test.astype('float32')

X_train /= 16
X_val   /= 16
```

```
X_test  /= 16
X_train
X_val
X_test
print(y_train[0])
print(y_train[34])
print(y_train[1075])
#One-Hot Encoding with Keras
y_train = keras.utils.to_categorical(y_train)
y_val = keras.utils.to_categorical(y_val)
print(y_train[0])
print(y_train[34])
print(y_train[1075])
num__of_digits = 10

model = Sequential()
model.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
model.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
model.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
model.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
model.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

# Input Layer - 4 Hidden Layer - Output Layer
print("MODEL SUMMARY\n")

model.summary()
# Defining learning rate
learning_rate = 0.1

sgd = optimizers.SGD(lr=learning_rate)

model.compile(optimizer='sgd',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
# Defining batch and epoch sizes
batch_size = 32
epochs = 25

history  =  model.fit(X_train, y_train,
                batch_size = batch_size,
                epochs = epochs,
                validation_data=(X_val, y_val),
                verbose=2)
best_val_accuracy = max(history.history['val_accuracy'])
print("Best Validation Accuracy: %",best_val_accuracy*100)
#Plot Accuracy and Loss

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
```

```
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()

#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
num__of_digits = 10

model2 = Sequential()
model2.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
model2.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
model2.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
model2.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
model2.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

# Input Layer - 4 Hidden Layer - Output Layer
print("MODEL SUMMARY\n")

model2.summary()
# Defining learning rate
learning_rate = 0.1

# We use Adam optimizer for Model 2.
adam = optimizers.Adam(lr=learning_rate)

model2.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
# Defining batch and epoch sizes
batch_size = 32
epochs = 25

history  =  model2.fit(X_train, y_train,
                batch_size = batch_size,
                epochs = epochs,
                validation_data=(X_val, y_val),
                verbose=2)
best_val_accuracy = max(history.history['val_accuracy'])
print("Best Validation Accuracy: %",best_val_accuracy*100)
#Plot Accuracy and Loss

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
```

```
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()

#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
num__of_digits = 10

model3 = Sequential()
model3.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
model3.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
model3.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
model3.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
model3.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

# Input Layer - 4 Hidden Layer - Output Layer
print("MODEL SUMMARY\n")

model3.summary()
# Defining learning rate
# We use lr=0.01 for Model 3.
learning_rate = 0.01

# We use Adam optimizer for Model 3.
adam = optimizers.Adam(lr=learning_rate)

model3.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
# Defining batch and epoch sizes
batch_size = 32
epochs = 25

history  =  model3.fit(X_train, y_train,
                batch_size = batch_size,
                epochs = epochs,
                validation_data=(X_val, y_val),
                verbose=2)
best_val_accuracy = max(history.history['val_accuracy'])
print("Best Validation Accuracy: %",best_val_accuracy*100)
#Plot Accuracy and Loss

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
```

```
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()

#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
num__of_digits = 10

# We add one mode hidden layer for 4th Model.

model4 = Sequential()
model4.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
model4.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
model4.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
model4.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
model4.add(Dense(100, activation='relu', name='Hidden_Layer_5'))
model4.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

# Input Layer - 5 Hidden Layer - Output Layer
print("MODEL SUMMARY\n")

model4.summary()
# Defining learning rate
learning_rate = 0.01

# We use Adam optimizer for Model 4.
adam = optimizers.Adam(lr=learning_rate)

model4.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
# Defining batch and epoch sizes
batch_size = 32
epochs = 25

history  =  model4.fit(X_train, y_train,
                batch_size = batch_size,
                epochs = epochs,
                validation_data=(X_val, y_val),
                verbose=2)
best_val_accuracy = max(history.history['val_accuracy'])
```

```
print("Best Validation Accuracy: %",best_val_accuracy*100)
#Plot Accuracy and Loss

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()

#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
num__of_digits = 10

# We add Dropout layers to prevent overfitting.

model5 = Sequential()
model5.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
model5.add(Dropout(0.3))
model5.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
model5.add(Dropout(0.3))
model5.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
model5.add(Dropout(0.3))
model5.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
model5.add(Dropout(0.3))
model5.add(Dense(100, activation='relu', name='Hidden_Layer_5'))
model5.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

