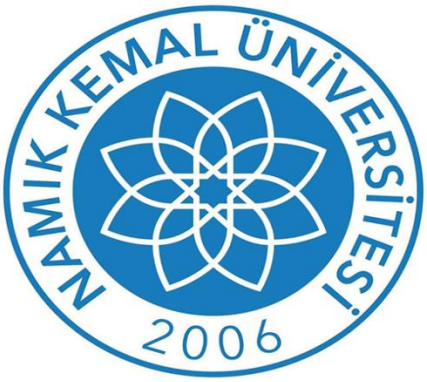
T.C.
NAMIK KEMAL ÜNİVERSİTESİ
ÇORLU MÜHENDİSLİK FAKÜLTESİ



BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİTİRME TEZİ ÇALIŞMASI

Fotoğraftan Duygu Tahmini

(FacEmo)

MEHMET ÇELEN

AYKUT AVCI

AHMET AKYÜZ

DANIŞMAN

Dr. Öğr. Üyesi HEYSEM KAYA

TEKİRDAĞ 2018

İÇİNDEKİLER

FİGÜRLER

TABLOLAR

ÖZET

Bu tez, bir insan yüzünden belirlenmiş duyguların otomatik tespit edilmesini sağlayan akıllı yazılım için gerekli bilimsel inceleme ve teknik geliştirme süreçlerini kapsamaktadır. İmgeden duygu tanıma sürecinde, Histogram of Oriented Gradients ve Local Binary Patterns yöntemlerinin resimlere uygulanarak öznitelik çıkartımını ve daha sonrasında bu özniteliklerle Sequential Minimal Optimization, J48 Ağaç ve Random Forest algoritmaları ile sınıflandırma yaparak sınıflandırma başarısı en yüksek olan yöntemleri seçmeyi amaçlayan deneyleri kapsamaktadır. Materyal olarak bulunan her biri 64x64 piksel boyutlarında, 672 imge eğitim veri seti ve 308 imge test veri seti olarak kullanılmıştır. Bu veri setlerinden öncelikle yukarıda bahsedilen öznitelik çıkarım yöntemleri uygulanarak öznitelikler her iki yöntem için çıkarılmıştır. Çıkarılan LBP ve HOG öznitelikleri hem ayrı ayrı hem de birleşik olarak ele alınmıştır. Sonrasında yukarıda bahsedilen sınıflandırıcı makine öğrenmesi algoritmaları kullanılarak model eğitilmiş ve sınıflandırma başarıları karşılaştırılmıştır. Sonuç olarak Histogram of Oriented Gradients yöntemi ile çıkarılan 1024 öznitelik k katlamalı çapraz geçerleme (k-fold cross-validation) k=8 tutularak yapılan sınıflandırma deneylerinde Sequential Minimal Optimization makine öğrenmesi algoritmasıyla 90.3 % başarıyla (672 resmin 607 tanesini doğru tahmin ederek) sınıflandırılmıştır. Deneylerin sonucunda en iyi yöntemler olarak Histogram of Oriented Gradients öznitelik çıkartıcı ve Sequential Minimal Optimizationmakine öğrenmesi sınıflandırıcı algoritması olarak belirlenmiştir. Yapılan çalışma ile literatüre yeni bir yöntem önerilmemiştir. Gerçekleştirilen çalışma ile bir araştırmanın sonuçları aktarılmıştır. Her ne kadar bu çalışma sonunda yeni bir yöntem önerisi getirilmese de bu çalışmalardan elde edilen veriler ışında üzerinde çalışılan yeni bir makine öğrenmesi yöntemi için temel karşılaştırma parametrelerini vermektedir. Başarılı bulunan yöntemlers kullanılarak bir web servisi programlanmış ve kullanıcıdan alınan resimler eğitilmiş modele verilerek bilgisayarın tahmin yapması sağlanmıştır. Bu tahminler kullanıcıya tekrar döndürülerek bir geri bildirim sağlanmıştır.

Anahtar kelimeler: LBP, SMO, HOG, Duygu Çıkarımı, Makine Öğrenmesi

ABSTRACT

This thesis covers the scientific research and technical development processes required for intelligent software that enables automatic determination of the emotions determined by a human being. In the process of emotional recognition, the histogram of Oriented Gradients and Local Binary Patterns are applied to the pictures to extract the features, and later the classification by these attributes with Sequential Minimal Optimization, J48 Tree and Random Forest algorithms, includes the experiments aiming to choose the methods with the highest success. Each of the materials was used as 672 image training dataset and 308 image test dataset with 64x64 pixel size. Attributes were extracted for both methods by applying the above feature extraction methods from these data sets first. The extracted LBP and HOG attributes are treated separately and in combination. Afterwards, the model was trained using the classifier machine learning algorithms mentioned above and the classification achievements were compared. As a result, 1024 features extracted by Histogram of Oriented Gradients were classified by k-fold k-8 k = 8 classifi- cation experiments using Sequential Minimal Optimization machine learning algorithm with 90.3% success rate (672 images estimated 607 correctly). As a result of the experiments, Histogram of Oriented Gradients feature extractor and Sequential Minimal Optimization Machine learning classifier algorithm are determined as the best methods. A new method has not been proposed in the literature. The results of a research were reported with the study. Although a new method proposal is not introduced at the end of this study, the data obtained from these studies give basic comparison parameters for a new method of learning machine learning. A web service has been programmed using the methods that have been found successful, and the images taken from the user are provided with trained models and the computer is able to make predictions. These estimates are returned to the user and a feedback is provided
Keywords: LBP, SMO, HOG, Emotion Recognition, Machine Learning

1.GİRİŞ

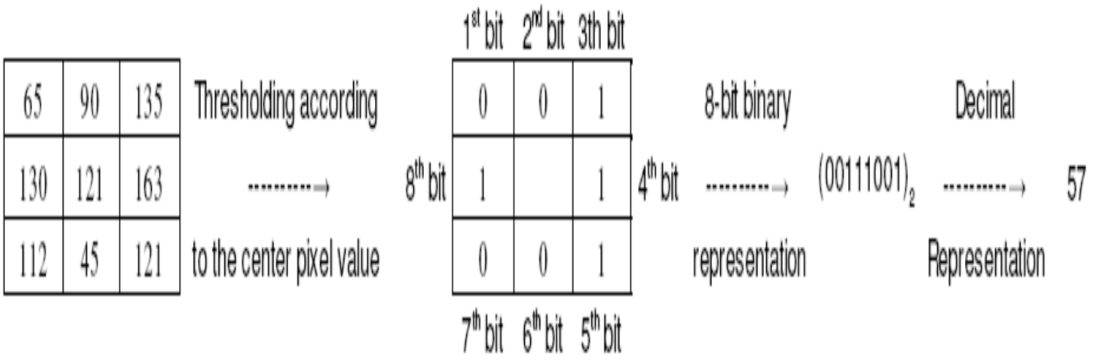
Mevcut teknolojilerle bir insanın yüz ifadesinden hangi duyguyu hissettiği bilgisayar tarafından yüksek başarımla tespit edilebilir. Literatürde projemize örnek olarak Microsoft Emotion API [30] verilebilir. Biz de bu projemizde anlık ya da daha önceden mevcut resimlerden duygu tahmin etmeye çalıstık. Bizi bu projeye iten günümüzde ve gelecekte bu ve bunun gibi işlerin revaşta olacağıdır. Projemiz daha iyi bir geliştirme ile özelikle müşteri memnuniyetini ölçebilir. Örnek olarak bir mağazaya kameradan anlık çekilen fotoğrafların dugu tahmını yapılarak müşterilerin duygu durumları hakkında bilgi toplamak mümkün olabilir. Toplanan bu bilgi çerçevesinde mağaza daha müşteri memnuniyetini analiz ederek çeşitli değerli bilgiler keşfedebilir. Projemiz aynı örnek üzerinden bir anket görevi görebilir.

