



EGE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAPAY ZEKÂ YÖNTEMLERİ (3+0)
2022-2023 BAHAR YARIYILI

PROJE-1 RAPORU

TESLİM TARİHİ

04/05/2023

HAZIRLAYANLAR

05190000114 – Mahmut Çelik

05190000027 – Özgür Bayraşa

05220000011 – Sedat Korkmaz

İçindekiler

1) 1) Algoritmalar, Tanımlar, Karşılaştırma, Araştırma ve Yorum.....	2
1.a Kalemle yazılan algoritmaların tarayıcı görüntüleri veya fotoğraflar	2
1.b Tanım ve Karşılaştırmalar	6
1.c Araştırma ve Yorum	10
2) Problem Çözme ve Kodlama	10
2.a Problemin Tanımı	10
2.b Çözüm Mekanizması ve Kaynak Kod	11
2.c Programın Ekran Görüntüleri.....	15
2.d Sonuç Tablosu.....	16
3) Genetik Algoritmalar ile Şifre Kırma.....	16
3.a İlgili Maddede İstenenler ve Karşılaştırma (kromozom sayısının etkisi)	16
3.b Kod; Çaprazlama ve Mutasyon Fonksiyonlarının Anlatımı	17
3.c Çözüm Süreleri Karşılaştırması	18
4) Makine Öğrenmesi	18
4.a Standardization ve Normalization Farkı ve Python Örneği	18
4.b Veri Setinin ve Problemin Kısa Anlatımı	19
4.c İki Farklı Sınıflandırıcı için Python Kodu	20
4.d İki Farklı Sınıflandırıcı için Sonuçlar: Hata Matrisleri, Tablo	21
4.e Kullanıcı tarafından verilen örneğin sınıflandırma ekran görüntüsü (konsol çıktıları)	22
4.f ChatGPT Kullanımı ve Yorumlama.....	24
5) Öz değerlendirme Tablosu	26

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

1.a Kalemle yazılan algoritmaların tarayıcı görüntüleri veya fotoğraflar

1-a

★ Tabu Search Algoritması

Mevcut çözüm, iterasyon sayısı ve tabu (yasak) listesi büyüklüğü girilir.

⊗ Tabu Search (Initial Solution, Iterations, Tabu-size)

En iyi aday ve en iyi çözüm, başlangıçta girilen çözümdür.

Bu çözüm random belirlenebilir.

best-solution = initial-solution

best-candidate = ~~initial-solution~~ initial-solution

Tabu (Yasak) listesi oluşturulur ve mevcut çözüm eklenir.

tabu-list = []

tabu-list.append(initial-solution)

~~while~~

iterasyon takibi için gereklidir.

iteration = 0

iterasyon sayısı kadar döngü

while (iteration < iterations)

En iyi aday için komşular bulunur. Komşular listesi alınır.

solution-neighborhood = get-neighbors(best-candidate)

Varsayılan en iyi aday olarak ilk komşu atanır.

best-candidate = solution-neighborhood[0]

Komşular dolaşarak en iyi komşu bulunur.

for (candidate in solution-neighborhood)

Bir komşu, en iyi aday komşudan daha uygunsa (fitness)

Ve yasak (tabu) listesinde değilse en iyi komşu olur.

if ((fitness(candidate) > fitness(best-candidate)) and
(candidate not in tabu-list))

best-candidate = candidate

En iyi komşu, mevcut çözümden daha uygunsa en iyi çözüm olur.

if (fitness(best-candidate) > fitness(best-solution))

best-solution = best-candidate

~~while~~

En iyi aday tabu (yasak) listeye eklenir.

tabu-list.append(best-candidate)

Listenin boyutu artırılmışsa ilk eleman (en eski olan) listeden çıkarılır.

if (tabu-list.size() > tabu-size)

tabu-list.remove(First)

İterasyonlar sonunda en iyi çözüm döndürülür.

return best-solution

★ Görüldüğü üzere 1 while ve 1 adet if for döngüsü vardır.

52 Bu yüzden Tabu Search için zaman karmaşıklığı $O(n^2)$ olarak ifade edilebilir.

1-a

★ A* Algoritması

★ # Algoritmada başlangıç ve hedef noktalar kullanılır.

A-Star (start, goal)

Başlangıç ^(düğümü) noktasının f değeri, başlangıç noktası olduğundan 0 olarak atanır.
start.f = 0

Açık (open) ve kapalı (closed) kümeler tanımlanır.

Açık (open) küme → Oluşturulmuş ancak henüz incelenmemiş düğümler

Kapalı (closed) küme → incelenmiş kümeler

openSet = { start } # Açık kümeye başlangıç düğümü eklenir.

closedSet = {}

Döngü, düğümlerin hepsi inceleninceye kadar, açık kümede eleman kalıncaya kadar
devam eder.


while (openSet.length > 0)

En düşük f değerine sahip düğüm, açık kümeden alınır.

currentNode = openSet.get(Node.LowestF())

Eğer mevcut düğüm hedeflenene aynıysa, başlangıca doğru yol yapı ve döndür.


if (current == goal)

return PathToBeginning(current) #  Başlangıca doğru, mevcut düğümü olan yol döner.

incelenerek olan düğüm açık kümenin kapalı kümeye alınır.

openSet.remove(current)

closedSet.add(current)

incelenerek olan düğümün komşularını (komşu düğümler)  alınır.

neighbors = current.neighbors

Komşular içinde g değeri en az olanı bulmak için döngü oluşturulur.

for neighbor in neighbors

Komşu zaten incelenmişse yani kapalı kümedeyse tekrar incelemeye gerek yok.

if neighbor in closedSet

continue

Bu komşu için g değerini hesaplanır.

temp_g = current.g + getDistance(current, neighbor)

Bu komşu oluşturulmuşsa (kayıldıysa) açık kümeye eklenir.

if (neighbor not in openSet)

openSet.add(neighbor)

Hesaplanan geçici g değeri, komşudan ulaşılabilecek olan g değerinden

daha büyükse diğer hesaplama gerek olmaz. Zaten daha maliyetli

olacaktır.

else if (temp_g > neighbor.g)

continue

Bu aşamada bulunan g değerinin, yeni maliyetin daha az olduğunu anlarız.

Gerekli hesaplamaları yapabiliriz.

neighbor.parent = current

neighbor.g = temp_g

neighbor.h = get_h(neighbor, goal)

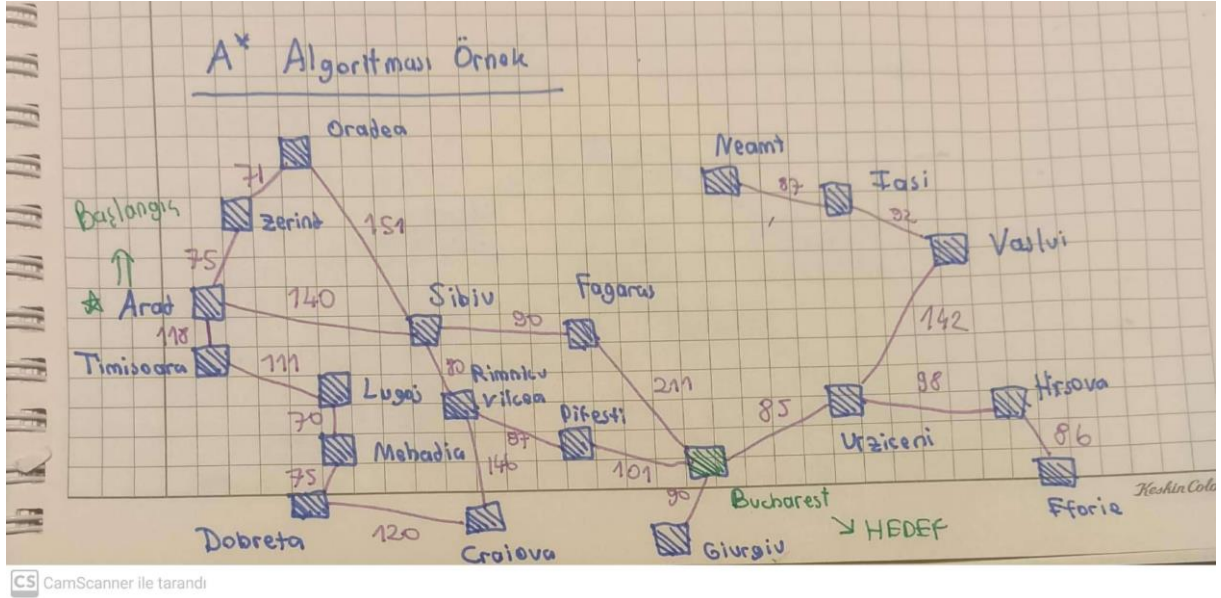
neighbor.f = neighbor.g + neighbor.h

Null değer dönerse, hedeflenen düğümü bir yol bulunamamıştır.

