MARKOV ZİNCİRLERİ KULLANILARAK YENİ BESTELER ÜRETİLMESİ VE ÜRETİLEN BESTELERİN SINIFLANDIRILMASI

Hakan GÜL

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Aralık, 2015

JÜRİ VE ENSTİTÜ ONAYI

Hakan GÜL'ün **"Markov Zincirleri Kullanılarak Yeni Besteler Üretilmesi ve Üretilen Bestelerin Sınıflandırılması"** başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 28.07.2015 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

		Adı Soyadı	**İmza**
Üye (Tez Danışmanı) :		**Yard. Doç. Dr. Muzaffer DOĞAN**	**……..**
Üye	**:**	**Yard. Doç. Dr. Mustafa ATANAK**	**……..**
Üye	**:**	**Yard. Doç. Dr. İbrahim YAKUT**	**……..**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …….. tarih ve …….. sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

MARKOV ZİNCİRLERİ KULLANILARAK YENİ BESTELER ÜRETİLMESİ VE ÜRETİLEN BESTELERİN SINIFLANDIRILMASI

Hakan GÜL

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Muzaffer DOĞAN

2015, 65 sayfa

Algoritmik besteleme, bilgisayar yardımıyla yeni besteler üretmektir. Algoritmik bestelemede yaygın olarak kullanılan yöntemlerden biri de Markov zincirleridir. Bu tezin ilk aşamasında, çok sayıda ve birbirinden farklı yeni besteler üretmeyi amaç edinen bir lisans tez çalışmasındaki (yazarın tezi) program kullanılmış ve buna ek olarak yeni bir program geliştirilmiş ve ikinci aşamasında ise, ilk aşamada kodlanan her iki programın ürettiği bestelerin çok iyi olmayanlarının elenmesini ve böylece dinleyiciye daha çok güzel beste sunulmasını amaç edinen sınıflandırma işlemleri yapılmıştır. İlk aşamadaki programda, Markov zincirleri arasında yaratıcılığa en elverişli algoritma olan birinci dereceden Markov zincirleri algoritması; üretilen bestelerin kalitelerini arttırmak için, buna benzer çalışmalarda kullanılabilecek şekilde geliştirilmiştir. İkinci aşamadaki sınıflandırma çalışmalarında ise on kişilik değerlendiricilerin, ilk aşamada üretilen bestelerin 40 adedine yaptığı değerlendirmeler veri olarak kullanılmıştır. En sonunda farklı deneylerden elde edilen sonuçlar analiz edilmiş, iki çalışma (bir lisans tezi çalışması ve bu çalışma) kıyaslanmış ve ilerideki çalışmalar için daha ne tarz geliştirmeler yapılabileceği irdelenmiştir.

Anahtar Kelimeler: Algoritmik besteleme, Markov zincirleri, Sınıflandırma.

ABSTRACT

Master of Science Thesis

PRODUCING NEW MUSICAL COMPOSITIONS USING MARKOV CHAINS AND CLASSIFYING THE COMPOSITIONS

Hakan GÜL

Anadolu University

Graduate School of Sciences

Computer Engineering Program

Supervisor: Assistant Prof. Dr. Muzaffer DOĞAN

2015, 65 pages

Algorithmic composition means producing new compositions with a computer. A common method used in algorithmic composition is Markov chains algorithm. In the first stage of this thesis, a program used which was made in a license thesis (writer's thesis) and additionally it improved and in the second stage, classification processes made whose aim was to eliminate the bad compositions produced by two programs in the first stage, for offering the listener a lot of good compositions. In the first stage; in the program, first degree Markov chains algorithm used and modificated, which has the most creativity feature in Markov chains algorithms, to improve the quality of the compositions as it can be used in the similar studies. It is observed that it improves the ratio of production of the good compositions. In the second stage; evalutions, made by 10 evaluators for 40 compositions produced in the first stage, were used as data during the classifications. In the last part, different results, obtained by different experiments, were analyzed and 2 studies were compared and any developments that can be done in the future was examined.

Keywords: Algorithmic composition, Markov chains, Classification.

TEŞEKKÜR

Yüksek lisans tez çalışmam boyunca tezimin olması gerektiği gibi özgür, esnek ama sağlıklı yürümesini sağlayan tez danışmanım Yard. Doç. Dr. Muzaffer DOĞAN'a teşekkürü bir borç bilirim.

Tezde kodlanan programın çalıştırılması sonucunda üretilen besteleri değerlendirmek için zaman ayıran ve bu tezdeki çalışmaya katkı sağlayan herkese teşekkürlerimi sunarım.

Anadolu Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünde görev yapmakta olan tüm hocalarımdan ve çalışma arkadaşlarımdan memnuniyetimi belirtirim.

Tercih ettiğim bölümün ve şu ana kadarki çalışma hayatımın bilgisayar mühendisliği dalında olmasının tek sebebi annemi ve babamı burada anlayışla ve de sevgiyle anmayı bir borç bilirim.

Hakan GÜL

Aralık, 2015

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Müzik çeşitli şekillerde tanımlanabilir. Bir tanımında müzik; duyguların sesler yardımıyla düzenlenmesi sanatıdır (Akbulut, 2006). İnsanların bazen eğlenmek bazense farklı ruh hallerine girmek için dinlediği müzik, konserlerde, şarkıcıların albümlerinde, reklamlarda, defilelerde ve bunun gibi pek çok alanda kullanılmaktadır.

Müzik çok önemli bir unsurdur. Bir önceki paragrafta bahsedilen mecralarda kullanılması bir yana, toplumun iyi veya kötü dönüşümünü belirleyen temel faktörlerden biridir. Zira iyi niyetle üretilen şarkılar bir de güzel bestelenmişse, sosyolojik olarak toplumun gelişmesine doğrudan etki eder. Tam tersi olarak; kötü niyetle üretilen müzikler bir de kötü bestelenmişse, sosyolojik olarak toplumun gerilemesine doğrudan etki eder (Günindi, Ersöz, 2002).

Müzik birçok temel parçadan oluşur. Bunların ilerleyen kısımlarda detaylı açıklamaları ve hangilerinin neden bu tezde kullanıldığı veya kullanılmadığı belirtilmiştir. Burada ise önemli bir başlangıç bilgisi olarak sadece nota değerleri ve nota uzunluklarının tek bir enstrümanla (rock gitar sesiyle) kullanıldığını belirtmekte yarar vardır. Bu tezin temel çalışma alanı algoritmik bestelemedir ve tek bir enstrümanla icra edilen ve ilerleyen kısımlarda belirtilecek notaların kaydedildiği ses dosyalarının kullanımı gerekli ve yeterlidir.

Algoritma, bir sonuca ulaşmak için yapılacak bir dizi işlemler veya kurallar dizisidir. Besteleme ise adı üstünde, sanatın müzik dalında yeni besteler üretme işidir. Algoritmik besteleme ise, yeni besteler üretirken algoritmaları kullanmaktır. Algoritmalar, özellikle ilk zamanlarda yaygın olarak kâğıt üstünde elle yazılarak bir besteci tarafından düzenlenebilmekteydi. Sonraları bilgisayarların yaygınlaşmaya başlamasıyla birlikte, beste yapmak için sadece bilgisayar kullanma veya bilgisayar destekli çalışma alternatifleri doğdu. Böylece algoritmik besteleme, beste yapmak için kâğıt üstünde çalışmaktan ziyade bilgisayarlardan faydalanmak anlamında kullanılmaya başlanmıştır (Simoni, M. 2003).

Algoritmik bestelemeden faydalanan çeşitli besteciler, film-dizi yapımcıları ve reklamcılar vardır. Diğer bir deyimle profesyonel müzik yapımında

bu alanın bestecilere, film-dizi yapımcılarına ve reklamcılara oldukça önemli yararları olduğu bilinmektedir. Faydalanan sektörler genişletilebilir.

Bu tezdeki çalışmanın önemi, bilgisayar yardımıyla beste üretiminde yeni bir yaklaşım; yani daha önceden araştırılan çalışmalarda gerçekleştirilen tarzlardan oldukça farklı bazı küçük iyileştirmelerin uygun bir şekilde gerçekleştirilmesinde yatmaktadır.

Algoritmik bestelemede kullanılan belli başlı yöntemler çeşitli şekillerde sınıflandırılmaktadır. Bu sınıflandırmalardan birinde bunlar Markov zincirleri, sembolik ve bilgi tabanlı sistemler, gramerler, yapay sinir ağları, evrimsel ve diğer popülasyon tabanlı metotlar, özbenzerlik ve hücresel otomata kategorilerine ayrılmaktadır (Fernández, J. D., & Vico, F. 2013).

Algoritmik bestelemede yaygın olarak kullanılan yöntemlerden biri de Markov zincirleridir. Bu tezdeki çalışmanın konusu da Markov zincirleri ile algoritmik bestelemedir.

Markov zincirleri algoritmasını algoritmik bestelemede kullanma konusunda pek çok çalışma bulunmaktadır. Bunlardan edinilecek tecrübelerle şöyle bir durum keşfedilmiştir: Derece sayısı arttıkça yaratıcılık düşmekte ve algoritmada giriş bestesi olarak kullanılan bestelere daha da benzerlik gösteren yeni besteler üretilmektedir. Bu durum, müzik için dezavantajdır. Bu sebeple Markov zincirlerine yaratıcılık açısından bakıldığında en elverişli olan birinci dereceden Markov zincirleri ile bu çalışma gerçekleştirildi (Nierhaus, G. 2009). Markov zincirlerinin algoritmik bestelemeye daha uygun olacağı şekilde, Markov zincirleri algoritmasında geliştirme yapıldı. Birinci dereceden Markov zincirleri algoritmasının algoritmik bestelemede kullanılması konusunda, daha yüksek derecedeki Markov zincirleri algoritmalarına göre bir dezavantajı bulunmaktadır; o da üretilen bestelerin fazla yaratıcı şekilde üretilmesi diğer bir deyimle uçuk kaçık besteler de çıkabilmesidir (Nierhaus, G. 2009). Aslında buna dezavantaj denmiştir ama, uçuk kaçık olduğu düşünülen besteler de nihayetinde yeni bestelerdir ve üstelik bir kısmı çok kişi tarafından beğenilebilir. Yine de birinci dereceden Markov zincirleri algoritması kullanılarak üretilen bestelerde insanlar tarafından iyi bulunmayanları eleme mantığı, bu çalışmada esas alınmıştır. Sınıflandırma algoritmaları ile iyi bir sınıflandırma yapılabilirse, algoritmadan

çıkacak yeni bestelerin kötü olarak sınıflandırılanlarının -yani çoğunluğun kanaati itibariyle veya uzmanlar tarafından kötü bulunacağı düşünülenlerinin- elenmesi ve iyi bestelerin kullanıcıya dinlettirilip kullanıcının daha az yorulması ve daha az vakit kaybetmesi ve hatta iyi olacağını düşündüğü bestelerde daha tutarlı karar vermesi sağlanabilir.

Bu çalışmanın alan için iki öneminin var olduğu düşünülmektedir. Birincisi, bu çalışmada Markov zincirleri algoritması; üretilen bestelerin kalitelerini arttırmak için, buna benzer çalışmalarda kullanılabilecek şekilde geliştirilmiştir ve iyi beste üretim oranının arttırıldığı gözlemlenmiştir.

İkinci önemi ise, önceki bahsedilen aşamaya ek olarak bir sonraki aşamada değerlendiricilerden faydalanmaktır. Oldukça çok sayıda çalışma incelenmiş ve değerlendiricilerin değerlendirmelerinin bu çalışmadakine benzer bir sınıflandırmaya tâbi tutularak geliştirme yapılmasına, başka yöntemlerde (mesela yapay sinir ağları ile algoritmik bestelemede) rastlanmış, ancak Markov zincirleri ile algoritmik bestelemede rastlanamamıştır.

Tezin sonraki bölümleri şu şekilde organize edilmiştir: Bölüm 2'de algoritmik besteleme için gerekli ve yeterli unsurların tercihi; algoritmik bestelemenin bir örnekle anlatımı; Markov zincirleri; sınıflandırma algoritmalarına genel bir bakış; yapay sinir ağları ile sınıflandırmaya genel bir bakış; Naive Bayes ile sınıflandırmaya genel bir bakış; eğitim ve test verilerinin seçimi; algoritmik besteleme alanında yapılan çalışmalar ile ilgili açıklamalar yer almaktadır. Bölüm 3'te giriş verisi olarak kullanılabilecek bestelerin formatı ve özellikleri; giriş verisi olarak program tarafından önerilen 8 bestenin yapısı ve özellikleri; Markov zinncirleri algoritmasının iyileştirilmesi ile daha iyi besteler üretilmesi; değerlendiricilere sunulacak beste üretiminin tasarlanması ve kodlanması ile ilgili açıklamalar yer almaktadır. Bölüm 4'te değerlendirici kısmının kodlanması ve çalıştırılması; her bir değerlendiricinin değerlendirmeleri üzerinde yapılan sınıflandırma işlemleri; değerlendiricilerin genel değerlendirmeleri üzerinde yapılan sınıflandırma işlemleri ile ilgili açıklamalar yer almaktadır. 5. bölümde ise sonuçlar yorumlanmış ve gelecek çalışmalar için göz önünde bulundurulmasının fayda sağlayacağı düşünülen hususlar ifade edilmiştir.

2. ÖN BİLGİLER VE İLGİLİ ÇALIŞMALAR

2.1. Algoritmik Besteleme İçin Gerekli ve Yeterli Unsurların Tercihi

Müzikte en önemli olan kısım, bestedir denebilir. Bir bestede de en önemli kısım nota değerleri ve nota uzunlukları olduğu rahatlıkla gözlemlenebilir. Müziğin farklı yapısal özellikleri üzerinde çalışmalar yapılmışsa bile, dinlenilen bir parçanın müzikal kalitesini belirlemedeki en önemli faktör olan nota değerlerini ve notaların uzunluklarını hesaba katmanın hem yeterli hem de daha avantajlı olduğu varsayılabilir. Diğer bir deyişle müziksel yapıları oluşturan ses perdesi, ritim, ses tınısı, ses seviyesi, akorlar, çok seslilik gibi aslında, nota değerlerinde ve uzunluklarında değiştirmeler yapılarak da ulaşılabilecek; veya nota değerlerinde ve uzunluklarında değiştirmeyle alâkası olmayacak ama bestenin kalitesini hiç etkilemeyecek müziksel yapılar algoritmaya yük getireceği için, tümünün algoritmada göz ardı edilmesinin avantaj olduğu kolaylıkla öngörülebilir. Bu yüzden bu çalışmada sadece nota değerleri ve nota uzunlukları kullanılarak çalışılmıştır. Bu paragrafta bahsedilen diğer müziksel yapıların bu tez çalışmasında neden kullanılmadığı şu şekilde mercek altına alınabilir:

Perde, aynı notanın farklı kalınlıklardaki karşılığıdır. Ses perdesindeki değişiklikler, nota değerlerinin değiştirilmesi ile elde edilebilir. Basit bir dille örnek vererek ifade etmek gerekirse: Kalın RE notasının perdesinin bir üst perdesi demek daha kalın RE notasıdır veya ince LA notasının perdesinin bir alt perdesi daha ince LA notasıdır ve nota değerlerine tekabül eden sayıdan belli bir sayı çıkarılarak elde edilebilir.

Ritim değişiklikleri de nota uzunluklarına tekabül eden sayıların değiştirilmesiyle olanaklı hâle gelebilir. Meselâ ritmin iki kat hızlanması, tüm notaların uzunluklarının yarı yarıya azaltılmasıyla sağlanabilir.

Ses tınısı, aynı sesin farklı müzik aletleriyle çalınmasında, müzik aletinden kaynaklanan ses farklılığını ifade eder. Ses tınısının değişmesi besteyi değiştirmez. Dolayısıyla bu çalışma içerisinde üzerinde durulmasına gerek yoktur.

Ses seviyesinin elbette beste ile uzaktan yakından bir ilgisi yoktur. Sonuçta bir beste istenilen yüksek veya alçak seste dinlenebilir. Ancak bazı

notalarda ses seviyesinin yükseltilmesi gerekirse, kullanıcı bestenin üretiminden sonra istediği notaların yüksek sesle vurgulanma durumunu belirleyebilir ve bu sebeple bu çalışmada ele alınmasına hiçbir gerek yoktur.

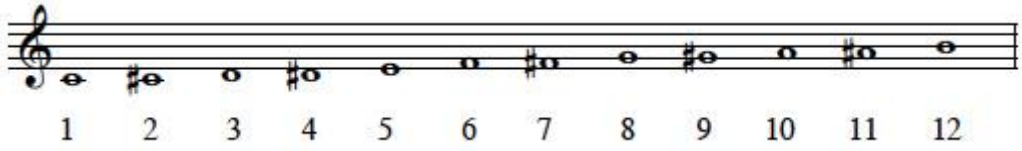
Akor, çok sesin bir arada tınlamasıdır. Genelde bir çok tarzda besteye eşlik etmesi amacıyla kullanılabilir. Akorlar, müzik tarzına göre önemlidir veya değildir. Caz ve klasik müzik gibi müzik tarzlarında bulunması gerekebilir ancak pop veya başka herhangi bir müzik tarzında göz ardı edilebilir. Daha doğrusu çoğu tarzda, beste üretildikten sonra besteyle uyumlu akorlar üretilebilir ve bu çalışmada üzerinde durulmasa da fazla eksikliği hissedilmez; sonuçta en önemli olay kulağa hoş gelmesi istenen ve birbirinin ardınca gelen tekli notalardır.

Çok seslilik, birden çok müzik aletinin birlikte çalınması ile oluşur. Bestenin hangi müzik aletiyle ve nasıl çalınması gerektiği bestecinin işi değildir ve dolayısıyla bu çalışmada göz ardı edilmiştir.

Sonuç olarak sadece nota değerleri ve nota uzunluklarının tek bir enstrümanla (rock gitar sesiyle) kullanılması, bu çalışmada ulaşılmak istenen beste üretme amacına hizmet eden gerekli ve yeterli unsurlardır.

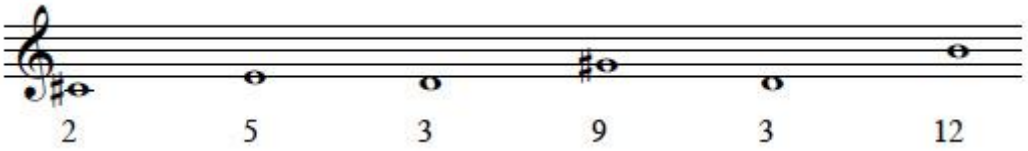
2.2. Algoritmik Bestelemenin Bir Örnekle Anlatımı

Algoritmik bestelemeyi daha somut anlatmak için bir örnek verilebilir. Bir zar oyunu olduğunu düşünülsün. Bu oyunda 12 yüzlü bir zar kullanıldığı varsayılsın -ki istense bu bilgisayarda yazılacak bir kodla da tasarlanabilir (Simoni, M. 2003).



