

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2023–2024 SPRING SEMESTER**

**PROJECT-1 REPORT**

**DELIVERY DATE**

15/04/2024

**PREPARED BY**

05210000195, Muhammed Said Golotooğlu

İçindekiler

[1) Algoritmalar, Tanımlar, Karşılaştırma, Araştırma ve Yorum 5](#_Toc164032146)

[1.a Tabu Search ve A\* Algoritmaları 5](#_Toc164032147)

[1.a.i Tabu Search ve A\* Algoritmalarının kağıda yazılmış görüntüsü(Tarama/Fotoğraf) 5](#_Toc164032148)

[1.a.ii Tabu Search algoritmasının anlatımı 6](#_Toc164032149)

[6](#_Toc164032150)

[1.a.iii İki algoritmanın farkları 6](#_Toc164032151)

[6](#_Toc164032152)

[1.a.iv Zaman karmaşıklıkları 6](#_Toc164032153)

[1.b A\* ve GenAI 6](#_Toc164032154)

[1.b.i A\* Kodunu GenAI aracına yazdırtma 6](#_Toc164032155)

[1.b.ii Kodun çalıştırılması ve test ekran görüntüleri 8](#_Toc164032156)

[8](#_Toc164032157)

[1.b.iii GenAI aracına kodun detayları ile alakalı sorulan sorular ve cevapları 8](#_Toc164032158)

[1.b.iv Aracın avantajları, katkıları ve başarısız olduğu yönler 11](#_Toc164032159)

[1.c Tanım ve Karşılaştırmalar 11](#_Toc164032160)

[1.c.i Ensemble Learning 11](#_Toc164032161)

[1.c.ii LLM (Large Language Model) 11](#_Toc164032162)

[1.c.iii Devin 11](#_Toc164032163)

[1.c.iv CycleGAN 12](#_Toc164032164)

[1.c.v GPT-4 vs Gemini vs Claude 3 12](#_Toc164032165)

[1.d Makine Öğrenmesi Mülakat Soruları 12](#_Toc164032166)

[1.d.i Makine Öğrenmesi alanında çalışan bir firmaya girişte sorulabilecek mülakat soruları ve cevapları (Tarama/Fotoğraf) 12](#_Toc164032167)

[1.d.ii Aynı soruların ChatGPT ve Gemini araçlarına sorulması ve cevaplarının notlandırılması 13](#_Toc164032168)

[2) Problem Çözme ve Kodlama 16](#_Toc164032169)

[2.a Sekiz Vezir Probleminin Hill Climbing ile Çözülmesi / Bridge and torch probleminin A\* ile çözülmesi 16](#_Toc164032170)

[2.a.i Kodlar ve Çıktılar 16](#_Toc164032171)

[2.a.ii Stochastic Hill Climbing nedir? Hill Climbing kaynak kodunun hangi kısmında nasıl bir değişiklik yapılması gerektiğini anlatınız. 19](#_Toc164032172)

[3) Genetik Algoritmalar ile Şifre Kırma 20](#_Toc164032173)

[3.a Kodlar 20](#_Toc164032174)

[26](#_Toc164032175)

[26](#_Toc164032176)

[3.b Koddaki çaprazlama, mutasyon ve seçilim fonksiyonlarının açıklanması 26](#_Toc164032177)

[3.c Farklı kelime için çözüm sürelerinin karşılaştırılması 27](#_Toc164032178)

[4)Python ile Makine Öğrenmesi [Sınıflandırıcı] 27](#_Toc164032179)

[4.a Python Standardization örneği 27](#_Toc164032180)

[4.a.i Kodlar 27](#_Toc164032181)

[4.a.ii Normalizasyondan Farklı 28](#_Toc164032182)

[4.b Veri seti bulma veya oluşturma 28](#_Toc164032183)

[4.b.i Veri setine ait bilgiler 28](#_Toc164032184)

[4.c İki farklı sınıflandırıcı ve sonuçlar 28](#_Toc164032185)

[4.c.i Kaynak Kod 28](#_Toc164032186)

[4.c.ii Confusion Matrix 30](#_Toc164032187)

[4.d Sınıfı bilinmeyen bir örneğin sınıfını tahmin ettirme 31](#_Toc164032188)

[4.d.i Kaynak Kod 31](#_Toc164032189)

[4.d.ii Ekran Görüntüleri 31](#_Toc164032190)

[4.f 10-Fold Cross Validation 31](#_Toc164032191)

[4.f.i Kaynak Kod 31](#_Toc164032192)

[4.f.ii 10-Fold Cross Validation ile Elde Edilen Başarımlar 32](#_Toc164032193)

[5) Harcanan Süreler ve İş Bölümü Detayları 32](#_Toc164032194)

[6) Öz değerlendirme Tablosu 32](#_Toc164032195)

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

## 1.a Tabu Search ve A\* Algoritmaları

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

metin, el yazısı, doküman, belge, kağıt içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, el yazısı, doküman, belge, kağıt içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

### metin, el yazısı, kitap, defter içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.a.iii İki algoritmanın farkları

### metin, el yazısı, kağıt, mürekkep içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.a.iv Zaman karmaşıklıkları

Yukarıdaki görselde var.