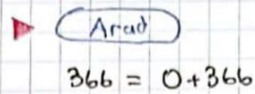
return null

Kaşkın Çelen

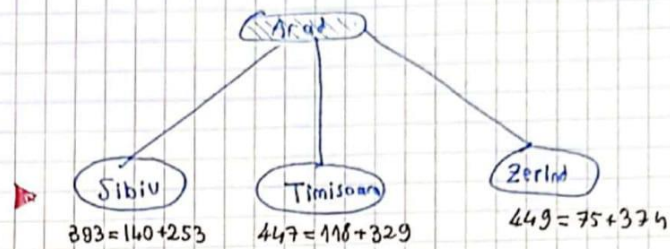
★ b : Ortalama dallanma (göbek) sayısı, d : Çözüm ağacı derinliği olmak üzere A^* Algoritması için zaman karmaşıklığı $O(b^d)$ olarak bulunur.



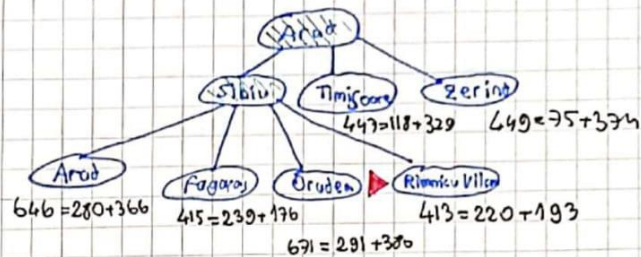
ADIM 1



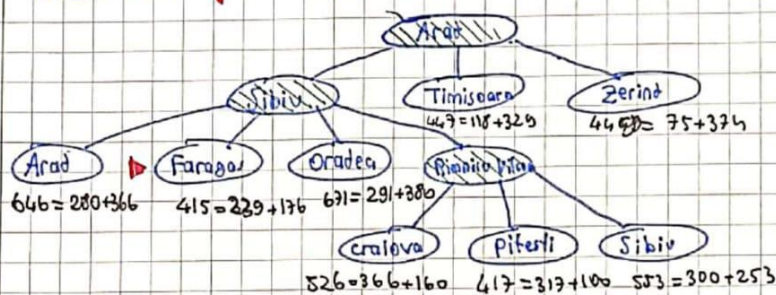
ADIM 2



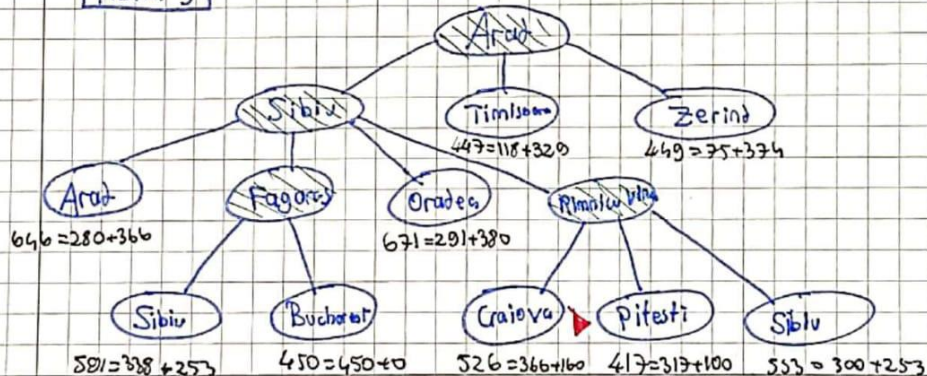
ADIM 3



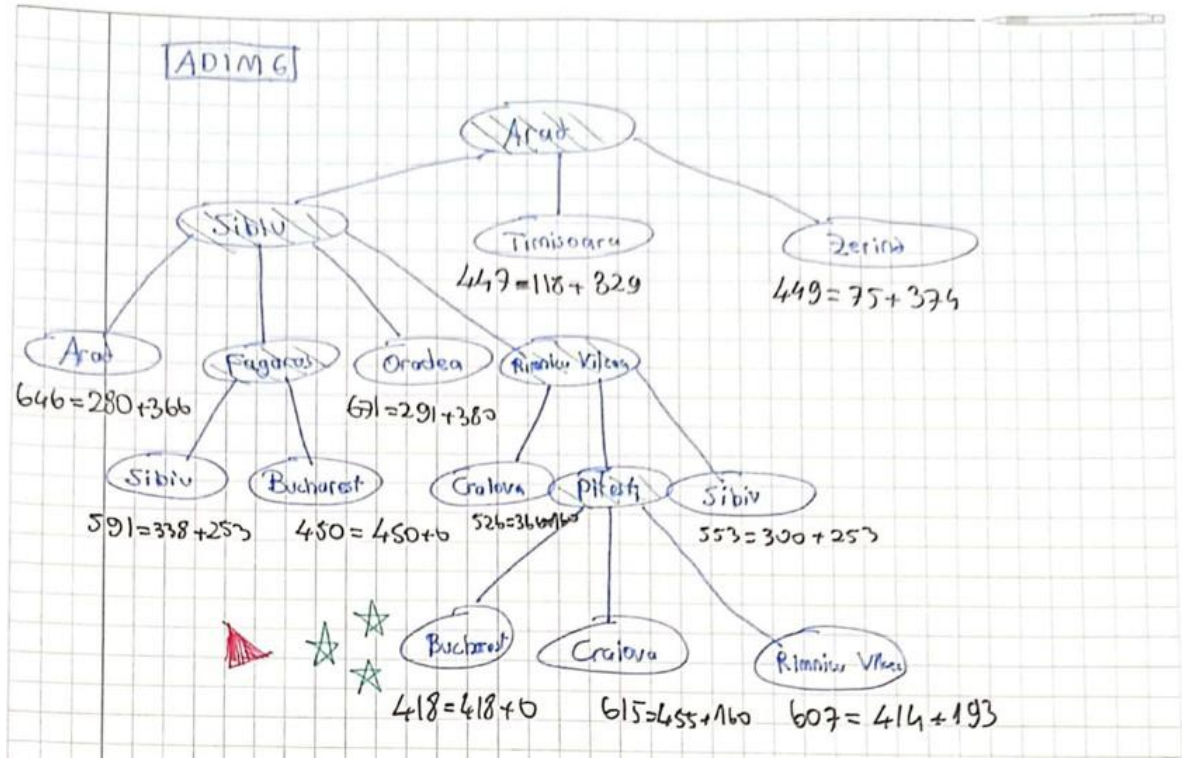
ADIM 4



ADIM 5



Kashin Color



1.b Tanım ve Karşılaştırmalar

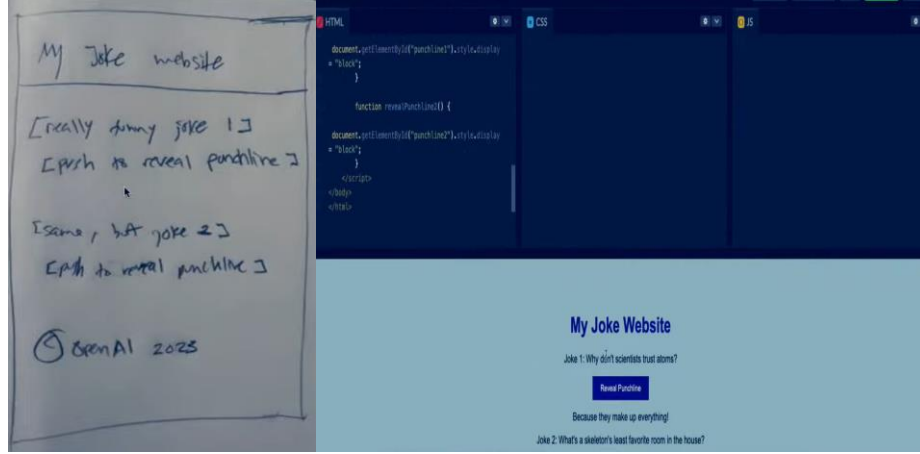
GPT-4

GPT (Generative Pre-trained Transformer), derin öğrenme tekniğinin kullanıldığı büyük bir dil modelidir. Bu model soruları yanıtlama, metni özetleme, kod satırları oluşturma gibi dil işleme görevlerini gerçekleştirebilmektedir.

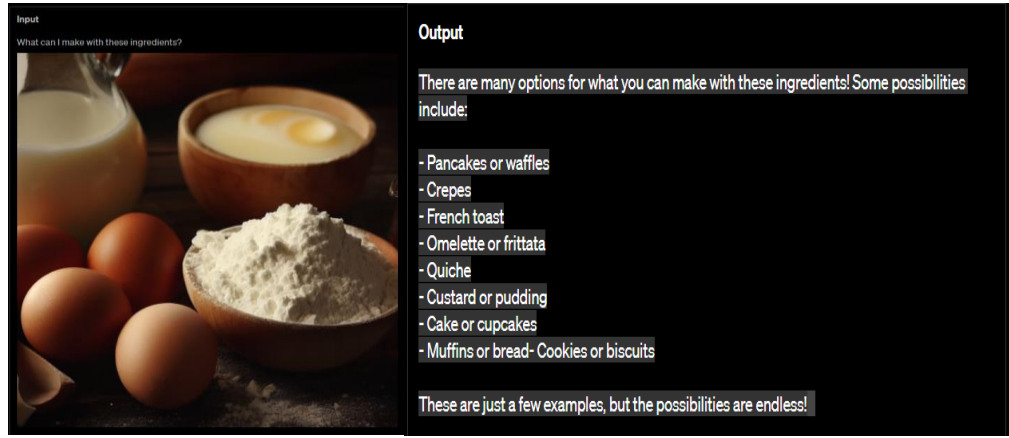
GPT-3 2020 yılında piyasaya sürülen, 175 milyar parametre üzerinde eğitilerek o dönemin en büyük sinir ağı haline gelmişti. Yakın zamanda ise GPT-4 tanıtılmıştır.

GPT-4'ün yeni özellikleri ise çığır açıcı olmuştur. Bunlardan birkaç örnek verebiliriz.

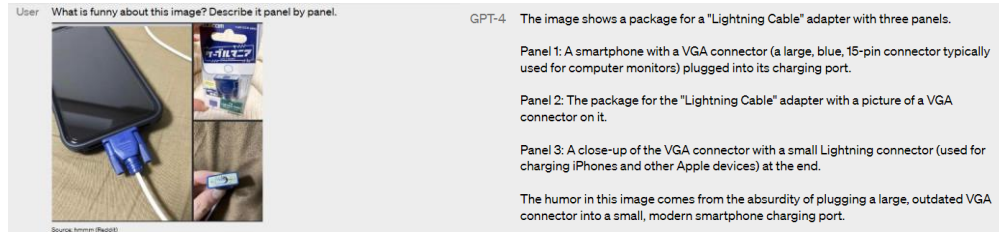
- Metinsel Girdinin Yerine Görsel Girdi Kabul Edilmesi
 - o Bir kullanıcının kağıda çizdiği web sitesi tasarımının kodlarını çıktı olarak verebilir.



- Bir kullanıcının girdi olarak yemek malzemeleri olan bir görseli koyması ve hangi yemekleri yapabileceğini sorması üzerine yemek tarifi verebilir.

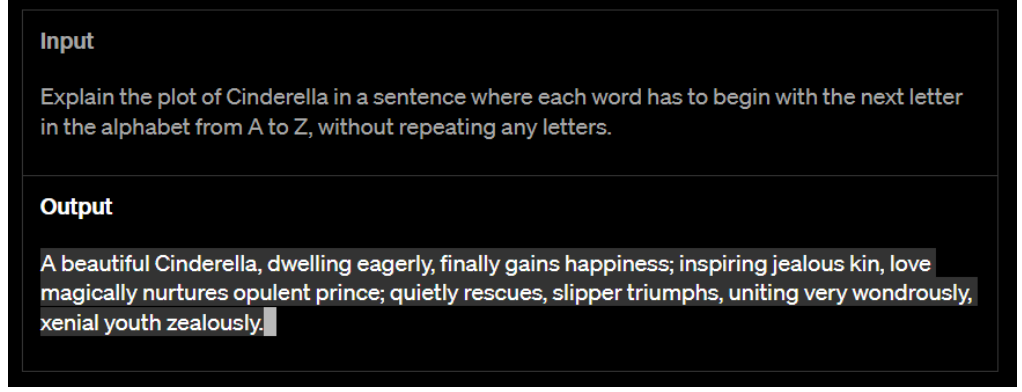


- Bir görsel üzerindeki espriyi anlatabilir.