Şekil 2.1. Porte üzerinde notalara değer atanması (Simoni, M. 2003)

Bu zar oyununa başlamak için öncelikle şekil 2.1'deki gibi müzik notaları puanlandırılsın. Burada en baştaki DO notasının değeri 1, ikinci sıradaki RE notasının 2, üçüncü sıradaki Mİ notasının 3… olsun. 6 kez rastgele atılan 12 yüzlü zarlar da sırasıyla 2, 5, 3, 9, 3, 12 olsun (Simoni, M. 2003).

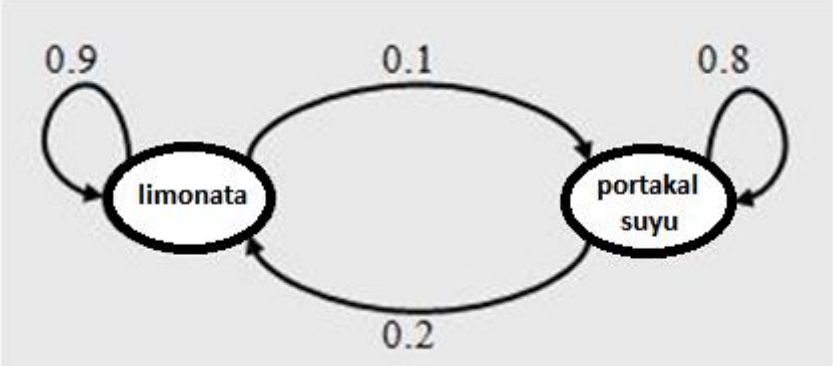


Şekil 2.2. Rastgele 6 kez atılan 12 yüzlü bir zarın atılması sonucundaki muhtemel sayısal değerler ve onlara tekabül eden notalar (Simoni, M. 2003)

Bu zar oyununda, zarın üstüne gelen muhtemel 6 değere tekabül eden notalar şekil 2.2'de gösterilmiştir. Bu, algoritmik bestelemenin izahı için oldukça basit şekilde uygulanmış bir başlangıç örneğidir (Simoni, M. 2003).

2.3. Markov Zincirleri

Markov zincirleri, en özetle bir durumdan sonra hangi durum veya durumların geleceğinin önceden belirlenmesidir. Bu durum belirlenirken ardışık durumlara geçme olasılıkları göz önünde bulundurulur. 0'dan büyük bir *n* doğal sayısının tanımlı olduğu varsayılırsa, o anki durumdan *n* adım sonraki tüm var olan durumların tespiti yapılır ve bu olasılıklara göre geçiş yapılır. Eğer *n* adım sonra benzer bir durum geçişi varsa, aynı olasılıkta yeni duruma geçilir. *n* sayısı dereceyi belirtir. Yani *n* sayısı 1 ise algoritmanın 1. dereceden Markov zincirleri; 5 ise 5. dereceden Markov zincirleri olduğu anlaşılır (Nierhaus, G. 2009).



Şekil 2.3. Markov zincirleri algoritmasına göre bir sonraki içecek içme durum geçişleri

Şekil 2.3'teki gibi iki içeceğin nasıl tüketildiğine ilişkin 2. dereceden Markov zincirleri algoritması oluşturulmak istendiğini düşünelim. İçeceğin birinin limonata ötekinin portakal suyu olduğu varsayılsın. Bu iki içecek arasındaki karar olasılıkları okların çevresinde belirtilsin. Yani portakal suyu içen bir kişinin bir sonraki içeceğinin yine portakal suyu olma olasılığı 0.8 ve limonata olma olasılığı 0.2, benzer şekilde limonata içen birisinin bir sonraki içeceğinin yine limonata olma olasılığı 0.9 ve portakal suyu olma olasılığı 0.1 olarak verilsin.

"Bir kişinin ikinci alış verişinde limonata içme olasılığı nedir?" sorusunu cevaplamak için ikinci dereceden Markov zincirlerini kullanmak gerekir. Bu amaçla kullanılacak Markov zincirleri olasılık matrisi şu şekilde gösterilir:

$$P=\begin{bmatrix}0.9 & 0.1\\ 0.2 & 0.8\end{bmatrix} \tag{2.1}$$

İkinci alışverişteki olasılığın hesaplanması için P matrisinin karesi alınır.

$$P^2=\begin{bmatrix}0.9 & 0.1\\ 0.2 & 0.8\end{bmatrix}\begin{bmatrix}0.9 & 0.1\\ 0.2 & 0.8\end{bmatrix}=\begin{bmatrix}0.83 & 0.17\\ 0.34 & 0.66\end{bmatrix} \tag{2.2}$$

P^2 matrisinin ikinci satır birinci sütundaki elemanı olan 0.34 değeri, kişinin ikinci alışverişinde portakal suyundan limonataya geçme olasılığını vermektedir.

2. adımdaki tüm olasılıkları hesaplamak için bu soruya benzer sorular oluşturulabilir. Bazıları şunlardır:

-Portakal suyu içen bir kişinin ikinci alış verişinde portakal suyu içme olasılığı nedir?

-Bir kişinin ikinci alış verişinde limonata içme olasılığı nedir?

-Bir kişinin iki alış verişinde de portakal suyu içme olasılığı nedir?

Bu soruların cevabına da (2.2)'deki P^2 matrisinden ulaşılabilir. 3. dereceden bir Markov zinciri için ise, P^3 matrisi hesaplanır ve 3 adım sonraki olasılık geçişlerine ulaşılabilir.

2.4. Sınıflandırma Algoritmalarına Genel Bir Bakış

Sınıflandırmayı kısaca tabir etmek gerekirse, elimizdeki etiketi; yani hangi sınıfa ait olduğu belli olan eğitim verilerinden yararlanarak aynı tarzdaki ama farklı test verilerini mümkün olan en yüksek doğrulukta sınıflandırmaktır.

Sınıflandırmalar aşağıda verilen örnekler gibi çok çeşitli alanlarda kullanılmaktadır.

- Kredi başvurusu değerlendirme
- Kredi kartı harcamasında sahtekârlık olup olmadığına karar verme
- Hastalık teşhisi
- Ses tanıma
- Karakter tanıma
- Gazete haberlerini konularına göre ayırma
- Kullanıcı davranışları belirleme vs.

Etiket taşımayan verilerde bazı veri gruplarının kendi içerisindeki belli benzerliklere göre gruplandırmalarına kümeleme denir ve sınıflandırmadan, belirli bir etikete sahip olmama yönüyle ayrılır. Bu tezdeki verilerin iki sınıfı olup sınıflarının bilindiği için kümelemeye tâbi tutulmamışlardır.

Bazı sınıflandırıcılar aşağıda verilmiştir.

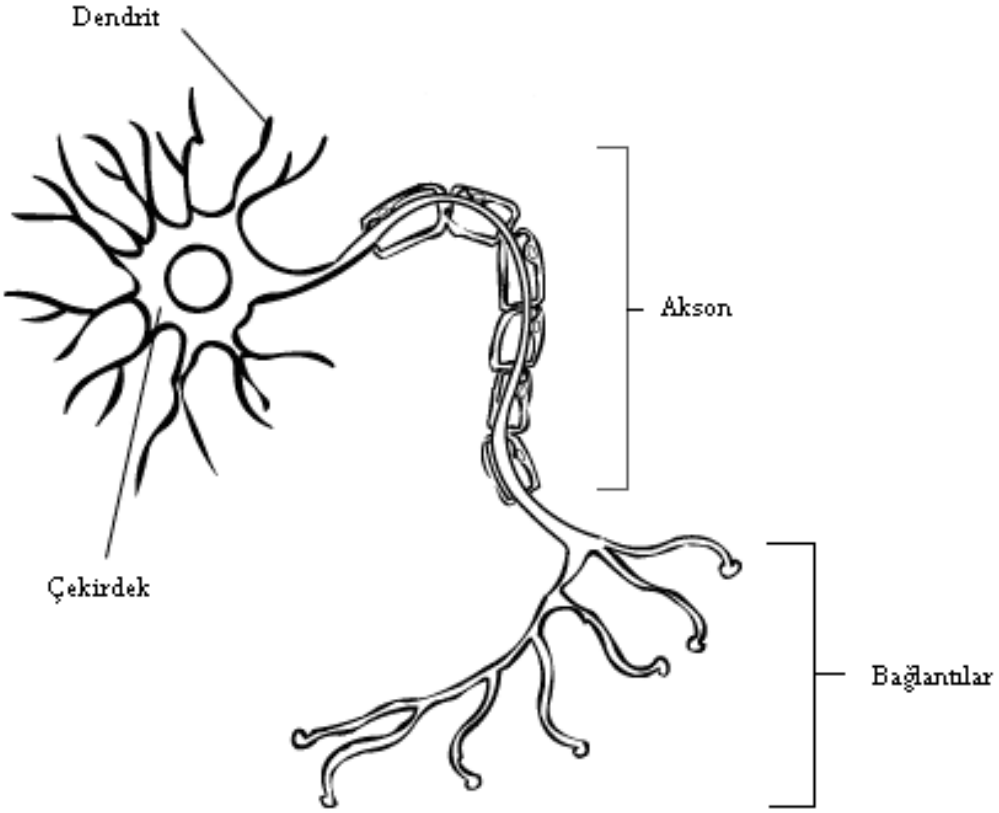
- Karar ağaçları
- Yapay sinir ağları
- Bayes sınıflandırıcılar
- Bayes ağları

Bu çalışma için Weka'da bir çok sınıflandırma algoritması denenmiş ve genel olarak birbirine yakın sonuçlar verdiği gözlemleniştir. (Weka sınıflandırmalara ve başka birtakım işlemlere olanak tanıyan bir programdır (Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. 2009).)

2.5. Yapay Sinir Ağları ile Sınıflandırmaya Genel Bir Bakış

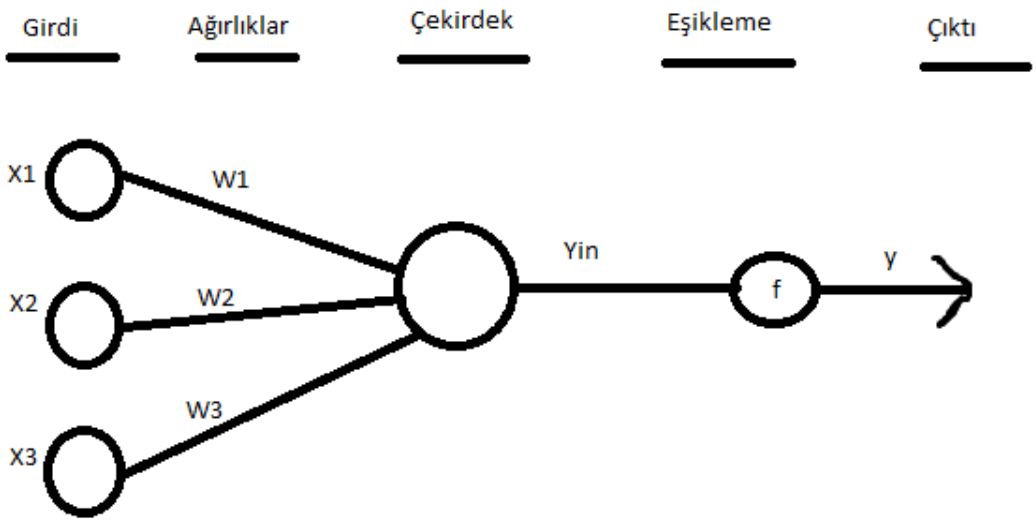
Tezin ikinci aşamasında yapılan sınıflandırmalardan biri de yapay sinir ağları ile sınıflandırmadır. Yapay sinir ağları arasında en çok kullanılan ve bu konuda da etkili olduğu gözlemlenen çok katmanlı yapay sinir ağları kullanılmıştır.

Şekil 2.4'te beyin hücrelerinin yapısı gösterilmektedir. Yapay sinir ağları da ilk olarak bu yapıdan esinlenerek tasarlanmıştır.



Şekil 2.4. İnsan beyninin hücre yapısı (Toraman, U. 2015)

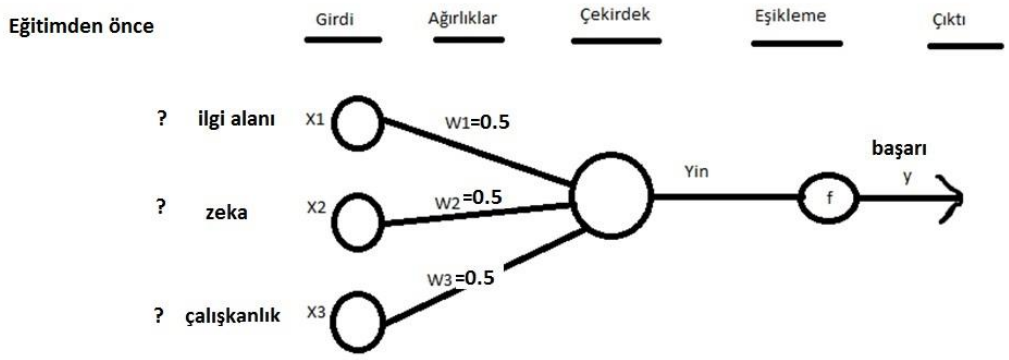
Yapay sinir ağlarının yapısı Şekil 2.5'teki gibidir. Her veri tek tek girilmekte ve her veri girişinde ağırlıklar, öğrenme katsayısına bağlı olarak değişmektedir.



Şekil 2.5. Yapay sinir ağlarının yapısı

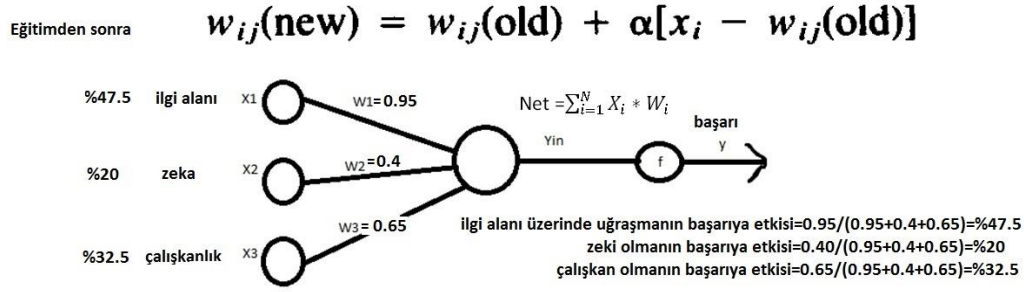
Bir örnek üzerinde yapay sinir ağının nasıl işlediği anlatmak için bir problem ortaya atılsın: 3 temel öğe olarak ilgi alanı, zekâ ve çalışkanlığın başarıya etkisinin ne olduğu bulunmak istenmektedir.

Yapay sinir ağları ile bu soruya çözüm bulmak için Şekil 2.6'daki gibi bir yapı oluşur ki; ilgi alanı, zekâ ve çalışkanlık giriş verileridir.



Şekil 2.6. Bir yapay sinir ağları uygulaması problemi

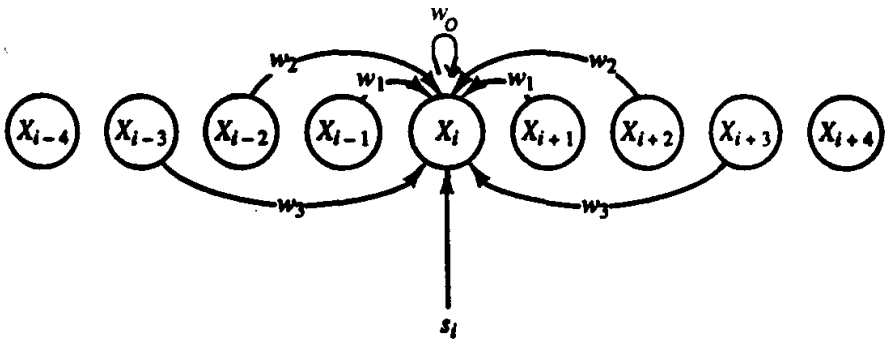
Eğitimden sonra ilgi alanı, zekâ ve çalışkanlık değerlerinin kullanıldığı çeşitli örnekler girilip de ağırlıklar değiştiğinde, yapay sinir ağındaki yeni ağırlıklar Şekil 2.7'deki formüle göre yeni değerlerden oluşur.



Şekil 2.7. Bir yapay sinir ağları uygulaması probleminin çözümü

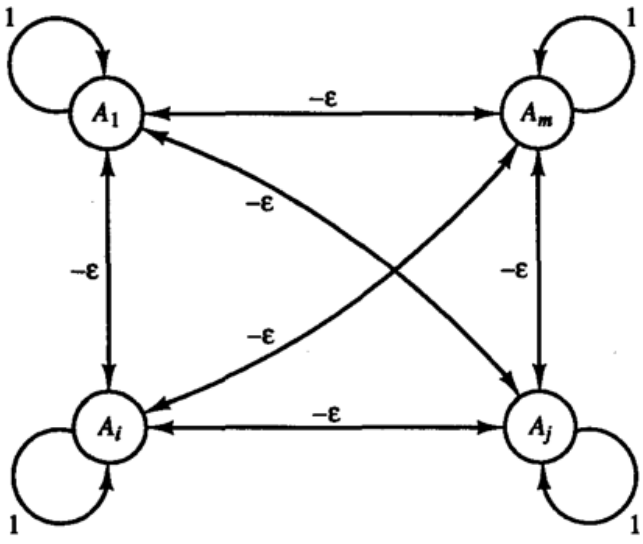
Çeşitli yapay sinir ağları yapıları mevcuttur. Mesela Şekil 2.8'de gösterilen Meksika şapkası bir yapay sinir ağı çeşididir (Mexican hat neural network) ve mantık olarak kullanıcının isteğine bağlı yakınlıktaki hücreler çıktıya pozitif yönde etki ederken yine kullanıcının isteğine bağlı uzaklıktaki hücreler çıktıya negatif yönde etki etmektedir ve bunların dışındaki daha uzaktaki hücreler çıktıya

etki edememektedir (Fausett, L. 1994). Bunlar, sırasıyla savaşmayıp iyi geçinen komşu iki ülkeye ve uzaktan gelip bomba yağdıran uzak bir ülkeye ve de milyonlarca ışık yılı uzakta olduğu için daha birbirini bile tanımayan iki galaksi ülkelerine benzetilebilir.