## 1.b A\* ve GenAI

### 1.b.i A\* Kodunu GenAI aracına yazdırtma

|  |
| --- |
| Promt: A\* algoritmasının kodunu yaz python'da  Answer:  import heapq  class Node:  def \_\_init\_\_(self, position, parent=None):  self.position = position  self.parent = parent  self.g = 0  self.h = 0  self.f = 0   def \_\_eq\_\_(self, other):  return self.position == other.position  def astar(start, end, grid):  open\_list = []  closed\_list = []   heapq.heappush(open\_list, (0, start))   while open\_list:  current\_node = heapq.heappop(open\_list)[1]  closed\_list.append(current\_node)   if current\_node == end:  path = []  while current\_node:  path.append(current\_node.position)  current\_node = current\_node.parent  return path[::-1]   children = []  for new\_position in [(0, -1), (0, 1), (-1, 0), (1, 0)]:  node\_position = (current\_node.position[0] + new\_position[0], current\_node.position[1] + new\_position[1])   if node\_position[0] > (len(grid) - 1) or node\_position[0] < 0 or node\_position[1] > (len(grid[len(grid)-1]) -1) or node\_position[1] < 0:  continue   if grid[node\_position[0]][node\_position[1]] == '#':  continue   new\_node = Node(node\_position, current\_node)   children.append(new\_node)   for child in children:  if child in closed\_list:  continue   child.g = current\_node.g + 1  child.h = ((child.position[0] - end.position[0]) \*\* 2) + ((child.position[1] - end.position[1]) \*\* 2)  child.f = child.g + child.h   for open\_node in open\_list:  if child == open\_node[1] and child.g > open\_node[1].g:  break  else:  heapq.heappush(open\_list, (child.f, child))   return None  def print\_path(path):  for position in path:  print(position)  def main():  grid = [  ['.', '.', '.', '.', '.', '.'],  ['.', '#', '#', '#', '#', '.'],  ['.', '.', '.', '.', '.', '.'],  ['.', '#', '#', '#', '#', '.'],  ['.', '.', '.', '.', '.', '.']  ]   start = Node((0, 0))  end = Node((4, 5))   path = astar(start, end, grid)  if path:  print("Path found:")  print\_path(path)  else:  print("No path found.")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

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

### metin, ekran görüntüsü, grafik yazılımı, multimedya yazılımı içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.b.iii GenAI aracına kodun detayları ile alakalı sorulan sorular ve cevapları

Prompt: Bulunan en kısa yolu bir graph şeklinde ifade edebilir miyiz? Edebiliyorsak grafiği döndüren kodu yaz.

import heapq  
import networkx as nx  
import matplotlib.pyplot as plt  
  
  
def path\_to\_graph(path):  
 graph = nx.Graph()  
  
 # Noktaları ekleyin  
 for i, node in enumerate(path):  
 graph.add\_node(i, pos=node)  
  
 # Kenarları ekleyin  
 for i in range(len(path) - 1):  
 graph.add\_edge(i, i + 1)  
  
 return graph  
  
  
def plot\_graph(graph):  
 pos = nx.get\_node\_attributes(graph, 'pos')  
  
 plt.figure(figsize=(5, 5))  
 nx.draw(graph, pos, with\_labels=True, node\_size=700, node\_color='lightblue', font\_size=10, font\_weight='bold',  
 edge\_color='gray', width=2)  
 plt.title("Shortest Path")  
 plt.show()  
  
  
class Node:  
 def \_\_init\_\_(self, position, parent=None):  
 self.position = position  
 self.parent = parent  
 self.g = 0  
 self.h = 0  
 self.f = 0  
  
 def \_\_eq\_\_(self, other):  
 return self.position == other.position  
  
  
def astar(start, end, grid):  
 open\_list = []  
 closed\_list = []  
  
 heapq.heappush(open\_list, (0, start))  
  
 while open\_list:  
 current\_node = heapq.heappop(open\_list)[1]  
 closed\_list.append(current\_node)  
  
 if current\_node == end:  
 path = []  
 while current\_node:  
 path.append(current\_node.position)  
 current\_node = current\_node.parent  
 return path[::-1]  
  
 children = []  
 for new\_position in [(0, -1), (0, 1), (-1, 0), (1, 0)]:  
 node\_position = (current\_node.position[0] + new\_position[0], current\_node.position[1] + new\_position[1])  
  
 if node\_position[0] > (len(grid) - 1) or node\_position[0] < 0 or node\_position[1] > (  
 len(grid[len(grid) - 1]) - 1) or node\_position[1] < 0:  
 continue  
  
 if grid[node\_position[0]][node\_position[1]] == '#':  
 continue  
  
 new\_node = Node(node\_position, current\_node)  
  
 children.append(new\_node)  
  
