



UZMAN SİSTEMLERE GİRİŞ PROJE RAPORU

ÖĞRENCİ 1
FATİH ÇOMAK
12011015

ÖĞRENCİ 2
HALİL İBRAHİM ÖZDEMİR
12011021

DERSİN YÜRÜTÜCÜSÜ
DOÇ. DR. MEHMET FATİH AMASYALI

Konu: Kendi Belirlediğimiz bir problem için uzman sistem oluşturma

TESLİM TARİHİ
19 ARALIK 2019

Ödevin Konusu

Google Play Store, android işletim sistemli mobil cihazların uygulama mağazasıdır. Bu mağazada bulunan uygulamalar, mağazada bulundukları yaşam döngüsünde belirli veriler elde ederler. Bunlardan bazıları uygulamaların indirilme sayısı, bu uygulamalara yapılan kullanıcı yorumları, boyut, isim bilgileri, yaş grubu, olumlu-olumsuz yorumlar ve puanlanmalar dikkate alınarak hesaplanan uygulama puanı gibi verilerdir.

Projemizde Google'un Play Store mağazasında sergilenen "Oyun" başlığı altında uygulamalar işlenecek. Oyunlardan edindinilen verileri kullanarak bu oyunların puanlarını tahmin etmek için uzman sistem tasarlayacağız.

Veri toplama işlemini Play Store'dan 1000 tane oyunun verilerini alarak yapacağız. Bilgi tabanı kaynağımız Google Play Store mağazasının uygulama deposu olmak durumundadır. Geliştirdiğimiz uzman sistem sayesinde bu 1000 tane oyunun verilerini kullanarak eğittiğimiz tahmin algoritması modeli, yine bu mağazadan seçtiğimiz trend oyunlar listesindeki 100 oyun üzerinde test edilecek. Tahmin edilen uygulama puanı ile gerçek uygulama puanı verisi karşılaştırılarak sistemin başarısı test edilmiş olacak.

Bilgi toplama sonrasında iki adet .csv dosyası elde edilmiş oldu. Bu .csv dosyasının içeriğinde toplam **963** oyuna ait bilgiler toplandı. Ayrıca test datası için de en çok indirilen bölmesinden **100** adet oyun verisi toplandı.

Dosyanın içeriğinin örnek görüntüsü şu şekildedir:

