

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**Artificial Intelligence Methods (3+0)**

**2024–2025 SPRING SEMESTER**

**PROJECT-1 REPORT**

**DELIVERY DATE**

--/--/---

**PREPARED BY**

Student\_number, Name Surname

İçindekiler

[1) Algoritmalar, Tanımlar, Karşılaştırma, Araştırma ve Yorum 3](#_Toc194926557)

[1.a Tabu Search ve İki veya Üç Boyutlu Bilgisayar Oyunu 3](#_Toc194926558)

[1.a.i Tabu Search ve A\* Algoritmalarının kağıda yazılmış görüntüsü(Tarama/Fotoğraf) 3](#_Toc194926559)

[1.a.ii Tabu Search algoritmasının anlatımı 3](#_Toc194926560)

[1.a.iii İki algoritmanın farkları 3](#_Toc194926561)

[1.a.iv Zaman karmaşıklıkları 3](#_Toc194926562)

[1.b İki veya Üç Boyutlu Bilgisayar Oyunu ve GenAI 3](#_Toc194926563)

[1.b.i İki veya Üç Boyutlu Bilgisayar Oyunu Kodunu GenAI aracına yazdırtma 3](#_Toc194926564)

[1.b.ii Kodun çalıştırılması ve test ekran görüntüleri 3](#_Toc194926565)

[1.b.iii Yeni özellikler eklenmesi için yazdığınız komutlar ve eklenen özellikler 4](#_Toc194926566)

[1.b.iv Gen AI kullanımının yarar & zararlarının açıklanması 4](#_Toc194926567)

[1.c Tanım ve Karşılaştırmalar 4](#_Toc194926568)

[1.c.i Self-attention GAN 4](#_Toc194926569)

[1.c.ii LLM (Large Language Model) 4](#_Toc194926570)

[1.c.iii Reasoning 4](#_Toc194926571)

[1.c.iv Weighted voting based ensemble learning 4](#_Toc194926572)

[1.c.v Deep Seek vs. Deep Seek DeepThink 4](#_Toc194926573)

[1.d Makine Öğrenmesi Mülakat Soruları 4](#_Toc194926574)

[1.d.i Makine Öğrenmesi alanında çalışan bir firmaya girişte sorulabilecek mülakat soruları ve cevapları (Tarama/Fotoğraf) 4](#_Toc194926575)

[1.d.ii ChatGPT’ye akıl yürüt modunda yazılan soru ve cevabı yazınız ve diğer araç ile cevaplarını karşılaştırınız. 4](#_Toc194926576)

[2) Problem Çözme ve Kodlama 4](#_Toc194926577)

[2.a Sekiz Vezir Probleminin First-Choice Hill Climbing ile Çözülmesi / Bridge and torch probleminin A\* ile çözülmesi 4](#_Toc194926578)

[2.a.i Kodlar ve Çıktılar 4](#_Toc194926579)

[2.a.ii First-Choice Hill Climbing nedir? Hill Climbing kaynak kodunun hangi kısmında nasıl bir değişiklik yapılması gerektiğini anlatınız. 5](#_Toc194926580)

[3) Genetik Algoritmalar ile Şifre Kırma 5](#_Toc194926581)

[3.a Kodlar 5](#_Toc194926582)

[3.b Koddaki çaprazlama, mutasyon ve seçilim fonksiyonlarının açıklanması 5](#_Toc194926583)

[3.c Farklı kelime için çözüm sürelerinin karşılaştırılması 5](#_Toc194926584)

[4)Python ile Makine Öğrenmesi [Sınıflandırıcı] 5](#_Toc194926585)

[4.a Python Standardization örneği 5](#_Toc194926586)

[4.a.i Kodlar 5](#_Toc194926587)

[4.a.ii Normalizasyondan Farkı 5](#_Toc194926588)

[4.b Veri seti bulma veya oluşturma 5](#_Toc194926589)

[4.b.i Veri setine ait bilgiler 5](#_Toc194926590)

[4.c İki farklı sınıflandırıcı ve sonuçlar 6](#_Toc194926591)

[4.c.i Kaynak Kod 6](#_Toc194926592)

[4.c.ii Veriseti Özet Bilgisi 6](#_Toc194926593)