Projemiz bir insan yüzü resminden matematiksel formülasyonların bilgisayar ortamında işlenecek algoritmalar ve fonksiyonlar yardımıyla çeşitli öznitelikler yani ölçülebilir ve seçilebilir sayısal nitelikler çıkartır. Bu adıma öznitelik çıkarımı denir. Farklı öznitelik çıkarım yöntemleriyle elde edilen öznitelikler bir veri seti oluşturur. Bu veri setini düzenledikten sonra sonraki adım için makine öğrenmesi yöntemleri kullanılır. Bu yöntemlerle oluşturulmuş veri seti kullanılarak bir sınıflandırma modeli oluşturulur. Sınıflandırma modeli oluşturulurken farklı yöntemler ve algoritmalar kullanılabilir. Genel olarak veri seti birbirinden bağımsız olarak eğitim ve test veri seti olmak üzere ikiye ayrılır. Eğitim veri seti kullanılarak model eğitilir ve test veri setiyle de model test edilerek sınıflandırma başarım oranları analiz edilir. LBP, HOG ve bunların kombinasyonları SVM ile nesne tanımada sıkça kullanılmaktadır [1]. Önceki çalışmalarda HOG ve SV kombinasyonu insan profilinin tanınmasında sıkça kullanılmaktadır [2]. Bu araştırmanı amacı da bu yöntemleri kullanarak deneyler yapmak ve sınıflandırma başarımı en yüksek olan kombinasyonları bulmak ve sonrasında bu çalışmayı son kullanıcının oluşturulmuş sisteme fotoğraf yükleyerek bilgisayarın yaptığı tahminleri cevap olarak alacağı bir web uygulamasına dönüştürmektir

2.MATERYAL

2.1 Yerel İkili Örüntü (Local Binary Pattern)

Yerel İkili Örüntü operatörü ilk defa Timo Ojola tarfından tanıtılmıştır [3] LBP güçlü bir tekstür tanımlayıcı olarak bilinmektedir[1]. Tekstür bir görüntünün piksel yoğunluğundaki mekansal varyasyonlara denir. LBP operatörü çok geniş yelpazeli uygulamalarda kullanılmaktadır [4,5]. Bu operatörün arkasındaki fikir, kenarlıklar, çizgiler, nokta gibi ortak özelliklerin belirli bir sayısal ölçekte bir değerle temsil edilmesidir. Bu nedenle, bir nesnenin içindeki nesneleri tanımlamak mümkündür[4]. LBP, tanıma sistemlerinde kullanılması uygundur [6]. LBP operatörü kullanılarak kişi tanıma (face recognition) [6], yüz yakalama (face detection) [7], cinsiyet belirleme (gender estimation) [8] ve yüz ifadelerinin tanınması (facial expression recognition) [9] gibi konularda çalışmalar yapılmıştır. LBP operatörü bir imgedeki her pixeli, kendisini çevreleyen 3x3'lük komşuluk bölgesinde bulunan komşu pixellerini merkez piksel değerine göre eşikleyerek etiketler. Eğer komşu pikselin değeri merkez piksel değerinden büyükse veya eşitse komşu piksel 1, küçükse 0 değerini alır.Bu şekilde bir komşuluk bölgesi için 8-bitlik bir LBP kodu yapılmış olur. Bu kodun onluk sistemdeki değeri ise merkez pikseli çevreleyen 3x3'lük bölgedeki yerel yapıyı gösterir.



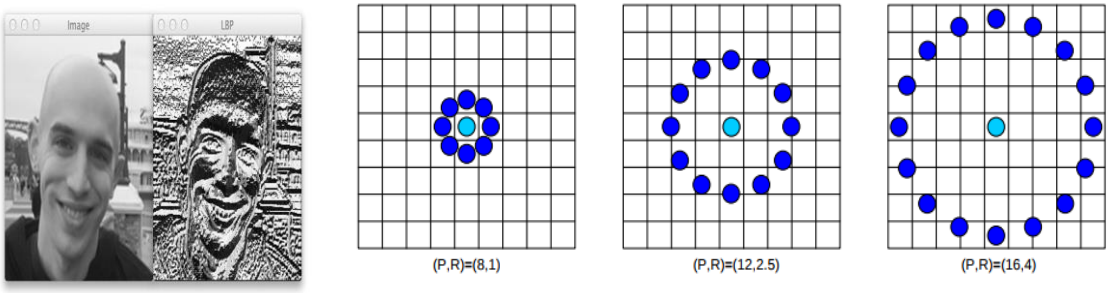
-Figür 1 8-bitlik *LBP* kodu -

Eşikleme işleminden sonra yukarıdaki ifadede (figür 1) sol üst köşedeki ikili sayı ilk basamak kabul edilerek saat yönü sırasıyla 8-bitlik *LBP* kodu oluşturulmaktadır.
Daha sonra bu kodun 10'lu sistemdeki değeri hesaplanarak merkez pikselimizin LBP değeri bulunmaktadır.

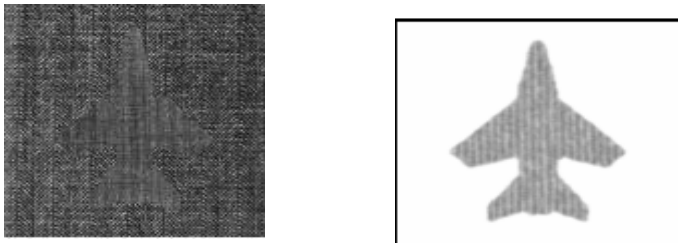
$$LBP_{P,R}(X_c)=\sum_{p=0}^{p} \mu(xp - xc)2p \quad , \quad \mu(y) = \begin{cases} 1, y > 0 \; ise \\ 0, y < 0 \; ise \end{cases}$$

LBP değerleri her pikselin komşuları arasındaki farkın basamak fonksiyonu ile ikilileştirilmesi ile elde edilir. [10,11,12] (Figür 2)

