Kocaeli Üniversitesi

Programlama Lab. 1. Proje

1st 200202087
Ferhat ARSLAN

ferhatarslnn117@gmail.com

2nd 200202002
Berkay MALKOÇ

malkocbrky@gmail.com

***Özet*—Multithreading kullanarak bize verilen veri tabanındaki istenilen verileri önce düzenleyip sonra istenilen multithreading oranında thread kullanarak karşılaştırmak hedeflenmektedir.**

I. Özet

Projemizde bize verilen hazır veri tabanını istenilen şekilde gereksiz sütunları, null değerleri ve stop wordleri sildirerek tekrar düzenledik. Bu düzenlediğimiz veri tabanını multithread kullanmak üzere python dilinde projemize aktardık. Her satırın kendi içinde birbiriyle karşılaştırılmasını, kullanıcı isterse bu aramayı daha da özelleştirebilmesini sağladık. Projemizin bizden istediği şekilde görsel bir arayüzle kullanıcıdan aldığımız inputlar sonucunda karşılaştırmamızı yapıp bunu ekrana yansıttık.

II. Giriş

Projemiz kapsamında bize verilen verilerin düzenlemesini, istenilen kriterlerde karşılaştırma yapılmasını, yapılan karşılaştırmanın özelleştirilebilmesini multithread kullanılarak projemizin bizden istediği şekilde gerçekleşmesini sağladık. Daha sonra bunu ekrana yansıttık.