[4.d Sınıfı bilinmeyen bir örneğin sınıfını tahmin ettirme 6](#_Toc194926594)

[4.d.i Kaynak Kod 6](#_Toc194926595)

[4.d.ii Confusion Matrix ve Ekran Görüntüleri 6](#_Toc194926596)

[4.e Özniteliklerin Hangi Sınıfa Ait Olduğunu Bulma 6](#_Toc194926597)

[4.f 5-Fold Cross Validation 6](#_Toc194926598)

[4.f.i Kaynak Kod 6](#_Toc194926599)

[4.f.ii 5-Fold Cross Validation ile Elde Edilen Başarımlar 6](#_Toc194926600)

[5) Harcanan Süreler ve İş Bölümü Detayları 6](#_Toc194926601)

[6) Öz değerlendirme Tablosu 6](#_Toc194926602)

# 1) Algoritmalar, Tanımlar, Karşılaştırma, Araştırma ve Yorum

## 1.a Tabu Search ve İki veya Üç Boyutlu Bilgisayar Oyunu

### 1.a.i Tabu Search ve A\* Algoritmalarının kağıda yazılmış görüntüsü(Tarama/Fotoğraf)

//Görüntü

### 1.a.ii Tabu Search algoritmasının anlatımı

i) Tabu Search, yönlendirilmiş bir arama algoritmasıdır. Yerel aramalarda takılıp kalmamak için geçmişte ziyaret edilen çözümleri "tabu listesi" (yasaklı liste) adında bir hafızada tutar ve kısa süreliğine bunlara dönüşü yasak getirir.  
Böylece, algoritma lokal minimum tuzaklarına düşmez ve daha geniş bir çözüm alanını keşfeder.

Başlıca adımları şöyle özetleyebiliriz:

1-Başlangıç çözümü oluşturulur.

2-Komşu çözümler üretilir.

3-En iyi komşu seçilir, tabu listesine eklenir.

4-Tabu listesi dolunca eski kayıtlar silinir (FIFO mantığı gibi).

5-İyileşme olana kadar devam edilir.

### 1.a.iii İki algoritmanın farkları

| **Özellik** | **Tabu Search** | **A\* Algoritması** |
| --- | --- | --- |
| Kullanım Alanı | Genel optimizasyon problemleri | Yol bulma ve minimum maliyet bulma |
| Arama Stratejisi | Geçmişe yasak koyarak arama yapar | En düşük toplam maliyeti önceliklendirir |
| Garantili Çözüm | Hayır (yaklaşık çözüm bulabilir) | Evet (doğru heuristic ile optimal çözüm bulur) |
| Bellek Kullanımı | Tabu listesine bağlı | Açık ve kapalı listelere bağlı |
| Hedef | En iyi çözümü bulmak | En kısa/ucuz yolu bulmak |

### -> Tabu Search daha çok büyük ve karmaşık problemlerde kullanılırken, A daha sistematik ve güvenli bir arama yöntemidir.

### 1.a.iv Zaman karmaşıklıkları

*Tabu Search:*  
-Genellikle **O(k \* n)** civarındadır. (k: iterasyon sayısı, n: komşu sayısı.)  
-Net bir garanti yoktur çünkü probleme ve parametrelere bağlıdır.

*A\* Algoritması:*

-En kötü durumda **O(b^d)** (b: dallanma faktörü, d: çözümün derinliği)  
-Eğer iyi bir heuristic kullanılırsa daha hızlı çalışır.