# Input Layer - 5 Hidden Layer - Output Layer
print("MODEL SUMMARY\n")

model5.summary()
# Defining learning rate
learning_rate = 0.01

# We use Adam optimizer for Model 5.
adam = optimizers.Adam(lr=learning_rate)

model5.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy'])
```

```
# Defining batch and epoch sizes
batch_size = 32
epochs = 25

history  =  model5.fit(X_train, y_train,
                batch_size = batch_size,
                epochs = epochs,
                validation_data=(X_val, y_val),
                verbose=2)
best_val_accuracy = max(history.history['val_accuracy'])
print("Best Validation Accuracy: %",best_val_accuracy*100)
#Plot Accuracy and Loss

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()

#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
def create_model5():

   num__of_digits = 10

   # We add Dropout layers to prevent overfitting.

   model5 = Sequential()
   model5.add(Dense(200, input_dim=64, activation='relu', name='Hidden_Layer_1'))
   model5.add(Dropout(0.3))
   model5.add(Dense(100, activation='relu', name='Hidden_Layer_2'))
   model5.add(Dropout(0.3))
   model5.add(Dense(100, activation='relu', name='Hidden_Layer_3'))
   model5.add(Dropout(0.3))
   model5.add(Dense(200, activation='relu', name='Hidden_Layer_4'))
   model5.add(Dropout(0.3))
   model5.add(Dense(100, activation='relu', name='Hidden_Layer_5'))
   model5.add(Dense(num__of_digits, activation='softmax', name='Output_Layer'))

   # Input Layer - 5 Hidden Layer - Output Layer
```

```
  # Defining learning rate
  learning_rate = 0.01

  # We use Adam optimizer for Model 5.
  adam = optimizers.Adam(lr=learning_rate)

  model5.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])

  return model5
  # Defining batch and epoch sizes
  batch_size = 32
  epochs = 25

  classifier = KerasClassifier(build_fn=create_model5, batch_size=batch_size, epochs=epochs)
  accuracies = cross_val_score(estimator=classifier, X=X_train, y=y_train, cv=10, verbose=2)
  #Cross Validation Scores
  Accuracies
  print('Max Cross-validation Accuracy  (10-Fold) : ',max(accuracies))
  print('Mean Cross-validation Accuracy (10-Fold) : ',accuracies.mean())
  model5.metrics_names
  y_test = dataset_labels_test

  #One-Hot Encoding with Keras
  y_test = keras.utils.to_categorical(y_test)

  model5.evaluate(X_test, y_test)
  predictions_test = model5.predict_classes(X_test)
  predictions_test
  predictions_test.shape
  print(classification_report(dataset_labels_test, predictions_test))
  from sklearn.metrics import confusion_matrix
  import seaborn as sns

  # Creating the confusion matrix
  conf_matrix = confusion_matrix(dataset_labels_test, predictions_test)
  f, ax = plt.subplots(figsize = (10,10))

  sns.heatmap(conf_matrix, annot=True, fmt='.0f', ax = ax, cmap='Blues')
  plt.xlabel('Predicted Labels')
  plt.ylabel('True Labels')
  plt.title('Confusion Matrix')
  plt.show();
  def classification_model(features):
    df = pd.DataFrame(data=features)
    df = df.iloc[:,:].values.reshape(1,df.shape[0])

    #Normalization
    df = df.astype('float32')
    df /= 16

    return print('Predicted Value : ', model5.predict_classes(df))
  new_sample_feature_values =
np.array([0,0,5,13,1,0,0,0,0,0,12,13,0,0,0,0,0,0,16,8,0,0,0,0,0,5,16,2,0,0,0,0,0,4,16,8,15,9,1,0,0,4,16,1
6,12,15,11,0,0,1,15,14,4,14,11,0,0,0,5,14,14,10,1,0])
```