- Yaratıcı Cevaplar Üretebilmesi

- Sindirella'nın olay örgüsünü, sırasıyla alfabedeki harfleri baştan sona sadece bir kez kullanarak anlatabilir.



- Daha Uzun Metinleri İşleyebilmesi
 - o GPT-4 25.000 kelimeye kadar olan metinleri işleyebilir. Kullanıcının dokümanı analiz etmesini, özet çıkartmasını kolaylaştırır.
- Güvenlik ve Performans Olarak Gelişmiş Olması

AutoML

AutoML, Machine Learning (Makine Öğrenimi) modellerinin tasarımı ve oluşturulması için insan müdahalesini azaltmak ve ortadan kaldırmak için kullanılan bir yapay zeka teknolojisidir. Verilen gören için en optimal olan makine öğrenimi algoritmasının kullanılmasını sağlar.

AutoML'in başlıca faydaları şunlardır;

- **Verimlilik:** Makine öğrenimi süreci basitleştirilir ve hızlandırılır. Makine öğrenimi modellerinin eğitim süresini azaltır.
- **Maliyet:** Firmalar daha verimli bir makine öğrenimi süreci geçireceği için maliyetten tasarruf ederler.
- **Performans:** AutoML algoritmaları, insan eliyle yazılan kodlardan daha iyi performans verebilirler.

Caption Generation

Bir görüntü için doğal dil cümleleri oluşturmak için kullanılan bir makine öğrenimi tekniğidir. **Görüntü tanıma (Image Recognition) ve doğal dil işleme (Natural Language Processing)** tekniklerinin birleşiminden oluşur.

Caption generation teknolojisi birçok farklı alanda kullanılabilir;

- **Görsel Arama Motorları:** Daha doğru bir şekilde arama sonuçları oluşturulabilir.
- **Sosyal Medya:** Kullanıcıların paylaştığı görüntüler için otomatik olarak açıklama veya hashtag eklenebilir.