## 1.b İki veya Üç Boyutlu Bilgisayar Oyunu ve GenAI

### 1.b.i İki veya Üç Boyutlu Bilgisayar Oyunu Kodunu GenAI aracına yazdırtma

|  |
| --- |
| ChatGPT’ye şu komutu verdim: ‘Python kullanarak basit bir 2D yılan oyunu oluştur’  Yanıt:  import pygame  import time  import random  # Pygame başlat  pygame.init()  # Renkler  white = (255, 255, 255)  black = (0, 0, 0)  red = (213, 50, 80)  green = (0, 255, 0)  # Oyun ekranı  width, height = 600, 400  screen = pygame.display.set\_mode((width, height))  pygame.display.set\_caption('Yılan Oyunu')  clock = pygame.time.Clock()  snake\_block = 10  snake\_speed = 15  font = pygame.font.SysFont("bahnschrift", 20)  def draw\_snake(snake\_list):      for x in snake\_list:          pygame.draw.rect(screen, green, [x[0], x[1], snake\_block, snake\_block])  def game\_loop():      game\_over = False      game\_close = False      x = width / 2      y = height / 2      dx = 0      dy = 0      snake\_list = []      length = 1      food\_x = round(random.randrange(0, width - snake\_block) / 10.0) \* 10.0      food\_y = round(random.randrange(0, height - snake\_block) / 10.0) \* 10.0      while not game\_over:          while game\_close:              screen.fill(black)              msg = font.render("Kaybettin! Q ile çık, C ile tekrar oyna", True, red)              screen.blit(msg, [width / 6, height / 3])              pygame.display.update()              for event in pygame.event.get():                  if event.type == pygame.KEYDOWN:                      if event.key == pygame.K\_q:                          game\_over = True                          game\_close = False                      elif event.key == pygame.K\_c:                          game\_loop()          for event in pygame.event.get():              if event.type == pygame.QUIT:                  game\_over = True              elif event.type == pygame.KEYDOWN:                  if event.key == pygame.K\_LEFT:                      dx = -snake\_block                      dy = 0                  elif event.key == pygame.K\_RIGHT:                      dx = snake\_block                      dy = 0                  elif event.key == pygame.K\_UP:                      dx = 0                      dy = -snake\_block                  elif event.key == pygame.K\_DOWN:                      dx = 0                      dy = snake\_block          if x >= width or x < 0 or y >= height or y < 0:              game\_close = True          x += dx          y += dy          screen.fill(black)          pygame.draw.rect(screen, red, [food\_x, food\_y, snake\_block, snake\_block])          snake\_head = [x, y]          snake\_list.append(snake\_head)          if len(snake\_list) > length:              del snake\_list[0]          for segment in snake\_list[:-1]:              if segment == snake\_head:                  game\_close = True          draw\_snake(snake\_list)          pygame.display.update()          if x == food\_x and y == food\_y:              food\_x = round(random.randrange(0, width - snake\_block) / 10.0) \* 10.0              food\_y = round(random.randrange(0, height - snake\_block) / 10.0) \* 10.0              length += 1          clock.tick(snake\_speed)      pygame.quit()      quit()  game\_loop() |
| Gayet başarılı bir şekilde yılan oyunu oluşturdu. Spesifik olarak söylemememe rağmen yem alma ve yılanın büyümesi gibi mekanikleri ekledi. Aynı zamanda oyun bitince tekrar başlatmak için kullanıcıya seçim veriyor. |

### 1.b.ii Kodun çalıştırılması ve test ekran görüntüleri

### ekran görüntüsü, yazılım, bilgisayar, multimedya yazılımı içeren bir resim Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, yazılım, bilgisayar, multimedya içeren bir resim Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

### 1.b.iii Yeni özellikler eklenmesi için yazdığınız komutlar ve eklenen özellikler

1-) Bu oyuna yılanın yediği her yem için bir skor artacak şekilde, ekranın sağ üstünde görünecek bir skorboard ekle.

2-)Şu an yılan ekranın sınırlarından çıktığında oyun bitiyor. Bunu değiştirelim ve ekrandan çıkıldığında tam tersindeki kenardan çıkılsın. Örneğin yılan ekranın en sağından dışarı çıktığında en solundan ve aynı satırdan çıksın ve oyun devam etsin.

Yanıtta bu dediğimiz geliştirmeleri ekledi ve tespit edilen bir sıkıntı çıkmadı.  
ekran görüntüsü, metin, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, yazılım, bilgisayar, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

### 1.b.iv Gen AI kullanımının yarar & zararlarının açıklanması

Gen AI kullanımı işimizi aşırı derece kolaylaştırır. Basit bir tabirle işin “hamallığını” bu şekilde kolayca halledebiliriz. 5 kişinin birleşip yazacağı kodu 1 kişinin yazabilmesine olanak sağlar. Zararlarına gelecek olursak, bilinçsizce ve istenen kodun genel mantığını anlamayan kullanıcılar bolca hata yaparlar ve bunu toparlayamazlar. Ayrıca yazılımcıların biraz tembelleşmesine olanak sağlar. Kısaca projeye hakim olmalı ve daha küçük parçaları yazarken yardım almalıyız.