Relation: Play Store Games											
No.	1: App	2: Reviews	3: Size	4: Installs	5: Type	6: Price	7: Content Rating	8: Genres	9: Last Updated	10: Rating	11: Rating
	Nominal	Nominal	Size	Size	Price	Price	Content Rating	Genres	Updated	Rating	Rating
143	Exterior A Play About Survival	11023.0	16.0	1000000.0	Paid	3.99	Teen	Adventure	2018.0	(=4.5)	
144	Superbrothers Sword & Sworcery	10515.0	44.0	1000000.0	Free	0.0	Teen	Casino	2018.0	(=4.5)	
145	FaFaFa's Gold Casino: Free slot machines	11258.0	44.0	500000.0	Free	0.0	Mature 18	Action	2016.0	(=4.5)	
146	Range Z: Multiplayer Zombie FPS Online Shooter	11379.0	20.0	500000.0	Free	0.0	Everyone 12	Arcade	2018.0	(=4.5)	
147	Pacific Navy Fighter C.E. (AS)	11408.0	96.0	1000000.0	Free	0.0	Teen	Action	2018.0	(=4.5)	
148	G-Switch 2	12693.0	19.0	1000000.0	Free	0.0	Everyone	Action	2018.0	(=4.5)	
149	Mendrisia	12736.0	56.0	1000000.0	Free	0.0	Everyone	Arcade	2017.0	(=4.5)	
151	Call of Duty Black Ops Zombies	13004.0	46.0	100000.0	Paid	6.99	Teen	Action	2016.0	(=4.0)	
152	Destiny Ninja Shall we die otome games love story	13079.0	30.0	500000.0	Free	0.0	Everyone 12	Adventure	2018.0	(=4.0)	
153	Law of Creation: A Playable Manga	13118.0	47.0	500000.0	Free	0.0	Teen	Adventure	2018.0	(=4.5)	
154	BombSquad Remote	13304.0	1.3	1000000.0	Free	0.0	Everyone	Arcade	2016.0	(=4.5)	
155	Zombie Avengers:(Dreams)/Stickman War Z	13604.0	96.0	1000000.0	Paid	0.99	Teen	Action	2018.0	(=4.5)	
156	Stickman Warriors Heroes 2	13714.0	41.0	1000000.0	Free	0.0	Everyone 12	Action	2017.0	(=4.5)	
157	Q*bert: Rebooted	13788.0	55.0	1000000.0	Free	0.0	Everyone	Arcade	2017.0	(=4.5)	
158	NDS Emulator - For Android 6	14002.0	19.0	1000000.0	Free	0.0	Everyone	Arcade	2018.0	(=4.5)	
159	Dr. Parker - Parking Simulator	14253.0	81.0	1000000.0	Free	0.0	Everyone	Racing	2018.0	(=5.0)	
160	Thai Sic Bo	14283.0	4.2	100000.0	Free	0.0	Teen	Casino	2017.0	(=4.0)	
161	The Fish Master!	15763.0	54.0	1000000.0	Free	0.0	Teen	Arcade	2018.0	(=4.0)	
162	The Vikings	15806.0	42.0	1000000.0	Free	0.0	Everyone	Arcade	2018.0	(=4.5)	
163	Five Nights at Freddy's: SL	16162.0	99.0	100000.0	Paid	2.99	Teen	Action	2017.0	(=4.5)	
164	Dubai Racing	16237.0	23.0	500000.0	Free	0.0	Everyone	Racing	2015.0	(=4.5)	
165	Fate/Grand Order (English)	16601.0	55.0	500000.0	Free	0.0	Teen	Role Playing	2018.0	(=4.5)	
166	Magnum 3.0 Gun Custom Simulator	16815.0	59.0	1000000.0	Free	0.0	Teen	Action	2018.0	(=4.5)	
167	V for Voodoo	16876.0	59.0	1000000.0	Free	0.0	Teen	Trivia	2018.0	(=4.5)	
168	Block Puzzle Classic Legend I	17039.0	4.9	5000000.0	Free	0.0	Everyone	Puzzle	2018.0	(=4.5)	
169	Block Puzzle Classic Legend I	17044.0	4.9	5000000.0	Free	0.0	Everyone	Puzzle	2018.0	(=4.5)	
170	AE Spider Solitaire	17263.0	47.265	5000000.0	Free	0.0	Everyone	Card	2017.0	(=4.5)	
171	LA Stories 4 New Order Sandbox 2018	17453.0	99.0	1000000.0	Free	0.0	Mature 18	Racing	2018.0	(=4.5)	
172	Don't Starve: Pocket Edition	17988.0	7.9	100000.0	Paid	4.99	Teen	Adventure	2018.0	(=4.5)	
173	Farming Simulator 18	18125.0	15.0	100000.0	Paid	4.99	Everyone	Simulation	2018.0	(=5.0)	
174	AE Lucky Fishing	18277.0	17.0	1000000.0	Free	0.0	Teen	Arcade	2016.0	(=5.0)	
175	The Game of Life	18621.0	63.0	100000.0	Paid	2.99	Everyone	Board	2018.0	(=5.0)	
176	The Game of Life	18552.0	63.0	100000.0	Paid	2.99	Everyone	Board	2018.0	(=3.5)	
177	Jungle Marble Blast	18985.0	23.0	5000000.0	Free	0.0	Everyone	Casual	2018.0	(=4.0)	
178	Dino War: Rise of Beasts	18996.0	81.0	1000000.0	Free	0.0	Teen	Strategy	2018.0	(=4.0)	
179	Car Driving Simulator Drift	19816.0	57.0	1000000.0	Free	0.0	Everyone	Racing	2017.0	(=4.0)	
180	Crazy Bike attack Racino New: motorcycle racing	20364.0	45.0	5000000.0	Free	0.0	Everyone	Racing	2018.0	(=4.5)	

Veri Seti

Play Store Games.csv , Play Store Games - Test.csv

Oluşturulan veri setlerine göre özelliklerimiz şöyledir:

Veri setinin içerisinde 1000 oyun örneğine ait **10** adet öznitelik bulunur. Bunlardan **5** tanesi nümeriktir ve **5** tanesi de kategoriktir. Attribute açıklamalarımız ise şu şekildedir:

- **App** (numeric): Uygulamanın isminin tutulduğu öznitelik.
- **Reviews** (numeric): Uygulamaya yapılan olumlu veya olumsuz yorumun tutulduğu ve verilen puanın alındığı alanın sayısıdır.
- **Size** (numeric): Uygulamanın Megabyte cinsinden boyut değeridir.
- **Installs** (numeric): Uygulamanın indirilme sayısıdır.
- **Type** (nominal): Free veya Paid olarak **2 kategoriye** ayrılmış şekilde olan, uygulamanın ücretlendirme tipini tutan özniteliktir.
- **Price** (numeric): Uygulamanın \$ kuru üzerinden ücretini ifade eder. Eğer tipi Free ise ücreti 0 \$'dır.
- **Content Rating** (nominal): Uygulamanın oynanabilmesi için önerilen yaş kitlesidir. 12 yaşına kadar olanlar için Everyone, 12 yaş için Everyone 12, 16 yaş için Teen, 18 yaş için de Mature 18 olmak üzere **4 kategori** kullanılmıştır.
- **Genres** (nominal): Oyunun türüdür. Kart, Aksiyon, Arcade, Tahta, Kelime, Bilgi Oyunları, Kasino, Macera, Puzzle, Müzik, Yarış, Klasik, Strateji, Simulasyon, Rol Yapma ve Spor olmak üzere **16 kategoriden** oluşur.
- **Last Updated** (numeric): En son güncellenme tarihini tutar. İlk başta gün-ay-yıl şeklinde girilen bu değer programın kolay kullanımı için nümerik değere dönüştürmek adına sadece yıla dönüştürülmüştür. Yıllar ise 2013-2019 arasındadır.
- **Rating** (nominal): Bu öznitelik, class attribute'üdür. İlk başta sayısal olarak girilmesine karşın, tahmin algoritmasının

çalıştırılması için ve oranı yükseltmek için, eldeki verilere göre 6 kategoriye ayrılmıştır. Bunlar;

- 2.1-2.5 puan arasında olanlar 2.5 olarak,
- 2.6-3.0 puan arasında olanlar 3.0,
- 3.1-3.5 puan arasında olanlar 3.5,
- 3.6-4.0 puan arasında olanlar 4.0,
- 4.1-4.5 puan arasında olanlar 4.5,
- 4.6-5.0 puan arasında olanlar 5.0 puan olmak üzere kategorilendirilmiştir.

Tüm bu attribute işlemleri yapılırken veri seti normalize de edilmiştir.

Amacımız uzman sistemi elde ettikten sonra, test datamızın verilerini verdikten sonra ilgili oyunun puan grubunun doğru tahmin edilmesidir. Bu sistemin başarımı da puan grubunun doğruluk tespiti sayısı kullanılarak oransal olarak sayısal bir değer olarak ifade edilecektir.

İşlemler

Öncelikle doğru algoritmayı seçip kodlayabilmek için Weka'daki algoritmaların başarımı denenmiştir.