- **Sağlık Hizmetleri:** Tıbbi görüntülerin analiz edilmesinde ve raporlanmasında kullanılabilir. Hastalıkların teşhisinde yardımcı olabilir.
- **E-Ticaret:** E-Ticaret sitelerinde, ürünlerin resimlerine otomatik olarak açıklama eklenmesinde kullanılabilir. Müşteriler böylelikle ürünleri daha iyi anlayabilirler.

Self-Attention GAN

Self-Attention GAN (SAGAN) öncesinde GAN'ın tanımına bakmak daha doğru olacaktır.

GAN (Generative Adversarial Network), derin öğrenme alanında kullanılan bir yapay sinir ağı modelidir. Bu modelde iki ayrı sinir ağı bulunur.

1. **Üretici Ağ (Generator):** Öğrenme verilerinden örnekler alarak **yeni veriler üretir**.
2. **Ayırt Edici Ağ (Discriminator):** Üretilen verilerin **gerçek veri mi yoksa üretilmiş veri mi** olduğunu ayırt eder. Bu sınıflandırma üretici ağına daha gerçekçi veriler üretmesini sağlar.

İki ağ birbirleriyle işbirliği yaparak eğitim verilerine benzer ancak onlarla aynı olmayan yeni veriler üretirler.

Self-Attention GAN, daha fazla dikkat mekanizması kullanarak **daha yüksek kaliteli görüntü üretmek amacıyla** geliştirilmiş bir GAN çeşididir.

Self-Attention mekanizması, farklı piksellerin birbirleriyle olan ilişkilerini değerlendirir. Pikseller arasındaki bağımlılıkları hesaplar. Dikkat verilmesi gereken bölgeleri belirler ve GAN'ların daha tutarlı ve gerçekçi sonuçlar üretmesine yardımcı olur. **GAN'ların** eğitim sürecinde, üretim kalitesi artırılır.

Self-attention GAN, genellikle yüksek çözünürlüklü görüntülerin üretilmesinde kullanılır.

Ensemble Learning vs. Random Forest

Ensemble Learning, birçok farklı modelin bir araya getirilmesi ile daha yüksek performans elde edilmesini sağlayan bir makine öğrenimi (machine learning) tekniğidir. Farklı özelliklere ve hiperparametrelere sahip farklı öğrenme algoritmaları bir araya getirilir. Bu algoritmalar **farklı veri kümelerine** uygulanır. Sonuçlar daha sonra birleştirilir. Ensemble learning, birden fazla modelin **bir arada çalışmasını sağlayarak tek bir modele göre daha yüksek bir doğruluk oranı elde edilmesine** olanak sağlar.

Random Forest ise bir Ensemble Learning yöntemidir. Random Forest için Ensemble Learning'in bir alt kümesi denebilir. Sınıflandırma, regresyon ve diğer öğrenme görevlerinde kullanılır. Birden çok karar ağacının bir araya getirilmesiyle **daha yüksek doğruluk oranları elde etmek** temel amaçtır.

1.c Araştırma ve Yorum

3 - Makine Öğrenmesi Alanında 3 Adet Mülakat Sorusu

1.) Overfitting nedir ve nasıl engellenebilir?

⇒ Overfitting, makine öğrenmesi modelinde eğitim verilerinin modele aşırı uyum sağlama ile ortaya çıkan bir problemdir. Modelin çok karmaşık olması, yeterli eğitim verisinin olmaması overfitting durumuna yol açabilir.

★ Daha fazla veri toplamak, overfitting problemini çözebilir.

★ Modelin karmaşıklığı azaltılarak overfitting önlenir.

★ Bazı nöronların rastgele seçilmesini sağlamak, modelin farklı özellikler öğrenmesine, bunun sonucunda overfitting'in önlenmesine yardımcı olabilir.

2.) False Positive ve False Negative kavramlarını örneklerle anlatabilir misin?

⇒ False Positive, gerçekte var olmayan bir örneğin var olarak sınıflandırılmasıdır. Dolayısıyla yanlış bir sınıflandırmadır.

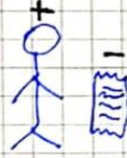
★ Bir virüse sahip ~~bir~~ olmayan bireyin, test sonucunda virüse sahip olduğunun sapıtılması False Positive örneğidir. Teste göre virüs vardır ama gerçek durumda virüs yoktur.

⇒ False Negative de yine yanlış sonuç üretmekle ortaya çıkar. Bu sefer hata olan bir değer olmaması (yokmuş) gibi gösterilmektedir.

★ Virüslü olmayan birey test sonucunda virüslü çıkarılan sonuç için False Positive denir.



FALSE POSITIVE



FALSE NEGATIVE

3.) Deep Learning ile Machine Learning ile ilgili farkları özetler misin?

★ Deep Learning'in çalıştığı veri kümeleri çok daha büyük ve karmaşıktır. Machine Learning ise daha küçük ve daha az karmaşık veri kümeleriyle çalışır.

★ Deep Learning dolayısıyla daha fazla donanım gücü gerektirir. Bu da daha fazla maliyet demektir.

★ Machine Learning genellikle tek katmanlı veya birkaç katmanlı yapay sinir ağı kullanır. Deep Learning ise çok katmanlı yapay sinir ağı kullanır.

CS CamScanner ile tarandı

2) Problem Çözme ve Kodlama

2.a Problemin Tanımı

Eight Queens problemi, 8 Vezir bulmacası 8*8 satranç tahtası üzerinde 8 vezirin birbirini yemeyecek şekilde konumlandırılmasıdır. (Aynı satır, aynı sütun veya aynı çarpaz sırada iki veya fazla vezir bulunamaz.)

2.b Çözüm Mekanizması ve Kaynak Kod

Çözüm mekanizması olarak, başlangıçta tüm vezirler array de 0 ile 7 arasında rastgele yerleştirilir. Başlangıç tahtasının çözülmüş bir şekilde olup olmadığı kontrol edilir. Kontrol için çapraz yeme ve yatay yeme durumlarına dikkat edilir. Herhangi bir yeme durumu olmuyorsa tahta çözülmüştür. Tahta çözülmemişse mevcut tahtanın neighbor state leri bulunur ve map içerisine yerleştirilir. Yerleştirilirken neighbor state lerdeki yeme durumları da hesaplanarak yerleştirilir. Yerleştirme sonrası en iyi durumdaki neighbor state e geçiş yapılır. Birden fazla en iyi durumda neighbor state varsa bu durumda random olarak birine geçilir. Neighbor state e geçilmeden önce kontrol mekanizması bulunuyor. Bu kontrolde şuanki state le geçilebilecek en iyi neighbor state teki yeme durumları karşılaştırılıyor. Değerde iyileşme olmuyorsa local min e takılmış oluyoruz ve random restart la tahtayı rastgele tekrar dizip baştan başlıyoruz. Değerde iyileşme varsa yeni state e geçilir ve bu şekilde ilerleyerek solution a ulaşırız.

```
package org.example;
public class Main {
    //Çözüm yolu olarak Map kullanmayı tercih ettik. Alternatif olarak
    temporary array de kullanabilirdik.
    //Aybars hocaya sorduğumuzda Map şeklinde kalabileceğini ve size de
    iletmemizi söyledi
    public static void main(String[] args) {
        EightQueens queens = new EightQueens();
        queens.solveNineTimes();
    }
}
```

```
package org.example;

import java.util.*;
import java.util.concurrent.atomic.AtomicInteger;

public class EightQueens {
    private static Integer[] board = new Integer[8]; // Chess Board ->
    Elements are columns and Indexes are rows
    private static final Map<Integer[], Integer> neighborStates = new
    LinkedHashMap<>();
    // Neighbor states of current board -> It holds neighbor state and
    number of eating each other count
    private final Random random = new Random();

    private static final Object[][] solutionTable = new Object[9][4];
    //Keeps our solutions results

    private static int randomRestartCount = 0; //Counts random restart for
    each solution tour

    private static Replacement replacement = new Replacement(new
    ArrayList<>(), 0);
    /*
     * Replacement has two fields
     * 1. field is a list that holds replacement counts for each random
    restart
     * 2. field is an int that holds current replacement count for that
    board
     * */
    /**
```



```

    * It calls solve() nine times and sets values in each loop
    * To calculate time -> System.nanoTime()
    * Finally prints the solutions to the console
    */
    public void solveNineTimes() {

        for (int x = 0; x < 9; x++) {
            double time = 0;
            long startTime = System.nanoTime();

            replacement.setCurrentReplacementCount(0); // Set 0 replacement
            - beacuse new board will start
            replacement.setReplacementCounts(new ArrayList<>()); //Refresh
            the replacement list beacyse new tour will start
            randomRestartCount = 0; // Set 0 random restart - beacuse new
            board will start

            solve();

            long endTime = System.nanoTime();

            time = (endTime - startTime) / 1000000000.0;

            solutionTable[x][0] = replacement.getSumOfReplacements();
            //Fill the table
            solutionTable[x][1] = (double) randomRestartCount;
            solutionTable[x][2] = time;
            solutionTable[x][3] = replacement.getReplacementCounts();

        }
        printSolutionTable();
    }

    /**
     * 1 -> Generate random chess table via randomReplacement() method
     * 2 -> While loop continues until a solution is found
     * 2.1 -> If program encountered a local min, method calls itself again
     * (Current state value -> 5 , neighbor states values are 5 or bigger -
     It means local min)
     * 3 -> If everthing is okay, State will change via changeState()
     * (Current state value -> 5 , neighbor states contain a number that
     lower than 5)
     */
    public void solve() {
        randomPlacement();
        while (numOfEating(board) != 0) { //RANDOM RESTART YAPTIKTAN SONRA
            ONCEKI TURDAKI REPLACEMENT LARI DA SAYMALI MIYIZ?
            replacement.increaseFinalReplacement();
            fillNeighbors();
            if (!neighborStates.values().stream().anyMatch(e -> e <
            numOfEating(board))) {
                replacement.stateChange();
                randomRestartCount++;
                solve();
                break;
            }
            changeState();
        }
    }

    /**