Şekil 2.8. Meksika şapkası yapay sinir ağı yapısı (West, L. 2012)

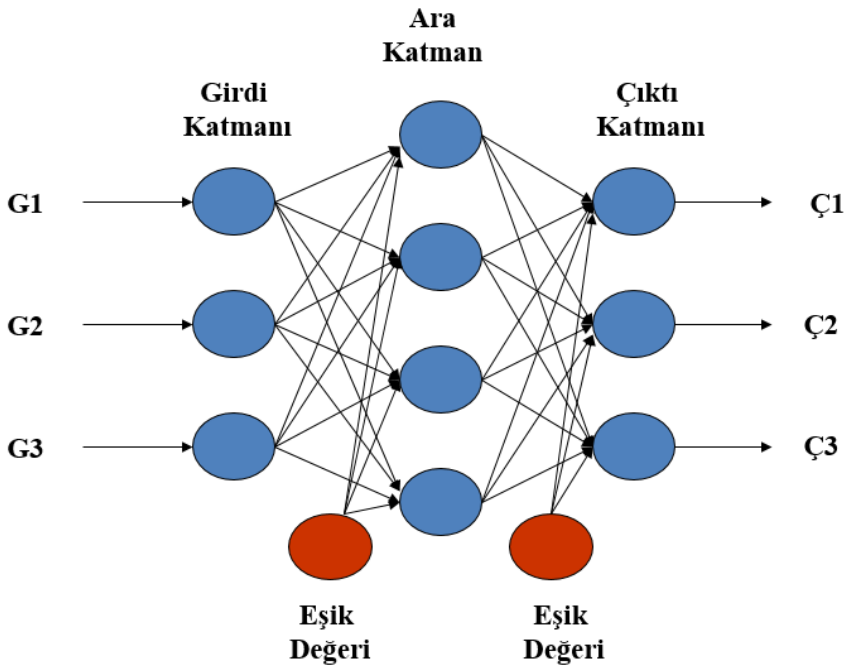
Bazı yapay sinir ağları da Şekil 2.9'daki gibi kendisinin pozitif etki ettiği ve geri kalan tüm hücrelerin negatif etki ettiği, en büyük değeri bulmayı amaç edinen yapay sinir ağlarını (Maxnet neural networks) oluşturur (Fausett, L. 1994).



Şekil 2.9. Maxnet yapay sinir ağı yapısı (West, L. 2012)

Yapay sinir ağlarının oldukça çok çeşidi vardır. Örnekler çoğaltılabilir.

Weka ile sınıflandırma aşamasında, çok katmanlı, etkili bir sınıflandırıcı kullanıldı. Çok katmanlı yapay sinir ağı Şekil 2.10'da gösterildiği üzere, çok katmanlı bir sınıflandırıcıdır.

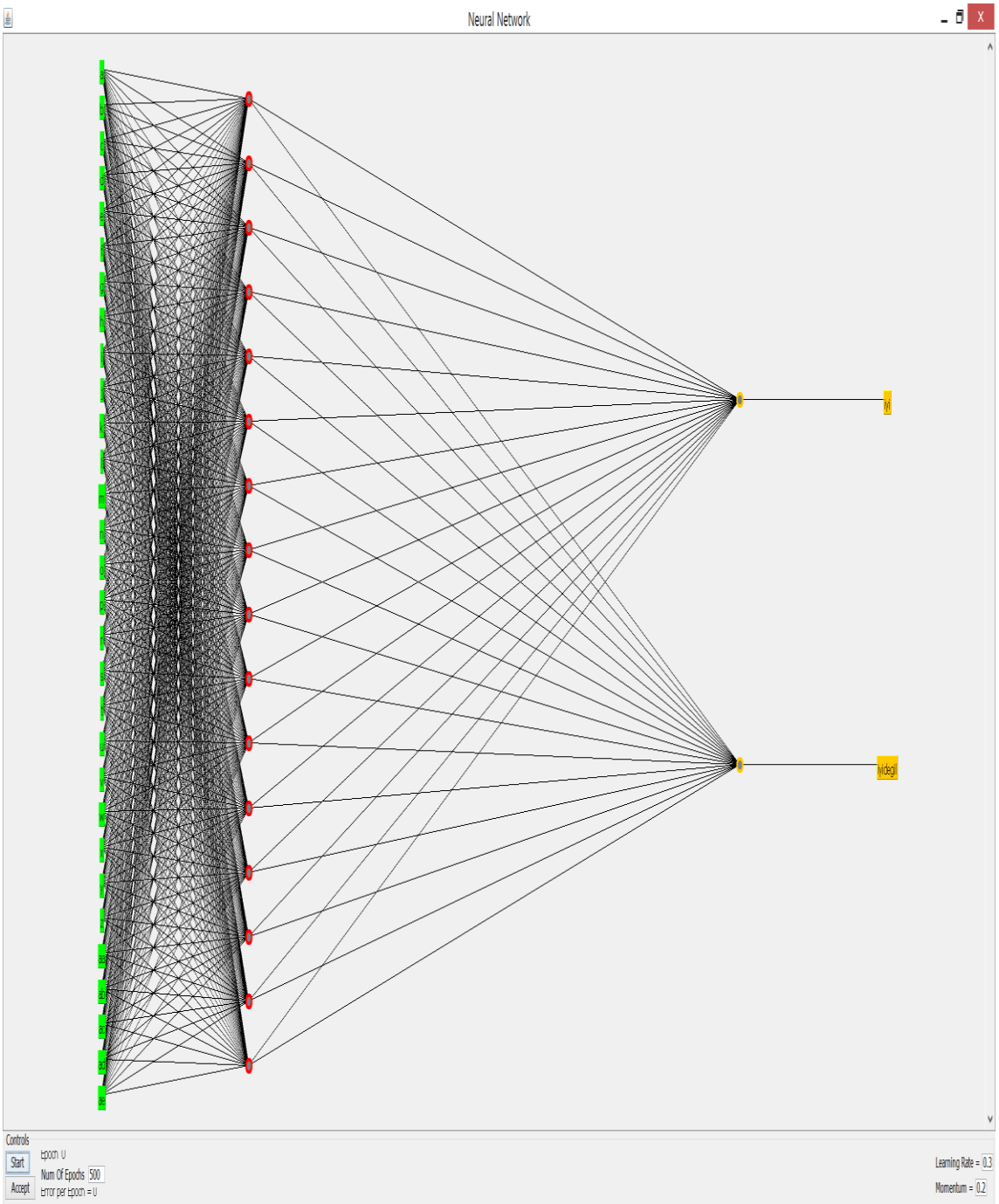


Şekil 2.10. Çok katmanlı yapay sinir ağı yapısı (Onder, E. 2011)

Şekil 2.10'daki çok katmanlı yapay sinir ağı yapısı örnek olması açısından gösterilmiştir. Çok katmanlı yapay sinir ağları en çok kullanılan yapay sinir ağı modelidir ve girdi katmanındaki, ara katmanındaki ve çıktı katmanındaki hücre sayısı elbette yukarıdaki şekildekinden farklı sayıda tasarlanabilir.

Çok kısa bir şekilde açıklamak gerekirse, yapay sinir ağları ile sınıflandırmada her eğitim verisinin sırayla girilip eğitilmesi ve bu eğitime bağlı olarak hücreler arası ağırlıkların değişmesi ve test verilerinin bu yeni ağırlıklara göre sınıflandırması yapılmaktadır. Eğitim ve test verileri çeşitli şekillerde seçilebilir.

Bu çalışmada sınıflandırma için, içinde çeşitli sınıflandırma algoritmalarının bulunduğu Weka programı kullanılmıştır. Şekil 2.11'de sınıflandırmaya örnek vardır.



Şekil 2.11. Çok katmanlı yapay sinir ağları ile sınıflandırma

Şekil 2.11'de rahatlıkla görülebileceği üzere giriş katmanında 30 adet giriş verisinden dolayı, çok katmanlı yapay sinir ağının giriş katmanı 30 hücreden, ara katmanı 16 hücreden ve çıktı katmanı ise 2 hücreden oluşmaktadır. Ağ 500 devir (epoch) çalışmakta ve ağın öğrenme katsayısı 0.3'tür.

2.6. Naive Bayes ile Sınıflandırmaya Genel Bir Bakış

Tezin ikinci aşamasında yapılan sınıflandırmalardan biri de Naive Bayes ile sınıflandırmadır. Bayes teoremine dayalı etkili bir sınıflandırıcıdır ve formülü aşağıdaki gibi gösterilebilir (Leung, K. M. 2007).

P(A) ve P(B) ; sırasıyla A ve B olaylarının olma olasılıkları,

P(A|B) ; B olayı gerçekleştiği durumda A olayının meydana gelme olasılığı,

P(B|A) ; A olayı gerçekleştiği durumda B olayının meydana gelme olasılığı olmak üzere;

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.3}$$

formülü geçerlidir.

2.7. Eğitim ve Test Verilerinin Seçimi

Eğitim ve test verilerinin seçiminde başta çapraz doğrulama (cross validation) ve ayırma (split) olmak üzere çeşitli seçim yöntemleri mevcuttur (Burman, P. 1989).

Ayırma metodu: Bu metotta eğitim ve test verileri istenilen yüzdelikte dilimler halinde ayrılabilir. %66.666 eğitim seti verisi ve %33.333 test seti verisi veya %50 eğitim seti verisi ve %50 test seti verisi olacak şekilde ayrılır gibi.

Çapraz doğrulama metodu: Bu metotta herhangi bir sayıya bölünen veri setinin önce bölünmeyen kısmı eğitilir, sonra bölünen kısmı test edilir ve her bir parça için bu işlemler tekrarlanarak ortalamaları alınır.

Mesela n= 10 olarak seçilmişse (ki bu tezde yapılan çalışmada, bu metoda ve bu sayıya göre veri seçimi yapılmıştır) verinin ilk aşamada ilk %90'lık kısmı eğitilir ve son %10'luk kısmı test edilir, ikinci aşamada ilk %80'lik ve son

%10'luk kısmı eğitilir sonra geri kalan %10'luk kısım test edilir … onuncu aşamada son %90'lık kısmı eğitilir ve ilk %10'luk kısmı test edilir. Bu on aşamadaki, doğruluk oranı ve benzer diğer parametrelerin aritmetik ortalaması yani 10 sayısı için, hepsinin toplamının 10'a bölümü alınarak sonuca ulaşılır.

2.8. Algoritmik Besteleme Alanında Yapılan Çalışmalar

Algoritmik besteleme nihayetinde bilgisayarla beste yapmanın planlandığı bir amacı karşılar. Bu amaca ulaşmak için de belli başlı yöntemler yaygın olarak kullanılmaktadır. Bunlar Markov zincirleri, sembolik ve bilgi tabanlı sistemler, gramerler, yapay sinir ağları, evrimsel ve diğer popülasyon tabanlı metotlar, özbenzerlik ve hücresel otomata gibi kategorilere ayrılmaktadır (Fernández, J. D., & Vico, F. 2013). Bu tez çalışmasında Markov zincirleri kullanılmıştır.

Markov zincirleri pek çok besteci tarafından; bu algoritma ile sayısı oldukça çok beste türetilebilmesi sebebiyle tercih edilmektedir. (Fernández, J. D., & Vico, F. 2013). Markov zincirlerinde kullanılan olasılık matris tablolarının; herhangi birkaç besteden değil de, manuel olarak kulanıcı tarafından oluşturulduğu sistemler de vardır (Ariza, 2006), (Zicarelli, 1987). Bu yaklaşımlar da oldukça önemlidir ancak, dışarıdan müdahale söz konusu olduğu için, iyi bestelerin yanında müziksel kalite açısından çok değer taşımayan oldukça çok sayıda düzensiz nota dizilimleri oluşabilir.

Verbeurgt et al. (2004) yaptığı çalışmada, Markov zincirleri algoritmasını kullanarak modellediği besteleri, yapay sinir ağlarına da uygulamış ve sisteme iyi müzik seçimini öğretmiştir. Band-out-of-the-Box sisteminde de (Thom, 2000) Markov zincirleri yapay sinir ağları ile eğitilmiştir. Bu ve buna benzer çalışmalarda iki yöntemi birlikte kullanmak daha avantajlı olsa bile, yapay sinir ağları, girilen örneklere dayalı olarak eğitilen yapıyı ezberleyen sistemlerdir ve bu tezde uygulanan çalışma gibi değerlendirici faktörünün sisteme dâhil edilmesinin daha büyük avantaj getirmesi söz konusudur. Werner ve Todd (1997) Markov zincirleri üzerine yine farklı bir bakış açısıyla yaklaşmış, evrimsel olarak gelişen

zincirler oluşturmuştur. Thornton (2009) çalışmasında bestelerden gramer benzeri kurallar çıkartmış ve bunları Markov zincirlerine uygulamıştır.

Bilgi tabanlı sistemler, bilginin yapılandırılmış sembollerle ifade edilebildiği sistemlerin tümüdür. Elbette müziksel semboller manipüle edilerek yeni müziksel semboller oluşturulabilir -ki bunlar yeni bestelerdir ve algoritmik bestelemedeki temel amaç da zaten budur.

Rothgeb (1968) doktora tez çalışmasında, on sekizinci yüzyıl müziğine dair yapılmış bilimsel araştırmaları incelemiş ve klasik kuralların eksik ve tutarsız olduğu sonucuna varmıştır. Thomas (1985) besteleri oluşturan notalara uygun akorları Lisp üzerinde, öğrencilerine gösterdiği müzik kurallarına bağlı olarak üretmiştir. Lisp'in kullanıldığı çalışmaların yanında, kendi kendini düzenleyen haritaların kullanıldığı oldukça farklı çalışmalar da yapılmıştır. Walker (1994) nesne tabanlı analiz-sentez motoru yapmıştır ki ilgili çalışmada, bir vokalin seslendirmesi sonucunda uygun caz müzik notaları basılmakta ve bu çalışma, doğaçlama olması dolayısıyla algoritmik bestelemenin çalışma sahasına girmektedir.

Ames ve Domino (1992) kurallar ve Markov zincirlerini birlikte kullanarak değişik popüler müzik tarzlarında besteler üretmiştir. Bunun gibi farklı algoritmalarla bir arada yapılan, algoritmik besteleme üzerinde çeşitli kural tabanlı sistem çalışmaları mevcuttur.

Düzenli diller ile beste üretiminde önemli bir ilk adım olarak, dilin kurallarını tanımlamak gereklidir. Bu alanda yapılan ilk zamanlardaki çalışmalar bu sorunu; müzik kuralları teorilerine dayalı olarak, kâğıt üstünde elle kurallar oluşturarak çözmüştür. Bundan başka bir grup besteden kurallar çıkarma, ağaç yapısı kullanma, evrimsel algoritmalar kullanma gibi çeşitli yöntemler de mevcuttur.

Olasılıklara göre dil kurallarının oluşturulduğu skolastik diller oldukça yaygındır. Ayrıca "A Generative Theory of Tonal Music" kitabı (Lerdahl, Jackendoff, 1985), müzik analizine dilsel bakış açısıyla yaklaşımı ve alanda prestij sahibi olup yüksek puanlandırılması dolayısıyla önemli bir kitap olma

özelliği taşımaktadır. Bu kitaptan ilham alınarak yapılan Pope (1991)'in "T_R Trees" çalışması ve Leach and Fitch (1995)'in "Event Trees" ve Hamanaka (2008)'in "Melody Morphing" çalışmaları vardır. Hamanaka; Lendhal'in iki besteden yeni beste üretilmesi sisteminde, türetilmiş ağaç yapısını değiştirerek geliştirme yapmıştır.

Yapay sinir ağları, insanın doğal beyin yapısından ilham alınarak geliştirilmiş algoritmalardır. Müzik alanında Todd (1989) üç katmanlı bir yapay sinir ağını kullanarak; yapay sinir ağlarının ilk defa algoritmik bestelemede kullanılmasına öncülük etti. Shibata (1991) melodiye en uygun akoru kullanıcıların değerlendireceği bir geri beslemeli yapay sinir ağı üzerinde çalıştı. Onun çalışmasındaki gibi en uygun akor seçimi popüler bir çalışma alanı olmakla birlikte, direkt olarak yeni beste üretmeyi amaç edinen çeşitli çalışmalar da yapılmıştır. Örnek olarak Toiviainen (1995) yapay sinir ağına eğittiği caz tarzında daha önceden var olan melodilerden yeni melodiler üretmiştir.

Yapay sinir ağları ile başka algoritmaların bir arada kullanıldığı çeşitli çalışmalar da mevcuttur. İlk örneklerden biri Hild (1992)'in HARMONET çalışmasıdır. Hild, çalışmasında üç aşamalı bir yapı kullanılmıştır. İlk aşama üç katmanlı bir yapay sinir ağından oluşmaktadır ve çıkış verileri, ikinci aşamadaki kural tabanlı bir kısıtlama algoritmasına girmektedir. Son aşamada ise, ikinci aşamadaki bahsi geçen sistemden çıkan çıkış verileri, yine başka bir yapay sinir ağına giriş verisi olarak girmekte ve bu son aşamada bestenin kulağa daha hoş gelmesi için birtakım notalar eklenmektedir. Başka bir benzer çalışma da Verbeurgt (2004) tarafından yapılmıştır. Verbeurgt çalışmasında önce Markov zincirleri algoritması ile oluşturduğu sisteme gelen giriş verilerinin sayısal nota değerleri, yapay sinir ağlarında eğitildikten sonra belirlenmekteydi. Yapay sinir ağları algoritmik bestelemede oldukça çeşitli şekillerde kullanılmıştır.

Evrimsel algoritmalar şu prensibi benimser: Aday çözümlerin üzerinde değişiklikler yapmak, aralarından seçim yapmak ve yeniden üretmek. Bu yöntem kullanılarak da algoritmik besteleme alanında çeşitli çalışmalar yapılmıştır. Örneğin Marques (2000) direk temsili genotipler kullanarak kısa polifonik

melodiler üretmiştir. Gartland-Jones (2002) önceden var olan iki beste üzerinde evrimsel bir algoritma uygulamıştır.

Algoritmik bestelemede evrimsel metotlar; Markov zincirleri, kural tabanlı sistemler ve hücresel otomata ile bir arada da kullanılmaktadır. Bu şekilde de yapılmış çeşitli çalışmalar vardır.