 for child in children:  
 if child in closed\_list:  
 continue  
  
 child.g = current\_node.g + 1  
 child.h = ((child.position[0] - end.position[0]) \*\* 2) + ((child.position[1] - end.position[1]) \*\* 2)  
 child.f = child.g + child.h  
  
 for open\_node in open\_list:  
 if child == open\_node[1] and child.g > open\_node[1].g:  
 break  
 else:  
 heapq.heappush(open\_list, (child.f, child))  
  
 return None  
  
  
def print\_path(path):  
 for position in path:  
 print(position)  
  
  
def main():  
 grid = [  
 ['.', '.', '.', '.', '.', '.'],  
 ['.', '#', '#', '#', '#', '.'],  
 ['.', '.', '.', '.', '.', '.'],  
 ['.', '#', '#', '#', '#', '.'],  
 ['.', '.', '.', '.', '.', '.']  
 ]  
  
 start = Node((0, 0))  
 end = Node((4, 5))  
  
 path = astar(start, end, grid)  
 plot\_graph(path\_to\_graph(path))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

ekran görüntüsü, metin, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.b.iv Aracın avantajları, katkıları ve başarısız olduğu yönler

Katkıları: Bir algoritmayı öğrenmek istediğimde YouTube, Web, www.geeksforgeeks.com, [www.wikipedia.com](http://www.wikipedia.com) vb. gibi yerlerden araştırma yapmak fazla zaman kaybına sebep oluyor. Artık şöyle çalışıyorum AI ile Python’da kodu oluşturuyorum oradaki iskeletten algoritmanın yapısını ve mantığını anlıyorum. Daha sonra ise başta bahsettiğim kaynakları kullanıyorum. Böylece verimim çok fazla artıyor.

Avantajları: Çok hızlı bir şekilde istediğim algoritmanın kodunu yazabiliyorum, Google’da çok fazla bilgi kirliliği olduğundan Generative AI’ları artık Google olarak kullanıyorum.

Dezavantajları: Çoğu bir dil modeli olduğundan prompt’a verilebilecek en olasılıklı kelimelerden cevapları seçiyorlar bu da aslında AI’ın beni anlamadığını ezberlediği cevabı söylediğini gösteriyor. Bu yüzden bazen yanlış cevap verebiliyorlar.

## 1.c Tanım ve Karşılaştırmalar

### 1.c.i Ensemble Learning

Ensemble Learning, birden fazla makine öğrenimi modelinin veya algoritmasının bir araya getirilerek kullanılmasıyla gerçekleştirilen bir makine öğrenimi tekniğidir. **Temel fikir, birçok zayıf öğrenicinin (genellikle temel modeller olarak adlandırılır) bir araya gelerek daha güçlü bir öğrenici oluşturmasıdır.** *Ensemble Learning'de kullanılan temel teknikler arasında Bagging, Boosting ve Stacking bulunur.* Örnek olarak, Random Forest algoritması, birden fazla karar ağacının bir araya gelerek daha güçlü bir sınıflandırıcı oluşturduğu bir Ensemble Learning örneğidir.

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

Büyük Dil Modeli, genellikle büyük bir metin veri kümesi üzerinde eğitilen ve dilin yapılarını, kalıplarını ve bağlamsal ilişkilerini öğrenen bir yapay zekâ modelidir. LLM'ler, genellikle milyarlarca parametreye sahip olan ve geniş bir dil anlama yeteneği kazanan derin öğrenme modelleridir. **Bu modeller, çeşitli doğal dil işleme görevlerini gerçekleştirmek için kullanılabilir, örneğin metin üretimi, çeviri, özetleme ve duyarlılık analizi gibi.**

### 1.c.iii Devin

"Devin" terimi, yapay zekâ (AI) alanında özellikle nesne tespiti ve sınıflandırma görevlerinde kullanılan bir derin öğrenme modelini ifade eder. Bu model, nesne tanıma görevlerinde kullanılan birçok farklı evrişimli sinir ağı (CNN) mimarisini içerebilir. Ayrıca Devin AI, Cognition AI şirketinin oluşturduğu ve verilen problemi çözmek için kendi yapay zekâ modellerini oluşturabilen bir AI’ı refere eder.

### 1.c.iv CycleGAN

CycleGAN, image-to-image translation problemlerini çözmek için kullanılan bir evrişimli sinir ağı (CNN) tabanlı bir tekniktir. Bu teknik, görüntülerin stilini veya içeriğini başka bir domaine (alana) dönüştürmek için kullanılır.

CycleGAN, iki temel bileşenden oluşur: Jeneratör (Generator) ve Ayrımcı (Discriminator). Jeneratörler, bir görüntü domeni için verilen bir görüntüyü diğer domene dönüştürürken, ayrımcılar bu dönüşümlerin ne kadar gerçekçi olduğunu değerlendirir.