## 1.c Tanım ve Karşılaştırmalar

### 1.c.i Self-attention GAN

Self-Attention GAN (SAGAN), GAN'lara (Generative Adversarial Network) "self-attention" (kendi dikkati) mekanizması ekleyerek çalışır.  
Klasik GAN'larda üretilen görüntüler daha çok lokal ilişkilere dayanır. Self-Attention modülü, her bir pikselin görüntünün uzak bölgelerindeki piksellerle ilişkisini dikkate alır. Bu sayede, daha tutarlı ve global anlamda uyumlu görüntüler üretilir. SAGAN, detaylar arasında uzun mesafeli bağıntıları öğrenir ve daha kaliteli sonuçlar üretir.  
(Kaynak: *Zhang et al., 2019, "Self-Attention Generative Adversarial Networks"*.)

### 1.c.ii LLM (Large Language Model)

Large Language Model (LLM), milyarlarca parametre içeren devasa yapay zeka dil modelleridir.  
Büyük veri kümeleri üzerinde eğitilerek, insan benzeri metinler üretir, soruları yanıtlar ve farklı görevlerde kullanılır. Örnekler: ChatGPT, GPT-4, Gemini, Claude, Mistral gibi modellerdir. LLM'ler, çok büyük dil bilgisi ve dünya bilgisi taşıyan, çok yönlü yapay zeka modelleridir.  
(Kaynak: *OpenAI, 2023, "GPT-4 Technical Report"*)

### 1.c.iii Reasoning

Reasoning, bir yapay zekanın var olan bilgilerden yeni sonuçlar çıkarabilme yeteneğidir.  
Örneğin, "Tüm insanlar ölümlüdür. Ali bir insandır. → Ali ölümlüdür." şeklinde akıl yürütme yapabilir. Reasoning, bilgiyi işleyip mantıklı sonuçlar çıkarma sürecidir.  
(Kaynak: *Russell & Norvig, 2021, "Artificial Intelligence: A Modern Approach"*.)

Reasoning ikiye ayrılır:

-Deductive Reasoning: Genel bilgiden özel sonuca gitme.

-Inductive Reasoning: Özel durumdan genelleme yapma.

### 1.c.iv Weighted voting based ensemble learning

Weighted Voting Based Ensemble Learning, birden fazla modelin oylarının, eşit ağırlıklı değil, belirli ağırlıklarla değerlendirilmesi anlamına gelir. Daha iyi modellerin görüşü daha güçlü şekilde hesaba katılır.  
(Kaynak: *Zhou, 2012, "Ensemble Methods: Foundations and Algorithms"*.)  
Her modelin güvenilirliğine göre farklı ağırlıklar atanır.

### 1.c.v Deep Seek vs. Deep Seek DeepThink

"Deep Seek, daha çok üretim odaklı bir LLM iken; Deep Seek DeepThink, üretimin ötesine geçerek akıl yürütmeye ve çözüm üretmeye odaklanmıştır. DeepThink, daha zor problemleri çözmede daha üstündür."

(Kaynaklar: *DeepSeek Developers Documentation, 2024*)

## 1.d Makine Öğrenmesi Mülakat Soruları

### 1.d.i Makine Öğrenmesi alanında çalışan bir firmaya girişte sorulabilecek mülakat soruları ve cevapları (Tarama/Fotoğraf)

//Görüntü

### 1.d.ii ChatGPT’ye akıl yürüt modunda yazılan soru ve cevabı yazınız ve diğer araç ile cevaplarını karşılaştırınız.

# i)1-)Overfitting (Aşırı Uydurma) Nedir ve Nasıl Önlenir?

ChatGBT’nin cevabı : Modelin amacı yeni verilerde başarılı olmaktır. Eğer eğitim verisindeki tüm detayları ve hataları ezberlerse, yeni verilerde başarısız olur; buna overfitting denir.  
Bunu önlemek için ya modeli daha sade tutarız, ya daha fazla veri kullanırız, ya da modelin karmaşıklığını cezalandıran yöntemler (regularization gibi) uygularız. Çünkü amaç, veriyi ezberlemek değil, genel bir desen öğrenmektir.