```

```

    * Randomly places queens
    */
    public void randomPlacement() {
        for (int x = 0; x < board.length; x++) {
            board[x] = random.nextInt(0, 8);
        }
    }

    /**
    * It has one parameter and it calculates the parameter's eating count
    * 1 - Cross eating control
    * 2 - Horizontal eating control
    */
    public Integer numOfEating(Integer[] val) {
        int count = 0;
        for (int x = 0; x < val.length - 1; x++) {
            for (int y = x + 1; y < val.length; y++) {
                if ((Math.abs(val[y] - val[x])) == (Math.abs(y - x)) ||
                    val[x] - val[y] == 0) {
                    count++;
                }
            }
        }
        return count;
    }

    /**
    * Calculates 56 neighbor states of current state
    */
    public void fillNeighbors() {
        neighborStates.clear();
        for (int x = 0; x < board.length; x++) {
            Integer[] copyBoard = board.clone();
            for (int y = 0; y < board.length; y++) {
                if (y != board[x]) {
                    copyBoard[x] = y;
                    neighborStates.put(copyBoard.clone(),
numOfEating(copyBoard));
                }
            }
        }
    }

    /**
    * It changes the current state to neighbor state that has min number
    value
    * ! -> If there are more than 1 state that has min number value,
    program will choose next state randomly
    */
    public void changeState() {
        int minNumber = findMin();
        List<Map.Entry<Integer[], Integer>> arr = neighborStates
            .entrySet()
            .stream()
            .filter(e -> e.getValue() == minNumber)
            .toList();

        board = arr.get(random.nextInt(arr.size())).getKey().clone();
    }

    /**

```

```

    * Finds the min number of eating in neighbor states values
    */
    private Integer findMin() {
        return neighborStates
            .values()
            .stream()
            .min(Integer::compare)
            .orElseThrow(NullPointerException::new);
    }

    /**
    * Printing solution table
    */
    public void printSolutionTable() {
        System.out.printf("-----\n");
        System.out.printf("| %-30s | %-30s | %15s |\n", "Replacement
Count", "Random Restart Count", "Time");
        System.out.printf("-----\n");
        for (int y = 0; y < solutionTable.length; y++) {
            System.out.printf("| %-30s | %-30s | %15s |\n",
solutionTable[y][0], solutionTable[y][1], solutionTable[y][2]);
        }
        System.out.printf("-----\n");
        System.out.println("| Replacement counts per Loop");
        for (int z = 0; z < solutionTable.length; z++) {
            System.out.println("| TOUR " + (z + 1) + ": " +
solutionTable[z][3].toString());
        }
        System.out.printf("-----\n");
    }
}

/**
* Below class has 2 fields to hold replacement count for table - To
provide each tours replacement count
*/
class Replacement {
    List<Integer> replacementCounts;
    Integer currentReplacementCount;

    public Replacement(List<Integer> replacementCounts, Integer
currentReplacementCount) {
        this.replacementCounts = replacementCounts;
        this.currentReplacementCount = currentReplacementCount;
    }

    public List<Integer> getReplacementCounts() {
        return replacementCounts;
    }

    public void setReplacementCounts(List<Integer> replacementCounts) {
        this.replacementCounts = replacementCounts;
    }

    public void setCurrentReplacementCount(Integer currentReplacementCount)
{
        this.currentReplacementCount = currentReplacementCount;
    }
}

```

```

    }

    //Increases number of replacement of current tour
    public void increaseFinalReplacement() {
        this.currentReplacementCount++;
    }

    //Calculates the final replacement count of tour
    public int getSumOfReplacements() {
        AtomicInteger sum = new AtomicInteger();
        replacementCounts.forEach(sum::addAndGet);
        return sum.get();
    }

    //While state changes, current replacement count will add to the list
    and set current rep. to 0
    public void stateChange() {
        replacementCounts.add(currentReplacementCount);
        currentReplacementCount = 0;
    }
}

```

2.c Programın Ekran Görüntüleri

```

-----
| Replacement counts per Loop
| TOUR 1: [3]
| TOUR 2: [4, 6, 4, 6, 4, 4, 4, 4, 5, 4, 5]
| TOUR 3: [5, 6, 3, 3, 4, 4, 5, 5, 4]
| TOUR 4: [2, 3, 4, 5, 5, 3, 2, 3, 4, 4]
| TOUR 5: [5, 5, 3]
| TOUR 6: [4, 5, 6, 5, 3]
| TOUR 7: []
| TOUR 8: [4, 5, 4, 3, 3, 6, 5, 3, 4, 3, 4]
| TOUR 9: [4, 3]
-----

```

Replacement Count	Random Restart Count	Time
3	1.0	0.0118671
50	11.0	0.0096545
39	9.0	0.0070749
35	10.0	0.0035603
13	3.0	0.0011702
23	5.0	0.0020013
0	0.0	3.562E-4
44	11.0	0.0033544
7	2.0	8.383E-4

2.d Sonuç Tablosu

Replacement Count	Random Restart Count	Time
3	1.0	0.0118671
50	11.0	0.0096545
39	9.0	0.0070749
35	10.0	0.0035603
13	3.0	0.0011702
23	5.0	0.0020013
0	0.0	3.562E-4
44	11.0	0.0033544
7	2.0	8.383E-4

3) Genetik Algoritmalar ile Şifre Kırma

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

Chromosome sayısının artması execution time da belirgin bir artışa sebep olmaktadır çünkü generation ları oluşturmak daha uzun sürecektir. Chromosome sayısındaki artış şifrenin kaç generation da bulunduğu sayısında düşüşü de sağlamaktadır. Daha fazla chromosome oluşturarak daha iyi genlere sahip chromosome elde etme şansımız ve asci aralığındaki değerlerin çoğuna sahip olmamızı sağlar.