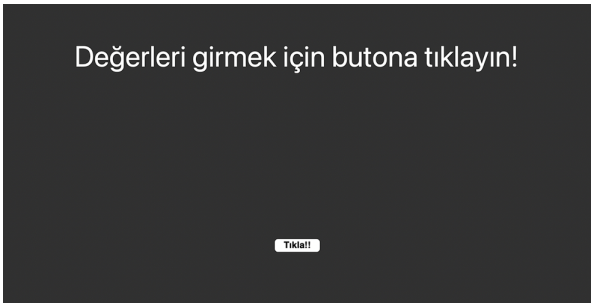
III. Yöntem

İlk önce bize verilen database'den veri setini (data frame'yi) csv dosyası olarak indirdik. Daha sonra projemizin bizden beklediği şekilde bu veri setindeki null değerleri dropna komutuyla dropladık. Ardından bizden istenildiği üzere veri setindeki noktalama işaretlerini replace komutu kullanarak sildirdik. Sonrasında veri setindeki stopword'leri apply komutuyla veri seti içerisinde aratarak yerlerini space ile değiştirdik. Düzenlenmiş olan veri setini kodu her çalıştırdığımızda tekrar işleme sokmamak, süreci hızlandırmak için sonradan verileri ordan çekmek için kullanmak üzere ayrı bir csv dosyası olarak kaydettik. Projemizin asıl işlemlerini gerçekleşeceği py dosyasına Tkinter, pandas, threading ve time kütüphanelerini import ettik. Tkinter kütüphanesini kullanarak ilk önce ana penceremizi oluşturduk. Tkinter kütüphanesi safe-thread olmadığı için yeni bir pencerede multithread işlemi yapamayacağımızdan yeni bir pencere açılmasını tetikleyen bir button oluşturduk ve o açılan yeni pencerede kullanıcıdan gerekli inputları aldık. Açılan yeni pencerede kullanıcılardan Entry blokları kullanılarak karşılaştırma yapılması istenen sütun ismi, yapılan karşılaştırmada kabul edilecek minimum benzerlik oranı, maksimum benzerlik oranı ve kullanıcının işlemi kaçlı thread kullanarak multithreada sokacağının bilgisini aldık. Ayrıca "Spesifik arama" yapmak için başka bir buton daha ekledik. Eğer kullanıcı bu butona basarda 2. Penceredeki entry blokları tekrar düzenleniyor ve artık kullanıcıdan Karşılaştırma yapılması istenilen sütunun ismi, taban benzerlik oranı, maksimum benzerlik oranı, multithread miktarı tekrar alınmakla beraber bu sefer ayrıyeten özellikle kullanıcının hangi etikete sahip ürünleri karşılaştırma yapmasını seçebilmesi için aynı olacak etiket sütununun bilgisini ve aynı olacak etiketi de kullanıcıdan input olarak aldık. (Burada etiketten kasıt veri setinin belli bir sütunundaki belli bir değerdir.) Daha sonra kullanıcı girdiği her inputu kenarda bulunan onayla butonundan onayladıktan sonra işlemi tamamla butonuna basarsa program ikinci pencereyi kapatıp ilk pencereye döner. Bu ilk pencerede bir ana_frame oluşturur. Bu frame'in içine koşulları sağlayan verilerin yansıtılması için bir canvas koyduk. Canvasın içinde de verilerin yazılacağı tek ve geniş text widgeti bulunuyor. Gerekli koşulları sağlayan her türlü veri sütun sütun ayrılmış düzgün bir şekilde bu text widgetine aktarılır ve kullanıcıya yansıtılır. Bu ekranda istenilen sütundaki 1. Veri sonraki 1000 veri ile karşılaştırılmıştır. Eğer kullanıcı pencerenin altında bulunan sonraki sayfa butonuna tıklarsa 1. Veri 1000 ile 2000 arasındaki veriyle kıyaslanır ve onlar ekrana yansıtılır. Bu şekilde programın 1. Satırı bütün 1,1 milyon satırla kıyaslanması sağlanmıştır. Ayrıca sayfanın altında bir de sonraki veri butonu bulunur. Eğer kullanıcı bu butona basarsa istenilen sütunun 1. Verisinden 2. Verisine geçiş yapılır ve bu veri ilk 1000 veriyle kıyaslanıp ekrana yansıtılır. Kullanıcı tekrar sonraki sayfaya basarsa bu sefer 2. Veri 1000 ile 2000 arasındaki verilerle kıyaslanır. Bu yöntemle veri setindeki her veri birbiriyle kıyaslanmış olur. Sayfa algoritması kurmak hem programın daha hızlı çalışmasını, hem de kullanıcının bekleme yaşamamasını sağlamıştır. Ayrıca sayfanın sağ üstünde kullanıcının seçmiş olduğu multithread miktarına bağlı olarak her threadın kaç ms sürdüğü gözükmektedir. Programın sayfaya yansıtmasının toplam kaç ms sürdüğü de yazmaktadır. Eğer farklı multithread değeleriyle test edilirse thread miktarı arttıkça programın daha hızlı çalıştığı gözlemlenir. Son olarak pencerenin alt kısmında

bir de yeniden değer bir butonu bulunmaktadır. Bu da kullanıcının programdan çıkmadan tekrar baştan veri girip başka bir çıktı almasına olanak sağlar. Bu şekil programımız bizden istenilen her türlü işlemi yapabilecek kapasiteye ulaşmıştır

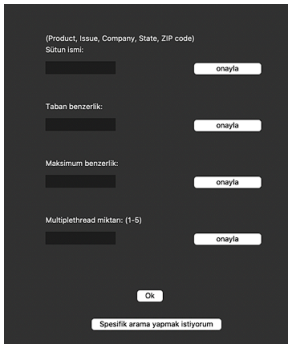
IV. Programın İşleyişi

1-) Öncelikle program açıldığında kullanıcıyı istediği değeleri girebilmesi için oluşacak pencereye yönlendirecek bir giriş sayfası oluşturduk



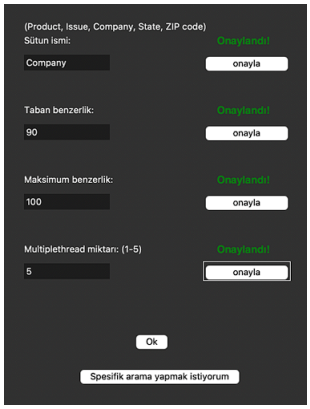

```
label22= Label(window, text="Değerleri girmek için butona tıklayın!", font= ('Helvetica 15 bold', 45))
label22.pack(pady=200, side=TOP)

button22= Button(window, text= "Tıkla!!", command= popupwin, font= ('Helvetica 14 bold'))
button22.pack(pady=50, side=TOP)
```

2-) Kullanıcının Tıkla yazan butona tıklaması sonucu kullanıcıyı değerleri gireceği panele yönlendirdik.