**Geleneksel GAN'lardan (Generative Adversarial Networks) farklı olarak, CycleGAN, eşleştirilmiş veriye ihtiyaç duymaz ve dönüşüm yapmak için döngüsel tutarlılık kaybını kullanır. Örneğin, bir atın resmini zebra gibi bir görüntüye dönüştürme veya yaz gecesi bir manzaranın resmini kış manzarasına dönüştürme gibi.**

### 1.c.v GPT-4 vs Gemini vs Claude 3

GPT-4: OpenAI tarafından geliştirilen GPT-4, GPT serisinin bir sonraki modelidir. GPT-4, milyarlarca parametreye sahip bir model olup, geniş bir dil anlama yeteneği sunar.

Gemini: Microsoft tarafından geliştirilen Gemini, GPT-4'e benzer şekilde büyük dil modellerinin bir örneğidir. **Ancak, GPT-4'ten farklı olarak, Gemini, dikkat odaklı mekanizmaları vurgular ve dil anlama görevlerinde daha fazla odaklanma yeteneğine sahip olabilir.** Gözlemlerime göre, genel sorularda GPT-4, yazılımsal sorularda (kod, algoritma vs.) Gemini daha doğru sonuçlar üretiyor.

Claude 3: Google tarafından geliştirilen Claude 3, büyük bir dil modelidir. **Claude 3, bilgi tabanlı öğrenme ve doğrudan bilgi çıkarma görevlerine odaklanırken, dil anlama yeteneği açısından diğer modellere göre farklı bir odak noktasına sahip olabilir.** Gözlemlerime göre, Claude 3 her iki dil modelinden de daha doğru sonuçlar üretiyor ve bilimsel kavramları çok iyi kavrıyor.

**Kalın çizili yerler GPT-3.5’tan alınmıştır.**

*İtalik çizili yerler Gemini’dan alınmıştır.*

## 1.d Makine Öğrenmesi Mülakat Soruları

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

1. Supervised ve Unsupervised arasındaki farklar nelerdir?

metin, el yazısı, doküman, belge, kağıt içeren bir resim

Açıklama otomatik olarak oluşturuldu

2. Association nedir, algoritması nasıl çalışır?

metin, el yazısı, sayı, numara, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.d.ii Aynı soruların ChatGPT ve Gemini araçlarına sorulması ve cevaplarının notlandırılması

ChatGpt:

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Gemini:

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

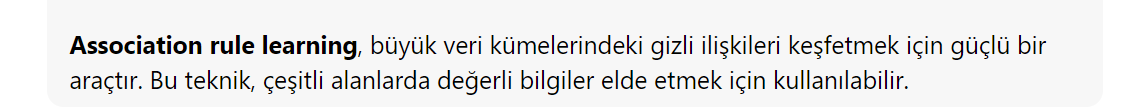
Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu



Puanlarım (ChatGpt, Gemini): 80, 95

# 2) Problem Çözme ve Kodlama

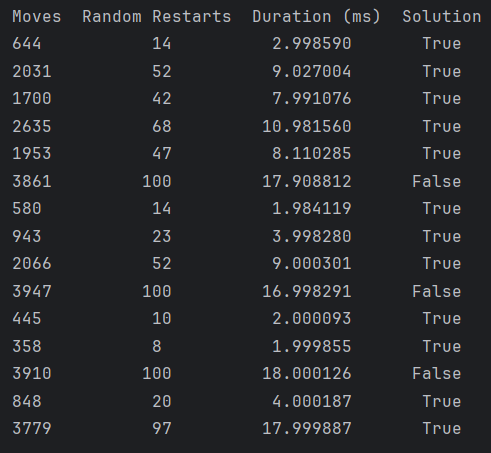
## 2.a Sekiz Vezir Probleminin Hill Climbing ile Çözülmesi / Bridge and torch probleminin A\* ile çözülmesi

### 2.a.i Kodlar ve Çıktılar

|  |
| --- |
| import random import time   class EightQueens:  def \_\_init\_\_(self):  self.n = 8  self.board = [random.randint(0, self.n - 1) for \_ in range(self.n)]   def get\_conflicts(self):  conflicts = 0  for i in range(self.n):  for j in range(i + 1, self.n):  if self.board[i] == self.board[j] or abs(self.board[i] - self.board[j]) == j - i:  conflicts += 1  return conflicts   def hill\_climbing(self, restart\_count):  min\_conflicts = self.get\_conflicts()  restarts = 0  moves = 0   while min\_conflicts > 0:  for i in range(self.n):  original\_row = self.board[i]  for j in range(self.n):  if j != original\_row:  self.board[i] = j  conflicts = self.get\_conflicts()  moves += 1  if conflicts < min\_conflicts:  min\_conflicts = conflicts  break  else:  self.board[i] = original\_row   if min\_conflicts == 0:  break   restarts += 1  if restarts >= restart\_count:  break   self.board = [random.randint(0, self.n - 1) for \_ in range(self.n)]  min\_conflicts = self.get\_conflicts()   return moves, restarts   def main():  results = []  for \_ in range(15):  start\_time = time.time()  problem = EightQueens()  moves, restarts = problem.hill\_climbing(100)  end\_time = time.time()  duration = (end\_time - start\_time) \* 1000  if problem.get\_conflicts() == 0:  print("Solution found: " + str(problem.board))  solution = True  results.append([moves, restarts, duration, solution])  else:  print("Solution not found.")  solution = False  results.append([moves, restarts, duration, solution])   print("\nResults:\n")  print(" Moves Random Restarts Duration (ms) Solution")  for moves, restarts, duration, solution in results:  moves\_str = str(moves).center(5)  restarts\_str = str(restarts).center(15)  duration\_str = "{:.6f}".format(duration).center(13)  solution\_str = str(solution).center(8)  print(f"{moves\_str}\t{restarts\_str}\t{duration\_str}\t{solution\_str}")   if \_\_name\_\_ == "\_\_main\_\_":  main() |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Çalıştırma No** | **Yer değiştirme sayısı** | **Random restart sayısı** | **İşlem Süresi** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |



metin, ekran görüntüsü, yazı tipi, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

\* Listedeki her bir indeks satranç tahtasının sütun numarasını ve her indeksteki değer de vezirlerin bulundukları satır numarasını ifade etmektedir

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

Stochastic Hill Climbing: Klasik Hill Climbing algoritmasında, bir sonraki adım, mevcut durumun tüm komşuları incelenerek ve bu komşulardan en iyi olanı seçilerek belirlenir. Ancak Stochastic Hill Climbing'de, komşular arasından rastgele birisi seçilir ve eğer seçilen komşu mevcut durumdan daha iyiyse, o konuma geçilir.

Bu stochastic seçim sayesinde algoritma, yerel optimumlara takılma ihtimalini azaltır ve daha geniş bir arama alanı elde eder. Fakat bu stochastic seçim, her zaman en iyi çözüme ulaşmayı garanti etmez.

Bu nedenle, **hill\_climbing** fonksiyonunda, her vezirin yeni bir konuma yerleştirilmeden önce, rastgele olarak birkaç farklı konum denemesi yapılmalıdır.

def hill\_climbing(self, restart\_count):  
 min\_conflicts = self.get\_conflicts()  
 restarts = 0  
 moves = 0  
  
 while min\_conflicts > 0:  
 for i in range(self.n):  
 original\_row = self.board[i]  
 for \_ in range(5):  
 j = random.randint(0, self.n - 1)  
 if j != original\_row:  
 self.board[i] = j  
 conflicts = self.get\_conflicts()  
 moves += 1  
 if conflicts < min\_conflicts:  
 min\_conflicts = conflicts  
 break  
 else:  
 self.board[i] = original\_row  
  
 if min\_conflicts == 0:  
 break  
  
 restarts += 1  
 if restarts >= restart\_count:  
 break  
  
 self.board = [random.randint(0, self.n - 1) for \_ in range(self.n)]  
 min\_conflicts = self.get\_conflicts()  
  
 return moves, restarts

\* Bizim problemimiz için bu yaklaşım daha kötü sonuçlar verdi

metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 3) Genetik Algoritmalar ile Şifre Kırma