20 Chromosome daki sonuçlar

START TIME	END TIME	EXECUTION TIME	GENERATION COUNT
3.116973762409E14	3.116979117199E14	535.479 ms	2180
3.116979131614E14	3.116980581559E14	144.9945 ms	1433
3.116980581965E14	3.116982920219E14	233.8254 ms	1994
AVERAGE GENERATION COUNT: 1869			

100 Chromosome daki sonuçlar

START TIME	END TIME	EXECUTION TIME	GENERATION COUNT
3.117510106245E14	3.117520115625E14	1000.938 ms	1179
3.117520121337E14	3.117524067685E14	394.6348 ms	777
3.117524068571E14	3.117528502728E14	443.4157 ms	795
AVERAGE GENERATION COUNT: 917			

1000 Chromosome daki sonuçlar

START TIME	END TIME	EXECUTION TIME	GENERATION COUNT
3.117880206205E14	3.117941778724E14	6157.2519 ms	577
3.117941783644E14	3.11800406204E14	6227.8396 ms	673
3.118004062451E14	3.118087185216E14	8312.2765 ms	884
AVERAGE GENERATION COUNT: 711			

3.b Kod; Çaprazlama ve Mutasyon Fonksiyonlarının Anlatımı

Seçilim: Seçilim için derste anlatıldığı gibi roulettWheel yöntemini kullandım. Generationdaki her bir chromosome un fitness değerini $1/n$ şeklinde toplayıp bunu yüzdelik hale çevirdim bu sayede her bir fitness değerinin yüzdelik miktarını hesaplamak için çarpılması gereken sayıyı buluyoruz. Sonrasında bir max değerden random sayı ürettim (Başlangıçta 100 sonrasında azalıyor) ve generation u dolaşarak başlangıçta hesapladığım çarpma değerini de kullanarak random oluşturduğum sayıdan çıkarak ilerledim ve negatife düşmediği sürece devam etti

(Başlangıçta 100 random 20 geldi, 1.Chromosome sonucunda random sayı 15 e düştü, 2.chromosome sonucunda random sayı 9 a düştü, 3.chromosome sonucunda random sayı 4 e düştü, 4.chromosome sonucunda random -1 e düştü burada durduk)

Negatife düştüğünde o chromosome seçilir ve maxRandomNumber (ilk başta 100 olandan) dan o chromosome un çarktaki yüzdelik oranı çıkarılır ($100 - 5 = 95$). 2. Chromosome un seçilmesi için 95 max olacak şekilde random sayı üretilir ve üstteki işlem gerçekleşir. Bu sayede 2 tane chromosome seçilmiş olundu.

Çaprazlama: Seçilimden gelen 2 tane chromosome un gen sayısının yarısından sonraları yer değiştirilir. (16 gen varsa 1.chromosome un son 8 geni ile 2.chromosome un son 8 geni yer değiştirir)

(1. abcdefgh – 2. qweqweqw ->1. abcdweqw – 2. qweqefgh)

Mutasyon: Mutasyon için öncelikle sayede two opt mutation yöntemini tüm chromosome lara uğratacak şekilde oluşturmuştum fakat belirli nesil tekrarından sonra belirli chromosome a doğru evrimleşmeye başladığı için normal mutation da ekledim bu sayede problem çözüldü. Generationdaki en iyi fitness değerine sahip chromosome u two opt mutation a, geri kalan chromosome ları da normal mutation a yolladım.

Two Opt Mutation: Chromosome un genleri aralarında sırayla yer değiştirir ve daha iyi bir fitness function a sahip bir dizilim bulunursa sabit kalır, bulunmazsa değişiklik geriye alınır. Tüm genler dolaşılır ve chromosome mevcut genleriyle en iyi dizilime sahip olacak şekilde dizilmiş olur.

(Gchtewr->ehctGwr)

Normal Mutation: Constant değerler içerisinde mutation olasılığı bulunuyor. Max 100 olacak şekilde random sayı oluşturuluyor ve random sayı olasılıktan küçükse chromosome normal mutation a uğruyor değilse uğramıyor. Normal mutationu chromosome un genlerinden 1 i rastgele seçilir ve asci tablosundan rastgele bir değerle değiştirilir. Bu sayede generation içindeki çeşitlilik sağlanmış olunur.

3.c Çözüm Süreleri Karşılaştırması

Gen sayısındaki yani şifredeki uzunluğun düşüşü şifrenin kaç generation da bulunduğu sayısında düşüş sağlar. Şifrenin uzunluğu azaldığı için daha az kombinasyonda sonuca ulaşabiliyoruz.

20 Chromosome 17 harfli çözüm

START TIME	END TIME	EXECUTION TIME	GENERATION COUNT
3.118552576819E14	3.118556506957E14	393.0138 ms	1974
3.11855653062E14	3.118557737105E14	120.6485 ms	1963
3.118557738583E14	3.11855866988E14	93.1297 ms	1089
AVERAGE GENERATION COUNT: 1675			

20 Chromosome 7 harfli çözüm

START TIME	END TIME	EXECUTION TIME	GENERATION COUNT
3.119074330374E14	3.119076229961E14	189.9587 ms	638
3.119076249415E14	3.119076564572E14	31.5157 ms	263
3.119076566253E14	3.119076815519E14	24.9266 ms	157
AVERAGE GENERATION COUNT: 352			

4) Makine Öğrenmesi

4.a Standardization ve Normalization Farkı ve Python Örneği

Standardizasyon ve normalizasyon, veri ön işleme (data preprocessing) adı verilen bir işlemde kullanılan iki farklı tekniktir. Bu teknikler, veri setlerini modelleme için hazırlamadan önce verilerin ölçeğini ve dağılımını standartlaştırmak için kullanılır.

Standardizasyon, verilerin ortalamasını 0 ve standart sapmasını 1 olarak değiştirir. Normalizasyon ise, verilerin belirli bir aralıkta (genellikle 0 ve 1 arasında) ölçeklendirilmesini sağlar.

Örneğin, bir veri seti verildiğinde, bir sütunun değerleri 100-1000 aralığında olabilirken, diğer bir sütunun değerleri sadece 0-1 aralığında olabilir. Bu nedenle, bu veri setindeki farklı sütunların değerleri arasındaki farklılıkları dengelemek için standardizasyon ve normalizasyon teknikleri

kullanılabilir.

```
import numpy as np
from sklearn import preprocessing

# Örnek veri seti
data = np.array([[10, 2.7, 3.6],
                 [-100, 5, -2],
                 [120, 20, 40]], dtype=np.float64)

# Standartlaştırma
data_standardized = preprocessing.scale(data)

# Normalizasyon
data_normalized = preprocessing.normalize(data, norm='l2')

print("Standartlaştırılmış veri:")
print(data_standardized)

print("Normalleştirilmiş veri:")
print(data_normalized)
```

```
Standartlaştırılmış veri:
[[ 0.          -0.85170713 -0.55138018]
 [-1.22474487 -0.55187146 -0.852133   ]
 [ 1.22474487  1.40357859  1.40351318]]
Normalleştirilmiş veri:
[[ 0.91192151  0.24621881  0.32829174]
 [-0.99855315  0.04992766 -0.01997106]
 [ 0.93704257  0.15617376  0.31234752]]
```

4.b Veri Setinin ve Problemin Kısa Anlatımı

Veri seti 2000 tabletin teknik özelliklerini ve fiyat aralığını belirten bir tabloya ait. Verilen veri setine, göre verilen tablet özelliklerini sınıflandırılmamız bekleniyor.

Veri seti;

20 Öznitelik ve 2000 gözlemden oluşuyor.

