EGE ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAPAY ZEKÂ YÖNTEMLERİ (3+0)

2021-2022 BAHAR YARIYILI

PROJE-1 RAPORU

TESLİM TARİHİ

13/05/2022

HAZIRLAYANLAR

05200000002, Mert Akçay

05200000001,Cem Ulus

05200000791, Deniz Üstüner

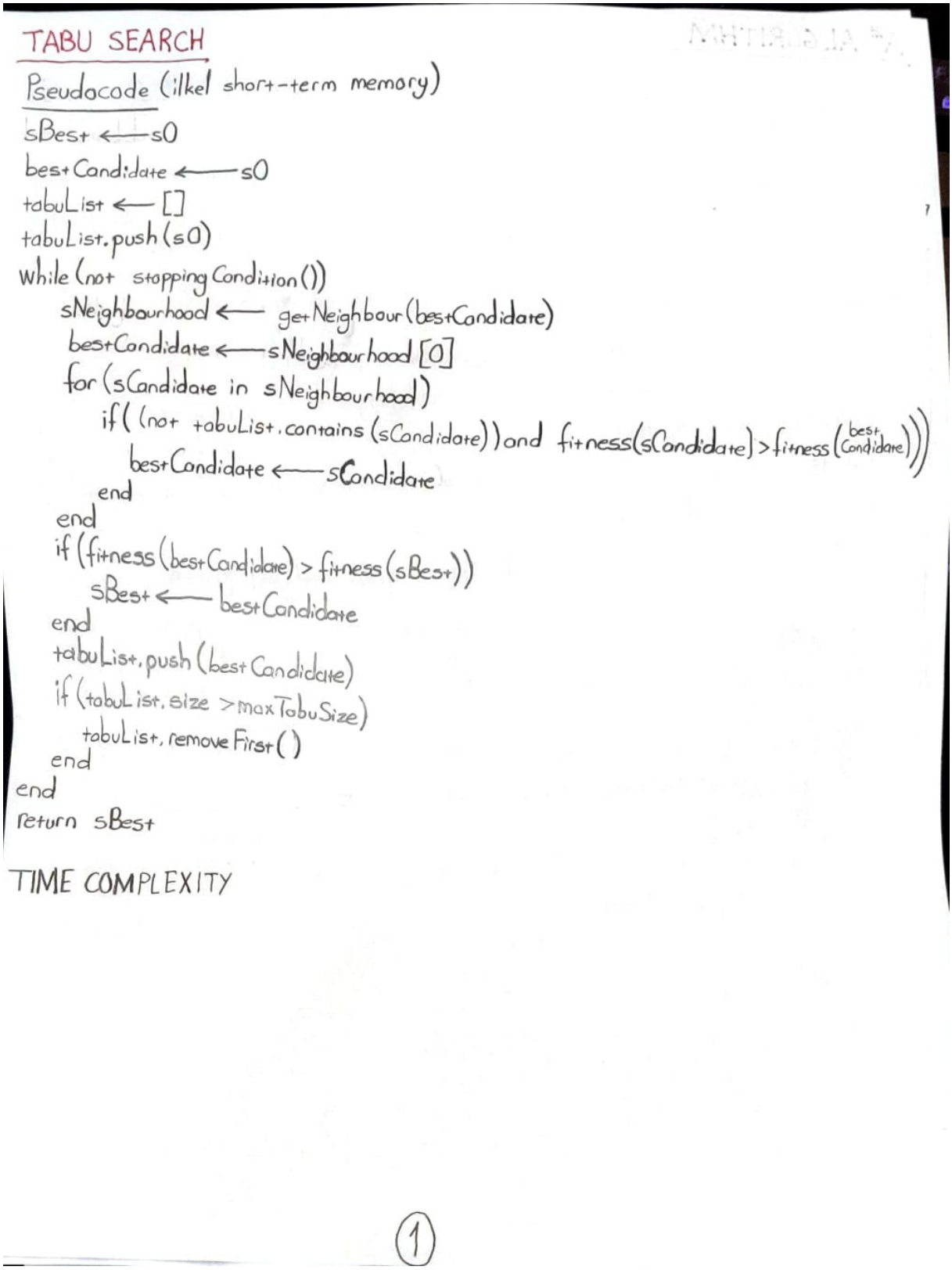
GITHUB LINKI

https://github.com/mertakcay/AI

İçindekiler

1) Algoritmalar, Tanımlar, Karşılaştırma, Araştırma ve Yorum

1.a) Tabu Search (Yasak Arama) ve A* Algoritmalarını araştırıp bilgilerinizi pekiştiriniz. Her iki algoritmayı da mürekkepli (veya kurşun) kalemle bir kağıda yazınız. Zaman karmaşıklıklarını belirtiniz. Sadece A* Algoritmasını anlatınız



TABU SEARCH

Pseudocode (ilkel short-term memory)

```
sBest ← s0
bestCandidate ← s0
tabuList ← []
tabuList.push(s0)
While (not stoppingCondition())
    sNeighbourhood ← getNeighbour(bestCandidate)
    bestCandidate ← sNeighbourhood[0]
    for (sCandidate in sNeighbourhood)
        if( (not tabuList.contains(sCandidate)) and fitness(sCandidate) > fitness(bestCandidate))
            bestCandidate ← sCandidate
        end
    end
    if (fitness(bestCandidate) > fitness(sBest))
        sBest ← bestCandidate
    end
    tabuList.push(bestCandidate)
    if (tabuList.size > maxTabuSize)
        tabuList.removeFirst()
    end
end
return sBest
```

TIME COMPLEXITY

(1)

A* ALGORITHM

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path
```

Belirli noktalar arasındaki yolu oluşturan fonksiyon. Hangi düğümden gelindiyse onları bularak geriye doğru yolu oluşturur.