Oluşturulan bütün LBP kodlarının kullanılması mümkün değildir. Bunun yerine 0'dan 1'e değişim sayısı iki veya daha az olanlar tercih edilir. Bu örüntüler uniform olarak isimlendirilir. Örneğin 11110011 veya 000111111 uniform örüntüler iken 01100110 veya 10100110 non-uniform örüntülerdir. 256 (0-255) LBP kodunun 58 tanesi uniform iken geri kalan 198 tanesi non-uniformdur.
Uniform örüntüler görüntü dokusunu tanımlamak için yeterlidir.
Uniform örüntüler kullanıldığında, tek bir hücre için feature vektörünün uzunluğu 256'dan 59'a düşer. (Non-uniform örüntüler için 1 değer tutulur.) LBP notasyonunda, R komşu piksellerin merkez piksele olan uzaklığını belirtirken, P ise işleme dahil edilen komşu piksel sayısını ifade eder.
LBP'nin farklı gösterimleri de vardır. Örneğin;
2 piksel uzaklık komşuluklu 8-bit ikili gösterim,
2 piksel uzaklık komşuluklu 16-bit ikili gösterim ve
3 piksel uzaklık komşuluklu 24-bit ikili gösterimi gibi.
2 piksel uzaklık komşuluklu 8-bit ikili gösteriminde 2 piksel uzaklıktaki gri seviye değerleri tam olarak bir piksel konumunun merkezine düşmezse yoğunluk değeri ara değerleme kullanılarak tahmin edilir.



Figür 2 Farklı P ve R parametrelerine göre LBP operatörleri



Figür 3 İki farklı Temsili LBP Örneği

2.2 Yönlü Gradyanlar Histogramı (Histogram of Oriented Gradients)

Nesne ve örüntü tanıma uygulamalarında, yüksek başarım elde eden Histogram of Oriented Gradient (HOG) algoritması yakın bir zamanda literatürde önerilmiştir. Nesne ve örüntü tanıma için yaygın olarak kullanılmaya başlanan Histogram of Oriented Gradient (HOG), bir çok konuda ve farklı şartlar altında çok yüksek bir başarımla çalışabilmektedir. HOG, ilk olarak Shashua ve diğerleri[13] tarafından yaya tanıma sistemlerinde kullanılabilecek tanımlayıcılar olarak önerilmiştir. Dalal ve Triggs[14] bu yeni tanımlayıcıları başarı ile kompleks ortamlarda insan tanıma problemine uyarladılar. Bu basit fakat etkin tanımlayıcı, başarılı uygulamalarından dolayı, literatürde yoğun bir ilgi kazanmış ve birçok uygulamada kullanılmaya başlanmıştır. HOG yöntemindeki temel amaç imgeyi bir grup lokal histogramlar olarak tanımlamaktır. Bu gruplar, imgenin lokal bir bölgesindeki gradyanların yönelimlerinde, gradyanların büyüklüklerinin toplandığı histogramlardır. Bir imgenin HOG değerlerinin çıkarılması için gerekli olan hesaplamalar şu şekilde gerçeklenmektedir. İlk olarak imgenin, yatay ve dikey Sobel filtreleri uygulanarak, I_X ve I_Y olmak üzere kenarları belirlenir.

$$I_x = I - S_y$$

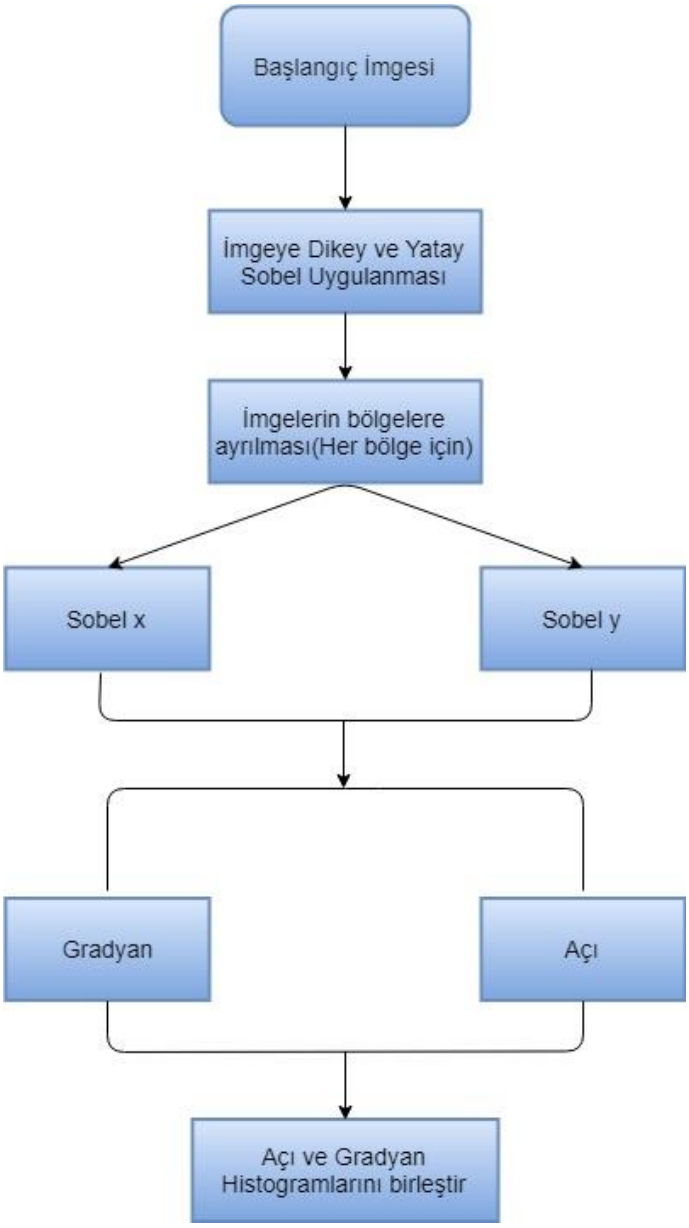
$$I_y = I - S_d$$

Sobel filtresi uygulanmış Ix ve Iy imgeleri kullanılarak, gradyan ve bu gradyanların yönelim açılarının hesaplanması için (G ve θ); formülleri kullanılır.

$$|G| = \sqrt{I^2_x + I^2_y}$$

$$\theta = arctan\frac{Ix}{Iy}$$

HOG algoritması uygulanırken oluşturulan histogramda piksellerin yönelim açılarının gruplanması ile daha anlaşılabilir sonuçlar elde edilebilmektedir. Bu gruplama olayı 0-360 aralığında yer alan açısal değerlerin istenilen bir aralığa çekilmesi ile mümkün olmaktadır.



Figür 4 HOG Algoritmasının Akış Diyagramı

Gradyan temelli bir öznitelik çıkarım metodu olan bu algoritma bir imgeyi yerel histogramların serisi olarak tanımlamaktadır. Yerel histogramların her biri, görüntü üzerinde hücre olarak tanımlanmış belirli bir alandan hesaplanan gradyanların, belirlenen yönlerde oluşma sayısının dağılımıdır. HOG algoritmasından öznitelik çıkarma işlemi sırasıyla üç aşamada gösterilmiştir.