## 3.a Kodlar

|  |
| --- |
| **package** project1;  **import** java.util.Arrays;  **import** java.util.Random;  **public** **class** GeneticAlgorithm {  **private** **static** **final** String ***TARGET*** = "(Generative AI) ";  **private** **static** **final** **int** ***POPULATION\_SIZE*** = 100;  **private** **static** **final** **double** ***MUTATION\_RATE*** = 0.01;  **private** **static** **final** Random ***random*** = **new** Random();  **public** **static** **void** main(String[] args) {  **double** avarageSolutionTime = 0;  **double** avarageSolutionGeneration = 0;  **for** (**int** i = 0; i < 2; i++) {  **long** startTime = System.*currentTimeMillis*();  **int** generation = *evolve*();  **long** endTime = System.*currentTimeMillis*();  System.***out***.println("Solution found in " + generation + " generations.");  System.***out***.println("Time taken: " + (endTime - startTime) + " milliseconds.");  System.***out***.println();  avarageSolutionTime += (endTime - startTime);  avarageSolutionGeneration += generation;  }  System.***out***.println("Avarage generation: " + (avarageSolutionGeneration / 2) + " generations.");  System.***out***.println("Avarage time taken: " + (avarageSolutionTime / 2) + " milliseconds.");  }  **private** **static** **int** evolve() {  **char**[][] population = **new** **char**[***POPULATION\_SIZE***][***TARGET***.length()];  **double**[] fitness = **new** **double**[***POPULATION\_SIZE***];  // Initialize population  **for** (**int** i = 0; i < ***POPULATION\_SIZE***; i++) {  population[i] = *generateRandomString*();  }  **int** generation = 0;  **int** flagOfClosestSolution = 0;  **while** (**true**) {  // Calculate fitness for each individual in population  **for** (**int** i = 0; i < ***POPULATION\_SIZE***; i++) {  fitness[i] = *calculateFitness*(population[i]);  **if** (fitness[i] == 1) {  System.***out***.print("Solution found: ");  System.***out***.println(population[i]);  **return** generation;  }  }    // Since printing all generations would take up unnecessary space,  // a generation is written once every 100 generations.  **if** (flagOfClosestSolution %100 == 0) {  *findClosestSolution*(fitness, population, generation);  }  // Create new generation  **char**[][] newPopulation = **new** **char**[***POPULATION\_SIZE***][***TARGET***.length()];  **for** (**int** i = 0; i < ***POPULATION\_SIZE***; i++) {  **char**[] parent1 = *selectParent*(population, fitness);  **char**[] parent2 = *selectParent*(population, fitness);  **char**[] child = *crossover*(parent1, parent2);  *mutate*(child);  newPopulation[i] = child;  }  population = newPopulation;  generation++;  flagOfClosestSolution++;  }  }    **private** **static** **void** findClosestSolution(**double**[] fitness, **char**[][] population, **int** generation) {  **int** bestIndividualIndex = 0;  **for** (**int** i = 1; i < ***POPULATION\_SIZE***; i++) {  **if** (fitness[i] > fitness[bestIndividualIndex]) {  bestIndividualIndex = i;  }  }  System.***out***.print("Generation " + generation + ": ");  System.***out***.println(population[bestIndividualIndex]);  }  **private** **static** **char**[] generateRandomString() {  **char**[] randomString = **new** **char**[***TARGET***.length()];  String validChars = "abcdefghıijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789!@#$%^&\*()\_+-=[]{};':,.<>/?";  Random random = **new** Random();  **for** (**int** i = 0; i < ***TARGET***.length(); i++) {  randomString[i] = validChars.charAt(random.nextInt(validChars.length()));  }  **return** randomString;  }  **private** **static** **double** calculateFitness(**char**[] candidate) {  **double** fitness = 0;  **for** (**int** i = 0; i < ***TARGET***.length(); i++) {  **if** (candidate[i] == ***TARGET***.charAt(i)) {  fitness++;  }  }  **return** fitness / ***TARGET***.length();  }  **private** **static** **char**[] selectParent(**char**[][] population, **double**[] fitness) {  **double** totalFitness = Arrays.*stream*(fitness).sum();  **double** randomFitness = ***random***.nextDouble() \* totalFitness;  **double** sum = 0;  **for** (**int** i = 0; i < ***POPULATION\_SIZE***; i++) {  sum += fitness[i];  **if** (sum >= randomFitness) {  **return** population[i];  }  }  **return** population[***POPULATION\_SIZE*** - 1];  }    **private** **static** **char**[] crossover(**char**[] parent1, **char**[] parent2) {  **char**[] child = **new** **char**[***TARGET***.length()];  **int** midpoint = ***random***.nextInt(***TARGET***.length());  **for** (**int** i = 0; i < ***TARGET***.length(); i++) {  **if** (i < midpoint) {  child[i] = parent1[i];  } **else** {  child[i] = parent2[i];  }  }  **return** child;  }  **private** **static** **void** mutate(**char**[] child) {  String validChars = "abcdefghıijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789!@#$%^&\*()\_+-=[]{};':,.<>/?";  **for** (**int** i = 0; i < ***TARGET***.length(); i++) {  **if** (***random***.nextDouble() < ***MUTATION\_RATE***) {  **int** randomIndex = ***random***.nextInt(validChars.length());  child[i] = validChars.charAt(randomIndex);  }  }  }  } |

### metin, yazı tipi, ekran görüntüsü, siyah beyaz içeren bir resim Açıklama otomatik olarak oluşturuldu

### metin, yazı tipi, beyaz, ekran görüntüsü içeren bir resim Açıklama otomatik olarak oluşturuldu

Kromozom sayısı arttıkça başarı artıyor ama süre de artıyor.

\* Kalabalıklık olmasın diye her 100 jenerasyonda bir en yakın sonuç yazdırıldı.

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

Crossover: İki ebeveyn kromozomdan rastgele bir nokta (midpoint) seçilir ve bu noktaya kadar olan genler birinci ebeveyn kromozomundan, kalan genler ise ikinci ebeveyn kromozomundan alınarak yeni bir çocuk kromozom oluşturulur.

Mutation: Çocuk kromozomu mutasyona uğratır. ***random***.nextDouble() ile seçilen 0 ve 1 arasındaki ondalıklı sayı eğer ***MUTATION\_RATE*** ‘den küçük ise validChars String’inden rastgele bir char seçilir ve çocuğun ilgili kısmı (burası da random seçiliyor) bu char ile değiştirilir.