```
classification_model(new_sample_feature_values)
# Plotting entering sample matrix form

digit = new_sample_feature_values.reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(digit, cmap='gray')
plt.title("Digit")
```

SVM Sınıflandırıcı

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

np.random.seed(42)

from sklearn.model_selection import train_test_split
from sklearn import svm, metrics
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
#Training Dataset
dataset_train = pd.read_csv('dataset/optdigits.tra', header=None)

dataset_feautes = dataset_train.iloc[:,:-1].values
dataset_labels = dataset_train.iloc[:,-1].values
#Training Dataset
dataset_train = pd.read_csv('dataset/optdigits.tra', header=None)

dataset_feautes = dataset_train.iloc[:,:-1].values
dataset_labels = dataset_train.iloc[:,-1].values
dataset_feautes.shape
# Plotting some samples matrix form

two = dataset_feautes[5,:].reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(two, cmap='gray')
plt.title("Digit 2")
# Plotting some samples matrix form

three = dataset_feautes[3000,:].reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(three, cmap='gray')
plt.title("Digit 3")
X_train, X_val, y_train, y_val = train_test_split(dataset_feautes, dataset_labels,
                                  test_size = 0.2,
                                  random_state = 42)
print(X_train.shape, X_val.shape)
X_test = dataset_feautes_test.reshape(1797, 64)
X_test.shape
X_train[1]
print((min(X_train[1]), max(X_train[1])))
# Feature Normalization
X_train = X_train.astype('float32')
```

```
X_val= X_val.astype('float32')
X_test = X_test.astype('float32')

X_train /= 16
X_val   /= 16
X_test  /= 16
X_train
X_val
X_test
classifier = svm.SVC(C=100, kernel='rbf', random_state=42)
classifier.fit(X_train, y_train)
predictions_val = classifier.predict(X_val)
predictions_val
print("Accuracy Score:",metrics.accuracy_score(y_val, predictions_val))
accuracies = cross_val_score(estimator=classifier, X=X_train, y=y_train, cv=10, verbose=1)
#Cross Validation Scores
Accuracies
print("KfoldCrossVal score using SVM is %s" %accuracies.mean())
y_pred = classifier.predict(X_test)
y_pred
y_pred.shape
print("Accuracy Score with Test Set:",metrics.accuracy_score(dataset_labels_test, y_pred))
print(classification_report(dataset_labels_test, y_pred))
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Creating the confusion matrix
conf_matrix = confusion_matrix(dataset_labels_test, y_pred)
f, ax = plt.subplots(figsize = (10,10))

sns.heatmap(conf_matrix, annot=True, fmt='.0f', ax = ax, cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show();
def classification_model_svm(features):
    df = pd.DataFrame(data=features)
    df = df.iloc[:,:].values.reshape(1,df.shape[0])

    #Normalization
    df = df.astype('float32')
    df /= 16

    return print('Predicted Value : ', classifier.predict(df))
new_sample_feature_values =
np.array([0,0,2,15,15,3,0,0,0,0,8,14,16,11,0,0,0,0,0,0,11,14,0,0,0,0,0,0,11,14,3,0,0,0,4,12,16,16,7,0,0,
0,11,16,12,1,0,0,0,0,1,14,6,0,0,0,0,0,4,12,1,0,0,0])
classification_model_svm(new_sample_feature_values)
# Plotting entering sample matrix form

digit = new_sample_feature_values.reshape(8,8)

plt.figure(figsize=(1, 1))
plt.imshow(digit, cmap='gray')
plt.title("Digit")
```

Özdeğerlendirme Tablosu (her bölümden kaç puan alabileceğinizi tahminleyiniz. Açıklama bölümünde, eksiklerinizi, hataları veya nelerin yapılabildiğini kısaca belirtiniz.)

	İstenen Özellik	**Var**	**Açıklama**	**Tahmini Not**
1a	Algoritmalar + Karmaşıklıklar (10)	☑	Zaman karmaşıklıklarI hesaplanmadı.	8
1b	Tanım ve Karşılaştırmalar (10)	☑	Yapıldı	10
1c	Araştırma ve Yorum (10)	☑	Yapıldı	10
2	Problem Çözme ve Kodlama (10)	☑	Yapıldı	10
3	Genetik Algoritmalar ile Şifre Kırma (15)	☑	Yapıldı	15
4	Makine Öğrenmesi (25)	☑	Yapıldı	25
	Rapor (20)	☑	Yapıldı	20
			100 üzerinden Toplam Not:	98

1. Soru -> Enes Yaşarbaş 4 Saat
2. Soru -> Güneş Hacıhalil 4 Saat
3. Soru -> Kemal Orhan 4 Saat
4. Soru -> Herkes 1 Hafta (Online Toplantı)