3. PROBLEM ANALİZİ VE ÖNERİLEN METOT

Algoritmik bestelemede Markov zincirlerini kullanırken şöyle bir durum mevcuttur: Birinci dereceden Markov zincirleri algoritmasının kullanılması ile üretilen bestelerin kendi aralarında ve Markov zincirlerine giriş verisi olan bestelere -bir avantaj olarak- neredeyse hiç benzememekte ancak zaman zaman harika olmayan besteler çıkabilmektedir. Yaratıcılık perspektifinden düşünüldüğünde bu mükemmel bir durumdur. Bununla birlikte Markov zincirlerinin derecesi ne kadar büyürse (mesela 10. dereceden Markov zincirleri gibi) yaratıcılık o oranda azalmaktadır; hatta çok yüksek dereceli Markov zincirleri algoritması sonucu oluşan yeni besteler, giriş verisi olan bestelere oldukça çok benzemektedir. Bu durum müzik açısından elbette ki büyük bir dezavantajdır. Sonuçta amaç çok uçlarda rastgelelik oluşturmadan yaratıcılığı olabildiğince zirveye çıkarmaktır. Bu sebeple bu tezde, yaratıcılığa en elverişli olduğu düşünülen en düşük dereceden yani birinci dereceden Markov zincirleri kullanıldı. (Önemli not: Bu tezdeki çalışmada, (Gül, H. 2010) çalışmasından yararlanılmış ve oldukça geliştirilmiştir.)

Daha önceden de bahsedildiği gibi Markov zincirlerinin tek başına uygulandığı veya çeşitli başka yöntemlerle beraber uygulandığı epey çalışma vardır. Yalnız bu tezde yapılan çalışmadaki değerlendirici faktörünün sisteme dahil edilerek algoritmik bestelemede kullanılmasına dönük, Markov zincirleri ile birlikte kullanıldığı herhangi başka bir çalışmaya rastlanmamıştır ve bu eksikliğin giderilmesi ve kötü bestelerin olabildiğince elenerek nihai amaç olan en iyi bestelerin elde edilmesi; yani bir nevi bir süzgeç oluşturarak bestelerin bu süzgeçten geçirilmesi amaçlanmıştır.

Bu tezde yapılan çalışmada kullanılan Markov zincirleri algoritmasında, daha iyi netice alınacak şekilde geliştirme yapılmış yani algoritmanın yapısında küçük bir değişiklik yapılmıştır. Devamında, daha da iyi netice alabilmek için sisteme değerlendirici sistemi eklemek ve buradan çıkacak sonuçla sistemde çeşitli sınıflandırma algoritmalarıyla sınamalar yapmak suretiyle, değerlendiriciler açısından çok iyi bulunmayan besteleri elemek ve değerlendiriciler açısından çok

iyi bulunan bestelerin müzikalite seviyesinde yeni besteler üretebilmek hedeflenmiştir.

Değerlendirici olarak 10 kişi seçilmiştir. 10 değerlendiriciden de, Markov zincirleri algoritmasına uygun 20 besteye ve Markov zincirleri algoritmasının biraz geliştirilmiş versiyonunun uygulanmasıyla oluşan 20 adet besteye iyi veya kötü olarak değerlendirmede bulunmaları istenmiştir.

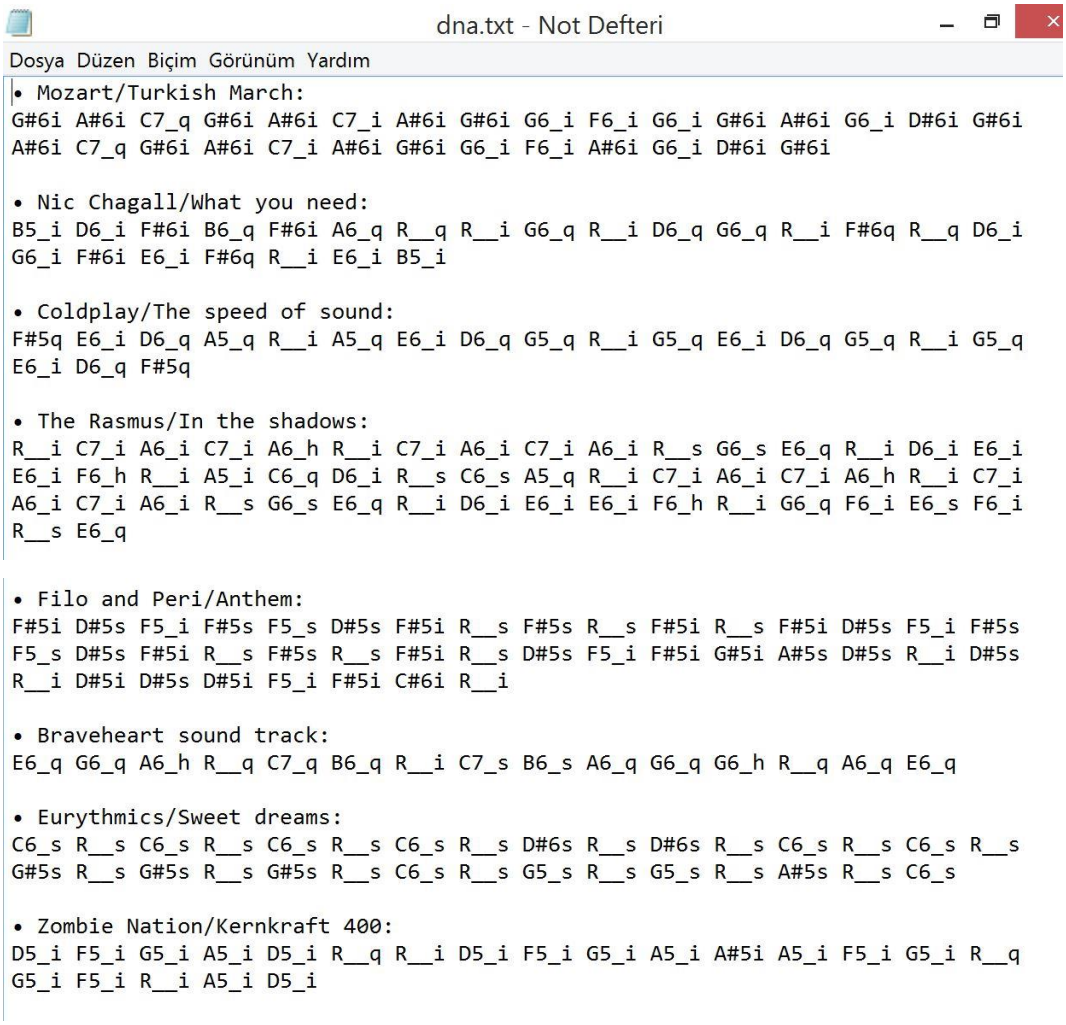
3.1. Giriş Verisi Olarak Kullanılabilecek Bestelerin Formatı ve Özellikleri

Herşeyden önce giriş verileri oluşturulmalıdır ki bu tezdeki proje oluşabilsin. Giriş verisi olarak beste girişi uygun görülmüştür. Tek bestenin Markov zincirlerinde kullanılması fazla fayda sağlamaz. Doğada birçok olay iki elemanlı süreçlerin sonucudur. İkişerli beste girişini kullanmak akıllıcadır. Bu sebeple bu tezde ikişerli beste girişinin kullanılması uygun görüldü. Bu beste çiftlerini seçmek içinse program;

-ya kullanıcının notalarını dışarıdan elle gireceği 2 besteyi

-ya da sisteme kaydedilen ve birbirine benzemeyen ama her biri popüler çeşitli bestelerin kaydedildiği bir kaynak dosyayı kullanır.

Aşağıdaki şekilde, bu tezdeki çalışmada kullanılan ve içeriği Şekil 3.1'de gösterilen "dna.txt" dosyasının içerisinde, nakarat bölümleri kullanılan 8 parça verilmiştir. G#6i A#6i C7_q … gibi aynı formattaki farklı ifadelerden oluşmaktadır.

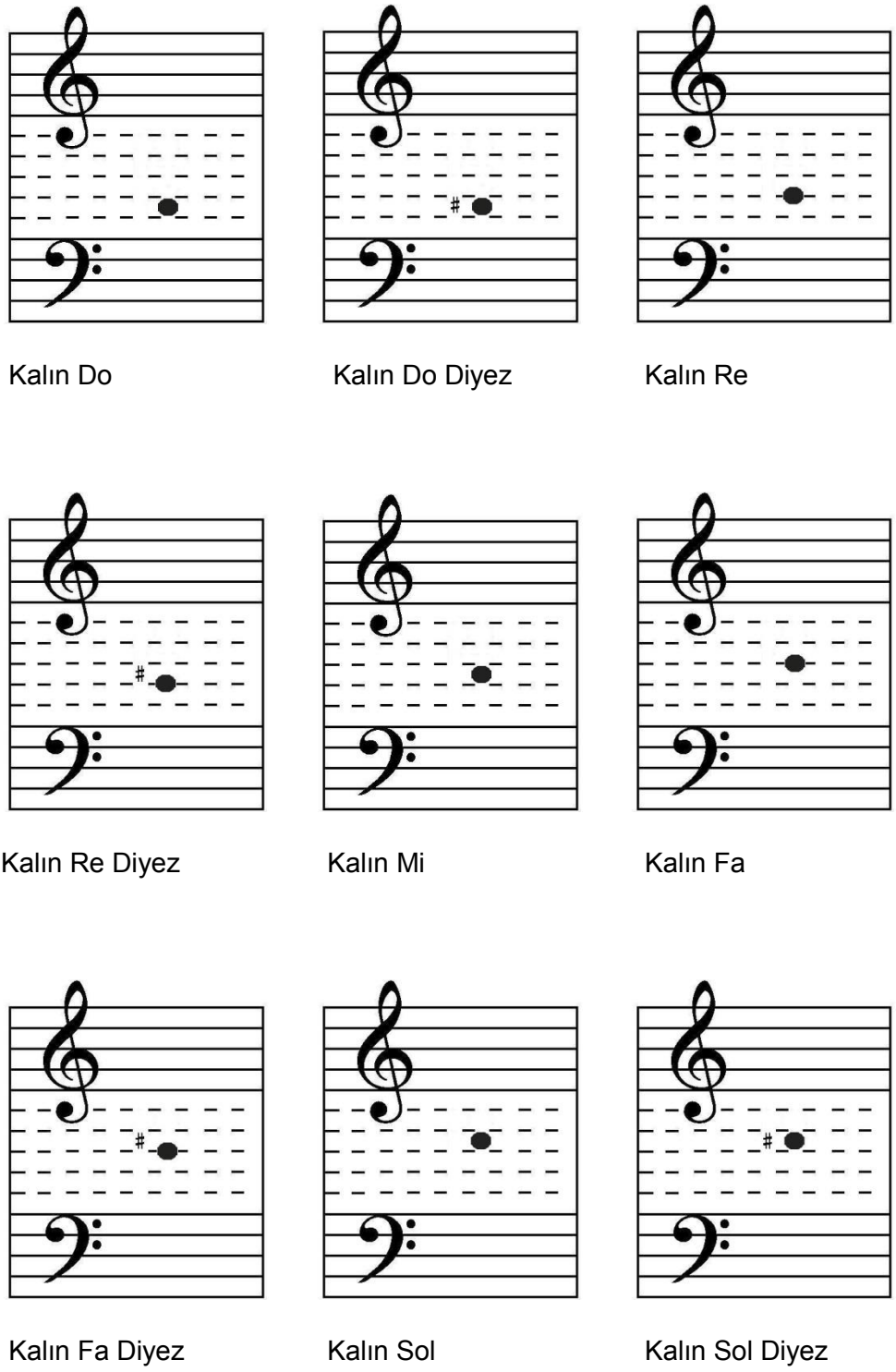


Şekil 3.1. "dna.txt" dosyasının içeriği

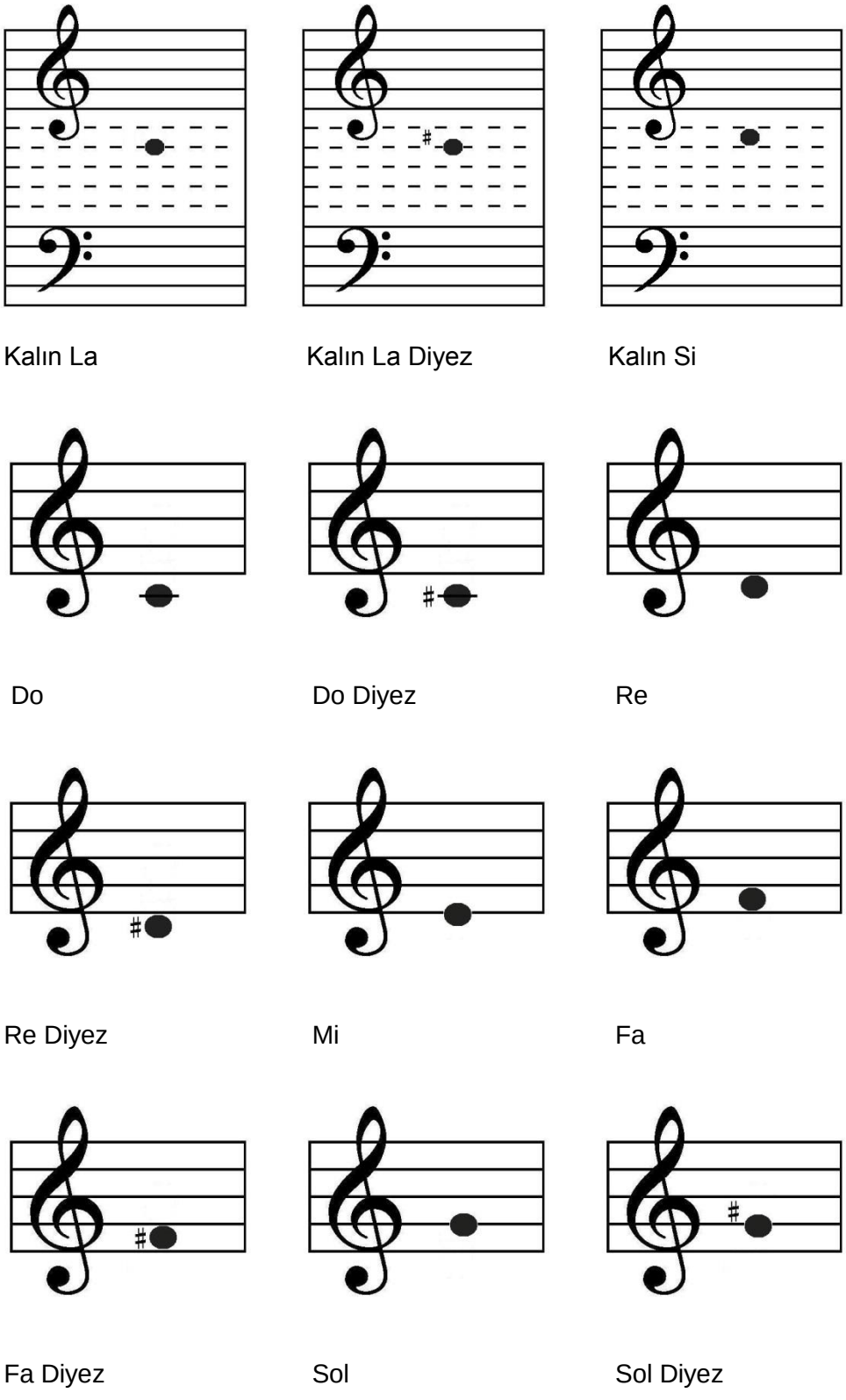
"dna.txt" dosyasında kullanılan formatın karşılığı Şekil 3.3'te belirtilmiştir. Bu format, dünyadaki en yaygın beste gösterim şeklidir.

Şekil 3.2. Yaygın olarak kullanılan müzik notasyonuna göre bir beste örneği

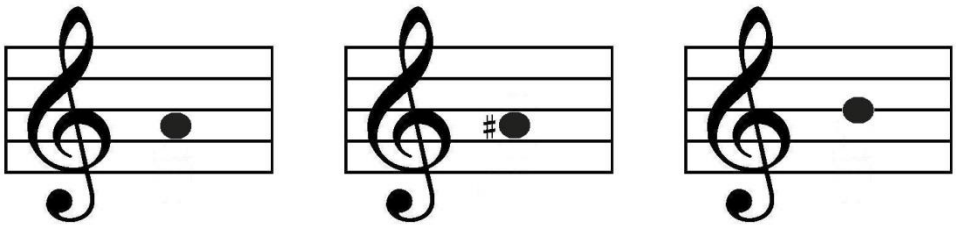
Bu projede kullanılan notasyona çevirmek için öncelikle notaların porte üzerindeki nota isimleri bilinmelidir.

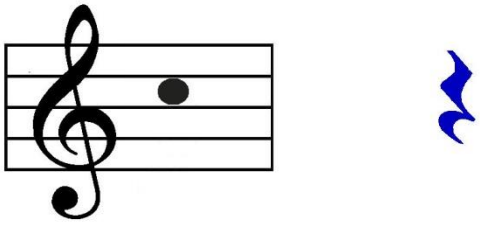


Şekil 3.3. Yaygın olarak kullanılan müzik notasyonuna göre bir beste örneği



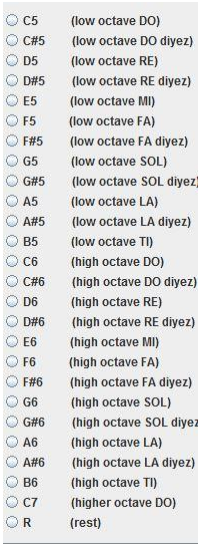
Şekil 3.3. (Devam) Yaygın olarak kullanılan müzik notasyonuna göre bir beste örneği





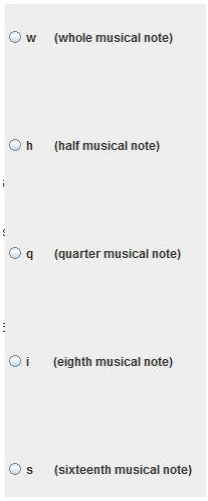
Şekil 3.3. (Devam) Yaygın olarak kullanılan müzik notasyonuna göre bir beste örneği

Şekil 3.3'teki notalar, 1 adet es (es, durak notası demektir; yani kullanıldığı yerde bir süreliğine sessizlik olacağı manasını taşır) ve 25 adet de es olmayan notadan oluşmaktadır ve bu tezdeki programda bunlar kullanılmıştır. Müzisyenler için tasarlanmış 25 tuşluk orglar mevcuttur. Bir orgtaki tuşların her birinin arasında yarım nota değerinde aralık bulunmaktadır. Bu 25 notanın özelliklerini ve hemen hemen tüm müzik bestelerinin (uygulanamayacak olanlar çok sayıda nota iniş çıkışının olduğu müziklerdir ki şarkıların çok çok küçük bir azınlığını oluşturmaktadır) uygulanabileceğini ve bu tezde tercih edilen formatın detaylarını açıklamak gerekirse öncelikle Şekil 3.4'te verilen nota değer kodlamalarını incelemekle başlamak uygun olur.