```
function A_star(start, goal, h)
    openSet := {start} // Yeniden genişletilmesi gerekebilecek keşfedilen düğümler kümesi
    // ilk başta yalnızca başlangıç düğümü biliniyor
    cameFrom := empty map
    // n düğümü için cameFrom[n] başlangıçtan itibaren en kısa yolda hemen önceki düğüm
    gScore := map with default value of Infinity
    gScore[start] := 0 // başlangıçtan başlangıca gidiş maliyeti 0

    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty // → öncelik sırası min-heap ise → O(Log(n))
        current := the node in openSet having the lowest fScore[] value
        if current == goal
            return reconstruct_path(cameFrom, current)
        openSet.remove(current)
        for each neighbour of current // tüm komşularda en küçük değeri bulmak için döngü
        // distance(current, neighbour) o andaki düğümden komşuya olan kenarın ağırlığı
            tentative_gScore := gScore[current] + d(current, neighbour)
            if tentative_gScore < gScore[neighbour] // bu yol önceki yollardan daha verimli ise güncelle
                cameFrom[neighbour] := current
                gScore[neighbour] := tentative_gScore
                fScore[neighbour] := tentative_gScore + h(neighbour)
                if neighbour not in openSet
                    openSet.add(neighbour)
    return failure // openSet boş ama hedefe ulaşılamadı ise
```

★ f = g + h

fScore değeri daha hesaplanmadığı için infinity değerler

★ current i bulmak için openSet de en düşük skorlu f değerini buluyoruz

Sezgisel arama yöntemi olan A* search algoritması optimal verimliliği nedeniyle bilgisayar biliminin birçok alanında sıklıkla kullanılan bir grafik geçiş ve yol arama algoritmasıdır. Bir önemli dezavantajı oluşturulan tüm düğümleri bellekte depoladığı için $O(b^d)$ yer karmaşıklığıdır. Tamdır, çözüm varsa bulunur. Zaman karmaşıklığı optimaldir. Sezgi ne kadar iyi olursa zaman da o kadar iyi olur.

En iyi durumda → h çok iyi → $O(d)$

En kötü durumda → $h=0$ → $O(b^d)$ → BFS ile aynı

BFS greedy yerine $f=g+h$ ile yapıldığında A*'a ulaşılır.

g: şu ana kadar olan maliyet fonksiyonu

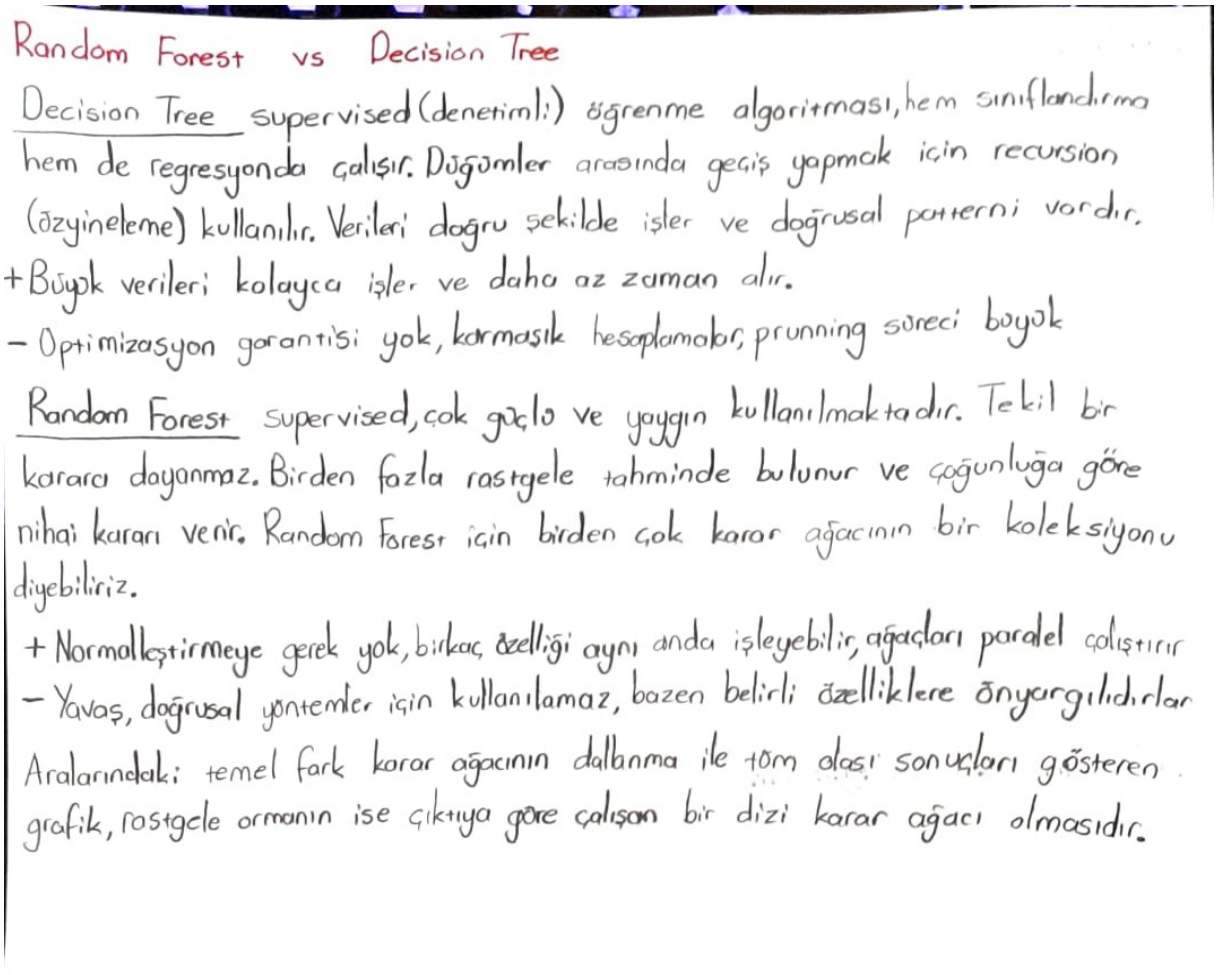
h: uygun sezgi fonksiyonu

- Eğer sezgisel h grafiğin her x,y kenarı için $h(x) \leq d(x,y) + h(y)$ koşulunu sağlıyor ise h → monoton/tutarlıdır.

(d(x,y): kenarın uzunluğu)

Tutarlı bir sezgi ile A* algoritmasının herhangi bir düğümü birden fazla işlemeden en optimal yolu bulması garanti edilir. Dijkstra algoritmasını düşük maliyetli çalıştırmaya eşdeğerdir.

1.b) Aşağıdaki kavramları araştırdıktan sonra (i-iii) maddelerini anlatınız, (iv-v) maddelerini kendi cümlelerinizle kısaca karşılaştırınız, alıntı yaptığınız yerlerde ilgili kaynaklara atıf veriniz



Random Forest vs Decision Tree

Decision Tree supervised (denetimli) öğrenme algoritması, hem sınıflandırma hem de regresyonda çalışır. Düğümler arasında geçiş yapmak için recursion (özyineleme) kullanılır. Verileri doğru şekilde işler ve doğrusal patterni vardır.

+ Büyük verileri kolayca işler ve daha az zaman alır.

- Optimizasyon garantisi yok, karmaşık hesaplamalar, prunning süreci büyük

Random Forest supervised, çok güçlü ve yaygın kullanılmaktadır. Tekil bir kararа dayanmaz. Birden fazla rastgele tahminde bulunur ve çoğunluğa göre nihai kararı verir. Random Forest için birden çok karar ağacının bir koleksiyonu diyebiliriz.