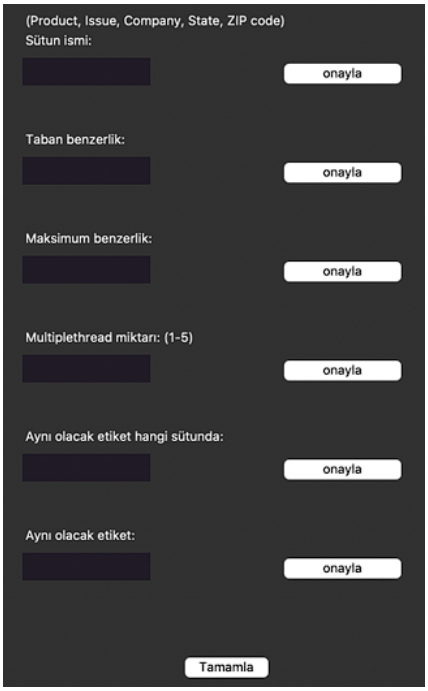

```
label0= Label(top, text="(Product, Issue, Company, State, ZIP code)")
label0.place(x=200, y=150)

label1= Label(top, text="Sütun ismi:")
label1.place(x=200, y=170)

label2= Label(top, text="Taban benzerlik:")
label2.place(x=200, y=270)

label3= Label(top, text="Maksimum benzerlik:")
label3.place(x=200, y=370)

label4= Label(top, text="Multiplethread miktarı: (1-5)")
label4.place(x=200, y=470)


input_txt = Entry(top, width= 13)
input_txt.place(x=200, y=200)

input_txt2 = Entry(top, width= 13)
input_txt2.place(x=200, y=300)

input_txt3 = Entry(top, width= 13)
input_txt3.place(x=200, y=400)

input_txt4 = Entry(top, width= 13)
input_txt4.place(x=200, y=500)

btn1 = Button(top, text = "onayla", command = clicked, height=1, width=10)
btn1.place(x=460, y= 200)
btn2 = Button(top, text = "onayla", command = clicked2, height=1, width=10)
btn2.place(x=460, y= 300)
btn3 = Button(top, text = "onayla", command = clicked3, height=1, width=10)
btn3.place(x=460, y= 400)
btn4 = Button(top, text = "onayla", command = clicked4, height=1, width=10)
btn4.place(x=460, y= 500)
```

3-) Kullanıcı istediği değerleri girip onaylaması durumunda programın bu değerleri global değişkenlere kaydedip kullanıcıya kaydedildiğine dair bir onaylandı mesajı göstermesini sağladık




```
def clicked():
 global sutunkac
 sutunkac = input_txt.get()
 if sutunkac == "ZIP Code":
     sutunkac = 4
 elif sutunkac == "Complaint ID":
     sutunkac = 5
 label= Label(top, text="Onaylandı!", font= ('Helvetica 15 bold'))
 label.config(fg= "green")
 label.place(x=477, y=170)

def clicked2():
 label= Label(top, text="Onaylandı!", font= ('Helvetica 15 bold'))
 label.config(fg= "green")
 label.place(x=477, y=270)
 global taban
 taban = input_txt2.get()
 taban = float(taban)/100

def clicked3():
 label= Label(top, text="Onaylandı!", font= ('Helvetica 15 bold'))
 label.config(fg= "green")
 label.place(x=477, y=370)
 global maksimum
 maksimum = input_txt3.get()
 maksimum = float(maksimum)/100

def clicked4():
 global threadkac
 threadkac = input_txt4.get()
 label= Label(top, text="Onaylandı!", font= ('Helvetica 15 bold'))
 label.config(fg= "green")
 label.place(x=477, y=470)
```