Play Store Games.arff , Play Store Games - Test.arff

Öncelikle sistem için kullanılacak algoritmanın belirlenmesi için Weka isimli programda veri setimiz öncelikle .arff dosyasına dönüştürülerek Weka Explorer'da çalıştırılmıştır. İlgili .arff dosyalarımızın text editöründeki örnek görüntüsü şöyledir:

```
1 @relation 'Play Store Games'
2
3 @attribute App string
4 @attribute Reviews numeric
5 @attribute Size numeric
6 @attribute Installs numeric
7 @attribute Type {Free,Paid}
8 @attribute Price numeric
9 @attribute 'Content Rating' {Everyone,Teen,'Everyone 12','Mature 18'}
10 @attribute Genres {Card,Action,Arcade,Board,Word,Trivia,Casino,Adventure,Puzzle,Music,Racing,Casual,Strategy,Simulation,'Role Playing',Sports}
11 @attribute 'Last Updated' numeric
12 @attribute Rating {<=4.5,<=3.5,<=4.0,<=5.0,<=2.5,<=3.0}
13
14 @data
15 'CJ Poker Odds Calculator',207,0.116,50000,Free,0,Everyone,Card,2011,<=4.5
16 'Mobile CS:GO',1015,0.643,100000,Free,0,Everyone,Action,2015,<=3.5
17 'FreeCell CY',387,1.1,50000,Free,0,Everyone,Card,2011,<=4.0
18 'BombSquad Remote',13304,1.3,1000000,Free,0,Everyone,Arcade,2016,<=4.5
19 'BW-Go Free',547,1.3,10000,Free,0,Everyone,Board,2015,<=5.0
20 'BW-Go',265,1.3,1000,Paid,3.49,Everyone,Board,2015,<=5.0
21 'DS Tower Defence',768,1.4,100000,Free,0,Everyone,Arcade,2013,<=3.5
22 'Word Search multilingual',32849,1.5,1000000,Free,0,Everyone,Word,2015,<=4.5
23 'BW-Joseki',705,1.8,10000,Free,0,Everyone,Board,2015,<=5.0
24 'BEBONCOOL GAMEPAD V1.0',404,2.2,100000,Free,0,Everyone,Arcade,2017,<=4.0
25 'What s Your CS:GO rank?',278,2.4,10000,Free,0,Teen,Trivia,2017,<=2.5
26 'Dino T-Rex',69115,2.4,10000000,Free,0,Everyone,Arcade,2018,<=4.0
27 'My Boy! Free - GBA Emulator',531074,2.5,10000000,Free,0,Everyone,Arcade,2018,<=4.5
28 'Four In A Line',22191,3,1000000,Free,0,Everyone,Board,2015,<=4.0
29 'Sudoku Master',58387,3.3,1000000,Free,0,Everyone,Word,2013,<=4.5
30 'Checkers',375996,3.3,10000000,Free,0,Everyone,Board,2018,<=4.5
```

Aslında class özneliğimiz olan Rating, ilk eklediğimizde nümerik bir değer ifade etmekteydi. Fakat Weka'da sınıflandırma algoritmalarını test edebilmek adına bu öznelikteki örnekler, **üst 0.5'lik yakın değerlerine yuvarlanarak gruplandırılmış ve nominal hale getirilmiştir**. Aynı sırada App özneliği yalnız ve yalnızca dosyaların isimlerini içerdiği için ve karar algoritmasına bir etkisi olmayacağı için yok sayılmıştır. Bir yandan da veri setindeki bazı outlier datalar temizlenmiş ve balanslanmış, duplicate olan örnekler arındırılmıştır. Son aşamada **963** örneğe kadar indirgenmiştir.

Algoritmalar

Veri seti ve test seti sınıflandırma algoritmalarına sokulduğunda lazy.Ibk algoritmasında 68%'lik bir başarı elde edilmiştir. Bu algoritma **K-nearest neighbours classifier** algoritmasına denk düşmektedir. Amacımız kodlamada da en az 68%'lik accuracy değerini yakalamaktır. Ayrıca ilgili algoritmanın karışıklık matrisi de şekilde görüldüğü üzere verilmiştir.

Classifier output

Correctly Classified Instances	68	68	%
Kappa statistic	0.3826		
Mean absolute error	0.1077		
Root mean squared error	0.3256		
Relative absolute error	62.9275	%	
Root relative squared error	114.9401	%	
Total Number of Instances	100		