(Deepseek ile karşılaştırdığımda bu cevaba 90/100 veriyorum. Doğruluk ve netlik olarak çok iyi fakar çok küçük eksiklikleri var ( birkaç yaygın yöntemden bahsetmemiş.)

2-)Bias-Variance Trade-off (Sapma-Varyans Dengesi) Nedir?

Model ne kadar basitse, o kadar az varyans ama fazla sapma yapar; ne kadar karmaşıksa, sapması azalır ama varyansı artar. Amaç, modelin ne çok ezber yapmasını (yüksek varyans) ne de gerçeği ıskalamasını (yüksek sapma) istemediğimiz için ikisi arasında bir denge kurmaktır.

( Bu cevabı deepseek ile karşılaştırdığımda 95/100 verdim. Biraz daha örnekleri artırabilmesi haricinde hiçbir kusur yok.)

# 2) Problem Çözme ve Kodlama

## 2.a Sekiz Vezir Probleminin First-Choice Hill Climbing ile Çözülmesi

### 2.a.i Kodlar ve Çıktılar

|  |
| --- |
| import random  import time  def generate\_board():      return [random.randint(0, 7) for \_ in range(8)]  def print\_board(board):      for row in range(8):          line = ""          for col in range(8):              line += "Q " if board[row] == col else ". "          print(line)      print("-" \* 16)  def calculate\_cost(board):      cost = 0      for i in range(len(board)):          for j in range(i + 1, len(board)):              if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j):                  cost += 1      return cost  def hill\_climbing\_with\_random\_restart():      random\_restart\_count = 0      move\_count = 0      start\_time = time.time()      board = generate\_board()      current\_cost = calculate\_cost(board)      while current\_cost != 0:          neighbors = []          for row in range(8):              for col in range(8):                  if board[row] != col:                      new\_board = list(board)                      new\_board[row] = col                      neighbors.append((new\_board, calculate\_cost(new\_board)))          neighbors.sort(key=lambda x: x[1])          best\_board, best\_cost = neighbors[0]          if best\_cost >= current\_cost:              board = generate\_board()              current\_cost = calculate\_cost(board)              random\_restart\_count += 1          else:              board = best\_board              current\_cost = best\_cost              move\_count += 1      end\_time = time.time()      elapsed\_time = end\_time - start\_time      return board, move\_count, random\_restart\_count, elapsed\_time  # Ana program  if \_\_name\_\_ == "\_\_main\_\_":      print("Yer değiştirme sayısı, random restart sayısı ve işlem sürelerini içeren tablo")      print(f"{'Çalıştırma No':<15}{'Yer değiştirme':<20}{'Random Restart':<20}{'İşlem Süresi (s)':<20}")      print("-" \* 75)      for i in range(1, 21):          board, moves, restarts, duration = hill\_climbing\_with\_random\_restart()          print\_board(board)          print(f"{i:<15}{moves:<20}{restarts:<20}{duration:.4f}") |

metin, ekran görüntüsü, astronomi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, metin, siyah içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, siyah, astronomi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, siyah içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.ekran görüntüsü, metin, siyah içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

#### Yer değiştirme sayısı, random restart sayısı ve işlem sürelerini içeren tablo

|  |  |  |  |
| --- | --- | --- | --- |
| **Çalıştırma No** | **Yer değiştirme sayısı** | **Random restart sayısı** | **İşlem Süresi** |
| 1 | 15 | 4 | 0.0050 |
| 2 | 7 | 2 | 0.0040 |
| 3 | 11 | 2 | 0.0045 |
| 4 | 8 | 2 | 0.0045 |
| 5 | 3 | 0 | 0.0013 |
| 6 | 7 | 1 | 0.0020 |
| 7 | 9 | 1 | 0.0020 |
| 8 | 32 | 8 | 0.0129 |
| 9 | 28 | 10 | 0.0100 |
| 10 | 19 | 4 | 0.0082 |
| 11 | 15 | 4 | 0.0130 |
| 12 | 6 | 1 | 0.0040 |
| 13 | 10 | 2 | 0.0100 |
| 14 | 32 | 7 | 0.0156 |
| 15 | 26 | 8 | 0.0101 |
| 16 | 33 | 10 | 0.0133 |
| 17 | 46 | 12 | 0.0162 |
| 18 | 5 | 0 | 0.0020 |
| 19 | 3 | 0 | 0.0010 |
| 20 | 4 | 0 | 0.0010 |

### 2.a.ii First-Choice Hill Climbing nedir? Hill Climbing kaynak kodunun hangi kısmında nasıl bir değişiklik yapılması gerektiğini anlatınız.