Fiyatlar için 4 tane sınıf bulunuyor (Çok ucuz, ucuz, pahalı, normal)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   BataryaGucu                          2000 non-null   int64
1   Bluetooth                            2000 non-null   object
2   MikroislemciHizi                     2000 non-null   float64
3   CiftHat                              2000 non-null   object
4   OnKameraMP                          1995 non-null   float64
5   4G                                    2000 non-null   object
6   DahiliBellek                        2000 non-null   int64
7   Kalinlik                            2000 non-null   float64
8   Agirlik                             2000 non-null   int64
9   CekirdekSayisi                      2000 non-null   int64
10  ArkaKameraMP                        2000 non-null   int64
11  CozunurlukYukseklk                 2000 non-null   int64
12  CozunurlukGenislik                  2000 non-null   int64
13  RAM                                  1988 non-null   float64
14  BataryaOmru                         2000 non-null   int64
15  3G                                    2000 non-null   object
16  Dokunmatik                          2000 non-null   object
17  WiFi                                 2000 non-null   object
18  FiyatAraligi                        2000 non-null   object
19  Renk                                 2000 non-null   object
dtypes: float64(4), int64(8), object(8)
memory usage: 312.6+ KB
```


Özniteliklerden 2 tanesi boş değerler içeriyor. Özniteliklerden FiyatAraligi her bir gözlem için sınıflarını belirtiyor. Bu sınıflar şu şekilde;

```
df["FiyatAraligi"].unique()

array(['Normal', 'Pahalı', 'Ucuz', 'Çok Ucuz'], dtype=object)
```

Bu veri seti sınıflandırıcılara verilmeden önce ön işleme ve eksik verilerinin tamamlanması aşamalarından geçmiştir. Bu sebeple kategorik veriler sayısal değerlere dönüştürülmüştür.

4.c İki Farklı Sınıflandırıcı için Python Kodu

Öncelikle veriyi test ve eğitim verisi olarak bölüyoruz.

```
y = df['FiyatAraligi']
X = df.drop(['FiyatAraligi'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.25,
                                                    random_state = 42)
```

Decision Tree

```
cart = DecisionTreeClassifier(random_state = 0)
cart_model = cart.fit(X_train, y_train)
```

```
y_pred = cart_model.predict(X_test)
accuracy_score(y_test, y_pred)
```

0.818

KNN

KNN sınıflandırıcısını eğitmeden önce en iyi sonuç alabileceğimiz komşu sayısını belirlemek için grid search kullanıyoruz

```
knn_params = {"n_neighbors": np.arange(2,15)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, knn_params, cv = 3)
knn_cv.fit(X_train, y_train)
```

```
GridSearchCV
  estimator: KNeighborsClassifier
    KNeighborsClassifier
```

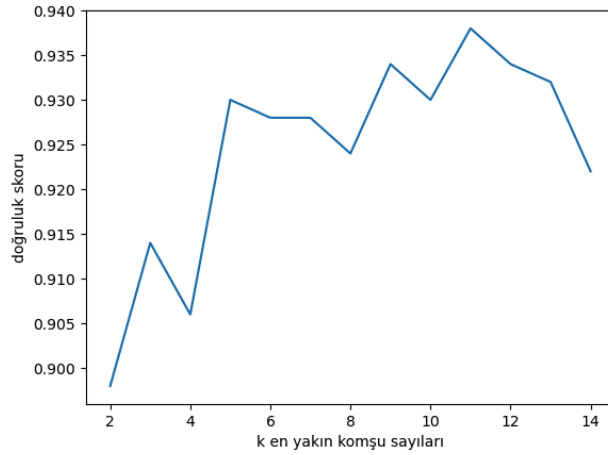
```
print("En iyi skor: " + str(knn_cv.best_score_))
print("En iyi parametreler: " + str(knn_cv.best_params_))
```

En iyi skor: 0.934
En iyi parametreler: {'n_neighbors': 9}

```
score_list = []

for each in range(2,15,1):
    knn2 = KNeighborsClassifier(n_neighbors = each)
    knn2.fit(X_train,y_train)
    score_list.append(knn2.score(X_test, y_test))

plt.plot(range(2,15,1),score_list)
plt.xlabel("k en yakın komşu sayıları")
plt.ylabel("doğruluk skoru")
plt.show()
```



Grafiğe de bakarak komşu sayısını 11 olarak belirliyoruz.

```
knn_tuned = KNeighborsClassifier(11)
knn_tuned = knn_tuned.fit(X_train, y_train)
y_pred = knn_tuned.predict(X_test)
accuracy_score(y_test, y_pred)

0.938
```

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

Desicion Tree

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.76	0.74	0.75	131
1	0.83	0.84	0.83	113
2	0.79	0.79	0.79	127
3	0.89	0.91	0.90	129
accuracy			0.82	500
macro avg	0.82	0.82	0.82	500
weighted avg	0.82	0.82	0.82	500

```
karmasiklik_matrisi = confusion_matrix(y_test, y_pred)
print(karmasiklik_matrisi)
cross_val_score(cart_model, X_test, y_test, cv = 10).mean()
```

```
[[ 97  20  14   0]
 [ 18  95   0   0]
 [ 13   0 100  14]
 [  0   0  12 117]]
```

0.8

KNN

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.94	0.89	0.91	131
1	0.94	0.98	0.96	113
2	0.91	0.91	0.91	127
3	0.95	0.98	0.97	129
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

```
karmasiklik_matrisi = confusion_matrix(y_test, y_pred)
print(karmasiklik_matrisi)
cross_val_score(knn_tuned, X_test, y_test, cv = 10).mean()
```

```
[[116  7  8  0]
 [  2 111  0  0]
 [  5  0 116  6]
 [  0  0  3 126]]
```

```
0.9040000000000001
```

4.e Kullanıcı tarafından verilen örneğin sınıflandırma ekran görüntüsü (konsol çıktıları)

Kategorik değişkenler sayısal değerlere dönüştürüldüğünden dolayı modele verilen örnek ve çıktı sayısal olacaktır sonuçların anlamlı olması için kategorik değişkenlerin sayısal karşılığı aşağıda verilmiştir.

Normal -> 0 , Pahalı -> 1, Ucuz -> 2, Çok Ucuz -> 3

Yok -> 1 , Var -> 0

Desicion Tree

```
BataryaGucu = 2500
Bluetooth = 1
MikroislemciHizi = 2.2
CiftHat = 0
OnKameraMp = 5
DortG = 0
DahiliBellek = 32
Kalinlik = 0.9
Agirlik = 120
CekirdekSayisi = 4
ArkaKameraMp = 12
CozunurlukYukseklık = 900
CozunurlukGenislik = 1200
RAM = 3000
BataryaOmru = 12
UcG = 0
Dokunmatik = 0
WiFi = 0
Renk = 1
cart_model.predict([[BataryaGucu,Bluetooth,MikroislemciHizi,CiftHat,OnKameraMp,DortG,DahiliBellek,Kalinlik,Agirlik,Ceki
array([1])
```

Ayrıca örnek 5 test verisinin gerçek fiyat aralığı ve modelin tahminleri şu şekildedir;

```
test = pd.DataFrame(X_test).copy()
test["fiyatAraligi"] = y_test
test["predicted_fiyatAraligi"] = y_pred
test.sample(5)
```

Calinlik	Agirlik	CekirdekSayisi	...	CozunurlukYukseklk	CozunurlukGenislik	RAM	BataryaOmru	3G	Dokunmatik	WiFi	Renk	fiyatAraligi	predicted_fiyatAraligi
0.9	97	3	...	179	772	523.0	4	1	0	0	4	3	3
0.1	153	8	...	98	977	3696.0	3	0	1	0	1	1	1
0.6	147	3	...	889	1635	440.0	13	0	1	1	11	3	3
0.1	159	8	...	613	650	990.0	6	1	0	0	6	3	3
0.1	81	7	...	88	1682	2144.0	12	0	0	0	9	0	0

KNN