Selection: İlk olarak, popülasyon içindeki her bireyin fitness değeri toplanır ve **totalFitness** değişkenine atanır. Ardından, bir rastgele sayı üretilir ve bu sayı, toplam fitness değerine göre normalize edilir (**totalFitness** ile çarpılır). Daha sonra, bu rastgele fitness değerine kadar olan toplam fitness değerleri kümülatif olarak hesaplanır. Bu kümülatif değer, rastgele fitness değerine ulaşana kadar döngü devam eder. Ulaştığı zaman o indeksteki birey seçilir. Eğer rastgele fitness değerine hiçbir bireyin fitness değeri ulaşamazsa, son birey seçilir.

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

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü üzere hedeflenen kelime kısaldıkça süre de azalıyor.

Dipnot: Genetik algoritmaların etkinliğini ölçmek için gönüllü olarak başka bir sınıf daha yazdım. StringGuesser sınıfım şöyle çalışıyor: rastgele bir kelime oluşturuluyor (target ile aynı uzunlukta), eşleşen char varsa eşleşen char dışındaki tüm char’lar tekrar rastgele olarak atanıyor (eşleşen char yoksa hepsi yine rastgele atanıyor), target bulunana kadar bu böyle devam ediyor.

Örneğin;

Target = “elma”

Prediction1= “elas”

Prediction2 = “elsr”

Prediction3 = “elms”

Prediction4 = “elma”

Dikkatimi çeken şey ise neredeyse genetik algoritmadan 10 kat daha hızlı çalışıyor (Eğer target uzunluğu çok fazla ise StringGuesser neredeyse 200-300 kat daha hızlı çalışıyor, target uzadıkça hız exponential olarak artıyor 1000-2000 kat gibi) . Genetik algoritmanın bu kadar işlevsiz olduğuna şaşırdım.

“(Generative AI)” için sonuçlar:

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 4)Python ile Makine Öğrenmesi [Sınıflandırıcı]

## 4.a Python Standardization örneği

### 4.a.i Kodlar

|  |
| --- |
| from sklearn.preprocessing import StandardScaler import numpy as np  data = np.array([[1.2, 3.4], [4.4, 2.2], [3.7, 4.3], [5.8, 6.0]])  scaler = StandardScaler() scaled\_data = scaler.fit\_transform(data)  print("Standartlaştırılmış veri: ") print(scaled\_data) |

### 4.a.ii Normalizasyondan Farklı

Standardizasyon: Bir veri kümesindeki her bir özelliğin (feature) değerlerini, ortalama değere göre ölçeklendirmek ve standart sapmaya göre normalize etmek anlamına gelir. Bu işlem, her bir özelliğin değerlerini o özelliğin ortalaması 0 ve standart sapması 1 olacak şekilde dönüştürür.

Normalizasyon: Veri özelliklerini belirli bir aralığa ölçeklemek için kullanılır, genellikle 0 ile 1 arasında veya -1 ile 1 arasında bir aralık oluşturulur. Örneğin, yapay sinir ağlarında softmax aktivasyon fonksiyonu çıktı değerleri 0 ve 1 arasında toplamları 1 olacak şekilde ölçeklendirir.

## 4.b Veri seti bulma veya oluşturma

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

**Veri sayısı: 8000**

**Öznitelik sayısı: 7**

**Öznitelikler: Size, Weight, Sweetness, Softness, HarvestTime, Ripeness, Acidity, Quality**

**Sınıf sayısı: 2**

**Sınıflar: “Good”, “Bad”**

## 4.c İki farklı sınıflandırıcı ve sonuçlar

### 4.c.i Kaynak Kod

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split, cross\_val\_score from sklearn.tree import DecisionTreeClassifier from sklearn.svm import SVC from sklearn.metrics import confusion\_matrix import matplotlib.pyplot as plt import pandas as pd import numpy as np import seaborn as sns   def print\_confusion\_matrix(y\_true, y\_pred):  cm = confusion\_matrix(y\_true, y\_pred)  plt.figure(figsize=(8, 6))  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Good', 'Bad'], yticklabels=['Good', 'Bad'])  plt.title('Confusion Matrix')  plt.xlabel('Predicted Label')  plt.ylabel('True Label')  plt.show()   def predict\_class(classifier, new\_instance):  prediction = classifier.predict([new\_instance])  return prediction[0]   def main():   banana\_file = 'C:\\Users\\Can\\PycharmProjects\\AI\\csv files\\banana\_quality.csv'  data = pd.read\_csv(banana\_file)   X = data.iloc[:, :-1]  y = data.iloc[:, -1]   X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=0, stratify=y)   # Karar Ağacı modeli  dtc\_model = DecisionTreeClassifier(random\_state=0)  dtc\_model.fit(X\_train, y\_train)   # Destek Vektör Makinesi modeli  svm\_model = SVC(random\_state=0)  svm\_model.fit(X\_train, y\_train)   new\_instance = [-2, 1, 1.5, 0.73, -0.34, 1.23, -1.2]   dtc\_prediction = predict\_class(dtc\_model, new\_instance)  svm\_prediction = predict\_class(svm\_model, new\_instance)   print("DTC Classifier Prediction:", dtc\_prediction)  print("SVM Prediction:", svm\_prediction)   # DTC Classifier için 10 katlı çapraz doğrulama  dtc\_accuracy\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='accuracy')  dtc\_precision\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='precision\_macro')  dtc\_recall\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='recall\_macro')   # SVM için 10 katlı çapraz doğrulama  svm\_accuracy\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='accuracy')  svm\_precision\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='precision\_macro')  svm\_recall\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='recall\_macro')   # Skorları yazdır  print("\nDTC Classifier:")  print("Accuracy:", np.mean(dtc\_accuracy\_scores))  print("Precision:", np.mean(dtc\_precision\_scores))  print("Recall:", np.mean(dtc\_recall\_scores))   print("\nSVM:")  print("Accuracy:", np.mean(svm\_accuracy\_scores))  print("Precision:", np.mean(svm\_precision\_scores))  print("Recall:", np.mean(svm\_recall\_scores))   # Karışıklık matrislerini yazdır  y\_pred\_dtc = dtc\_model.predict(X\_test)  y\_pred\_svm = svm\_model.predict(X\_test)   print\_confusion\_matrix(y\_test, y\_pred\_dtc)  print\_confusion\_matrix(y\_test, y\_pred\_svm)   main() |