4-) Kullanıcı spesifik arama yapmak istiyorum yazan butona tıklarsa kullanıcının belli bir sütundaki belli bir değere sahip olan değerleri, istediği sütundaki istediği benzerlik oranına sahip değerlerle listelemesini sağlayacak girdileri kullanıcıdan aldık.



```
label0= Label(top, text="(Product, Issue, Company, State, ZIP code)")
label0.place(x=200, y=50)

label1= Label(top, text="Sütun ismi:")
label1.place(x=200, y=70)

label2= Label(top, text="Taban benzerlik:")
label2.place(x=200, y=170)

label3= Label(top, text="Maksimum benzerlik:")
label3.place(x=200, y=270)

label4= Label(top, text="Multiplethread miktarı: (1-5)")
label4.place(x=200, y=370)

label5= Label(top, text="Aynı olacak etiket hangi sütunda:")
label5.place(x=200, y=470)

label5= Label(top, text="Aynı olacak etiket:")
label5.place(x=200, y=570)


input_txt = Entry(top, width= 13)
input_txt.place(x=200, y=100)

input_txt2 = Entry(top, width= 13)
input_txt2.place(x=200, y=200)

input_txt3 = Entry(top, width= 13)
input_txt3.place(x=200, y=300)

input_txt4 = Entry(top, width= 13)
input_txt4.place(x=200, y=400)

input_txt5 = Entry(top, width= 13)
input_txt5.place(x=200, y=500)

input_txt6 = Entry(top, width= 13)
input_txt6.place(x=200, y=600)
```

5-) Kullanıcının tamamla butonuna tıklayıp 2. Pencereyi pencere_kapat fonksiyonuyla kapattıktan sonra normal arama yapmak istediği durumda arka planda bastır fonksiyonu içindeki threadyazdir fonksiyonuna giren programla "girdimi" değişkeni sayesinde kullanıcının normal arama mı spesifik arama mı yapmak istediğini, "threadkac" değişkeni sayesinde kaçlı multithread yapmak istediğini anlayıp ona göre multiyazdir veya multiyazdir2 fonksiyonunu çalıştırdık.

```
if int(threadkac) == 1 and girdimi == 0:
    startt = time.time()
    t1 = threading.Thread(target=multiyazdir, args=(satirsayisi, baslangic))
    t1.start()
    t1.join()
    text = Text(ikinci_frame, width=20, height=4, bg = "Red")
    text.grid(row=1,column=7)
    mskac = 1
    #toplamms = 0
    for bam in mslist:
        text.insert(INSERT, bam* 10**3)
        text.insert(END, 'ms\nthread')
        text.insert(END, mskac)
        text.insert(END, '\n')
        mskac += 1
        #toplamms += bam
    endd = time.time()
    text.insert(END, (endd-startt)* 10**3)
    text.insert(END, 'ms\n(Toplam Geçen Süre)')
    mslist = []
elif int(threadkac) == 1 and girdimi == 1:
    startt = time.time()
    t1 = threading.Thread(target=multiyazdir2, args=(satirsayisi, baslangic))
    t1.start()
    t1.join()
    text = Text(ikinci_frame, width=20, height=4, bg = "Red")
    text.grid(row=1,column=7)
    mskac = 1
    #toplamms = 0
    for bam in mslist:
        text.insert(INSERT, bam* 10**3)
        text.insert(END, 'ms\nthread')
        text.insert(END, mskac)
        text.insert(END, '\n')
        mskac += 1
        #toplamms += bam
    endd = time.time()
    text.insert(END, (endd-startt)* 10**3)
    text.insert(END, 'ms\n(Toplam Geçen Süre)')
    mslist = []
```

```
def pencere_kapa(top):
    global bnz_st1
    top.destroy()
    label22.destroy()
    button22.destroy()
    bnz_st1 = []
    bastir()
```