+ Normalleştirmeye gerek yok, birkaç özelliği aynı anda işleyebilir, ağaçları paralel çalıştırır

- Yavaş, doğrusal yöntemler için kullanılamaz, bazen belirli özelliklere önyargılıdırlar

Aralarındaki temel fark karar ağacının dallanma ile tüm olası sonuçları gösteren grafik, rastgele ormanın ise çıktıya göre çalışan bir dizi karar ağacı olmasıdır.

Alpha-Beta Pruning

Recursive minimax algoritmasına karşı tarafın en iyi hamlesini yapacağını varsayarak iki tarafın da ulaşabileceği en iyi skoru takip eden alpha (α) ve beta (β) parametreleri eklenerek optimize edilmiş halidir.

Arama ağacının sonuçla ilgili olmayan dalları budanır. Ağacın herhangi bir derinliğinde uygulanabilir ve bazen sadece yaprakları değil tüm alt ağacı budar. Bir yol şimdiki durumdan daha kötüyse elenir.

	Initial Value
α → Maximize etme yolu boyunca herhangi bir noktada en iyi seçim	$-\infty$
β → Minimize etme yolu boyunca herhangi bir noktada en iyi seçim	$+\infty$

★ Budama sonucu değiştirmez, algoritmayı yavaşlatan tüm düğümleri kaldırarak hızlandırır.

Worst-case performans → $O(b^m)$

Best-case performans → $O(b^{m/2})$

★ Satranç gibi oyunlarda acemi ve uzman oyuncu arasındaki farkı belirler.

GPT-3

Generative Pre-trained Transformer 3, insan benzeri metinler üretmek için derin öğrenmeyi kullanan otoregresif bir dil modelidir.

Open AI tarafından oluşturulan GPT-3 ün önemli kabul edilmesini sağlayan şey kendisine beslenen veri büyüklüğünden ziyade bilgiler arasında bağlantı kurabilmesini sağlayan 175 milyar makine öğrenme parametresine sahip olmasıdır.

GPT-3 sorulara cevap verebiliyor, tasarım yapabiliyor, yazarları taklit edebiliyor hatta kendisine tarif edilen uygulamayı birkaç saniyede yazıp kodu gösterebiliyor. Sadece bilgi anlamında değil insanları davranış biçimi olarak da taklit edebiliyor.

GPT-3 Eğitim Verileri →

Dataset	Tokens	Training Mix Ağırlığı
Common Crawl*	410 billion	%60
WebText2	19 billion	%22
Books1	12 billion	%8
Books2	55 billion	%8
Wikipedia	3 billion	%3

*Common Crawl, webi tarayan ve arşivlerini, datasetlerini halka ücretsiz olarak sağlayan kuruluş

AutoML

Otomatikleştirilmiş makine öğrenimi, makine öğrenimini gerçek dünya sorunlarına uygulamayı otomatikleştirme sürecidir. Potansiyel olarak ham bir veri setinden son uygulamaya hazır bir makine öğrenimi modeli oluşturmaya kadar her aşamayı içerir. AutoML artan makine öğrenimi uygulama zorluğuna yapay zeka tabanlı bir çözüm olarak önerilmiştir.

Kullanılan yaygın teknikler

Hiperparametre optimizasyonu
Meta-öğrenme
Sinir mimarisi araması

Standart makine öğrenimi uygulamasında ham verilerin tüm algoritmaların uygulanabileceği biçime getirilmesi için bir uzmanın çeşitli yöntemler uygulaması gerekebilir ve bu süreç zorlu olabilir. AutoML, bu adımları basitleştirmeyi ve daha verimli hale getirmeyi amaçlar.

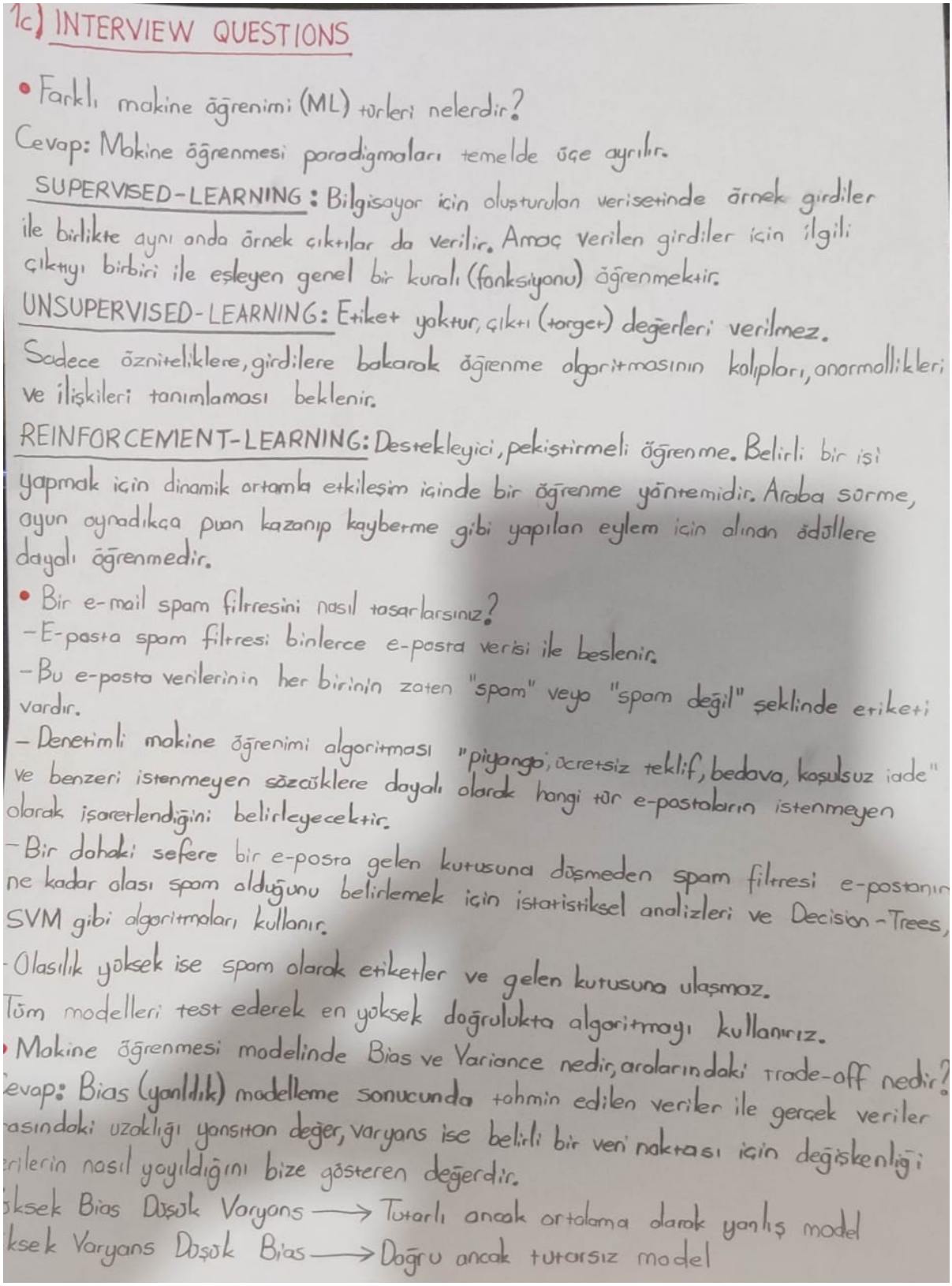
CycleGan vs StyleGan

CycleGAN bir resmi farklı bir konsepte dönüştürmeyi amaçlar. Bu çalışma öbki sistem her verinin iki durumda da nasıl görüneceğini gösteren eşli veriye gerek duymaz. İç içe geçmiş 2 tane generative adversarial network kullanarak resmi önce istenen içeriğe sonra eski haline dönüştürerek sonucun yeni içeriğe ne kadar uyum sağladığını hesaplar ve kayıpları azaltmayı hedefler.