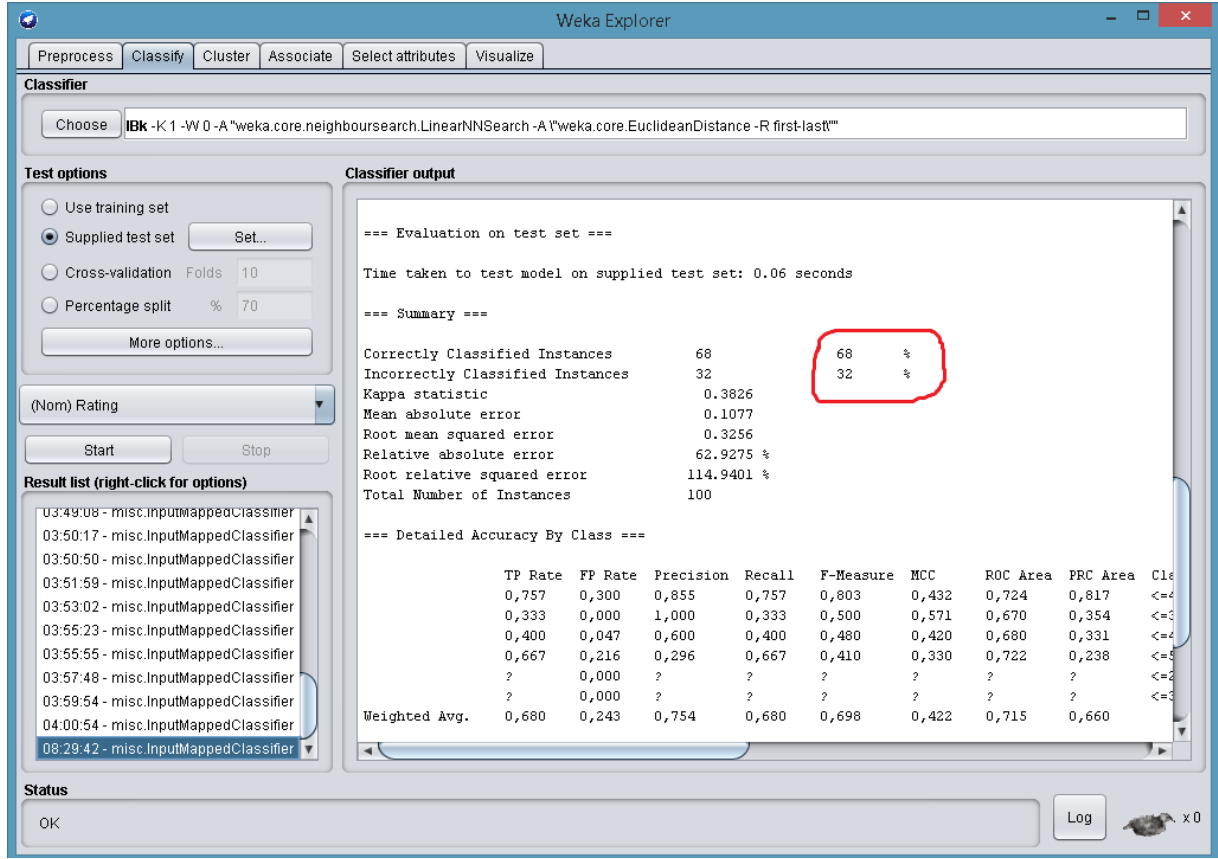
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,757	0,300	0,855	0,757	0,803	0,432	0,724	0,817	<=4.5
	0,333	0,000	1,000	0,333	0,500	0,571	0,670	0,354	<=3.5
	0,400	0,047	0,600	0,400	0,480	0,420	0,680	0,331	<=4.0
	0,667	0,216	0,296	0,667	0,410	0,330	0,722	0,238	<=5.0
	?	0,000	?	?	?	?	?	?	<=2.5
	?	0,000	?	?	?	?	?	?	<=3.0
Weighted Avg.	0,680	0,243	0,754	0,680	0,698	0,422	0,715	0,660	

=== Confusion Matrix ===

a	b	c	d	e	f	<-- classified as
53	0	2	15	0	0	a = <=4.5
2	1	0	0	0	0	b = <=3.5
5	0	6	4	0	0	c = <=4.0
2	0	2	8	0	0	d = <=5.0
0	0	0	0	0	0	e = <=2.5
0	0	0	0	0	0	f = <=3.0

for 1 nearest neighbours WEKA output



The screenshot shows the Weka Explorer interface. The 'Classifier' tab is selected, and the 'IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A "weka.core.EuclideanDistance -R first-last"' classifier is chosen. The 'Test options' section shows 'Supplied test set' selected. The 'Classifier output' section displays the evaluation results on the test set.