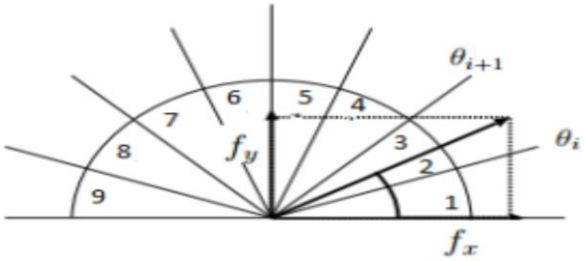
Gradyan Hesaplama

Bu aşamada, ilk olarak imge üzerindeki her bir noktanın yatay ve dikey gradyan değerleri f (x, y) x ve f (x, y) y aşağıdakieşitliklerle hesaplanır. Bu bileşenlerin hesaplanmasında genellikle Sobel filtreleri kullanılır.

$f_x(x, y) = I(x+1, y) - I(x-1, y)$ $\quad m(x, y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2}$

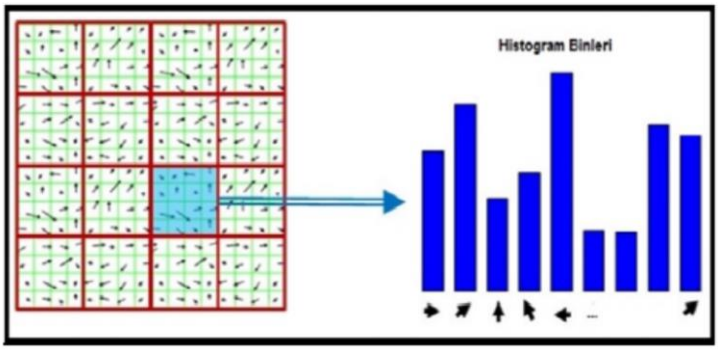
$f_y(x, y) = I(x, y+1) - I(x, y-1)$ $\quad \theta(x, y) = \arctan\frac{f_x(x,y)}{f_y(x,y)}$

Yukarıdaki eşitliklerde kullanılan I(x, y) ifadesi (x, y) noktasındaki parlaklığı ifade eder. Hesaplanan gradyanların büyüklüğü m(x, y) ve gradyan yönü θ (x, y) yine yukarıdaki eşitliklerle ifade edilir. Histogram hesaplanırken, θ (x, y) uygulamaya göre 0–180 ya da 0– 360 derece arasında eşit bölgelere ayrılabilir. Bu çalışmada gradyan yön bölgeleri 0-180 derece arasında 20 derecelik açı farkıyla 9 bölge olarak alınmıştır. Aşağıdaki şekilde (Figür 5) gradyan yön belgeleri gösterilmiştir.



-Figür 5 Gradyan yön belge-

Histogramlar oluşturulurken, hücreiçerisindeki her bir pikselin gradyan büyüklüğü sahip olduğu açıya göre histogram bölgelerine belirli biryönteme göre dağıtılır (Figür-6). Bu dağıtım genellikle lineer ve trilineerinterpolasyon yöntemiyle yapılır.



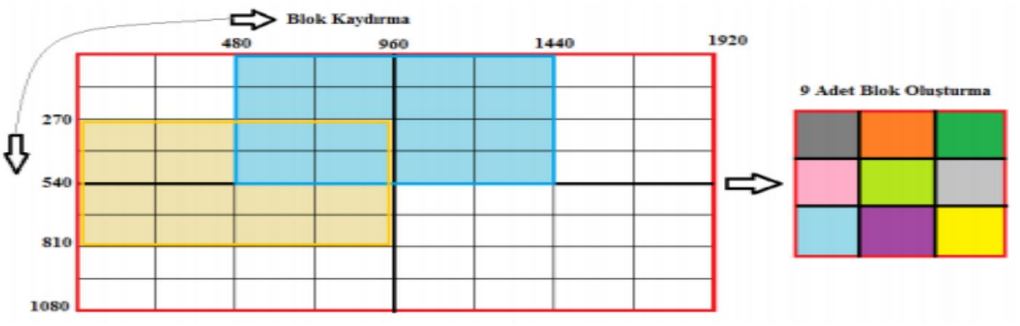
-Figür 6 Pikselin gradyan büyüklüğü sahip olduğu açıya göre histogram bölgeleri-

Sonuç olarak, bir blok içerisinde oluşturulan tüm histogramların birleştirilmesiyle büyük bir histogram elde edilir. Daha sonra bu histogram (Figür 11) kullanılarak normalize edilir.

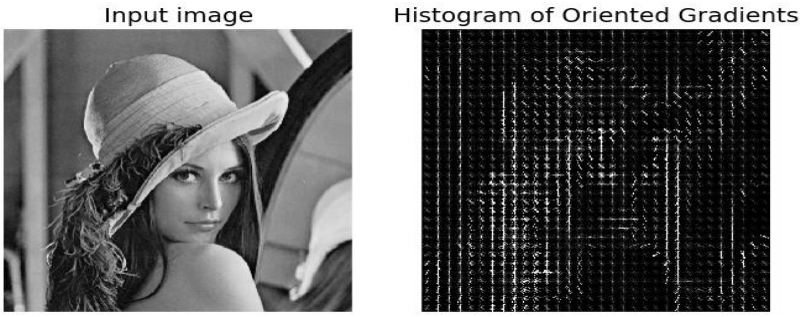
$$V = \frac{Vk}{\sqrt{|Vk|2+1}}$$

Burada *Vk* bir bloktan elde edilen büyük histogram vektörü iken *v* ise normalize edilmiş HOG öznitelik vektörünü gösterir. Örnek bir çalışmada histogram bölmeleri, dikdörtgen hücreler (R-HOG) üzerinden işaretsiz gradyanların hesaplanmasıyla oluşturulmuştur. Normalizasyon için L2-Norm kullanılmıştır. Daha sonra, imgenin tamamı 3x3 diktörgen hücreye bölünerek her bir hücreden (540x960 piksel) bir adet yerel histogram oluşturulmuştur. Sonuç olarak hücre başına 9 histogram bölmesi (*180°/20°=9*) olmak

üzere toplam 9 yerel histogram birleştirilerek *(9x9) = 81* birim uzunluğunda HOG öznitelik vektörü sınıflandırıcıların girişlerine verilmek üzere hesaplanmıştır *(Figür 12)*.