StyleGAN, NVIDIA tarafından yapılan çalışmanın amacı gerçekçi yüz resimleri üretmek. Kullanılan yöntemler sadece yüz üretme için sınırlı olmasa da en büyük başarıyı bu alanda sağladığı için özelliği ile ön plana çıkıyor. Sistem yüzün açısı iller, saç rengi gibi bir çok niteliği unsupervised (gözetimsiz) öğrenebiliyor.

er seferinde yeni bir sahte yüz üreten site ⟶ thispersondoesnotexist.com

1.c) Yapay Zeka (ML ve Derin Öğrenme gibi alanları da içermektedir) alanında bir staja, şirkete veya üniversiteye başvurduğunuzda sorulabilecek mülakat sorularını internet üzerinden araştırınız ve 3 tanesini yazınız, ardından kendi cümlelerinizle kağıda yazarak cevaplandırınız veya çözümünü yapınız.



1c) INTERVIEW QUESTIONS

• Farklı makine öğrenimi (ML) türleri nelerdir?

Cevap: Makine öğrenmesi paradigmaları temelde üçe ayrılır.

SUPERVISED-LEARNING: Bilgisayar için oluşturulan verisetinde örnek girdiler ile birlikte aynı anda örnek çıktılar da verilir. Amaç verilen girdiler için ilgili çıktıyı birbiri ile eşleyen genel bir kuralı (fonksiyonu) öğrenmektir.

UNSUPERVISED-LEARNING: Etiket yoktur, çıktı (target) değerleri verilmez. Sadece özniteliklere, girdilere bakarak öğrenme algoritmasının kalıpları, anormallikleri ve ilişkileri tanımlaması beklenir.

REINFORCEMENT-LEARNING: Destekleyici, pekiştirmeli öğrenme. Belirli bir işi yapmak için dinamik ortamla etkileşim içinde bir öğrenme yöntemidir. Araba sürme, oyun oynadıkça puan kazanıp kaybetme gibi yapılan eylem için alınan ödüllere dayalı öğrenmedir.

• Bir e-mail spam filtresini nasıl tasarlarsınız?

- E-posta spam filtresi binlerce e-posta verisi ile beslenir.
- Bu e-posta verilerinin her birinin zaten "spam" veya "spam değil" şeklinde etiketi vardır.
- Denetimli makine öğrenimi algoritması "piyango, ücretsiz teklif, bedava, koşulsuz iade" ve benzeri istenmeyen sözcüklere dayalı olarak hangi tür e-postaların istenmeyen olarak işaretlendiğini belirleyecektir.
- Bir dahaki sefere bir e-posta gelen kutusuna düşmeden spam filtresi e-postanın ne kadar olası spam olduğunu belirlemek için istatistiksel analizleri ve Decision-Trees, SVM gibi algoritmaları kullanır.
- Olasılık yüksek ise spam olarak etiketler ve gelen kutusuna ulaşmaz.

Tüm modelleri test ederek en yüksek doğrulukta algoritmayı kullanırız.

• Makine öğrenmesi modelinde Bias ve Variance nedir, aralarındaki trade-off nedir?

Cevap: Bias (yanlılık) modelleme sonucunda tahmin edilen veriler ile gerçek veriler arasındaki uzaklığı yansıtan değer, varyans ise belirli bir veri noktası için değişkenliği verilerin nasıl yayıldığını bize gösteren değerdir.

Yüksek Bias Düşük Varyans ⟶ Tutarlı ancak ortalama olarak yanlış model

Yüksek Varyans Düşük Bias ⟶ Doğru ancak tutarsız model

2) Problem Çözme ve Kodlama

A* için Brige&Torch Problem

2.a Problemin Tanımı

Köprüden karşıya minimum maliyetle geçmeye dayanmaktadır. Her bir kişi farklı hızlara sahip ve her geçişte maximum 2 kişi ile 1 torch ile geçmek zorundadır.

2.b Programın Özellikleri, Yöntemi Ve Orjinallikleri

Özellikleri: Probleme özel olarak her bir karşıya geçiş için tekrar tekrar en hızlı birim için soldan sağa ve sağdan sola işlemini gerçekleştirmiyoruz.Sadece en yavaş birim geçişi sağlar.

Yöntem Ve Orjinallikler: Her bir geçişte maximum 2 kişi geçiceği ve 1 kişi (mümkün olduğunca en hızlı olan) geri döneceği için her bir hareket başına reel olarak en yavaş kişiyi karşıya geçirip, 2 kişinin toplam maliyetini tutmayı gerçekleştirdim. İlgili kod satırında açıklamalar gösterilmiştir.

2.c Programın Ekran Görüntüleri