Şekil 3.4. Programdaki nota değer kodlamaları

Şekil 3.4'te, programda kullanılan nota değer kodlamaları verilmiştir. Programın içindeki kodda sayısal olarak; ilkin "C5" nota değeri "1" sayısal değerine, ikinci olarak "C#5" nota değeri "2" sayısal değerine …(her bir yarım aralık inceldiğinde birer birer artmaktadır)… son olarak "C7" nota değeri "25" sayısal değerine tekabül etmektedir. Böylece bilgisayar programı; geliştirilen Markov zincirinin bu tez için geliştirilmiş algoritmasının matematiksel işlemlerini uygulayabilmektedir. Müzikte duraklamayı belirten es notası olmadan müzik düşünülemez ve bu nota, "R" ile gösterilmiştir.

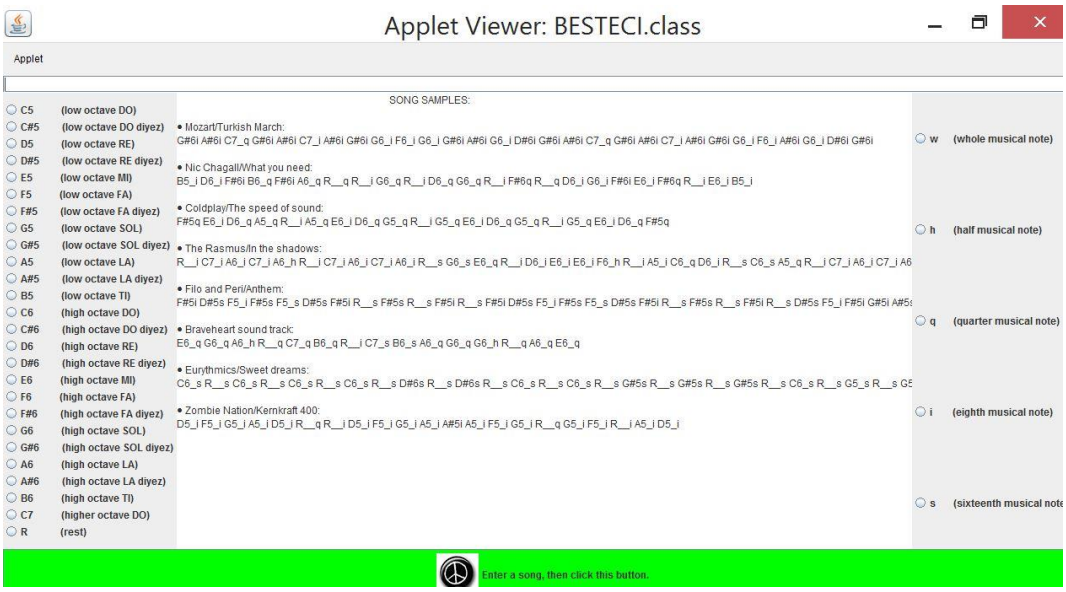


Şekil 3.5. Programdaki nota uzunluk kodlamaları

Şekil 3.6'da, programda kullanılan nota uzunluk kodlamaları gösterilmiştir. "w" tam nota uzunluğunu belirtmektedir ve "h" yarım nota uzunluğunu, "q" çeyrek nota uzunluğunu, "i" 1/8'lik nota uzunluğunu ve "s" 1/16'lık nota uzunluğunu göstermek için kullanılmıştır.

Şekil 3.6'da ise program çalıştırıldığında gelen ilk ekran gösterilmiştir. Buna göre, tez için yazılmış programda ilk önce sol taraftan kullanıcının seçtiği nota değeri ile sağ taraftan seçtiği nota uzunluğu girilmektedir ve üstteki kısma bu otomatik olarak yazılmaktadır. Tüm notalar için bunlar yapılmaktadır. Böylece ilk bestenin notalarının girilmesi tamamlanmakta ve bitince kullanıcı, arayüzün aşağısındaki tuşa basıp ikinci beste için aynı işlemleri yapabilmektedir.

İki besteyi girmek için izlenebilecek bir diğer yöntem de ortada gözüken ve "dna.txt" dosyasından okunan 8 besteden herhangi ikisini seçmektir.



Şekil 3.6. Beste yapan programın arayüzü

3.2. Giriş Verisi Olarak Program Tarafından Önerilen 8 Bestenin Yapısı ve Özellikleri

Giriş verisi olarak 8 beste numune olarak verilmiş ve programın yapısı dolayısıyla istenirse kullanıcı kendisi başka herhangi bir beste girebilmektedir. Şekil 3.6'da bu 8 bestenin bulunduğu ve kodlanan program açıldığında orta kısımda yer alan görüntüsü verilmiştir. Bu bestelerin her biri farklı uzunluklardadır.

Bu bestelere dikkat edildiğinde hepsi de farklı tarzda ve kimisi hafif müzik, kimisi ise tempolu müzik şeklindedir. Bu bestelerin ortak özelliği, çeşitli müzik tarzlarında ama çok sayıda insan tarafından beğenilen besteler olmalarıdır.

Bu besteler aşağıda verilmiştir:

1. Beste:

- Sanatçı: Mozart
- Parça: Turkish March
- Tarz: Klasik
- Tempo: Hızlı
- Uzunluk: 29 nota

2. Beste:

- Sanatçı: Nic Chagall
- Parça: What you need
- Tarz: Trans
- Tempo: Hızlı
- Uzunluk: 23 nota

3. Beste:

- Sanatçı: Coldplay
- Parça: The speed of sound
- Tarz: Brit Pop
- Tempo: Yavaş
- Uzunluk: 19 nota

4. Beste:

- Sanatçı: The Rasmus
- Parça: In the shadows
- Tarz: Alternative rock
- Tempo: Yavaş
- Uzunluk: 50 nota

5. Beste:

- Sanatçı: Filo and Peri
- Parça: Anthem
- Tarz: Trans
- Tempo: Hızlı
- Uzunluk: 40 nota

6. Beste:

- Sanatçı: James Horner
- Parça: Braveheart
- Tarz: Film müziği
- Tempo: Yavaş
- Uzunluk: 15 nota

7. Beste:

- Sanatçı: Eurythmics
- Parça: Sweet dreams
- Tarz: Synth pop
- Tempo: Yavaş
- Uzunluk: 31 nota

8. Beste:

- Sanatçı: Zombie Nation
- Parça: Kernkraft 400
- Tarz: Tekno
- Tempo: Hızlı
- Uzunluk: 21 nota

3.3. Markov Zincirleri Algoritması ile İyi Bestelerin Üretilmesi

Markov Zincirleri Algoritması ile iki giriş bestesinden yeni beste üretim algoritmasının açıklanması, iki aşamada ele alınacaktır. Bunlar ilk notanın üretilmesi ve ilk nota üretimi bittikten sonra diğer notaların üretilmesidir. İlk nota üretimi işlemi için önce kullanıcının girdiği her iki bestenin ayrı ayrı ortalama notaları incelenir. Ortalama notayı bulmak için ilk önce, girilen iki bestenin her birinin notalarının sayısal değerlerinin toplamı, ilgili bestedeki nota sayısına bölünmesiyle elde edilen iki değer elde edilir. İki bestenin her biri için bulunan bu iki değer toplanır ve ikiye bölünür. Buraya kadar yapılan diğer bir deyimle, ilk bestenin ortalama notasıyla ikinci bestenin ortalama notasının ortalamasının bulunmasıdır. Bulunan rasyonel sayı hangi doğal sayıya yakınsa o doğal sayıya karşılık gelen değere sahip nota yeni bestenin ilk notasıdır. "EK-1 İLK NOTA ÜRETİMİ" başlığı altında ilgili kodlar ve açıklamaları verilmiştir (Gül, H. 2010).

Çizelge 3.1. Notaların sayısal değerleri

İSİM	OKTAV	DEĞER
Rest	-	**0**
DO	5	**1**
DO diyez	5	**2**
RE	5	**3**
RE diyez	5	**4**
Mİ	5	**5**
FA	5	**6**
FA diyez	5	**7**
SOL	5	**8**
SOL diyez	5	**9**
LA	5	**10**
LA diyez	5	**11**
Sİ	5	**12**
DO	6	**13**
DO diyez	6	**14**
RE	6	**15**
RE diyez	6	**16**
Mİ	6	**17**
FA	6	**18**
FA diyez	6	**19**
SOL	6	**20**
SOL diyez	6	**21**
LA	6	**22**
LA diyez	6	**23**
Sİ	6	**24**
DO	7	**25**

İlk notadan sonraki notaların üretimi için öncelikle programda, girilen bestelerin her biri için es ve es olmayan nota sayıları hesaplanmaktadır. Bunun yapılma amacı ilk etapta, yeni bestenin üretilecek yeni notasının es olup olmayacağını belirlemektir. Bu belirleme işlemine ve bu başlığın geri kalan açıklamalarında değinilecek çalışmaları içeren kodlara ve onların detaylı açıklamalarına "EK-2 İLK NOTADAN SONRAKİ NOTALARIN ÜRETİMİ" başlığı altında bakılabilir (Gül, H. 2010). Eğer yeni nota es ise es notanın uzunluğunu belirleme işlemine geçilebilir. Eğer yeni nota es ise program es notanın uzunluğunun belirlenmesi işlemini tamamlamıştır. Eğer yeni nota es değilse, bir bestenin yapılmasında genellikle yeterli olan 25 adet notadan biridir demektir. Yeni seçilecek notanın hangi nota olacağı belirlenirken temel görev, notaların sayısal değerleri üzerinde işlem yapmak ve işlem sonucunda bulunan yeni sayısal değere karşı gelen notayı bulup yeni nota olarak belirlemektir. Eğer yeni notanın es olmadığı belirlenmiş ise notanın önceki notaya göre artıyor mu azalıyor mu yoksa değişmiyor mu; buna bakılır. Artıyorsa bu demek olur ki, yeni notayı tanımlayan sayısal değer bir önceki notaya göre artacaktır; eğer azalıyorsa tam tersi; eğer değişmiyorsa nota önceki notayla aynı sayısal değeri alacaktır. Bundan sonra sıra, es olmayan notanın ne kadar değiştiğini tespit etmektedir. Eğer değişmiyorsa önceki notayla aynı sayısal değeri alır. Eğer değişiyorsa ne kadar değiştiğine bakılar: Eğer nota artıyorsa önceki notanın sayısal değerine o değer eklenir; eğer azalıyorsa önceki notanın sayısal değerinden çıkarılır. Sonra sayısal değeri belli olan notanın hangi nota olduğu, bu sayısal değere karşılık gelen notadır. Es olmayan notanın bu bilgilerini temin ettikten sonra sıra, es olmayan notanın süresinin uzunluğunu tespit etmektedir. Bu işlemde de, es notada uzunluk nasıl tespit edilmişse aynı yöntem kullanılır. Yukarıdaki anlatılanların hepsi bir Java'da bir metotta kodlanmıştır ve ilk nota belirleyen metot çağrıldıktan sonra bu metodun tekrar tekrar çağırılmasıyla üretilen notalarla yeni bir beste üretilmiş olur. Yeni beste üretimi istendiği kadar yapılabilir ve bu da çok sayıda farklı beste üretimini karşılar ve kullanıcı açısından istenilen amacı karşılar. Programın kaliteli beste üretimi için, sınıflandırma işlemleri ile kullanıcıya daha büyük avantajlar sağlanacaktır ve bu, ilerleyen kısımlarda anlatılmaktadır.

3.4. Markov Zincirleri Algoritmasının İyileştirilmesi ile Daha İyi Besteler Üretilmesi

"dna.txt" dosyasında, daha önceden bahsedildiği gibi 8 beste vardır. 8 besteden 2'şerli beste kullanarak toplamda $\frac{8!}{2!6!} = 28$ adet farklı beste ikilisi çeşidi ile 8 adet aynı (yani girilecek iki bestenin de aynı olması durumunda) beste ikilisi çeşidi olmak üzere toplamda 36 adet beste ikilisi çeşidi girilebilir. Bunların her birinin girilmesi sonucu oluşan bestelerden oldukça çok sayıda örnek dinlenip Markov zincirleri algoritmasının algoritmik bestelemede uygulanmasında biraz değişiklik yapılmasına karar verildi. Temel sorun, es notaların olması gerekenden oldukça uzun olarak üretildiği fark edildi. Çizelge 3.2'de gösterilen değişiklikler gerçekleştirilerek tekrar bir kodlamaya gidildi. Dikkat edilirse bu değişiklikte, bestede "s" uzunluğunda bir es nota hariç geri kalan tüm es notalar yarıya indiriliyor. Bir tek "s" uzunluğundaki es notalar sabit kalıyor. (Çünkü sisteme "s" uzunluğundan daha kısa bir ses dosyası yüklenmemiştir. Zaten "s" uzunluğu oldukça kısa bir uzunluk olduğu için gerek de duyulmamıştır.) Bu dönüşümden sonra sistem yeni bestelerde daha kısa es notaları üretmekte ve kulağa daha hoş gelen yeni bestelerin oranı artmaktadır.

Çizelge 3.2. Yeni nota uzunluk dönüşümleri

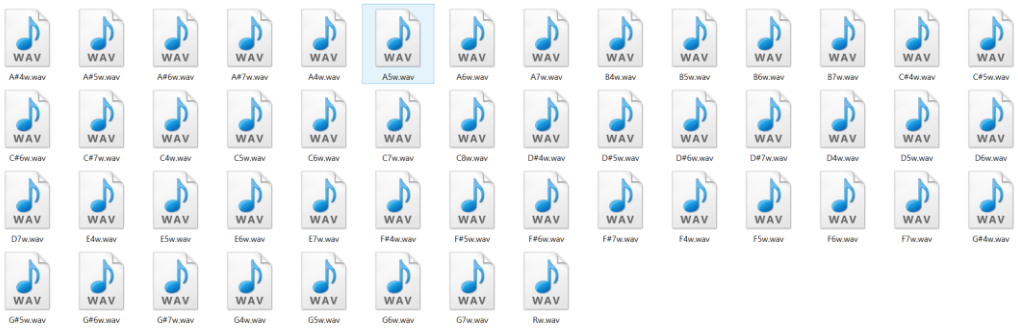
Eski Uzunluk	**Yeni Uzunluk**
w	h
h	q
q	i
i	s
s	s

Müzik salt analitik bir bakış açısıyla çözülemeyecek kadar belirsizdir. Bu nedenle bazı deneyler yapıldı. Belirtilen bu fikrin tasarlanmasından ve uygulanmasından önce; deneme yanılma yöntemiyle;

- Sırada es nota varsa es notayı ihmal etme, es nota varsa hem onu ihmal edip hem de ilgili es nota veya notalardan bir önceki notayı tekrar ettirme,
- Es nota varsa hem onu ihmal edip hem de ilgili es nota veya es notalardan bir önceki notayı tekrar ettirmeksizin notanın uzunluğunu arttırma,
- Çizelge 3.2'deki dönüşümlerden çok daha farklı dönüşümler yapma…

gibi çeşitli yöntemler denenmiş fakat her seferindeki sınamalarda Çizelge 3.2'deki dönüşümün daha uygun olduğu sonucuna varılmıştır. Belki daha iyi sonuçlara farklı bir bakış açısıyla ulaşılabileceği de tahmin dâhilindedir.

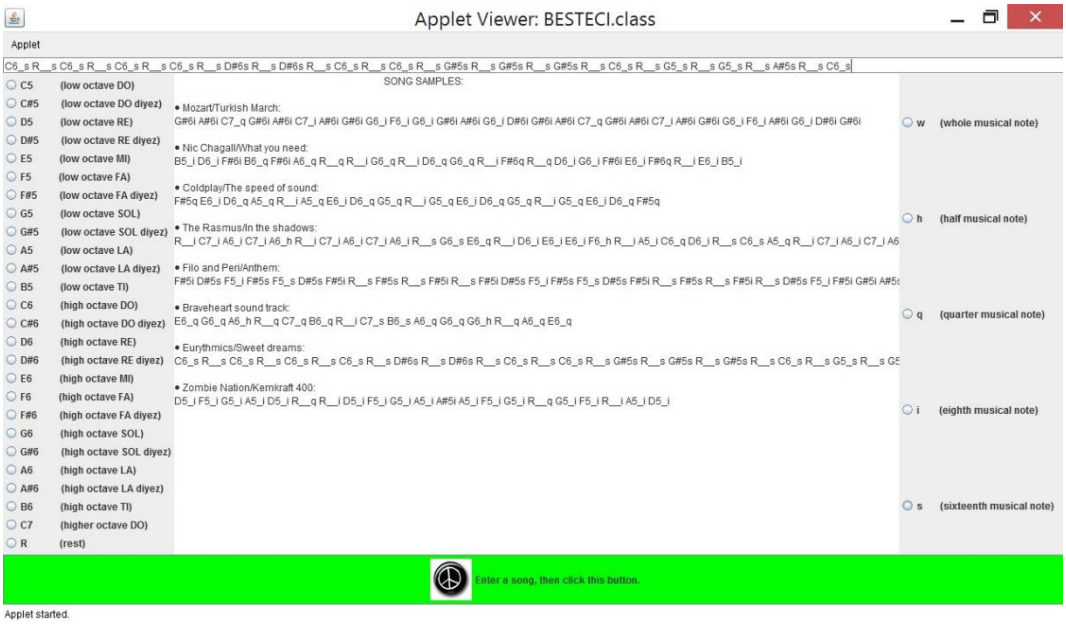
Programda girilen ve üretilen notaların çalınmasında kullanılan dosyada bulunan, "A#4.wav" … "Rw.wav" dosyalarından oluşan 50 adet ".wav" uzantılı ses dosyası Şekil 3.7'de verilmiştir. Sadece tam notalar kullanılmıştır. Eğer programda mesela "A5" notasının çeyrek uzunluğundaki nota çalınacaksa, programda, ilgili notanın tam nota ses dosyası olan "A5.wav" dosyası bulunmakta ve çeyrek nota uzunluğu süresince çalınmakta ve beklemeksizin yeni notaya geçilmektedir.