First-Choice Hill Climbing, klasik hill climbing'in daha hızlı ama biraz daha rastgele çalışan bir versiyonudur. Klasik hill climbing’de tüm komşular (yani tüm olası hamleler) denenir ve en iyisi seçilir.

Ama First-Choice Hill Climbing şunu yapar:

1-Komşuları tek tek dener,

2-Daha iyi bir komşu bulduğu anda hemen oraya geçer ve diğerlerini kontrol etmez.

neighbors.sort(key=lambda x: x[1])

   best\_board, best\_cost = neighbors[0]

-Kodumuzu çevirmek için yukarıdaki iki satır yerine aşağıdaki kodu yapıştırabiliriz :

improved = False

for row in range(8):

    for col in range(8):

        if board[row] != col:

            new\_board = list(board)

            new\_board[row] = col

            new\_cost = calculate\_cost(new\_board)

            if new\_cost < current\_cost:

                board = new\_board

                current\_cost = new\_cost

                move\_count += 1

                improved = True

                break  # En yakın iyileştirmeyi bulduk → diğerlerini geç

    if improved:

        break

# 3) Genetik Algoritmalar ile Şifre Kırma

## 3.a Kodlar

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz |

### 3.b Koddaki çaprazlama, mutasyon ve seçilim fonksiyonlarının açıklanması

//Açıklamalar

### 3.c Farklı kelime için çözüm sürelerinin karşılaştırılması

//Açıklamalar

# 4)Python ile Makine Öğrenmesi [Sınıflandırıcı]

## 4.a Python Standardization örneği

### 4.a.i Kodlar

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz |

### 4.a.ii Normalizasyondan Farkı

//Açıklama

## 4.b Veri seti bulma veya oluşturma

### 4.b.i Veri setine ait bilgiler

//Açıklama

## 4.c İki farklı sınıflandırıcı ve sonuçlar

### 4.c.i Kaynak Kod

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz |

### 4.c.ii Veriseti Özet Bilgisi

// Açıklama

## 4.d Sınıfı bilinmeyen bir örneğin sınıfını tahmin ettirme

### 4.d.i Kaynak Kod

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz |

### 4.d.ii Confusion Matrix ve Ekran Görüntüleri

//Ekran görüntüleri

## 4.e Özniteliklerin Hangi Sınıfa Ait Olduğunu Bulma

//Açıklama ve Ekran Görüntüsü

## 4.f 5-Fold Cross Validation

### 4.f.i Kaynak Kod

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz |

### 4.f.ii 5-Fold Cross Validation ile Elde Edilen Başarımlar

//Tablo

# 5) Harcanan Süreler ve İş Bölümü Detayları

//Açıklama

# 6) Öz değerlendirme Tablosu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **İstenen Özellik** | **Var** | **Açıklama** | **Tahmini Not** |
| **1a** | Algoritmalar + Karmaşıklıklar (10) |  |  |  |
| **1b** | Gen AI ile Bilgisayar Oyunu Geliştirme (10) |  |  |  |
| **1c** | Tanım ve Karşılaştırma (10) |  |  |  |
| **1d** | Araştırma ve Yorum (10) |  |  |  |
| **2** | Problem Çözme ve Kodlama (10) |  |  |  |
| **3** | Genetik Algoritmalar ile Şifre Kırma (15) |  |  |  |
| **4** | Makine Öğrenmesi (15) |  |  |  |
| **5** | İşbölümü Detayları (5) |  |  |  |
| **6** | Özd. Tablosu + Rapor (15) |  |  |  |
| **100 üzerinden Toplam Not:** | | | |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**

**Not: Raporu ve kodları sisteme yüklenmeyen proje değerlendirilmemektedir.**