```
PS C:\Users\mert0\OneDrive\Masaüstü\Notes\Ege\AI> python '.\Bridge&Torch.py'
Middle Cost: 0 - 4 - 2
Middle Cost: 6 - 5 - 2
Middle Cost: 13 - 5 - 2
Middle Cost: 20 - 7 - 2
Middle Cost: 29 - 9 - 2
Middle Cost: 40 - 14 - 2
Middle Cost: 56 - 15 - 2
Middle Cost: 73 - 17 - 2
Middle Cost: 92 - 19 - 2
Middle Cost: 113 - 24 - 2
Middle Cost: 139 - 25 - 2
Middle Cost: 166 - 27 - 2
Middle Cost: 195 - 29 - 2
Middle Cost: 226 - 35 - 2
Middle Cost: 263 - 45
---------------------
Total Cost: 308
---------------------
0 : Left2Right: Mert & Begüm Cost: 4
1 : Right2Left: Mert Cost: 2
2 : Left2Right: Mert & Cem Cost: 5
3 : Right2Left: Mert Cost: 2
4 : Left2Right: Mert & Serpil Cost: 5
5 : Right2Left: Mert Cost: 2
6 : Left2Right: Mert & Deniz Cost: 7
7 : Right2Left: Mert Cost: 2
8 : Left2Right: Mert & Burak Cost: 9
9 : Right2Left: Mert Cost: 2
10 : Left2Right: Mert & Begüm Cost: 14
11 : Right2Left: Mert Cost: 2
12 : Left2Right: Mert & Cem Cost: 15
13 : Right2Left: Mert Cost: 2
14 : Left2Right: Mert & Deniz Cost: 17
15 : Right2Left: Mert Cost: 2
16 : Left2Right: Mert & Burak Cost: 19
17 : Right2Left: Mert Cost: 2
18 : Left2Right: Mert & Begüm Cost: 24
19 : Right2Left: Mert Cost: 2
20 : Left2Right: Mert & Serpil Cost: 25
21 : Right2Left: Mert Cost: 2
22 : Left2Right: Mert & Deniz Cost: 27
23 : Right2Left: Mert Cost: 2
24 : Left2Right: Mert & Burak Cost: 29
25 : Right2Left: Mert Cost: 2
26 : Left2Right: Mert & Cem Cost: 35
27 : Right2Left: Mert Cost: 2
28 : Left2Right: Mert & Serpil Cost: 45
```

2.d Çözüm Başarısı/Çözüm Adımları

Bir node listesinde bulunan kişiler içinden minimum olarak karşıya geçicek ikiliyi arayarak problemi çözmeye başladım daha sonra bu iki kişi karşıya geçirerek en yavaş kişiyi karşıda bırakıp en hızlı kişiyi geriye dönmesini sağladım(Torch ile). f+g olarak belirlenen fonksiyon ise f 2 kişinin karşıya geçme maliyeti iken g en hızlı kişinin geri dönme maliyeti olarak belirlenmektedir. Bu şekilde tüm köprüyü karşıya geçirmiş olurum.

Github Linki : https://github.com/mertakcay/AI/blob/master/Bridge%26Torch.py

3) Genetik Algoritmalar ile Şifre Kırma

3.a)İlgili Maddede İstenenler ve Karşılaştırma (kromozom sayısının etkisi)

popülasyon sayısı: 16

1. deneme: 1914 jenerasyon, 130 milisaniye, "Deep Learning 2022"