6-) Programımızın işleyişi gereği global değişkenler oluşturup bu değişkenlerle multiyazdir fonksiyonunda belirli sütunlarda belirli satırlar arası karşılaştırma yapılabilmesini sağladık. Karşılaştırılıp istenilen şartları sağlayan satırları art arda gelecek şekilde append komutuyla baz_st1 dizisine atadık. Ayrıca threadin ne kadar sürdüğünü ölçmek adına bu kısımda 2 farklı time değişkeni oluşturup farklarını zaman ölçümlerini tutan bir diziye atadık.

```
global satirsayisi
satirsayisi = 10500
global baslangic
baslangic = 0
global satirx
satirx = 0
global varmi
varmi = 0
```

```
def multiyazdir(aralikmax, i):
    global bnz_st1
    global mslist
    global satirx
    global varmi

    x=satirx
    y=i+1
    satir = (aralikmax-1)
    st1 = time.time()
    k = 0
    j=1
    for j in range(satir):
        if maksimum <= string_hesaplama(str(df.iloc[x][sutunkac]).split(),str(df.iloc[y][sutunkac]).split()) <= taban and x < len(df. index)-1 and len(bnz_st1)%7 == 0:
            varmi = 1
            oranx = "%" and string_hesaplama(str(df.iloc[x][sutunkac]).split(),str(df.iloc[y][sutunkac]).split())
            oranx = float(oranx)*100

            bnz_st1.append(str(df.iloc[x]["Product"]))
            bnz_st1.append(str(df.iloc[x]["Issue"]))
            bnz_st1.append(str(df.iloc[x]["Company"]))
            bnz_st1.append(str(df.iloc[x]["State"]))
            bnz_st1.append(str(df.iloc[x]["ZIP code"]))
            bnz_st1.append(str(df.iloc[x]["Complaint ID"]))
            bnz_st1.append(str(oranx))
            #---
            bnz_st1.append(str(df.iloc[y]["Product"]))
            bnz_st1.append(str(df.iloc[y]["Issue"]))
            bnz_st1.append(str(df.iloc[y]["Company"]))
            bnz_st1.append(str(df.iloc[y]["State"]))
            bnz_st1.append(str(df.iloc[y]["ZIP code"]))
            bnz_st1.append(str(df.iloc[y]["Complaint ID"]))
            bnz_st1.append(str(oranx))

            if y+1 <= len(df. index)-1:
                y = y+1
                k = 1

        if y+1 < len(df. index)-1 and k == 0:
            y = y+1
        k=0

    end1 = time.time()
    mslist.append(float(end1-st1))
```

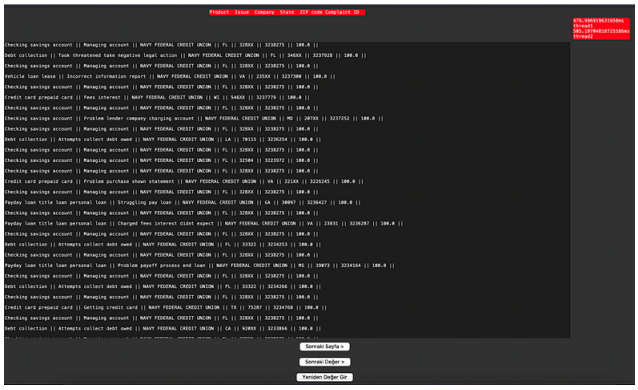
7-) Kullanıcının spesifik arama yapmayı istemesi durumunda bu sefer girdimi değişkeni sayesinde programın multiyazdir2 fonksiyonunu çalıştırmasını sağladık. Buradaki en temel fark if'in verilerin kullanıcının istediği etiketle eşleşip eşleşmemesini kontrol etmesi oldu. Spesifik değişkeni sütunu temsil derken aynietiket kullanıcının aynı olması istediği değeri temsil etmektedir.

```
) and df.iloc[x][spesifik] == aynietiket and df.iloc[y][spesifik] == aynietiket:
```