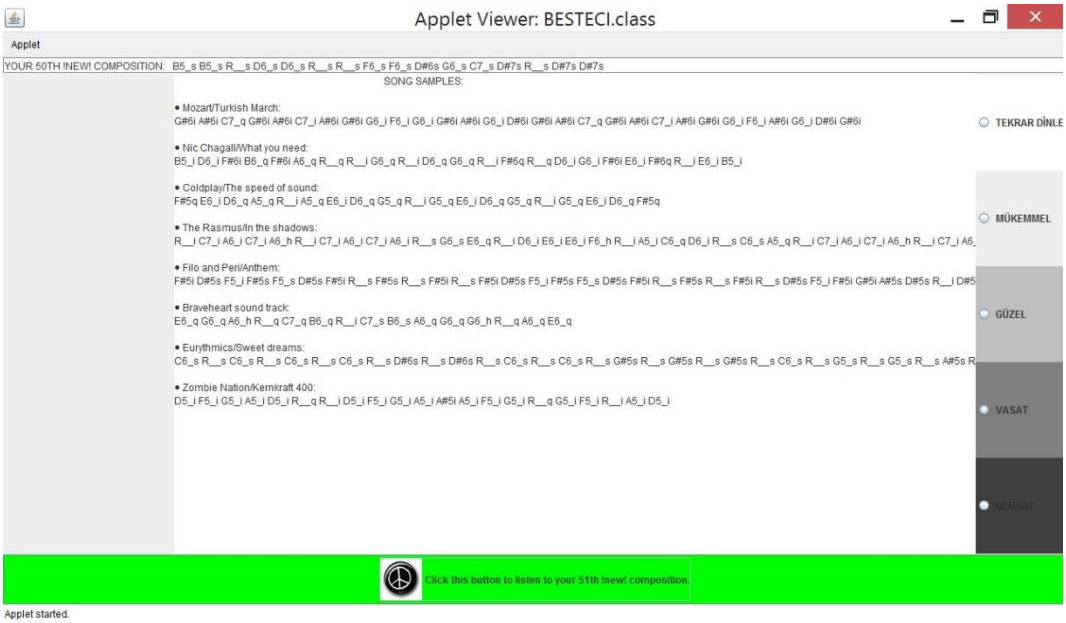
Şekil 3.7. Programda çalınan ".wav" uzantılı ses dosyaları

Birinci aşamada yani beste üretim aşamasında, iki bestenin sırasıyla girilmesi ve yeni besteler elde edilmesiyle ilgili ekran görüntüleri sırasıyla Şekil 3.7'de, 3.8'da, 3.9'da verilmiştir. Dikkatli bakılırsa ilk şekilde girilen beste, arayüz ekranının en yukarısında metin olarak yazılmakta ve dinlenilmektedir (beste girilmeden önce de her nota girişinde notanın doğruluğunu kontrol amaçlı

nota yazılmakta ve dinlenmektedir), sonuç bestesi ise onun bir altındaki (yani alttaki ikinci sıradaki) şekilde, arayüz ekranının yukarısına metin olarak yazılmakta ve dinlenilmektedir. Programda, arayüz ekranının en altındaki tuşa istenildiği kadar basılıp yeni beste üretilebilir.



Şekil 3.8. İki bestenin girilmesi esnasındaki program arayüzü



Şekil 3.9. Yeni bestenin (bu şekilde 50. bestenin) üretilmesi esnasındaki program arayüzü

3.5. Değerlendiricilere Sunulacak Bestelerin Üretilmesi

20 beste Markov zincirleri algoritmasına göre üretilmiş geçmişteki bir çalışmadan, 20 beste de Markov zincirleri algoritmasının geliştirilmesi ile kodlanan bu çalışmadan olmak üzere toplamda 40 adet yeni beste üretildi. Bunlar, 10 değerlendiriciye sunuldu. Besteler dijital müzik formatında kaydedildi ve değerlendiricilerden Onlardan her bir yeni besteyi "iyi" veya "kötü" şeklinde değerlendirmeleri istendi.

4. DENEYSEL ÇALIŞMALAR

4.1. Sınıflandırma İşlemleri

Farklı varyasyonlarla sınıflandırma deneyleri yapıldı. En sonunda her bir beste için oylanan 10 adet oylamadan; en az 8'sinin iyi olarak oylandığı besteler "iyibeste", en fazla 2'sinin iyi olarak oylandığı besteler "kötübeste", bu iki grup dışında kalanlar ise "ortabeste" gruplarında sınıflandırıldı. Böylece hangi bestenin iyi, orta veya kötü olacağını oy çokluğu belirtmiş olacaktır.

Besteler (tek bir kod satırında çok küçük bir kod değişikliğiyle istenilen nota sayısından oluşmaktadır; ama bu tezdeki sınıflandırmalarda kullanılmak üzere standart olması amacıyla) 16'şar notadan oluşmaktadır. Markov zincirlerine girecek olan veriler de bu 16'şar notadan üretilen 30'ar veri ile, (bahsedilen 30 verinin nasıl üretildiği bu konu başlığının devamında açıklanacaktır) sınıflandırıldığı iki gruptan hangisi olduğunu belirten 1'er veriden oluşmaktadır. Bu, her bir bestede toplamda 31'er adet veri eder.

Öncelikle es notasına hangi değer verilmesi gerektiği bir sorundu. Yapılan denemelerde verilen değerin pek de sonucu etkilemedi sonucuna varıldı.

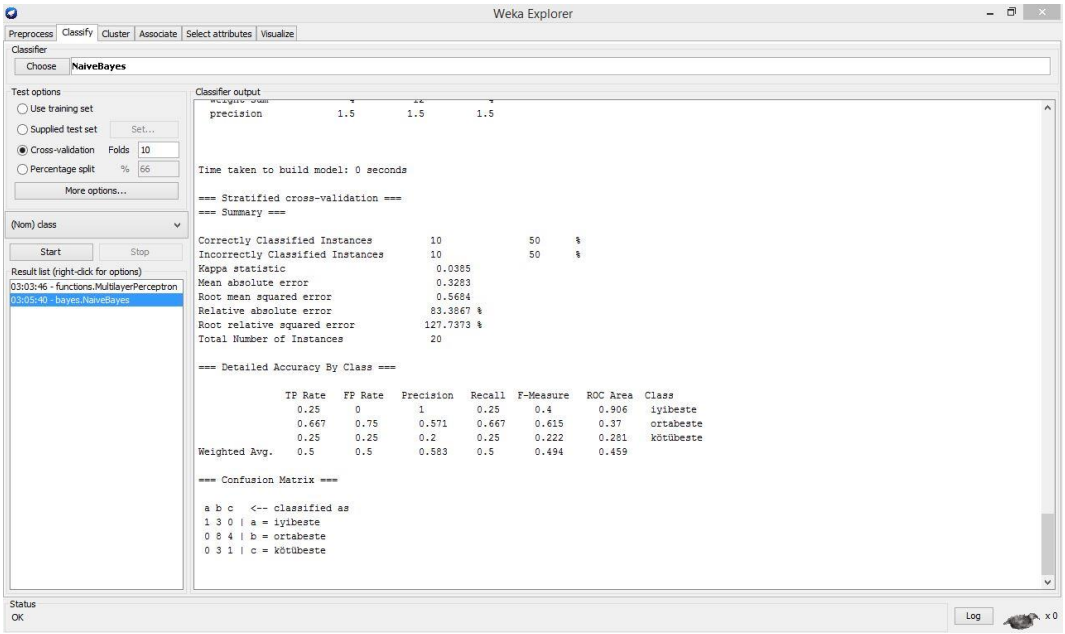
Sınıflandırma işlemi için veri setinin oluşturulması aşamasında; 16 notalık her bir yeni bestenin analizinden oluşacak olan verilerin her biri 30 birimlik oldu. Bunun için gerekli kodlar yazılıp, her bir yeni beste için aşağıdaki kurallara göre veriler elde edilip metin dosyasına kaydedildi.

- Bu 30 birimin 1.'si; ardışık 2 notanın sayısal değerindeki (4 oktav do ile 4 oktav re'nin farkı meselâ) artma veya azalma olacaktır. (Ardışık 2 nota; 2. nota eğer es notaysa, 1. notadan sonraki ilk es nota ile 1. nota arasındaki fark hesaplanacaktır.)
- Bu 30 birimin 2.'si; ardışık 2 notanın uzunluk değerleri (tam nota, yarım nota gibi) toplamı olacaktır. Es notada da aynı yöntem uygulanacaktır.

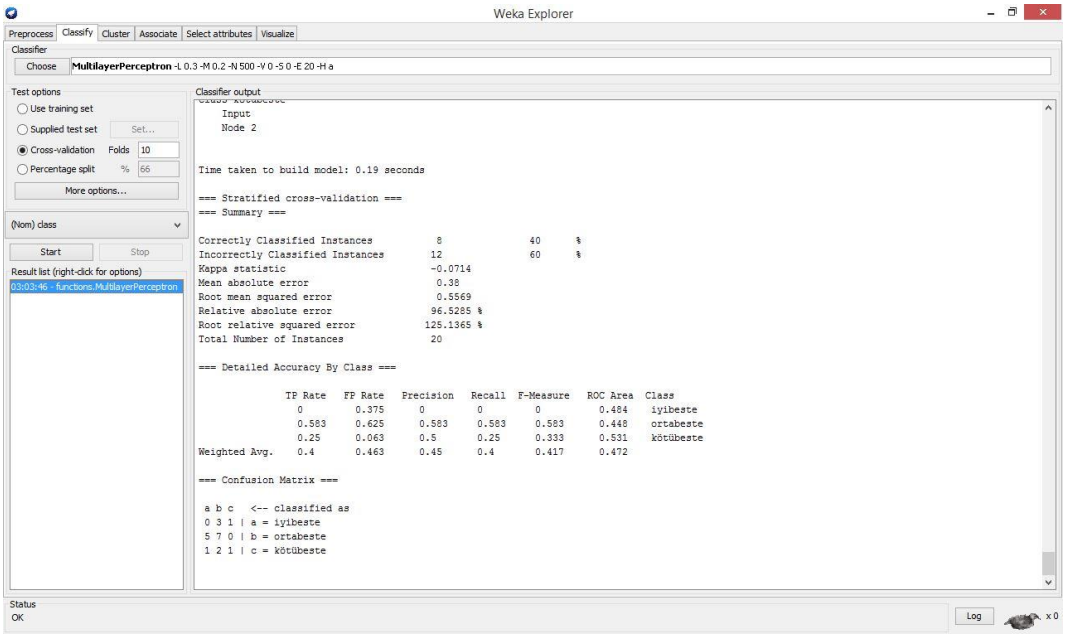
Yukarıdaki 2 kural, 16 notanın oluşturduğu 15'erlik ardışık 2'li nota çiftleri için 15 kez tekrar edecek ve sonuçta 30 birimlik veri oluşacaktır.

Verinin 31. elemanı olarak ise, daha önceden bahsedildiği üzere; 10 değerlendiricinin yaptığı değerlendirmelere göre, iki gruptan hangisine ait olduğunu belirten 1'er veriden oluşmaktadır. Bu, her bir beste için toplamda 31'er adet veri eder.

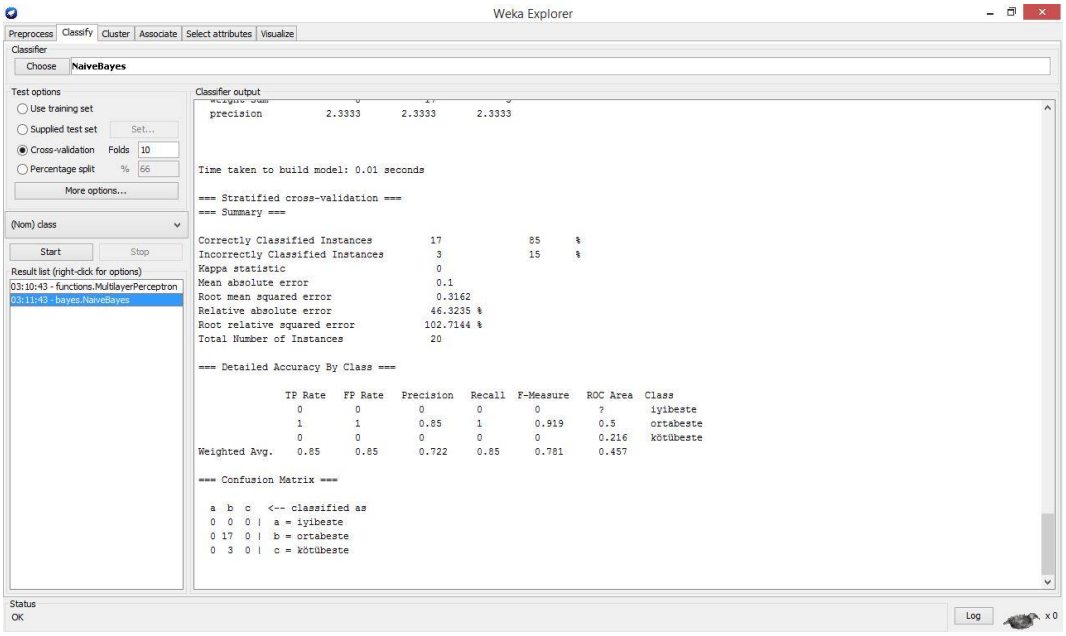
10 değerlendiricinin aynı 40 adet bestede yaptığı değerlendirmelere dayanılarak oluşturulan veriler, Weka programı üzerinde çalışılabilecek şekilde ".arff" uzantılı dosya tipi şeklinde en son hâlini aldı.



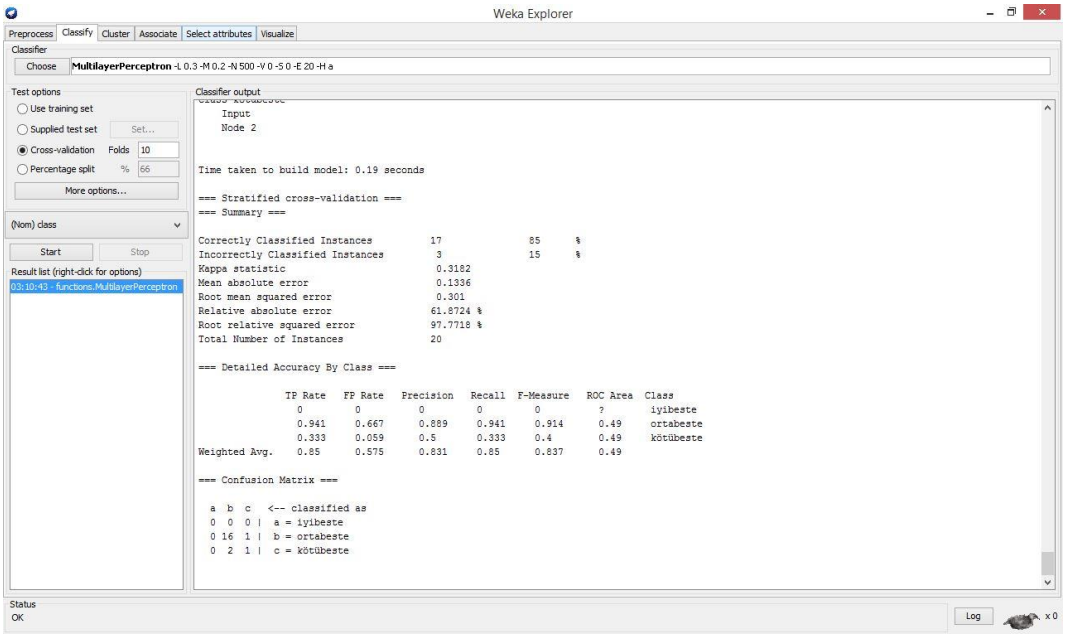
Şekil 4.1. Markov zincirleri algoritmasına göre üretilen 20 beste için yapılan Naive Bayes sınıflandırması



Şekil 4.2. Markov zincirleri algoritmasına göre üretilen 20 beste için yapılan çok katmanlı yapay sinir ağları sınıflandırması



Şekil 4.3. Markov zincirleri algoritmasının geliştirilmiş versiyonuna göre üretilen 20 beste için yapılan Naive Bayes sınıflandırması



Şekil 4.4. Markov zincirleri algoritmasının geliştirilmiş versiyonuna göre üretilen 20 beste için yapılan çok katmanlı yapay sinir ağları sınıflandırması

Bu 4 şekildeki değer ölçütlerinin belli başlılarını kısaca tanımlamak gerekirse:

-Doğru sınıflandırılma oranı: Olması gereken sınıflar ile birebir aynı olarak sınıflandırılan verilerin sayısının toplam veri sayısına bölünmesi ile elde edilir.

-Duyarlılık (Precision): Hangi sınıfla ilgili duyarlılık ele alınıyorsa; ele alınan sınıfta doğru olarak sınıflandırılanların, yine aynı sınıftaki toplam veri sayısına bölünmesiyle elde edilir.

-F ölçüsü (F-measure): Kesinlik değeri; hangi sınıfla ilgili kesinlik ele alınıyorsa; ele alınan sınıfta doğru olarak sınıflandırılanların, yine aynı sınıfta sınıflandırılan toplam veri sayısına bölünmesiyle elde edilir. İşte F ölçüsü de (4.1)'deki formüle göre duyarlılığa ve kesinliğe bulunur (Coşkun, C., Baykal, A., 2011).

$$F\ \text{Ölçüsü} = \frac{2 \times Duyarlılık \times Kesinlik}{Duyarlılık + Kesinlik} \qquad (4.1)$$

Çizelge 4.1. Değerlendirme sonuçları

ALGORİTMA		Doğru Sınıflandırılma	Duyarlılık	F ölçüsü
Markov zincirleri algoritması	Naive Bayes	50	0.571 (ortabeste) 1 (iyibeste)	0.615 (ortabeste) 0.4 (iyibeste)
	Çok katmanlı yapay sinir ağları	40	0.583 (ortabeste) 0 (iyibeste)	0.583 (ortabeste) 0 (iyibeste)
Markov zincirleri algoritmasının geliştirilmiş versiyonu	Naive Bayes	85	0.85 (ortabeste) 0 (iyibeste)	0.919 (ortabeste) 0 (iyibeste)
	Çok katmanlı yapay sinir ağları	85	0.889 (ortabeste) 0 (iyibeste)	0.914 (ortabeste) 0 (iyibeste)

Duyarlılık oranları, bu tezde yapılan çalışma açısından doğru sınıflandırma oranlarından da fazla önemlidir. Sonuçta değerlendiricilerin kötü besteleri elemesi ve geri kalan bestelerinse çoğunun gerçekten de iyi olması; yeni üretilecek bestelerin kötü olarak sınıflandırılmayanlarının beste olarak üretilip kullanıcıya dinletilmesi ile bu dinletilecek bestelerdeki iyi oranının fazla olmasında ve dolayısıyla kullanıcının iyi beste seçiminde daha az zaman ve daha da önemlisi daha az efor harcamasında birebir etkili olacaktır.