-Figür 7 HOG öznitelik vektörü sınıflandırıcıları-

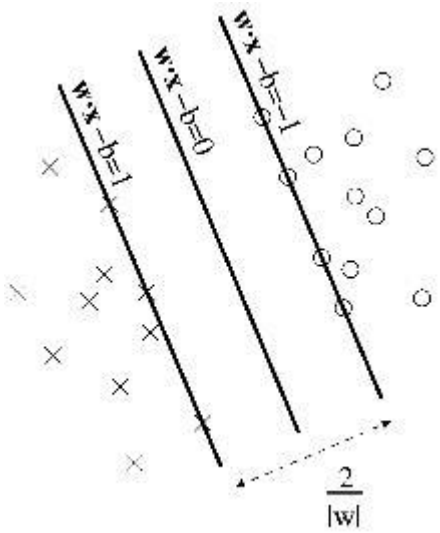


-Figür 8 Temsili HOG Örneği

2.3 Destek Vektör Makineleri(Support Vector Machine)

Sınıflandırma(Classification) konusunda kullanılan oldukça etkili ve basit yöntemlerden birisidir. Sınıflandırma için bir düzlemde bulunan iki grup arasında bir sınır çizilerek iki grubu ayırmak mümkündür. Bu sınırın çizileceği yer ise iki grubun da üyelerine en uzak olan yer olmalıdır. İşte SVM bu sınırın nasıl çizileceğini belirler.

Bu işlemin yapılması için iki gruba da yakın ve birbirine paralel iki sınır çizgisi çizilir ve bu sınır çizgileri birbirine yaklaştırılarak ortak sınır çizgisi üretilir. Örneğin aşağıdaki şekildeki iki grubu ele alalım:



-Figür 9 Girdilerin SVM ile sınıflama örneği-

Bu şekilde iki grup iki boyutlu bir düzlem üzerinde gösterilmiştir. Bu düzlemi ve boyutları birer özellik olarak düşünmek mümkündür. Yani basit anlamda sisteme giren her girdinin (input) bir özellik çıkarımı (feature extraction) yapılmış ve sonuçta bu iki boyutlu düzlemde her girdiyi gösteren farklı bir nokta elde edilmiştir. Bu noktaların sınıflandırılması demek, çıkarılmış olan özelliklere göre girdilerin sınıflanması demektir.

Yukarıda her iki sınıf arasında oluşan aralığa tolerans (offset) demek mümkündür. Bu düzlemdeki her bir noktanın tanımı aşağıdaki gösterim ile yapılabilir:

$$D=\{(x_i,c_i)\,|\,x_i \pounds R^p, c_i \pounds\{-1,1\}\}^{n}{}_{i=1}$$

Yukarıdaki gösterimi şu şekilde okumak mümkündür. Her x,c ikilisi için X vektör uzayımızdaki bir nokta ve c ise bu noktanın -1 veya +1 olduğunu gösteren değeridir. Bu noktalar kümesi i= 1 ‘den n’e kadar gitmektedir. Yani bu gösterim bir önceki şekilde olan noktaları ifade etmektedir.

Bu gösterimin bir aşırı düzlem (hyperplane) üzerinde olduğunu düşünürsek. Bu gösterimdeki her noktanın :

wx – b = 0

denklemi ile ifade edilmesi mümkündür. Buradaki w aşırıdüzleme dik olan normal vektörü ve x noktanın değişen parametresi ve b ise kayma oranıdır. Bu denklemi kalsik ax+b doğru denklemine benzetmek mümkündür.

Yine yukarıdaki denkleme göre b/||w|| değeri bize iki grup arasındaki mesafe farkını verir. Bu mesafe farkına daha önce tolerans (offset) ismini de vermiştik. Bu mesafe farkı denklemine göre mesafeyi en yüksek değere çıkarmak için yukarıdaki ilk şekilde gösterilen 0, -1 ve +1 değerlerine sahip 3 doğruyu veren denklemde 2/||w|| formülü kullanılmıştır. Yani doğrular arası mesafe 2 birim olarak belirlenmiştir.

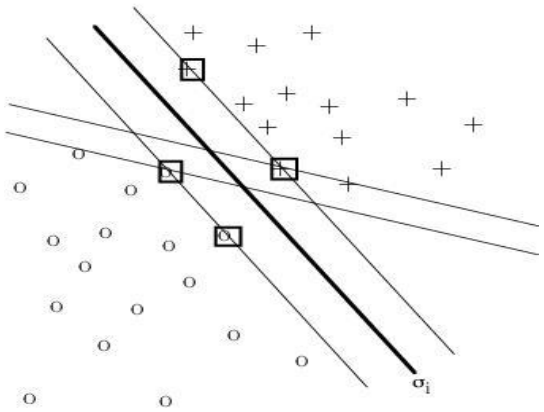
Bu denkleme göre elde edilen iki doğru denklemi:

wx – b = -1

wx + b = 1

olarak bulunmuştur. Aslında bu denklemler doğruların kaydırılması sonucunda elde edilen en yüksek değerlerin bulunması işleminin bir sonucudur. Aynı zamanda bu denklemlerle problemin doğrusal ayrılabilir (linearly seperable) olduğu da kabul edilmiş olur.

Tahmin edileceği üzere iki grup arasındaki aşırıdüzlemin (hyperplane) tek yönlü olması mümkün değildir. Aşağıda bu duruma bir örnek gösterilmiştir:



-Figür 10 İki farklı hiperdüzlem (aşırı düzlem) olasılığı bulunmasına karşılık SVM yöntemi -

Yukarıdaki şekilde iki farklı hiperdüzlem (aşırı düzlem) olasılığı bulunmasına karşılık SVM yönteminde bu olasılıklardan en büyük toleransa (offset) sahip olanı alınır.

2.4 Rassal Ormanlar(Random Forest)

Boosting (Shapire, 1996) ve Bagging (Breiman, 1996) ağaçların sınıflandırılmasında toplu öğrenme için çok iyi iki yöntem olarak bilinir. Bagging'de eğitim verisi kullanılarak her bir ağaç inşa edilir. Ardışık gelen ağaçlar bir öncekinden bağımsızdır ve en büyük oy tahmin için alınır. Boostingde, ardışık gelen ağaçlar bir öncekine bağımlıdır. Bir önceki öncüller tarafından yanlış tahmin edilmiş noktalar için ekstra ağırlık verilir. Sonrasında ağırlıklı oy tahmin için alınır (Liaw ve Wiener, 2002). RO, Bagging yöntemini kullanmaktadır (Breiman, 2001).

1996'da Breiman tarafından ilk amaçlanan Bagging tekniği kullanılarak her bir ağacın birbirinden bağımsız olarak eğitim verileri ile oluşturulmasıdır. Bunun yanında tahminler rastgele seçim metodu (Ho, 1998) ile oluşturulur. Ek olarak yeni verilerin tahmini için toplu öğrenme metodu süresince oluşturulan verilerden her bir değişkenin önem dereceleri ölçülür. Bu model özellikle çok sayıda öngörücü olduğunda indirgeme için yararlı olabilir. RO, çok ağaç üretmek için tekrarlanan bölümleme ve parçalama kullanan bir toplu öğrenme metodudur (ChemModLab, 2008).