```
Generations: 1914
Time taken: 130 ms
```

2. deneme: 930 jenerasyon, 112 milisaniye, "Deep Learning 2022"

```
Generations: 936
Time taken: 112 ms
```

3. deneme: 1876 jenerasyon, 127 milisaniye, "Deep Learning 2022"

```
Generations: 1876
Time taken: 127 ms
```

ortalama: 1573 jenerasyon, 123 milisaniye

popülasyon sayısı: 160

1. deneme: 147 jenerasyon, 118 milisaniye, "Deep Learning 2022"

```
Generations: 147
Time taken: 118 ms
```

2. deneme: 296 jenerasyon, 172 milisaniye, "Deep Learning 2022"

```
Generations: 296
Time taken: 172 ms
```

3. deneme: 191 jenerasyon, 164 milisaniye, "Deep Learning 2022"

```
Generations: 191
Time taken: 164 ms
```

ortalama: 211 jenerasyon, 151 milisaniye

popülasyon sayısı: 1600

1. deneme: 22 jenerasyon, 170 milisaniye, "Deep Learning 2022"

```
Generations: 22
Time taken: 170 ms
```

2. deneme: 34 jenerasyon, 207 milisaniye, "Deep Learning 2022"

```
Generations: 34
Time taken: 207 ms
```

3. deneme: 37 jenerasyon, 236 milisaniye, "Deep Learning 2022"

```
Generations: 37
Time taken: 236 ms
```

ortalama: 31 jenerasyon, 204 milisaniye

Popülasyon sayısı arttıkça jenerasyon sayısının azaldığı ve geçen sürenin arttığı gözlendi.

3.b)Çaprazlama ve Mutasyon Fonksiyonlarının Anlatımı

Çaprazlama:
Girdi olarak iki parent kromozomu alır, rastgele başlangıç ve bitiş noktaları belirler ve bu noktaları kullanarak parent kromozomları parçalar, sonrasında bu parçaları birleştirerek child kromozomu oluşturur.

Mutasyon:
Popülasyondaki her bir child kromozomun üzerinden geçer ve yüzde on ihtimalle seçtiği child kromozomlarının rastgele seçilen konumlarındaki karakteri rastgele bir karakterle değiştirir.

Seçilim:

Child yaratma fonksiyonu içerisinde, ilk önce elitism metodu ile seçilen belirli sayıda kromozom child listesine eklenir. Eklenen kromozomlar skoru en yüksek olanlardır.

3.c)Çözüm Süreleri Karşılaştırması

popülasyon sayısı: 16

1. deneme: 1026 jenerasyon, 82 milisaniye, "DeepLearning"

```
Generations: 1026
Time taken: 82 ms
```

2. deneme: 1600 jenerasyon, 118 milisaniye, "DeepLearning"

```
Generations: 1600
Time taken: 118 ms
```

3. deneme: 1155 jenerasyon, 110 milisaniye, "DeepLearning"

```
Generations: 1155
Time taken: 110 ms
```

ortalama: 1399 jenerasyon, 103 milisaniye
A ile fark: -174 jenerasyon, -20 milisaniye

popülasyon sayısı: 160

1. deneme: 75 jenerasyon, 67 milisaniye, "DeepLearning"

```
Generations: 75
Time taken: 67 ms
```

2. deneme: 133 jenerasyon, 102 milisaniye, "DeepLearning"

```
Generations: 133
Time taken: 102 ms
```

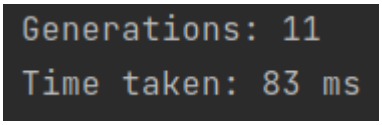
3. deneme: 124 jenerasyon, 114 milisaniye, "DeepLearning"

```
Generations: 124
Time taken: 114 ms
```

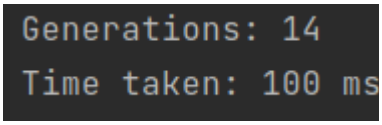
ortalama: 110 jenerasyon, 94 milisaniye
A ile fark: -101 jenerasyon, -57 milisaniye

popülasyon sayısı: 1600

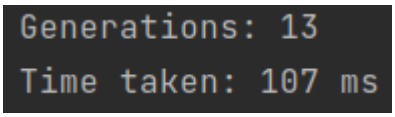
1. deneme: 11 jenerasyon, 83 milisaniye, "DeepLearning"



2. deneme: 14 jenerasyon, 100 milisaniye, "DeepLearning"



3. deneme: 13 jenerasyon, 107 milisaniye, "DeepLearning"



ortalama: 12 jenerasyon, 96 milisaniye
A ile fark: -19 jenerasyon, -108 milisaniye

4) Makine Öğrenmesi

4.a) Python ile makine öğrenmesi konusunu Ders Notlarından inceleyiniz. Normalizasyon konusunu araştırınız, hangi durumlarda gerektiğini ve nasıl kullanıldığını öğreniniz.

Normalizasyon elimizdeki verinin istenilen aralığa ve istenilen ortalamaya çekilme işlemidir. 0 Ortalama ve 0-1 aralığına çekmemiz ise standartizasyon olarak söyleyebiliriz. Kullanım amacı ise Verimizde bazı aralıklar ve ortalama daha büyük olduğunda diğer küçük aralıklara baskın olup öğrenmeyi zorlaştırmakta hatta imkansız hale getirmektedir. Algoritma üzerinden örnek vermem gerekirse KNN de bildiğiniz gibi uzaklık üzerinden öğrenme sağlanmaktadır X^1 kümesinin range aralığı 100 birim iken X^2 nin uzaklığı 0.1 birim olduğunu düşünürsek öğrenme baskınlığı X^1 üzerinden gerçekleşirken X^2 nin baskınlığı düşük olur. Hangi algoritmalarada kullanılır noktasında ise KNN, öklid uzaklığı gibi uzaklık metrikleri üzerinden gerçekleşen öğrenme ve/veya boyut düşürme algoritmalarında kullanılmaktadır.

4.b İki Farklı Sınıflandırıcı için Sonuçlar: Hata Matrisleri, Tablo

4.b.1) Öğrenme İşlemini gerçekleştirebileceğiniz derste anlatılandan (Zambak) farklı bir veriseti bulunuz (veya kendiniz oluşturunuz)

https://www.kaggle.com/datasets/yasserh/wine-quality-dataset dataset adresi.

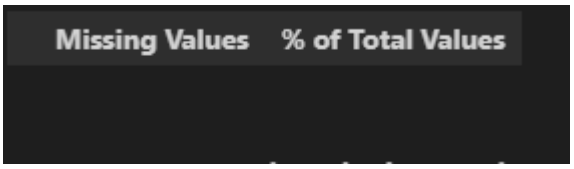
4.b.2) Verisetini inceleyip özet bilgileri rapora yazınız: Veri (örnek) sayısını, öznitelik (girdi) sayısını, özniteliklerin neler olduğunu, sınıf sayısını ve sınıfların neler olduğunu rapora yazınız.

11 Farklı feature ve 10 farklı label'a sahip olunan bir datasete sahibiz.

Data Type

#	Column	Non-Null Count	Dtype
0	fixed acidity	1143 non-null	float64
1	volatile acidity	1143 non-null	float64
2	citric acid	1143 non-null	float64
3	residual sugar	1143 non-null	float64
4	chlorides	1143 non-null	float64
5	free sulfur dioxide	1143 non-null	float64
6	total sulfur dioxide	1143 non-null	float64
7	density	1143 non-null	float64
8	pH	1143 non-null	float64
9	sulphates	1143 non-null	float64
10	alcohol	1143 non-null	float64
11	quality	1143 non-null	int64

Missing Value Yüzdeleri



Class Yüzdeleri