```
BataryaGucu = 2500
Bluetooth = 1
MikroislemciHizi = 2.2
CiftHat = 0
OnKameraMp = 5
DortG = 0
DahiliBellek = 32
Kalinlik = 0.9
Agirlik = 120
CekirdekSayisi = 4
ArkaKameraMp = 12
CozunurlukYukseklk = 900
CozunurlukGenislik = 1200
RAM = 3000
BataryaOmru = 12
UcG = 0
Dokunmatik = 0
WiFi = 0
Renk = 1
knn_tuned.predict([BataryaGucu,Bluetooth,MikroislemciHizi,CiftHat,OnKameraMp,DortG,DahiliBellek,Kalinlik,Agirlik,CekirdekSayisi,ArkaKameraMp,CozunurlukYukseklk,CozunurlukGenislik,RAM,BataryaOmru,UcG,Dokunmatik,WiFi,Renk])
array([1])
```

Ayrıca örnek 5 test verisinin gerçek fiyat aralığı ve modelin tahminleri şu şekildedir

```
test = pd.DataFrame(X_test).copy()
test["fiyatAraligi"] = y_test
test["predicted_fiyatAraligi"] = y_pred
test.sample(5)
```

Calinlik	Agirlik	CekirdekSayisi	...	CozunurlukYukseklk	CozunurlukGenislik	RAM	BataryaOmru	3G	Dokunmatik	WiFi	Renk	fiyatAraligi	predicted_fiyatAraligi
0.9	184	3	...	1438	1593	262.0	20	0	0	1	2	3	3
0.3	186	8	...	103	646	3396.0	7	0	1	1	3	0	0
0.6	88	6	...	831	1713	1179.0	18	1	1	0	8	2	2
0.9	167	7	...	1096	1155	2812.0	17	1	1	1	2	0	0
0.2	118	3	...	186	1810	1152.0	20	1	0	0	2	2	2

4.f ChatGPT Kullanımı ve Yorumlama

Chatgpt'ye öncelikle veri seti hakkında bilgilendirdik ve her iki model için skoru nasıl yükseltebileceğimizi sorduk.

Decision Tree

ChatGpt Decision Tree modelinin skorunu yükseltmek için öz niteliklerinin önemine bakmamızı istedi.

```
importance = cart_model.feature_importances_  
for i,v in enumerate(importance):  
    print('Öznitelik: %0d, Skor: %.5f' % (i,v))
```

```
Öznitelik: 0, Skor: 0.11739  
Öznitelik: 1, Skor: 0.00450  
Öznitelik: 2, Skor: 0.00705  
Öznitelik: 3, Skor: 0.00089  
Öznitelik: 4, Skor: 0.00954  
Öznitelik: 5, Skor: 0.00089  
Öznitelik: 6, Skor: 0.00898  
Öznitelik: 7, Skor: 0.00980  
Öznitelik: 8, Skor: 0.01539  
Öznitelik: 9, Skor: 0.00551  
Öznitelik: 10, Skor: 0.00840  
Öznitelik: 11, Skor: 0.07387  
Öznitelik: 12, Skor: 0.07097  
Öznitelik: 13, Skor: 0.64845  
Öznitelik: 14, Skor: 0.00747  
Öznitelik: 15, Skor: 0.00207  
Öznitelik: 16, Skor: 0.00119  
Öznitelik: 17, Skor: 0.00000  
Öznitelik: 18, Skor: 0.00764
```

Bu sonuçlara bakarak düşük etkili öz nitelikleri eğitim ve test verisinden çıkardık.

```
X = df[['BataryaGucu', 'CozunurlukGenislik', 'RAM']]  
y = df['FiyatAraligi']  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    test_size = 0.25,  
                                                    random_state = 42)
```

Ardından tekrar eğittiğimizde accuracy skorumuzun 0.818 ' den 0.838'e çıktığını gözlemledik. Sonrasında daha fazla yükseltmek için neler yapabileceğimizi sorduk ve hipertarametre ayarları yapmamızı önerdi.

```
param_grid = {'max_depth': [3, 5, 7, 10],  
              'min_samples_split': [2, 5, 10, 20]}  
  
# GridSearchCV ile hipertarametre arama ve en iyi parametreleri bulma  
grid_search = GridSearchCV(model, param_grid=param_grid, cv=5, scoring='accuracy')  
grid_search.fit(X, y)  
  
# En iyi parametreleri kullanarak final modelini oluşturma  
best_params = grid_search.best_params_  
final_model = DecisionTreeClassifier(max_depth=best_params['max_depth'],  
                                     min_samples_split=best_params['min_samples_split'])  
  
# Final modeli eğitme  
final_model.fit(X, y)
```

```
DecisionTreeClassifier  
DecisionTreeClassifier(max_depth=7, min_samples_split=20)
```

Grid search ile en iyi parametrelerle modeli eğittiğimizde accuracy skorumuz 0.916 ya çıktığını gözlemledik

Diğer skorlar ise şu şekilde;

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.89	0.89	0.89	131
1	0.93	0.93	0.93	113
2	0.92	0.87	0.90	127
3	0.93	0.98	0.95	129
accuracy			0.92	500
macro avg	0.92	0.92	0.92	500
weighted avg	0.92	0.92	0.92	500

```
karmasiklik_matrisi = confusion_matrix(y_test, y_pred)
print(karmasiklik_matrisi)
cross_val_score(cart_model, X_test, y_test, cv = 10).mean()
```

```
[[116  8  7  0]
 [  8 105  0  0]
 [  6  0 111 10]
 [  0  0  3 126]]
0.8140000000000001
```

KNN

KNN modelini öncelikle gereksiz öz niteliklerin çıkarıldığı veri setini kullanarak tekrar eğittik. Fakat KNN için aynı veri setinde daha düşük skorlar elde ettik. Bu nedenle feature selection işlemi uygulamadan ChatGpt nin bize önerdiği en iyi parametreleri bulma yoluyla modeli eğittik

```
knn_tuned = KNeighborsClassifier(metric = "euclidean", n_neighbors = 9)
knn_tuned = knn_tuned.fit(X_train, y_train)
y_pred = knn_tuned.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.934
```

```
karmasiklik_matrisi = confusion_matrix(y_test, y_pred)
print(karmasiklik_matrisi)
cross_val_score(knn_tuned, X_test, y_test, cv = 10).mean()
```

```
[[116  7  8  0]
 [  3 110  0  0]
 [  4  0 116  7]
 [  0  0  4 125]]
0.906
```

Fakat sonuçlarda değişiklik gözlemlemedik.

Veri ön işleme için scale etmemizi önerdi fakat scale ettiğimizde skorların daha da düştüğünü gözlemledik.

Yorumlarımız:

Bu proje için ChatGpt kullanımımızda özellikle farklı girdiler denememize rağmen tekrara düştüğünü gözlemledik. Veri seti hakkında olabildiğince bilgilendirmeye çalıştık fakat bir sonraki cevaplarında veri setini hatırlamadığını gözlemledik.

Önerilerinin hep tekrara düşmesi sebebiyle çoğunlukla çok iyi sonuçlar alamadık. Decision tree için verdiği örnekler modelin başarısını oldukça yükseltti fakat KNN için aynı sonuçları alamadık.

Sonuç olarak ChatGpt nin en iyi sonuçları vermesi için girdilerin çok önemli olduğunu düşünüyoruz. En iyi girdilerde bile bir asistan olmaktan daha ileriye gittiğini gözlemleyemedik.

5) Öz değerlendirme Tablosu

Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır.

	İstenen Özellik	Var	Açıklama	Tahmini Not
1a	Algoritmalar + Karmaşıklıklar (10)	✓	YAPILDI	10
1b	Tanım ve Karşılaştırmalar (10)	✓	YAPILDI	10
1c	Araştırma ve Yorum (10)	✓	YAPILDI	10
2	Problem Çözme ve Kodlama (10)	✓	YAPILDI	10
3	Genetik Algoritmalar ile Şifre Kırma (15)	✓	YAPILDI	15
4	Makine Öğrenmesi (25)	✓	YAPILDI	25
	Rapor (20)	✓	YAPILDI	20
100 üzerinden Toplam Not:				100