Toplu öğrenmede her bir ağacı geliştirmek için rastgele vektörler oluşturulur. Örneğin, 1996'da Breiman'ın geliştirdiği Bagging yöntemi, eğitim verisinden elde edilen örneklerin rastgele seçimiyle ağaç geliştirilmesi ilkesine dayanır. Diğer bir örnek (Dietterich 1998)'in Rastgele Split Seçimi'dir. Her bir düğümdeki splitler, K en iyi splitler arasından rastgele olarak seçilir. Breiman 1999'da, orijinal eğitim verisini rastgele hale getirerek yeni eğitim verisi oluşturmuştur. Bu yaklaşımların hepsinde, k ağaç için rastgele bir θ K vektörü oluşturulur. Oluşturulan θ K 'lar birbirinden bağımsızdır. Aynı dağılıma sahip θ K ve eğitim verisi kullanılarak bir ağaç geliştirilir. Eğitim verisindeki örnek sayısı N'dir. Rastgele vektörler, rastgele olarak N tane kutuya yerleştirilir. Sonrasında rastgele split seçimi yapılır. Rastgele split seçimi θ , 1 ile K arasında yer alan bağımsız, rastgele, integer bir sayıdır. θ 'nın boyutu ve yapısı, oluşturulacak ağacın yapısına ve kullanımına bağlıdır. Geniş sayıda ağaçlar oluşturulduktan sonra en popüler sınıf için oylanır. RO sınıflandırıcısı;

$\{h(x, \theta_K)k=1\}$

şeklindedir. Burada, x, girdi verisini; θ K rastgele vektörü temsil etmektedir. Her bir ağaç en popüler sınıf için bir oya atanır. Bu işlem adımlarına Rastgele Orman denir.

Breiman 2001'in Rastgele Orman yönteminde, bagging rastgele özellik seçimi ile birlikte ele alınır. Orijinal veri setinden yer değiştirmeli olarak yeni bir eğitim veri seti oluşturulur. Sonrasında, rastgele özellik seçimi kullanılarak yeni eğitim setinden bir ağaç geliştirilir. Bu geliştirilen ağaçlar budanmaz. Bagging metodunun tercih edilmesinin iki önemli nedeni vardır; birincisi, bagging işleminde rastgele özellik kullanıldığından doğruluğun artması; ikincisi, genelleştirilmiş hataları hesaplamasıdır. Bu hatalar out-of-bag (OOB) hatalarıdır (Brieman, 2001).

RO, birçok sınıflandırılmış ağaç geliştirir. Girdi verisinden yeni bir objeyi sınıflandırmak için girdi verisini ormandaki her bir ağaca yerleştirir. Her bir ağaç bir sınıflandırma verir. O sınıf için ağaç oyları belirlenir. Orman en yüksek oya sahip olan sınıflandırmayı seçer.

1- N sayıda orijinal veriden yer değiştirmeli olarak N sayıda rastgele eğitim verisi elde edilir.
2- 2- Her bir düğüm için M toplam girdi değişkenlerinden rastgele m ≤ M olacak şekilde değişkenleri seçilir. Bu m değeri orman geliştirme süresince sabittir.
3- 3- Her bir ağaç muhtemel en geniş oranda geliştirilir (Breiman ve Cutler, 2009). Sınıflandırma sırasında, kuralı durdurma ya da budama işlemleri yapılmaz (Archer, 2008). Bu durum, RO'yu diğer karar ağaçları metotlarından ayıran en önemli avantajdır (Pal, 2005).

m azalınca korelasyon ve güç azalır, m artınca korelasyon ve güç artar. Bu m değeri, bulunan OOB hatalarına göre ayarlanabilir (Breiman ve Cutler, 2009). RO sınıflandırıcı ile bir ağaç üretmek için kullanıcı tarafından tanımlanan 2 parametre gereklidir. Bu parametreler, en iyi bölünmeyi belirlemek için her bir düğümde kullanılan değişkenlerin sayısı (m) ve geliştirilecek ağaçların sayısı (N)'dir (Pal, 2005).

Kullanıcı tarafından başlangıç m değeri rastgele seçilir sonraki m'ler OBB hatalara göre artırılır ya da azaltılır. Bu şekilde en uygun m bulunur ve sınıflandırma duyarlığı artar, hata azalır. RO'dan elde edilen 3 parametre vardır. Bu parametreler OOB hatası, değişken önemi (variable importance) ve yakınlık analizi (proximity analysis) (Chen, 2008). T eğitim verisinden Tk yer değiştirmeli yeni eğitim verisi üretilir. Yeni eğitim veri seti kullanılarak ( ) Tk h x, sınıflandırıcısı oluşturulur. Sınıflandırıcı ile çantaya atılmış tahminlerden oylama yapılır. Eğitim verisindeki her x, y için sadece bu sınıflandırıcı ile oylama yapılır. Tk , x ve y' yi içermez. Bu sınıflandırıcı, OOB sınıflandırıcısı olarak da adlandırılır. Bu sınıflandırma doğruluğunun anlaşılmasına yardımcı olur (Beriman, 2001).

RO algoritması, OOB verisindeki verilerin yerleri değiştirildiğinde tahmin hatasının ne kadar olduğunu inceleyerek değişkenlerin önemini, etkilerini (variable importance) hesaplar (Liaw ve Wiener, 2002). Değişken önemi ölçümlerinde, kullanılan değişkenlerin ne kadar önemli olduğu değişkenlerin yerleri değiştirilerek yapılır. Değişimler sonucunda oluşan hatalar o değişkenin işlemdeki önemini ortaya koyar.

RO, budama olmadan maksimum boyutta ağaç geliştirmek için CART (Classification and Regression Tree) algoritmasını kullanmaktadır (Beriman, 2001). CART algoritmasında, bir düğümde belirli bir ölçüt uygulanarak bölünme işlemi gerçekleştirilir. Bunun için önce tüm niteliklerin var olduğu değerler göz önüne alınır ve tüm eşleşmelerden sonra iki bölünme elde edilir. Bu bölünmeler üzerinde seçme işlemi uygulanır (Özkan, 2008).

Bölünme işlemlerinde homojen sınıf dağılımına sahip düğümler tercih edilir. Düğüm homojenliğinin ölçümünde; Gini Index, Entropy, Misclassification Error, Gain Ratio Criteria gibi ölçütler kullanılmaktadır. RO yöntemi, Gini indeksini kullanmaktadır. Verilen bir t düğümü için Gini indeksi;