### 4.c.ii Confusion Matrix

DTC:

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

SVM:

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

### 4.d.i Kaynak Kod

|  |
| --- |
| def predict\_class(classifier, new\_instance):  prediction = classifier.predict([new\_instance])  return prediction[0]  new\_instance = [-2, 1, 1.5, 0.73, -0.34, 1.23, -1.2]  dtc\_prediction = predict\_class(dtc\_model, new\_instance) svm\_prediction = predict\_class(svm\_model, new\_instance)  print("DTC Classifier Prediction:", dtc\_prediction) print("SVM Prediction:", svm\_prediction) |

### 4.d.ii Ekran Görüntüleri

metin, yazı tipi, ekran görüntüsü, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 4.f 10-Fold Cross Validation

### 4.f.i Kaynak Kod

|  |
| --- |
| # DTC Classifier için 10 katlı çapraz doğrulama dtc\_accuracy\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='accuracy') dtc\_precision\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='precision\_macro') dtc\_recall\_scores = cross\_val\_score(dtc\_model, X, y, cv=10, scoring='recall\_macro')  # SVM için 10 katlı çapraz doğrulama svm\_accuracy\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='accuracy') svm\_precision\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='precision\_macro') svm\_recall\_scores = cross\_val\_score(svm\_model, X, y, cv=10, scoring='recall\_macro') |

### 4.f.ii 10-Fold Cross Validation ile Elde Edilen Başarımlar

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 5) Harcanan Süreler ve İş Bölümü Detayları

Tüm projeyi tek başıma yaptım.

Birinci soru: 3 saat

İkinci soru: 9 saat

Üçüncü soru: 6,5 saat

Dördüncü soru: 2 saat

Beşinci soru ve öz değerlendirme formu: 0,5 saat

# 6) Öz değerlendirme Tablosu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **İstenen Özellik** | **Var** | **Açıklama** | **Tahmini Not** |
| **1a** | Algoritmalar + Karmaşıklıklar (10) |  | Yapıldı, internetten bilgileri topladım ve derste işlediklerimizle karşılaştırdım. Sonra, sentezleyip yazdım. | 9 |
| **1b** | Gen AI ile A\* Öğrenme ve Kodlama (10) |  | Yapıldı, çok basitti ChatGpt’ye ve Gemine’a sordum. | 10 |
| **1c** | Tanım ve Karşılaştırma (10) |  | Yapıldı, Generative AI’lardan yardım aldım ve uzun süredir Generative AI kullandığımdan dolayı oluşan tecrübelerime göre karşılaştırdım. | 10 |
| **1d** | Araştırma ve Yorum (10) |  | Yapıldı, sorular için [www.coursera.com’dan](http://www.coursera.com’dan) yararlandım. | 10 |
| **2** | Problem Çözme ve Kodlama (10) |  | Yapıldı, en zor kısımdı benim için satranç tahtasını oluşturmak ve düzgün hill climbing algoritması yazmak baya uğraştırdı. | 9 |
| **3** | Genetik Algoritmalar ile Şifre Kırma (15) |  | Yapıldı, en zevk aldığım kısımdı. Hatta genetic algoritmaları test etmek için başka bir algoritma daha yazdım. | 15 |
| **4** | Makine Öğrenmesi (15) |  | Yapıldı, daha önce birçok kez uğraştığım için kodum neredeyse hazırdı sadece iki tane classifier ekledim. Eski kodumda tek classifier vardı. | 15 |
| **5** | İşbölümü Detayları (5) |  | Yapıldı, projeyi kendi başıma yaptığımdan doldurmak kolay oldu. | 5 |
| **6** | Özd. Tablosu + Rapor (15) |  | Yapıldı, genel olarak sonda tüm projeyi değerlendirmeyi seviyorum. | 15 |
| **100 üzerinden Toplam Not:** | | | | 98 |

**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.**