8-) Ana ekrandaki bütün widgetleri silip sütun isimlerinin yer aldığı bir başlık ve verilerin yer alacağı bir text widgeti oluşturduk. Doğru şekilde dizimize atanan değerleri bir for döngüsüyle text widgetinin içine aktarıp ekrana yansıttık. Daha sonra bnz_st1 listesini daha sonra tekrar kullanabilmek adına boşalttık.

```
text = Text(ikinci_frame, width=55, height=1, bg = "Red")
text.grid(row=0,column=0)
text.insert(INSERT, "Product  Issue  Company  State  ZIP code Complaint ID\n\n\n")

threadyazdir(satirsayisi, baslangic)

global bnz_st1
bam = 0
gecici = 0
text333 = Text(ikinci_frame, width=200, height=1000)
for bam in bnz_st1:
        gecici += 1
        text333.insert(END,  bam + ' ' + "|| ")
        text333.grid(row=3, column=0)
        if (gecici%7) == 0:
            text333.insert(END,'\n\n')
bnz_st1 = []
```

9-) Kullanıcı tamamla butonuna bastıktan sonra arkada bu işlemlerin dönmesini ve ekrandaki text widgetine istenilen değerlerin yazdırılmasını sağladık. Görüldüğü üzere önceden aldığımız thread sürelerini ve toplam çalışma zamanını da ekranın sağ üst köşesine yerleştirdik. Ayrıca programın veri işlemeye devam edebilmesi için sonraki sayfa ve sonraki veri diye iki farklı buton ekledik.



10-) Kullanıcının sonraki sayfa butonuna basması sonucunda global değişken olarak tuttuğumuz satirsayisi değişkeni kendini baştaki değeri kadar arttırıp ilk başta 0-10500 arası arama yaparken sonraki sayfada ise 10500-21000 verileri arası arama yapmasını sağladık. Ayrıca sonrakisayfa fonksiyonuna gelen argüman sayesinde kullanıcının spesifik arama mı normal arama mı yapmak istediğini gözlemleyebildik.

```
def SonrakiSayfa(atlakal):
    global satirsayisi
    global baslangic
    global bnz_st1
    global satirx
    baslangic = satirsayisi
    satirsayisi += 10500
    for widgets in ikinci_frame.winfo_children():
        widgets.destroy()
    if atlakal == 1:
        satirx += 1
        satirsayisi = 10500
        baslangic = 0

    text = Text(ikinci_frame, width=55, height=1, bg = "Red")
    text.grid(row=0,column=0)
    text.insert(INSERT, "Product  Issue  Company  State  ZIP code Complaint ID\n\n\n")
    bnz_st1 = []
    threadyazdir(satirsayisi, baslangic)
    bam = 0
    gecici = 0
    text333 = Text(ikinci_frame, width=200, height=1000)
    for bam in bnz_st1:
            gecici += 1
            text333.insert(END,  bam + ' ' + "|| ")
            text333.grid(row=3, column=0)
            if (gecici%7) == 0:
                text333.insert(END,'\n\n')
    bnz_st1 = []
```

```
global button31
global button32
global button33
button32= Button(window, text= "Yeniden Değer Gir", command= popupwin)
button32.pack(pady=5, side=BOTTOM)
button33= Button(window, text= "Sonraki Değer >", command= lambda:SonrakiSayfa(1))
button33.pack(pady=5, side=BOTTOM)
button31= Button(window, text= "Sonraki Sayfa >", command= lambda:SonrakiSayfa(0))
button31.pack(pady=5, side=BOTTOM)
```

11-) Kullanıcının Sonraki Değer butonuna basması halinde karşılaştırma yapılan kısımda sonraki sayfa fonksiyonu içerisinde 2. Değere geçmesini ve bu sefer onu 1,1 milyon veriyle kıyaslamasını sağladık.

12-) Yeniden değer gir yazan butona basılması halinde programın ilk açıldığı andaki değer girmek için tıklanan butona basıldığında gerçekleşen popupwin fonksiyonunun tekrar çalışmasını ve programın kapanmaya gerek duymadan kullanıcıdan tekrar veri alabilmesini sağladık.