$$2\ GINI(t)=1-\sum [\ p(j\backslash t)]^2$$

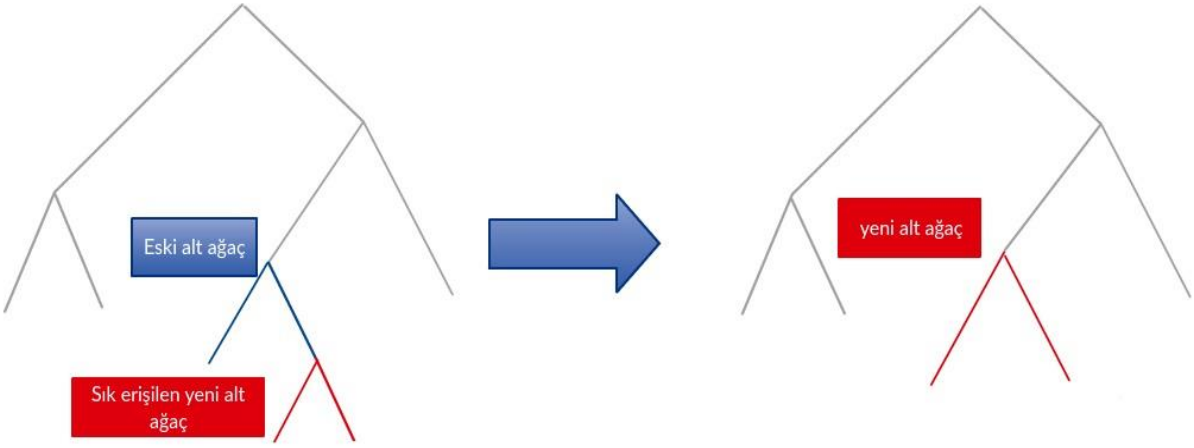
Eşitliğinde, p(|), jt t düğümündeki sınıfına ait bağıl olasılığı göstermektedir. En küçük gini indeksine sahip olan bölünme pozisyonu belirlenir. (Takçı, 2008). Oluşturulan eğitim verileri kullanılarak belirlenen bölünme kriterlerine göre Şekil 1'deki gibi düğümler splitlere ayrılmakta ve ağaç yapıları oluşmaktadır.

Kaç tane ağaç üretmek istenirse her düğüm için en iyi split belirlenerek o kadar ağaç üretilir. Belirlenen ağaçlar arasında oylama yapılır ve en iyi oyu alan ağaç bir sınıfa atanır (Liaw and Wiener, 2002). RO yönteminin sabit bir modeli, kısıtlaması, kalıbı yoktur. Kullanıcı ne kadar ağaç isterse o kadar ağaçla çalışır, hızlıdır.

K En Yakın Komşu, Destek Vektör Makineleri ve Yapay Sinir Ağları gibi çoğu öğrenme tabanlı yöntemler sınıflandırma için kullanışlıdır. Ancak bu yöntemler sınıflandırıcıya göre hangi değişkenlerin en önemli olduğuna yönelik bilgi vermezler. Lineer Diskriminant Analiz gibi diğer yöntemler de sınıflandırıcı üretirken kullanılan öncüllerde, indirgeme yapılmış öncüllere sahip tahmin uzayı gerektirir. RO yönteminde böyle bir indirgeme söz konusu değildir ve her bir adayın tahmini için değişken önem derece ölçümleri yapılır bu da sınıflandırma duyarlığını artırmaktadır (Archer, 2008).

2.5 C4.5 Ağacı (C4.5 Tree)

Bu yazının amacı, karar ağaçlarına (decision tree) bir örnek olaran C4.5 ağacını açıklamaktır. C4.5 ağacı, ID3 ağacının geliştirilmiş bir hali olarak düşünülebilir ve daha önce bu konuda yayınlanan ID3 ağacı başlıklı yazıyı okumanızda yarar vardır. Bu yazıda iki ağacı karşılaştırarak konu anlatılacaktır.
C4.5 ağacının ID3 ağacından en büyük farkı normalleştirme (normalization) kullanıyor olmasıdır. Yani ID3 ağacı üzerinde e ntropi hesabı yapılır (veya bilgi kazanımı (information gain)) ve bu değere göre karar noktaları belirlenir. C4.5 ağacında ise entropi değerleri birer oran olarak tutulur. Ayrıca ağaç üzerinde erişim sıklıklarına göre alt ağaçların (subtree) farklı seviyelere taşınması da mümkündür. C4.5 ağacının diğer bir farkı ise tam bu noktada orataya çıkar ID3 ağacının yaklaşımından farklı olarak C4.5 ağacında budama (prunning) işlemi yapılmaktadır.



Figür 11 C4.5 Ağaç örneği

Bu farklılıklardan bahsettikten sonra nasıl çalıştığını hızlıca açıklamaya çalışalım.

- Her adımda bütün özellikler kontrol edilir
- Her özelliğin normalize edilmiş bilgi kazanımı (information gain) hesaplanır
- En iyi bilgi kazanımını veren özellik karar ağacında karar olarak taşınır.
- Ardından bu yeni karar düğümünün altında bir alt liste oluşturularak alt karar ağacı inşa edilir.

  Konuyu bir örnek üzerinden anlamaya çalışalım. Örnek olarak aşağıdaki veri kümesini (dataset) ele alalım:

Özellik 1	Özellik 2	Özellik 3	Sınıf
Ali	70	Geçti	Sınıf1
Ali	90	Geçti	Sınıf2
Ali	85	Kaldı	Sınıf2
Ali	95	Kaldı	Sınıf2
Ali	70	Kaldı	Sınıf1
Evren	90	Geçti	Sınıf1
Evren	78	Kaldı	Sınıf1
Evren	65	Geçti	Sınıf1
Evren	75	Kaldı	Sınıf1
Şadi	80	Geçti	Sınıf2
Şadi	70	Geçti	Sınıf2
Şadi	80	Kaldı	Sınıf1
Şadi	80	Kaldı	Sınıf1
Şadi	96	Kaldı	Sınıf1

Tablo 1 C4.5 Veri kümesi

Yukarıdaki veri kümesi için bir C4.5 ağacı oluşturmak istiyor olalım. Yapacağımız ilk adım bilgi kazanımını (information gain) hesaplamaktır. Bu adımda bilgi kazanımının formülünü hatırlayalım. Bilgi kazanımı hesaplanırken, o anda veri kümesinde bulunan bütün veriler ve hesaplanması istenen belirli bir verinin üzerinden gidilir. Bu hesaplaması yapılacak olan belirli veriye örnekleme (misal, sampling) ismi verilir ve bütün veri kümesi üzerinden bu örneklemeye ait hesaplama yapılır.

$$\text{Bilgi(M)} = -\sum_{i=1}^{k}((\text{frekans(Sj,M)}/|\text{M}|) \,.\, \log 2\ \text{frekans(S,M)}/|\text{M}|))$$

Bilgi (information) hesaplaması sırasında kullanılacak olan formül yukarıdaki şekildedir. Buna göre herhangi bir misal (M ile gösterilmiştir) için o sınıftaki (S ile gösterilmiştir) değerlere göre frekansına bakılır. Ayrıca yukarıdaki formülde |M| değeri, o sınıftaki misallerin sayısını ifade etmektedir.

Yukarıdaki şekilde her örnek için bilgi değeri hesaplandıktan sonra kazanım (gain) hesaplanması mümkündür.

Genelde tam bu adımda bilgi parçalara bölünür ve bölünen parçalar (partition) üzerinden işlem yapılır. Bu durum için ise hesaplama aşağıdaki şekilde yapılabilir:

$$Bilgi(x) = \sum_{i=1}^{n}\left(\left(\frac{|\text{P}|}{|\text{P}|}\right).\text{Bilgi(Pi)}\right)$$

Yukarıdaki formülde her bir i parçası için yapılan bilgi hesaplaması verilmektedir.