Şekil 4.1, Şekil 4.2, Şekil 4.3 ve Şekil 4.4 incelendiğinde; bu tez için tasarlanan ve kodlanan Markov zincirleri algoritmasının geliştirilmiş versiyonunun kullanılması ile ortaya çıkan doğru sınıflandırılma oranları ile bu çalışma için önemli bir kıstas olan "kötübeste" sınıfının dışındaki sınıflandırılma

sonucunda elde edilen duyarlılık oranlarının, Markov zincirleri algoritmasının kullanılması ile ortaya çıkan bestelere nazaran daha iyi çıktığı sonucuna varılmıştır. F ölçüsünde de aynı şekildedir. Tüm bunlar, bu çalışmanın sonucundaki bestelerin sınıflandırılmasının daha sağlıklı yapılabileceği ve bu durum da daha iyi bestelerin üretileceğini ispatlamaktadır.

5. SONUÇ VE DEĞERLENDİRME

Müzik elbette kişiden kişiye değişen bir olgudur. Ancak, popüler şarkıların kitleler üzerindeki tesiri bilinmektedir. Bu popüler şarkıların özellikle melodi yapısı milyonlarca insan üzerinde etki bırakmaktadır. Melodiyi besteci yapar; besteci ya kendi çabasıyla bu işi gerçekleştirir ya da daha pratik bir yol olan; belki aklının ucundan dahi geçmeyecek çok güzel melodileri bilgisayar yardımıyla yani algoritmik bestelemeyle edinir. Algoritmik besteleme bu açıdan önemlidir.

Pek çok bestecinin, muazzam sayıda çok yeni beste üretilebilmesi sebebiyle beste yaparken Markov zincirleriyle üretilmiş besteleri dinlediği ve bunların içinden iyi bulduklarını değiştirmeksizin veya birkaç nota değişikliği ile kendi besteleri olarak tescil ettirdikleri bilinmektedir. Markov zincirlerinin algoritmik bestelemede kullanımı bu açıdan önemlidir.

Bu tezde algoritmik bestelemede kullanılan Markov zincirlerinde, daha iyi beste üretimine yönelik geliştirilme yapılmıştır. Ayrıca çalışma, Markov zincirleriyle beste üretimine, diğer yöntemlerde kullanıldığı tespit edilen ancak Markov zincirlerinde kullanıldığına rastlanmayan değerlendirici mantığı eklenerek kullanıcıya daha az efor ve zaman harcattırmakta ve bunlardan da önemlisi, kötü besteler belli bir oranda elendiği için; kullanıcının daha net bir şekilde iyi beste seçmesine olanak tanımaktadır. Bu tez bu açılardan önemlidir.

Bu tez açısından, doğru sınıflandırılma oranı, duyarlılık, F ölçüsü gibi ölçütler arasında bu tezdeki çalışmanın mantığına göre en önemli değer duyarlılıktır, çünkü "kötübeste" olarak sınıflandırılmayan besteler kullanıcıya dinletilecektir ve esas başarının bunların arasındaki doğru sınıflandırmanın oranının tatmin edici olmasının gerekliliğidir.

Besteci eğer bu yapıyı kullanarak beste elde etmeye kalkarsa, daha önce bahsedildiği gibi daha kısa zamanda daha az efor sarf ederek daha güzel besteler dinleyecek ve esas amaç olan "yeni" ve "mükemmele yakın" besteler elde edecek ve kendi müzik zevkine göre aralarından kendine göre mükemmel bir seçim yapacaktır.

Buna benzer yapılacak çalışmalarda duyarlılık oranını arttırmak ve besteciye daha da az efor, zaman harcatmak ve daha güzel besteler sunmak için

iki fikir üzerinde durulabilir: Birinci fikir, değerlendirici sayısını arttırmaktır. Müziğin, çoğunluğun beğenisine dayalı olması istendiğinde bu yöntem etkilidir. Ne kadar çok insanla bu değerlendirmeler yapılırsa, oluşacak yeni bestelerin o oranda çok sayıda insan tarafından beğenilme yani dünya çapında popüler olma olasılığı artar. İkinci fikir ise değerlendiricileri müzik alanında otorite, mümkünse güzel besteler yapabilen ve/veya çok şarkı dinleyerek müzik kulağını geliştirmiş profesyonellerden seçmektir.

KAYNAKLAR

Akbulut, E. (2006). Günümüz Müzik Eğitimcisi Nasıl Olmalıdır?.*Pamukkale Üniversitesi Eğitim Fakültesi Dergisi*, *20*(20), 23-28.

Ariza, C. (2006). Beyond the transition matrix: A language-independent, string-based input notation for incomplete, multiple-order, static Markov transition values. *Unpublished manuscript, available online at www. flexatone. net/docs/btmimosmtv. pdf*.

Ames, C., & Domino, M. (1992, August). Cybernetic Composer: an overview. In *Understanding music with AI* (pp. 186-205). MIT Press.

Beyls, P. (1989). The musical universe of cellular automata. In *Proceedings of the International Computer Music Conference*, pp. 34–41.

Burman, P. (1989). A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, *76*(3), 503-514.

Coşkun, C., & Baykal, A. (2011). Veri Madenciliğinde Sınıflandırma Algoritmalarının Bir Örnek Üzerinde Karşılaştırılması. *Akademik Bilişim, Malatya*.

Díaz-Jerez, G. (2000). *Algorithmic Music Using Mathematical Models*. Ph.D. thesis, Manhattan School of Music.

Fausett, L. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc..

Fernández, J. D., & Vico, F. (2013). AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 513-582.

Gartland-Jones, A. (2002). Can a genetic algorithm think like a composer?. In *Proceedings of the Generative Art Conference*.

Gül, H., & Doğan, M. (2015), "Markov zincirleri kullanılarak yeni besteler üretilmesi ve üretilen bestelerin sınıflandırılması," In *Proceedings of the International Conference on Research in Education and Science (ICRES).*

Gül, H. 2010., *Besteci*, Lisans Tezi, Pamukkale Üniversitesi, Fen Bilimleri Enstitüsü, Denizli, 2010.

Günindi-Ersöz, A. (2002). Popüler kültür ürünlerinden müzik videolarının gençler üzerindeki olumsuz etkileri. *Yazı İşleri Müdürü*, 4.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18.

Hamanaka, M., Hirata, K., & Tojo, S. (2008). Melody morphing method based on GTTM. In *Proceedings of the International Computer Music Conference*, pp. 155–158.

Hild, H., Feulner, J., & Menzel, D. (1992). HARMONET: a neural net for harmonising chorales in the style of J.S. Bach. In *Proceedings of the Conference on Neural Information Processing Systems*.

Hunt, A., Kirk, R., & Orton, R. (1991). Musical applications of a cellular automata workstation. In *Proceedings of the International Computer Music Conference*.

Leach, J., & Fitch, J. (1995). Nature, music, and algorithmic composition.

Computer Music Journal, *19* (2), 23–33.

Lerdahl, F., & Jackendoff, R. (1985). *A generative theory of tonal music*. MIT press.

Leung, K. M. (2007). Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*.

Marques, V. M., Oliveira, V., Vieira, S., & Rosa, A. C. (2000). Music composition using genetic evolutionary algorithms. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 714–719.

Millen, D. (1990). Cellular automata music. In *Proceedings of the International Computer Music Conference*.

Onder, E. (2011). *Çok Katmanlı Yapay Sinir Ağı Modelleri (slayt no:8)*, http://slideplayer.biz.tr/slide/2298469/

Phon-Amnuaisuk, S. (2010). Investigating music pattern formations from heterogeneous cellular automata. *Journal of New Music Research*, *39* (3), 253–267.

Pope, S. T. (1991). A tool for manipulating expressive and structural hierarchies in music (or: "T-R trees in the MODE: A tree editor based loosely on Fred's theory"). In *Proceedings of the International Computer Music Conference*.

Rothgeb, J. (1968). *Harmonizing the unfigured bass: A computational Study*. Ph.D. thesis, Yale University.

Shibata, N. (1991). A neural network-based method for chord/note scale association with melodies. *NEC Research and Development*, *32* (3), 453–459.

Simoni, M. (2003). Algorithmic Composition: A Gentle Introduction to Music Composition Using Common LISP and Common Music. *SPO Scholarly Monograph Series*.

Thom, B. (2000, June). BoB: an interactive improvisational music companion. In *Proceedings of the fourth international conference on Autonomous agents*(pp. 309-316). ACM.

Thomas, M. T. (1985). Vivace: A rule based AI system for composition. In *Proceedings of the International Computer Music Conference*, pp. 267–274.

Thornton, C. (2009). *Hierarchical markov modeling for generative music*. Ann Arbor, MI: MPublishing, University of Michigan Library.

Todd, P. M. (1989). A connectionist approach to algorithmic composition. *Computer Music Journal, 13* (4), 27–43.

Toiviainen, P. (1995). Modeling the target-note technique of bebop-style jazz improvisation: an artificial neural network approach. *Music Perception: An Interdisciplinary Journal, 12* (4), 399–413.

Toraman, U. (2015). *MESLEKİ TERMİNOLOJİ Artificial Intelligence YAPAY ZEKA (slayt no:41),* http://slideplayer.biz.tr/slide/2397207/

Verbeurgt, K., Fayer, M., & Dinolfo, M. (2004). A hybrid Neural-Markov approach for learning to compose music by example. In *Advances in Artificial Intelligence* (pp. 480-484). Springer Berlin Heidelberg.

Werner, G. M., & Todd, P. M. (1997). Too many love songs: Sexual selection and the evolution of communication. In *Fourth European Conference on Artificial Life* (pp. 434-443). MIT Press.

Walker, W. F. (1994). *A conversation-based framework for musical improvisation*. Ph.D. thesis, University of Illinois.

West, L. (2012). *Neural Networks based on Competition CHAPTER 4 (slayt no:11, slayt no: 16),* http://slideplayer.com/slide/5266950/

Zicarelli, D. (1987). M and jam factory. *Computer Music Journal*, 13-29.Fernández, J. D., & Vico, F. (2013). AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 513-582.

EK-1 İLK NOTA ÜRETİMİ

```
    public static void ilkNota()
    {
        //notaOrtalamasi1 kullanıcının girdiği 1. bestenin,
notaOrtalamasi2 kullanıcının girdiği 2. bestenin nota sayısal
değerlerinin ortalamasıdır. Bu ikisini toplayıp 2'ye böldüğümüzde
iki bestenin ortalama nota sayısal değerini buluruz.
        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=1     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <1.5 )
        tut_yepyenibesteGec_nota=1;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=1.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <2.5 )
        tut_yepyenibesteGec_nota=2;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=2.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <3.5 )
        tut_yepyenibesteGec_nota=3;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=3.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <4.5 )
        tut_yepyenibesteGec_nota=4;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=4.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <5.5 )
        tut_yepyenibesteGec_nota=5;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=5.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <6.5 )
        tut_yepyenibesteGec_nota=6;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=6.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <7.5 )
        tut_yepyenibesteGec_nota=7;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=7.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <8.5 )
        tut_yepyenibesteGec_nota=8;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=8.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <9.5 )
        tut_yepyenibesteGec_nota=9;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=9.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <10.5 )
        tut_yepyenibesteGec_nota=10;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=10.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <11.5 )
        tut_yepyenibesteGec_nota=11;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=11.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <12.5 )
        tut_yepyenibesteGec_nota=12;

        if(     (notaOrtalamasi1+notaOrtalamasi2)/2     >=12.5     &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <13.5 )
        tut_yepyenibesteGec_nota=13;
```

```
        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=13.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <14.5 )
        tut_yepyenibesteGec_nota=14;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=14.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <15.5 )
        tut_yepyenibesteGec_nota=15;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=15.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <16.5 )
        tut_yepyenibesteGec_nota=16;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=16.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <17.5 )
        tut_yepyenibesteGec_nota=17;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=17.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <18.5 )
        tut_yepyenibesteGec_nota=18;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=18.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <19.5 )
        tut_yepyenibesteGec_nota=19;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=19.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <20.5 )
        tut_yepyenibesteGec_nota=20;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=20.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <21.5 )
        tut_yepyenibesteGec_nota=21;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=21.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <22.5 )
        tut_yepyenibesteGec_nota=22;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=22.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <23.5 )
        tut_yepyenibesteGec_nota=23;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=23.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <24.5 )
        tut_yepyenibesteGec_nota=24;

        if(    (notaOrtalamasi1+notaOrtalamasi2)/2    >=24.5    &&
(notaOrtalamasi1+notaOrtalamasi2)/2 <=25 )
        tut_yepyenibesteGec_nota=25;
        //Yeni   bestenin   ilk   notası   ve   uzunluğu   aşağıda
atanmaktadır.
        yepyenibesteGec_nota[tsk]=tut_yepyenibesteGec_nota;
        yepyenibesteUzunluk[tsk]="w";

        yepyenibestenotaCevircumleye();
        tutCevircumleye();

        System.out.println(" Girilen iki bestenin ortalama notası:
"+yepyenibesteNota[tsk]);
    }
```