V. Yalancı Kod

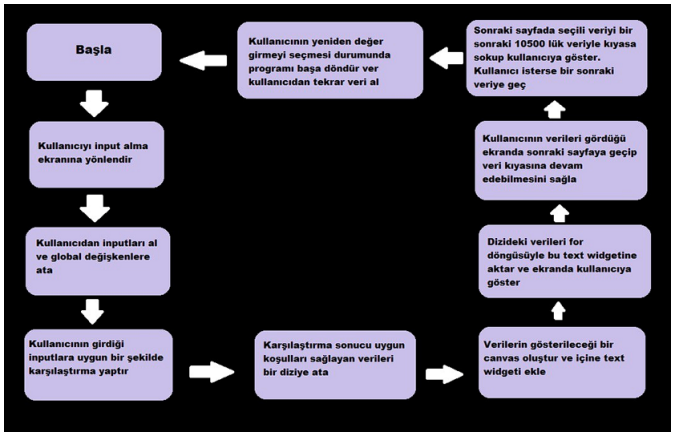
1) Kullanıcıyı input gireceği ekrana yönlendiren bir ana sayfa oluştur
2) Kullanıcının Sütun ismi, Taban benzerlik, Maksimum benzerlik ve Multithread sayısı girebileceği bir pencere oluştur.
3) Kullanıcının isteği halinde spesifik arama yapabileceği bir pencere oluştur ve önceki pencereden bu pencereye yönlendirilebilmesini sağla
4) Kullanıcı gerekli inputları girdikten sonra programın normal arama mı spesifik arama mı olduğunu algılayıp ona göre bir kıyas yapmasını sağla
5) Yapılan kıyaslama sonucu istenilen verileri bir diziye ata.
6) Kıyas yapılırken thread sürelerini tutan bir dizi oluştur
7) Verilerin olduğu diziyi ana ekranda bir text widgetinin içine aktar
8) Karşılaştırma ve aktarma işlemi bitince bir for döngüsüyle verileri ve thread sürelerini ekrana yazdır
9) Verilerin olduğu ekranda armaya devam edilebilmesi için sonraki sayfa ve sonraki veri butonları ekle
10) Sonraki sayfa ve sonraki değer butonlarının işlevsel olabilmesi için global değişkenler kullan
11) Kullanıcının programdan çıkmadan yeni veri girişi yapabilmesi için Yeni değer girin butonu eklei

VI. Deneysel Sonuçlar

Görsel bir arayüz kullanılarak kullanıcıdan gerekli isterler alındı. Program isterler doğrultusunda karşılaşma işlemlerini gerçekleştirebildi. Karşılaştırılan verilerden istenilen koşulları sağlayanların bir diziye atanması yardımıyla ekrana yazdırılması sağlandı. Kullanıcının süre kaybı, donma vs. Yaşamaması için verileri sayfa sayfa görme algoritması başarıyla oluşturuldu. Kullanıcının uygulamadan çıkmadan yeniden veri girişi yapılabilmesi sağlandı. Projenin bizden istediği bütün isterler, senaryolar başarıyla yerine getirildi.

VII. Akış Diyagramı

Akış Diyagramı:



VIII. Sonuç

Kullanıcı istediği inputlarla istediği satırlarda istediği spesifik koşullarda kıyaslama yapabildi. Kullanıcıdan alınan inputlara göre bir kıyaslama yapıldı. Yapılan kıyas kullanıcıya görsel bir arayüz yardımıyla sunuldu. Kullanıcının kıyas yapmaya devam edebilmesi veya yeni koşul girebilmesi sağlandı.

IX. Oluşturma Ortami

Programımızın Python dili kullanılarak, Visual Studio Code programı üzerinde yapımı tamamlandı. Rapor hazırlamak için LaTeX kullanıldı.

Kaynaklar

[1] https://mertmekatronik.com/thread-ve-multithread-nedirZ
[2] https://www.tutorialspoint.com/operating_system/os_multi_threading.htm
[3] https://www.geeksforgeeks.org/multithreading-python-set-1/
[4] https://www.kaggle.com/datasets/selener/consumer-complaint-database (Veri setinin alındığı kaynak)