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.06 seconds

=== Summary ===

Correctly Classified Instances	68	68	%
Incorrectly Classified Instances	32	32	%
Kappa statistic	0.3826		
Mean absolute error	0.1077		
Root mean squared error	0.3256		
Relative absolute error	62.9275	%	
Root relative squared error	114.9401	%	
Total Number of Instances	100		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0,757	0,300	0,855	0,757	0,803	0,432	0,724	0,817	<=4	
0,333	0,000	1,000	0,333	0,500	0,571	0,670	0,354	<=5	
0,400	0,047	0,600	0,400	0,480	0,420	0,680	0,331	<=6	
0,667	0,216	0,296	0,667	0,410	0,330	0,722	0,238	<=7	
?	0,000	?	?	?	?	?	?	<=8	
?	0,000	?	?	?	?	?	?	<=9	
Weighted Avg.	0,680	0,243	0,754	0,680	0,698	0,422	0,715	0,660	

The 'Result list (right-click for options)' section shows a list of classifiers, with '08:29:42 - misc.InputMappedClassifier' selected. The 'Status' section shows 'OK'.

Kodlama Aşaması

Kodlama için Python programlama dili ve Knn classifier algoritması kullanılmıştır.

Fakat veri setinde kullandığımız nominal değerlere knn uygulayabilmek için sembolik integer dönüşümü yapılmıştır. Sonra da kodlamada da Knn'i uygulayabilmek için, Weka'da olduğu gibi işlemle başlanmıştır.

- App(oyunların isimlerinin tutulduğu) string özniteliği çıkarılmıştır.
- Type özniteliği için Free label'ına '0' ve Paid label'ına '1' dönüşümü yapılmıştır.
- Content Rating için {Everyone,Teen,'Everyone 12','Mature 18'} label'ları sırasıyla {1,2,3,4} olarak değiştirildi.
- Genres için {Card,Action,Arcade,Board,Word,Trivia,Casino,Adventure,Puzzle,Music,Racing,Casual,Strategy,Simulation,Role Playing,Sports} label'ları sırasıyla {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}'yı temsil edecek şekilde dönüştürülmüştür.

Veri setinin son halinin temsili görüntüsü:

```
Reviews,Size,Installs,Type,Price,'Content Rating',Genres,'Last Updated',Rating
207,0.116,50000,0,0,1,1,2011,4.5
1015,0.643,100000,0,0,1,2,2015,3.5
387,1.1,50000,0,0,1,1,2011,4
13304,1.3,1000000,0,0,1,3,2016,4.5
547,1.3,10000,0,0,1,4,2015,5
265,1.3,1000,1,3.49,1,4,2015,5
768,1.4,100000,0,0,1,3,2013,3.5
32849,1.5,1000000,0,0,1,5,2015,4.5
705,1.8,10000,0,0,1,4,2015,5
404,2.2,100000,0,0,1,3,2017,4
278,2.4,10000,0,0,3,6,2017,2.5
69115,2.4,10000000,0,0,1,3,2018,4
531074,2.5,10000000,0,0,1,3,2018,4.5
22191,3,1000000,0,0,1,4,2015,4
58387,3.3,1000000,0,0,1,5,2013,4.5
375996,3.3,10000000,0,0,1,4,2018,4.5
783,3.4,100000,0,0,1,6,2014,4
42182,3.4,1000000,0,0,1,3,2018,4
211,3.6,50000,0,0,1,2,2018,4.5
4575,3.8,500000,0,0,1,3,2017,3.5
```

Görüldüğü üzere tüm örnekler ve öznitelik label'ları sayısal ifadeler etmektedir. Son attribute yine class label'larıdır.

knnClassifier.py dosyasına ait Kodlar:

```
import numpy as np

import operator

import pandas

#bu kısımda csv dosyası okunuyor

def load_csv(filename):

    dataset = np.array(pandas.read_csv(filename,sep=','))

    for i in range(len(dataset)):

        dataset[i] = [float(x) for x in dataset[i]]

    return dataset

#bu kısımda euclidean distance hesaplanıyor

def euclideanDistance(instance1, instance2, length):

    distance = 0

    for x in range(length):

        distance += pow((instance1[x] - instance2[x]), 2)

    return np.sqrt(distance)

#en yakın k tane komşusu bulunuyor

def find_knn(k,item,data):

    dist=[]

    for i in range(len(data)):

        d=euclideanDistance(item,data[i],len(data[0])-1)
```



```
        dist.append((i,d))
    dist.sort(key=operator.itemgetter(1))
    return dist[0:k]
```