```
Quality: 3, Count: 6, Percent of Dataset: 0.005249343832020997
Quality: 4, Count: 33, Percent of Dataset: 0.028871391076115485
Quality: 5, Count: 483, Percent of Dataset: 0.4225721784776903
Quality: 6, Count: 462, Percent of Dataset: 0.4041994750656168
Quality: 7, Count: 143, Percent of Dataset: 0.12510936132983377
Quality: 8, Count: 16, Percent of Dataset: 0.01399825021872266
```

Temel İstatistikler

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.914698	0.996730	3.311015	0.657708	10.442111	5.657043
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.782130	0.001925	0.156664	0.170399	1.082196	0.805824
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000000	0.995570	3.205000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000000	0.996680	3.310000	0.620000	10.200000	6.000000
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000000	0.997845	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

4.b.3) İki farklı sınıflandırıcı (MLP Classifier YSA, SVM, k-NN, Decision Tree, Random Forest …) kullanarak sınıflandırma işlemini yapan Python kodunu yazınız.

Github adresinde tüm kodlar paylaşılmıştır.Ek olarak rarda kodlar bulunmaktadır.

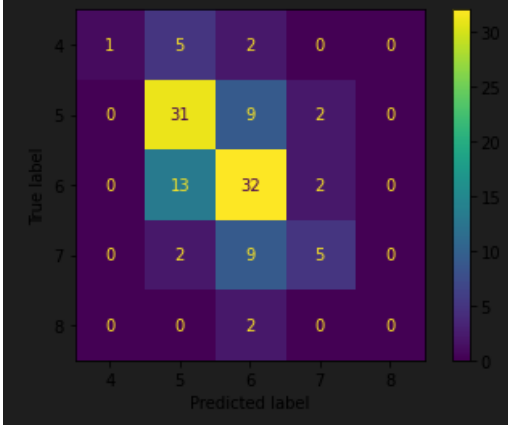
Link:https://github.com/mertakcay/AI/blob/master/HomeworkML.ipynb

4.b.4)Hata (Confusion) matrislerini elde ederek rapora ekleyiniz. Sınıflandırıcıların başarılarını Accuracy, Precision ve Recall cinsinden ölçerek bir tabloya kaydediniz fold cross validation kullanınız. Sonuçları yorumlayınız.

XGBoost Oversample Olmadan Çıktıları

Accuracy: 60.00%

	precision	recall	f1-score	support
4	1.00	0.12	0.22	8
5	0.61	0.74	0.67	42
6	0.59	0.68	0.63	47
7	0.56	0.31	0.40	16
8	0.00	0.00	0.00	2
accuracy			0.60	115
macro avg	0.55	0.37	0.38	115
weighted avg	0.61	0.60	0.57	115



K-Fold Acc Değerleri

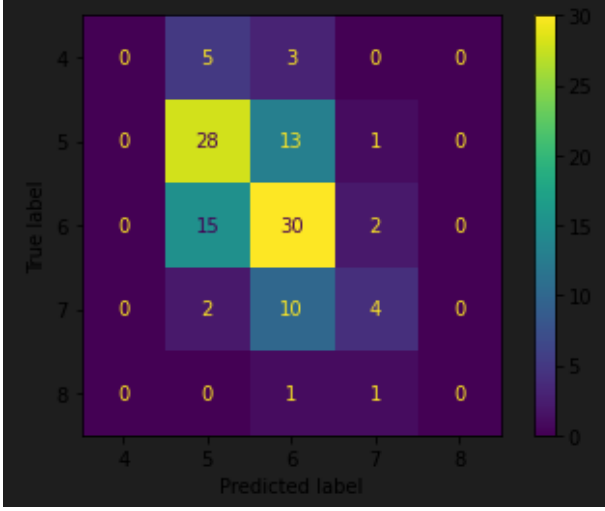
```
Scores: [0.7184466  0.66990291 0.70873786 0.59223301 0.5631068  0.59223301
 0.61165049 0.61165049 0.67647059 0.70588235]
```

Catboost Oversample Olmadan Çıktıları