EK-2 İLK NOTADAN SONRAKİ NOTALARIN ÜRETİMİ

```
//(Önemli not: Her açıklama kendinden önceki notla kendisi
arasındaki kod kısmını açıklamak için kullanılmıştır.)

    public static void uret()
    {

            tut1_1=(int) ( (double)indisBosnota1/indisNota1*100 );
            tut1_2=(int) ( (double)indisBosnota2/indisNota2*100 );
//bu kod kısmındaki mantık şöyledir: İlkin "tut1_1" ve "tut1_2"
değişkenleri sırasıyla 1. ve 2. bestenin boş notalarının
sayısının, toplam nota sayılarına oranının yüzdesini tutmaktadır.

            for(int i=0;i<tut1_1;i++)
            randDizi1[i]=1; //Boş

            for(int i=tut1_1;i<100;i++)
            randDizi1[i]=3; //Dolu
//sonra randDizi1[] dizisine, randDizi1[0]'dan randDizi1[tut1_1]'e
kadar boş nota olduğunu belirtecek sayı olan 1,
randDizi1[tut1_1]'den randDizi1[100]'e kadar dolu nota olduğunu
belirtecek sayı olan 3 atanmaktadır. Burada ve programın buna
benzer ileriki kod kısımlarında amaçlanan 0'dan 100'e kadar bir
dizinin elemanlarına gerekli değerleri atayarak, daha sonra
yapılacak 0'den 200'e kadarki rasgele seçimde, oranı diğerine göre
daha yüksek olan sayıya oranıyla orantılı olarak daha çok şans
vermek; oranı diğerine göre daha düşük olan sayıya da oranıyla
orantılı olarak daha az şans vermektir. (Burada 0'den 100'e kadar
olan seçim 1. beste için yapılmıştır. İki beste eşit hakka sahip
olacağı için bir sonraki açıklamada da belirtileceği üzere 100'den
200'e kadar olan seçim de 2. beste için yapılacaktır.)

            for(int i=100;i<100+tut1_2;i++)
            randDizi1[i]=1; //Boş

            for(int i=100+tut1_2;i<200;i++)
            randDizi1[i]=3; //Dolu
            //Bu kısımda 100'den 200'e kadar olan seçim 2. beste
için yapılmıştır.

            Random sss1=new Random();
            randSayi1=sss1.nextInt(200);
            //Bu kısımda 0'dan 200'e kadar rasgele bir sayı
üretilmiştir.

System.out.print("     Yeni     nota     boş     mu     dolu     mu:
"+randDizi1[randSayi1]+"; rasgele sayı:"+randSayi1);

            if(randDizi1[randSayi1]==1)
            System.out.print(" ***** "+tsk+". yeni nota boş\n");

            if(randDizi1[randSayi1]==3)
            System.out.print(" ***** "+tsk+". yeni nota dolu\n");

            if(randDizi1[randSayi1]==1)
            yepyenibesteNota[tsk]="R";
```

```
//yeni seçimde rasgele sayı boşu işaret ediyorsa, yeni bestenin
tsk'nıncı notasının nota değer stringine (Bir notastringi; nota
değer stringi ve nota uzunluk stringinden oluşmaktadır) R
atanmıştır. Buradan da anlaşılacağı gibi yeni bestenin tsk. notası
boş nota yani resttir.

            if(randDizi1[randSayi1]==1) //boşsa
            {

                                tut2_1_1=(int)
(sayBosuzunluk1[1]/bosOlaninsayisi1*100);
                                for(int j=0; j<tut2_1_1 ;j++)
                                randDizi2[j]=1;


                                tut2_1_2=(int)
(sayBosuzunluk1[2]/bosOlaninsayisi1*100);
                                for(int                j=tut2_1_1;
j<tut2_1_1+tut2_1_2 ;j++)
                                randDizi2[j]=2;


                                tut2_1_3=(int)
(sayBosuzunluk1[4]/bosOlaninsayisi1*100);
                                for(int       j=tut2_1_1+tut2_1_2;
j<tut2_1_1+tut2_1_2+tut2_1_3 ;j++)
                                randDizi2[j]=3;


                                tut2_1_4=(int)
(sayBosuzunluk1[8]/bosOlaninsayisi1*100);
for(int                              j=tut2_1_1+tut2_1_2+tut2_1_3;
j<tut2_1_1+tut2_1_2+tut2_1_3+tut2_1_4 ;j++)
                                randDizi2[j]=4;


                                tut2_1_5=(int)
(sayBosuzunluk1[16]/bosOlaninsayisi1*100);
                                for(int
j=tut2_1_1+tut2_1_2+tut2_1_3+tut2_1_4; j<100 ;j++)
                                randDizi2[j]=5;
//Bu kısımda 0'den 100'e kadar olan seçim 1. beste için
yapılmıştır. (Boş nota gelirse boş notanın uzunluğunun ne olacağı
belirleniyor.)



                                tut2_2_1=(int)
(sayBosuzunluk2[1]/bosOlaninsayisi2*100);
                                for(int   j=100;    j<100+tut2_2_1
;j++)
                                randDizi2[j]=1;


                                tut2_2_2=(int)
(sayBosuzunluk2[2]/bosOlaninsayisi2*100);
                                for(int            j=100+tut2_2_1;
j<100+tut2_2_1+tut2_2_2 ;j++)
                                randDizi2[j]=2;
```

```
                                tut2_2_3=(int)
(sayBosuzunluk2[4]/bosOlaninsayisi2*100);
for(int  j=100+tut2_2_1+tut2_2_2;  j<100+tut2_2_1+tut2_2_2+tut2_2_3
;j++)
                                randDizi2[j]=3;

                                tut2_2_4=(int)
(sayBosuzunluk2[8]/bosOlaninsayisi2*100);
for(int                          j=100+tut2_2_1+tut2_2_2+tut2_2_3;
j<100+tut2_2_1+tut2_2_2+tut2_2_3+tut2_2_4 ;j++)
                                randDizi2[j]=4;


                                tut2_2_5=(int)
(sayBosuzunluk2[16]/bosOlaninsayisi2*100);
                                for(int
j=100+tut2_2_1+tut2_2_2+tut2_2_3+tut2_2_4; j<200 ;j++)
                                randDizi2[j]=5;



                                Random sss11=new Random();
                                randSayi2=sss11.nextInt(200);
System.out.println("      Yeni      boş      notanın      uzunluğu:
"+randDizi2[randSayi2]+"; rasgele sayı:"+randSayi2);
//Bu kısımda 100'den 200'e kadar olan seçim 2. beste için
yapılmıştır. (Boş nota gelirse boş notanın uzunluğunun ne olacağı
belirleniyor.)
                }



                if(yepyenibesteNota[tsk].compareTo("R")==0      &&
randDizi2[randSayi2]==1)
                yepyenibesteUzunluk[tsk]="s";

                if(yepyenibesteNota[tsk].compareTo("R")==0      &&
randDizi2[randSayi2]==2)
                yepyenibesteUzunluk[tsk]="i";

                if(yepyenibesteNota[tsk].compareTo("R")==0      &&
randDizi2[randSayi2]==3)
                yepyenibesteUzunluk[tsk]="q";

                if(yepyenibesteNota[tsk].compareTo("R")==0      &&
randDizi2[randSayi2]==4)
                yepyenibesteUzunluk[tsk]="h";

                if(yepyenibesteNota[tsk].compareTo("R")==0      &&
randDizi2[randSayi2]==5)
                yepyenibesteUzunluk[tsk]="w";
//yeni seçimde, nota rest ise; rasgele sayı hangi uzunluğu işaret
ediyorsa, yeni bestenin tsk'nıncı notasının nota uzunluk stringine
işaret eden string atanmıştır. Buradan da anlaşılacağı gibi yeni
bestenin tsk. notası eğer boşsa uzunluğu da şu ana kadar
belirlenmiştir.
```

```
                if(randDizi1[randSayi1]==3)//doluysa
                {
                    tut3_1_1=(int)
(artanNotasayisi1/(indisBosolmayannota1-1)*100);
                    for( int i=0; i<tut3_1_1; i++)
                    randDizi3[i]=3;

                    tut3_1_2=(int)
(azalanNotasayisi1/(indisBosolmayannota1-1)*100);
                    for( int i=tut3_1_1 ;i<tut3_1_1+tut3_1_2; i++)
                    randDizi3[i]=-3;

                    tut3_1_3=(int)
(degismeyenNotasayisi1/(indisBosolmayannota1-1)*100);
                    for( int i=tut3_1_1+tut3_1_2 ;i<100 ;i++)
                    randDizi3[i]=1;


                    tut3_2_1=(int)
(artanNotasayisi2/(indisBosolmayannota2-1)*100);
                    for( int i=100;i<100+tut3_2_1;i++)
                    randDizi3[i]=3;

                    tut3_2_2=(int)
(azalanNotasayisi2/(indisBosolmayannota2-1)*100);
                    for(            int            i=100+tut3_2_1;
i<100+tut3_2_1+tut3_2_2; i++)
                    randDizi3[i]=-3;

                    tut3_2_3=(int)
(degismeyenNotasayisi2/(indisBosolmayannota2-1)*100);
                    for( int i=100+tut3_2_1+tut3_2_2; i<200; i++)
                    randDizi3[i]=1;



                    Random sss21=new Random();
                    randSayi3=sss21.nextInt(200);
System.out.print(" Yeni dolu nota artıyor mu azalıyor mu değişiyor
mu: "+randDizi3[randSayi3]+"; rasgele sayı:"+randSayi3);

                    if(randDizi3[randSayi3]==3)
                    System.out.print("   *****   yeni   dolu   nota
artıyor\n");

                    if(randDizi3[randSayi3]==-3)
                    System.out.print("   *****   yeni   dolu   nota
azalıyor\n");

                    if(randDizi3[randSayi3]==1)
                    System.out.print("    *****yeni    dolu    nota
değişmiyor\n");

                }


                 rand_Degismiyor=0;
```

```
if(randDizi1[randSayi1]==3  &&  randDizi3[randSayi3]==1)  //doluysa
ve değişmiyorsa
                rand_Degismiyor=1;


if(yepyenibesteNota[tsk].compareTo("R")!=0                          &&
rand_Degismiyor==1)//doluysa ve değişmiyorsa
                {
                    tutCevircumleye();
                    yepyenibesteNota[tsk]=tut_yepyenibesteNota;
                }

if(yepyenibesteNota[tsk].compareTo("R")!=0  &&  rand_Degismiyor==0)
//doluysa ve artıyor ya da azalıyorsa
                {

tut4_1_1=(int)           (notaFarkdeg1[1]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
                                for(int j=0; j<tut4_1_1 ;j++)
                                randDizi4[j]=1;


tut4_1_2=(int)           (notaFarkdeg1[2]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
                                for(int                j=tut4_1_1;
j<tut4_1_1+tut4_1_2 ;j++)
                                randDizi4[j]=2;


tut4_1_3=(int)           (notaFarkdeg1[3]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
                                for(int       j=tut4_1_1+tut4_1_2;
j<tut4_1_1+tut4_1_2+tut4_1_3 ;j++)
                                randDizi4[j]=3;


tut4_1_4=(int)           (notaFarkdeg1[4]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int                             j=tut4_1_1+tut4_1_2+tut4_1_3;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4 ;j++)
                                randDizi4[j]=4;


tut4_1_5=(int)           (notaFarkdeg1[5]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int                      j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5 ;j++)
                                randDizi4[j]=5;


tut4_1_6=(int)           (notaFarkdeg1[6]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int            j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6 ;j++)
                                randDizi4[j]=6;


tut4_1_7=(int)           (notaFarkdeg1[7]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
```

```
for(int   j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7
;j++)
                                randDizi4[j]=7;


tut4_1_8=(int)           (notaFarkdeg1[8]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8 ;j++)
                                randDizi4[j]=8;


tut4_1_9=(int)           (notaFarkdeg1[9]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9 ;j++)
                                randDizi4[j]=9;


tut4_1_10=(int)          (notaFarkdeg1[10]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10 ;j++)
                                randDizi4[j]=10;


tut4_1_11=(int)          (notaFarkdeg1[11]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11 ;j++)
                                randDizi4[j]=11;


tut4_1_12=(int)          (notaFarkdeg1[12]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12 ;j++)
                                randDizi4[j]=12;


tut4_1_13=(int)          (notaFarkdeg1[13]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
```

```
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13 ;j++)
                                randDizi4[j]=13;


tut4_1_14=(int)         (notaFarkdeg1[14]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
;j++)
                                randDizi4[j]=14;


tut4_1_15=(int)         (notaFarkdeg1[15]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15 ;j++)
                                randDizi4[j]=15;


tut4_1_16=(int)         (notaFarkdeg1[16]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16 ;j++)
                                randDizi4[j]=16;


tut4_1_17=(int)         (notaFarkdeg1[17]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17 ;j++)
                                randDizi4[j]=17;


tut4_1_18=(int)         (notaFarkdeg1[18]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17;
```

```
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18 ;j++)
                                randDizi4[j]=18;


tut4_1_19=(int)         (notaFarkdeg1[19]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19 ;j++)
                                randDizi4[j]=19;


tut4_1_20=(int)         (notaFarkdeg1[20]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20 ;j++)
                                randDizi4[j]=20;


tut4_1_21=(int)         (notaFarkdeg1[21]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21 ;j++)
                                randDizi4[j]=21;


tut4_1_22=(int)         (notaFarkdeg1[22]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21+tut4_1_22 ;j++)
                                randDizi4[j]=22;


tut4_1_23=(int)         (notaFarkdeg1[23]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
```

```
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21+tut4_1_22;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21+tut4_1_22+tut4_1_23 ;j++)
                                randDizi4[j]=23;


tut4_1_24=(int)         (notaFarkdeg1[24]/(indisBosolmayannota1-1-
notaFarkdeg1[0])*100);
for(int
j=tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21+tut4_1_22+tut4_1_23;
j<tut4_1_1+tut4_1_2+tut4_1_3+tut4_1_4+tut4_1_5+tut4_1_6+tut4_1_7+t
ut4_1_8+tut4_1_9+tut4_1_10+tut4_1_11+tut4_1_12+tut4_1_13+tut4_1_14
+tut4_1_15+tut4_1_16+tut4_1_17+tut4_1_18+tut4_1_19+tut4_1_20+tut4_
1_21+tut4_1_22+tut4_1_23+tut4_1_24 ;j++)
                                randDizi4[j]=24;



tut4_2_1=(int)          (notaFarkdeg2[1]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
                                for(int   j=100;   j<100+tut4_2_1
;j++)
                                randDizi4[j]=1;


tut4_2_2=(int)          (notaFarkdeg2[2]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
                                for(int           j=100+tut4_2_1;
j<100+tut4_2_1+tut4_2_2 ;j++)
                                randDizi4[j]=2;


tut4_2_3=(int)          (notaFarkdeg2[3]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int  j=100+tut4_2_1+tut4_2_2;  j<100+tut4_2_1+tut4_2_2+tut4_2_3
;j++)
                                randDizi4[j]=3;


tut4_2_4=(int)          (notaFarkdeg2[4]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int                          j=100+tut4_2_1+tut4_2_2+tut4_2_3;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4 ;j++)
                                randDizi4[j]=4;


tut4_2_5=(int)          (notaFarkdeg2[5]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
```

```
for(int                  j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5 ;j++)
                                randDizi4[j]=5;


tut4_2_6=(int)           (notaFarkdeg2[6]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int        j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6 ;j++)
                                randDizi4[j]=6;


tut4_2_7=(int)           (notaFarkdeg2[7]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7 ;j++)
                                randDizi4[j]=7;


tut4_2_8=(int)           (notaFarkdeg2[8]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8 ;j++)
                                randDizi4[j]=8;


tut4_2_9=(int)           (notaFarkdeg2[9]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9 ;j++)
                                randDizi4[j]=9;


tut4_2_10=(int)          (notaFarkdeg2[10]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10 ;j++)
                                randDizi4[j]=10;


tut4_2_11=(int)          (notaFarkdeg2[11]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11 ;j++)
                                randDizi4[j]=11;
```

```
tut4_2_12=(int)          (notaFarkdeg2[12]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12 ;j++)
                                  randDizi4[j]=12;


tut4_2_13=(int)          (notaFarkdeg2[13]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13 ;j++)
                                  randDizi4[j]=13;


tut4_2_14=(int)          (notaFarkdeg2[14]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14 ;j++)
                                  randDizi4[j]=14;


tut4_2_15=(int)          (notaFarkdeg2[15]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15 ;j++)
                                  randDizi4[j]=15;


tut4_2_16=(int)          (notaFarkdeg2[16]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16 ;j++)
                                  randDizi4[j]=16;


tut4_2_17=(int)          (notaFarkdeg2[17]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
```

```
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17 ;j++)
                                randDizi4[j]=17;


tut4_2_18=(int)          (notaFarkdeg2[18]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18 ;j++)
                                randDizi4[j]=18;


tut4_2_19=(int)          (notaFarkdeg2[19]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19 ;j++)
                                randDizi4[j]=19;


tut4_2_20=(int)          (notaFarkdeg2[20]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20
;j++)
                                randDizi4[j]=20;


tut4_2_21=(int)          (notaFarkdeg2[21]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21 ;j++)
                                randDizi4[j]=21;
```

```
tut4_2_22=(int)          (notaFarkdeg2[22]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21+tut4_2_22 ;j++)
                                randDizi4[j]=22;


tut4_2_23=(int)          (notaFarkdeg2[23]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21+tut4_2_22;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21+tut4_2_22+tut4_2_23 ;j++)
                                randDizi4[j]=23;


tut4_2_24=(int)          (notaFarkdeg2[24]/(indisBosolmayannota2-1-
notaFarkdeg2[0])*100);
for(int
j=100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21+tut4_2_22+tut4_2_23;
j<100+tut4_2_1+tut4_2_2+tut4_2_3+tut4_2_4+tut4_2_5+tut4_2_6+tut4_2
_7+tut4_2_8+tut4_2_9+tut4_2_10+tut4_2_11+tut4_2_12+tut4_2_13+tut4_
2_14+tut4_2_15+tut4_2_16+tut4_2_17+tut4_2_18+tut4_2_19+tut4_2_20+t
ut4_2_21+tut4_2_22+tut4_2_23+tut4_2_24 ;j++)
                                randDizi4[j]=24;



                                Random sss31=new Random();
                                randSayi4=sss31.nextInt(200);
System.out.println("   Yeni   dolu   notanın   değişme   miktarı:
"+randDizi4[randSayi4]+"; rasgele sayı:"+randSayi4);
                  }
//Bu kısımda önceki açıklamalar paralelinde yeni nota için 0'den
200'e kadar seçimler yapılmakta ve yeni nota doluysa, yeni notanın
bir önceki notayla değer olarak artan mı azalan mı yoksa aynı mı
olduğu belirlenir. (Bu işlem yapılırken daha önceki açıklamaların
da paralelinde art arda iki notanın değerlerinin artan mı azalan
mı yoksa aynı mı olduğunun yoğunluğuna bakılır.)
```

```
if(yepyenibesteNota[tsk].compareTo("R")!=0                          &&
randDizi3[randSayi3]==3)//artıyorsa
                {

yepyenibesteGec_nota[tsk]=tut_yepyenibesteGec_nota+randDizi4[randS
ayi4];

tut_yepyenibesteGec_nota=yepyenibesteGec_nota[tsk];
                    yepyenibestenotaCevircumleye();
                    tutCevircumleye();
                }

                if(yepyenibesteNota[tsk].compareTo("R")!=0     &&
randDizi3[randSayi3]==       -3)//azalıyorsa
                {
yepyenibesteGec_nota[tsk]=tut_yepyenibesteGec_nota-
randDizi4[randSayi4];

tut_yepyenibesteGec_nota=yepyenibesteGec_nota[tsk];
                    yepyenibestenotaCevircumleye();
                    tutCevircumleye();
                }


                if(randDizi1[randSayi1]==3) //doluysa
                {

tut5_1_1=(int) (sayBosolmayanuzunluk1[1]/bosOlmayaninsayisi1*100);
                                for(int j=0; j<tut5_1_1 ;j++)
                                randDizi5[j]=1;


tut5_1_2=(int)
(sayBosolmayanuzunluk1[2]/bosOlmayaninsayisi1*100);
                                for(int               j=tut5_1_1;
j<tut5_1_1+tut5_1_2 ;j++)
                                randDizi5[j]=2;


tut5_1_3=(int) (sayBosolmayanuzunluk1[4]/bosOlmayaninsayisi1*100);
                                for(int       j=tut5_1_1+tut5_1_2;
j<tut5_1_1+tut5_1_2+tut5_1_3 ;j++)
                                randDizi5[j]=3;


tut5_1_4=(int) (sayBosolmayanuzunluk1[8]/bosOlmayaninsayisi1*100);
for(int                             j=tut5_1_1+tut5_1_2+tut5_1_3;
j<tut5_1_1+tut5_1_2+tut5_1_3+tut5_1_4 ;j++)
                                randDizi5[j]=4;


tut5_1_5=(int)
(sayBosolmayanuzunluk1[16]/bosOlmayaninsayisi1*100);
                                for(int
j=tut5_1_1+tut5_1_2+tut5_1_3+tut5_1_4; j<100 ;j++)
                                randDizi5[j]=5;
```

```
tut5_2_1=(int) (sayBosolmayanuzunluk2[1]/bosOlmayaninsayisi2*100);
                                for(int    j=100;    j<100+tut5_2_1
;j++)
                                randDizi5[j]=1;


tut5_2_2=(int) (sayBosolmayanuzunluk2[2]/bosOlmayaninsayisi2*100);
                                for(int            j=100+tut5_2_1;
j<100+tut5_2_1+tut5_2_2 ;j++)
                                randDizi5[j]=2;


tut5_2_3=(int) (sayBosolmayanuzunluk2[4]/bosOlmayaninsayisi2*100);
for(int  j=100+tut5_2_1+tut5_2_2;  j<100+tut5_2_1+tut5_2_2+tut5_2_3
;j++)
                                randDizi5[j]=3;


tut5_2_4=(int) (sayBosolmayanuzunluk2[8]/bosOlmayaninsayisi2*100);
for(int                          j=100+tut5_2_1+tut5_2_2+tut5_2_3;
j<100+tut5_2_1+tut5_2_2+tut5_2_3+tut5_2_4 ;j++)
                                randDizi5[j]=4;


tut5_2_5=(int)
(sayBosolmayanuzunluk2[16]/bosOlmayaninsayisi2*100);
                                for(int
j=100+tut5_2_1+tut5_2_2+tut5_2_3+tut5_2_4; j<200 ;j++)
                                randDizi5[j]=5;



                                Random sss41=new Random();
                                randSayi5=sss41.nextInt(200);
System.out.println("     Yeni     dolu     notanın     uzunluğu:
"+randDizi5[randSayi5]+"; rasgele sayı:"+randSayi5);

              }



                if(yepyenibesteNota[tsk].compareTo("R")!=0      &&
randDizi5[randSayi5]==1)
                yepyenibesteUzunluk[tsk]="s";

                if(yepyenibesteNota[tsk].compareTo("R")!=0      &&
randDizi5[randSayi5]==2)
                yepyenibesteUzunluk[tsk]="i";

                if(yepyenibesteNota[tsk].compareTo("R")!=0      &&
randDizi5[randSayi5]==3)
                yepyenibesteUzunluk[tsk]="q";

                if(yepyenibesteNota[tsk].compareTo("R")!=0      &&
randDizi5[randSayi5]==4)
                yepyenibesteUzunluk[tsk]="h";
```

```
                if(yepyenibesteNota[tsk].compareTo("R")!=0      &&
randDizi5[randSayi5]==5)
                yepyenibesteUzunluk[tsk]="w";
//Bu kısımda önceki açıklamalar paralelinde yeni nota için 0'den
200'e kadar seçimler yapılmakta ve yeni nota doluysa, yeni notanın
bir önceki notayla değer olarak artan mı azalan mı yoksa aynı mı
olduğu belirlenmişse, eğer değişmemişse önceki notayla aynı değer
verilir, eğer artmış ya da azalmışsa ne kadar değiştiği oransal
olarak rasgele seçim yaptırılır ve yeni bulunan değer eğer
artmışsa yeni bestenin bir önceki notasına ilave edilir, eğer
azalmışsa çıkarılır. Yeni nota değer stringi belirlenmektedir.. Bu
arada yeni bestenin nota uzunluk stringi de oransal olarak aynı
yöntemle belirlenmektedir.

System.out.println("     Yeni     bestenin     "+tsk+".     notası:
"+yepyenibesteNota[tsk]+yepyenibesteUzunluk[tsk]+"\n");

                ++tsk;
//Bu kısımda tsk değeri 1 arttırılmaktadır. (Yani yeni bestenin
bir sonraki notasına geçilir.)
  }
```