#knnClassifier uygulanıyor

```
def knnClassifier(X,Y,k=1):
    acc = 0
    flag = 0
    for y in Y:
        idx = find_knn(k,y,X)
        for i in idx:
            #print(i,'|',X[i[0]][-1],'|',y[-1])
            if(X[i[0]][-1]==y[-1]):
                flag+=1
            #print(flag , k)
        if(flag>=int(k/2)+1):
            acc+=1
        flag=0
    return acc /len(Y)
```

```
X,Y=load_csv('playStoreGamesTrain.csv'),load_csv('playStoreGamesTest.csv')
```

```
print('accuracy is: %',knnClassifier(X,Y,1)*100)
```

Ekran Görüntüleri:

✓ for_1_nearest_neighbours_output

```
—»for x in range(length):
—»—»distance += pow((instance1[x] - instance2[x]), 2)
—»return np.sqrt(distance)

def find_knn(k,item,data):
    dist=[]
    for i in range(len(data)):
        d=euclideanDistance(item,data[i],len(data[0])-1)
        dist.append((i,d))
    dist.sort(key=operator.itemgetter(1))
    return dist[0:k]

def knnClassifier(X,Y,k=1):
    acc = 0
    for y in Y:
        idx = find_knn(k,y,X)
        for i in idx:
            #print(i, '/',X[i[0]][-1], '/',y[-1])
            if(X[i[0]][-1]==y[-1]):
                acc+=1
    return acc /len(Y) / k

Y,X=load_csv('edited2.csv'),load_csv('edited2Test.csv')

print('accuracy is: %',knnClassifier(X,Y,1)*100)
```

accuracy is: % 50.36344755970924

k=1 değeri için başarımlar değeri 73% olarak ölçüldü.

✓ for_3_nearest_neighbours_output

```
def euclideanDistance(instance1, instance2, length):
    —»distance = 0
    —»for x in range(length):
    —»—»distance += pow((instance1[x] - instance2[x]), 2)
    —»return np.sqrt(distance)

def find_knn(k,item,data):
    dist=[]
    for i in range(len(data)):
        d=euclideanDistance(item,data[i],len(data[0])-1)
        dist.append((i,d))
    dist.sort(key=operator.itemgetter(1))
    return dist[0:k]

def knnClassifier(X,Y,k=1):
    acc = 0
    for y in Y:
        idx = find_knn(k,y,X)
        for i in idx:
            #print(i, '/',X[i[0]][-1], '/',y[-1])
            if(X[i[0]][-1]==y[-1]):
                acc+=1
    return acc /len(Y) / k

Y,X=load_csv('edited2.csv'),load_csv('edited2Test.csv')

print('accuracy is: %',knnClassifier(X,Y,3)*100)
```

accuracy is: % 50.778816199376955

k=3 değeri için başarımlar değeri 58% olarak ölçüldü.

✓ for_5_nearest_neighbours_output

```
—>for x in range(length):
—>—>distance += pow((instance1[x] - instance2[x]), 2)
—>return np.sqrt(distance)

def find_knn(k,item,data):
    dist=[]
    for i in range(len(data)):
        d=euclideanDistance(item,data[i],len(data[0])-1)
        dist.append((i,d))
    dist.sort(key=operator.itemgetter(1))
    return dist[0:k]

def knnClassifier(X,Y,k=1):
    acc = 0
    for y in Y:
        idx = find_knn(k,y,X)
        for i in idx:
            #print(i, '{',X[i[0]][-1], '{',y[-1])
            if(X[i[0]][-1]==y[-1]):
                acc+=1
    return acc /len(Y) / k

Y,X=load_csv('edited2.csv'),load_csv('edited2Test.csv')

print('accuracy is: %',knnClassifier(X,Y,5)*100)
```

accuracy is: % 51.422637590861896

k=5 değeri için başarımlar değeri 54% olarak ölçüldü.

Sonuç

Sonuç olarak k için birçok değer denenmiş ve en yüksek oran veren değer k=1 olarak belirlenmiştir ve 73%'lik accuracy ölçülmüştür Sistemin başarısı, puanı doğru grupta tahmin edilen oyun ve bu oyunun gerçek puanı baz alınarak hesaplanmıştır.

Kaynaklar

Kaynak olarak da yardımı için internetteki çeşitli siteler ve bilgi toplanması için de Google Play Store mağazası kullanılmıştır.

SON