```
Accuracy: 53.91%
              precision    recall  f1-score   support

           4       1.00      0.12      0.22         8
           5       0.61      0.74      0.67        42
           6       0.59      0.68      0.63        47
           7       0.56      0.31      0.40        16
           8       0.00      0.00      0.00         2

    accuracy                           0.60       115
   macro avg       0.55      0.37      0.38       115
weighted avg       0.61      0.60      0.57       115
```

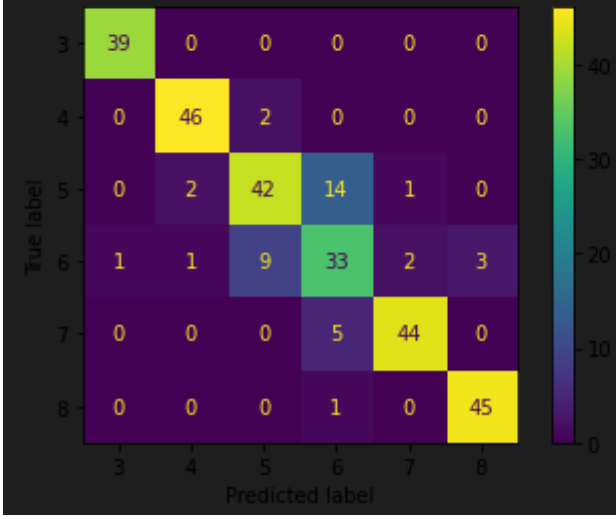


K-Fold Acc Değeleri

```
Scores: [0.62135922 0.55339806 0.59223301 0.54368932 0.50485437 0.59223301
 0.62135922 0.58252427 0.62745098 0.62745098]
```

XGBoost Oversample Çıktıları

	precision	recall	f1-score	support
3	0.97	1.00	0.99	39
4	0.94	0.96	0.95	48
5	0.79	0.71	0.75	59
6	0.62	0.67	0.65	49
7	0.94	0.90	0.92	49
8	0.94	0.98	0.96	46
accuracy			0.86	290
macro avg	0.87	0.87	0.87	290
weighted avg	0.86	0.86	0.86	290

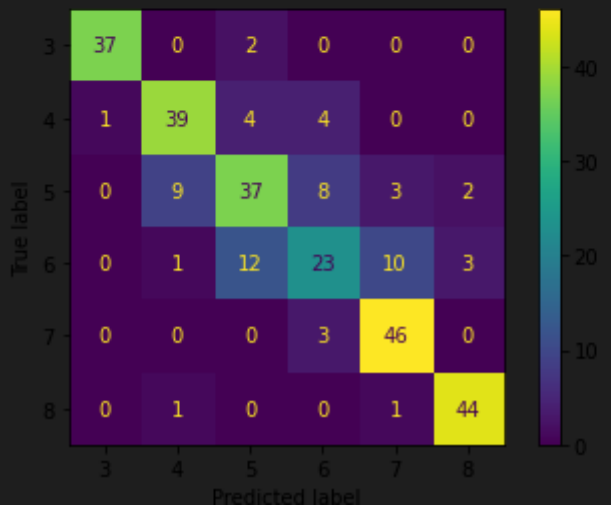


K-Fold Acc Değeleri

```
Scores: [0.87739464 0.85057471 0.86590038 0.83141762 0.85440613 0.87356322
 0.85823755 0.85823755 0.86538462 0.86923077]
```

Decision Tree Oversample Çıktıları

	precision	recall	f1-score	support
3	0.97	0.95	0.96	39
4	0.78	0.81	0.80	48
5	0.67	0.63	0.65	59
6	0.61	0.47	0.53	49
7	0.77	0.94	0.84	49
8	0.90	0.96	0.93	46
accuracy			0.78	290
macro avg	0.78	0.79	0.78	290
weighted avg	0.77	0.78	0.77	290



K-fold Acc Değerleri

```
Scores: [0.77777778 0.77394636 0.78544061 0.80842912 0.78544061 0.74712644
 0.79310345 0.79693487 0.76923077 0.78461538]
```

4.b.4) Kullanıcının girdiği yeni (sınıfı bilinmeyen) bir örneğe ilişkin özniteliklerin yani örüntünün (pattern) hangi sınıfa ait olduğunu buldurunuz. Test veri seti eğitimden ve validasyon verilerinden ayrı tutulduğu için tüm acc ve precision recall değerleri alınmıştır.

5) Öz değerlendirme Tablosu

Tabu search te çok farklı implementasyonlar ve problem üzerinden yaklaşımlar olduğu için time complexity' si boş bırakılmıştır.

Tüm kodlar

	İstenen Özellik	Var	Açıklama	Tahmini Not
1a	Algoritmalar + Karmaşıklıklar (10)	X	Tabu Search'te farklı yaklaşımlar bulunduğu için net bir time complexity yazamadık.	9
1b	Tanım ve Karşılaştırmalar (10)	X	Tanımları hepimiz araştırdıktan sonra discord üzerinden birbirimize aktardık ve deniz kağıda geçirdi bundan dolayı atıf ve kaynak noktasında çok dağınık çalıştığımız için bu kısımda eksikliğimiz bulunuyor.	9
1c	Araştırma ve Yorum (10)	X		10
2	Problem Çözme ve Kodlama (10)	X		10
3	Genetik Algoritmalar ile Şifre Kırma (15)	X		15
4	Makine Öğrenmesi (25)	X	Kod içinde bazı feature selection methodlarını denedik fakat ödev kapsamında istenmediği için model eğitimde ekleme gereği duymadık. Ayrıca elimizdeki veri oldukça küçük ve imbalance olduğu için oversample yapmak zorunda kaldık.	25
	Rapor (20)	X	Open Source editörden dolayı bazı bozukluklar oluşuyor.	18
			100 üzerinden Toplam Not:	96