Kazanım ise bu durumda aşağıdaki şekilde hesaplanabilir:

$$\text{Kazanım(Özerlik X)} = \text{Bilgi(P)} - \text{Bilgi}_x\text{(P)}$$

Yani herhangi bir X özelliği için kazanım değeri, o özelliğin bağlı olduğu bütün parça ve sadece o özelliği ilgilendiren parça arasındaki farka eşittir. Bu iki değerin hesabı da yukarıda verilmiştir (yazıdaki ilk ve ikinci formüller).
Şimdi yukarıda anlattığımız bu değerlerin gerçek bir uygulama üzerinden nasıl hesaplandığını görelim.
Örnek veri kümemiz Tablo 1 olsun:

Örneğin sınıf değerinin bilgi kazanımını (information gain) hesaplamak istiyor olalım. Yukarıdaki formüle göre, 14 toplam satırdan 5 tanesi sınıf 2 ve 9 tanesinin sınıf 1 olduğunu dikkate alarak aşağıdaki eşitliği yazıyoruz. Önce bilgi değerlerini hesaplayacak sonra da kazanımı bulacağız:

$$\text{Bilgi(P)} = -9/14 * \log_2(9/14) - 5/14 * \log_2(5/14)$$

$$= 0.940 \text{ bit}$$

İlk bilgi değeri bütün parçanın hesaplandığı yani 14 satırın tamamının dikkate alındığı ve 9/14 ve 5/14 olarak iki ihtimalin hesaba katıldığı durumdur. Bu durum aynı zamanda entropi olarak da düşünülebilir.
İkinci bilgi hesabımızda özellik 1 kullanılacak. Buna göre veri kümemizin ilk 5 satırında Ali, sonraki 4 satırında Evren ve son 5 satırında Şadi özellikleri var. Buna göre tabloyu 3 parçaya bölersek:

Özellik 1	Özellik 2	Özellik 3	Sınıf
Ali	70	Geçti	Sınıf1
Ali	90	Geçti	Sınıf2
Ali	85	Kaldı	Sınıf2
Ali	95	Kaldı	Sınıf2
Ali	70	Kaldı	Sınıf1
Evren	90	Geçti	Sınıf1
Evren	78	Kaldı	Sınıf1
Evren	65	Geçti	Sınıf1
Evren	75	Kaldı	Sınıf1
Şadi	80	Geçti	Sınıf2
Şadi	70	Geçti	Sınıf2
Şadi	80	Kaldı	Sınıf1
Şadi	80	Kaldı	Sınıf1
Şadi	96	Kaldı	Sınıf1

Tablo 2 Veri Kümesinin her özelik ayırdık

Yukarıdaki yeni tabloya göre her özellik parçasının ayrı ayrı hesaplanarak denklemde yerine yazılması gerekir:

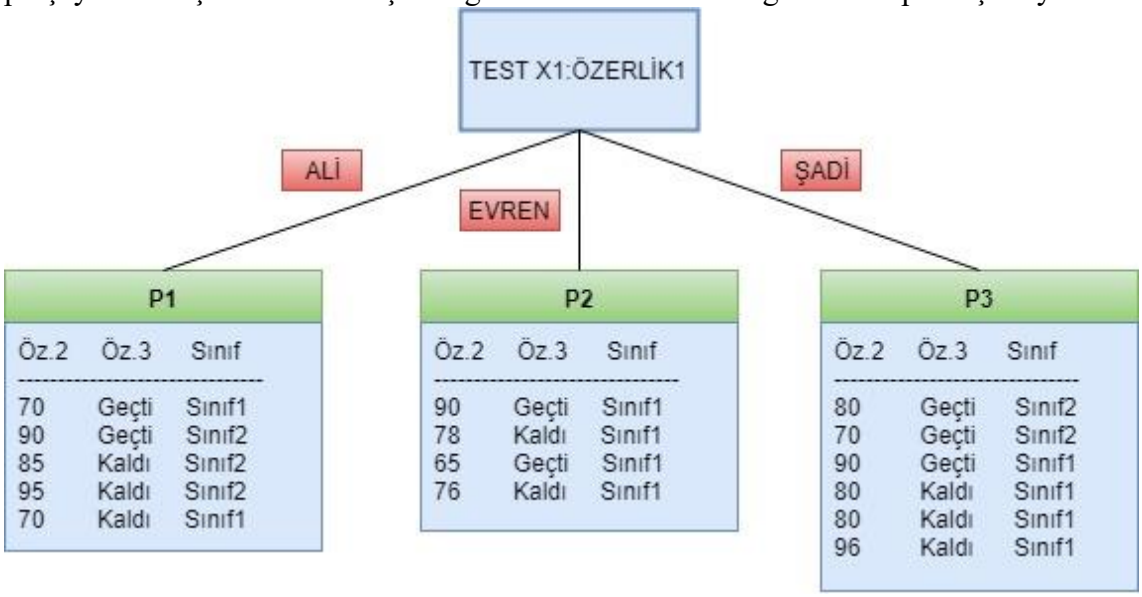
$$Bilgi_{x1}(P)=5/14(-2/5*\log_2(2/5)\ -3/5*\log_2(3/5))$$
$$+4/14(-4/4*\log_2(4/4)\ -0/4*\log_2(0/4))$$
$$+5/14(-3/5*\log_2(3/5)\ -2/5*\log_2(2/5))$$
$$=0.694\ bit$$

Yukarıdaki formülde mavir renkle belirtilen durum 1. özellik için (x1) 5, 4 ve 5 parçadan oluşan ve her parça için ayrı ayrı Sınıf1 ve Sınıf2 değerlerinin sayıldığı durumdur. Yani ilk 5 satırlık parçanın 2 satırı Sınıf1 ve 3 satırı Sınıf2 olduğu için 2/5 ve 3/5 şeklinde iki değer alınmıştır. Diğerleri de benzer şekilde hesaplanmıştır.
Son adımda bu iki değer arasındaki farkı hesaplayabiliriz:

$$Kazanım(x1)=0.940-0.694=0.246\ bit$$

olarak bulunur.

Yukarıda bulunan bilgi kazanımı, bütün veri kümesindeki Özellik1 için bütün sınıflar arasındaki kazanımı göstermektedir. Bu değerleri kullanarak aslında veri kümemizi 3 parçaya bölmüş ve her birisi için bilgi kazanımının olası değerini hesaplamış oluyoruz:



Figür 12 Karar ağacının alabileceği olasılıklardan birisi

Yukarıdaki gösterim karar ağacının alabileceği olasılıklardan birisidir ve ağacın özellik 1'de bulunan isimlere göre karar noktası oluşturulması halinde alacağı vaziyeti gösterir.

$$Bilgi(P)=-9/14*\log2(9/14)-5/14*\log2(5/14)$$

$$=0.940bit$$

$$Bilgi_{Ö3}(P)=6/14(-3/6*\log_2(3/6)-3/6*\log_2(3/6))$$

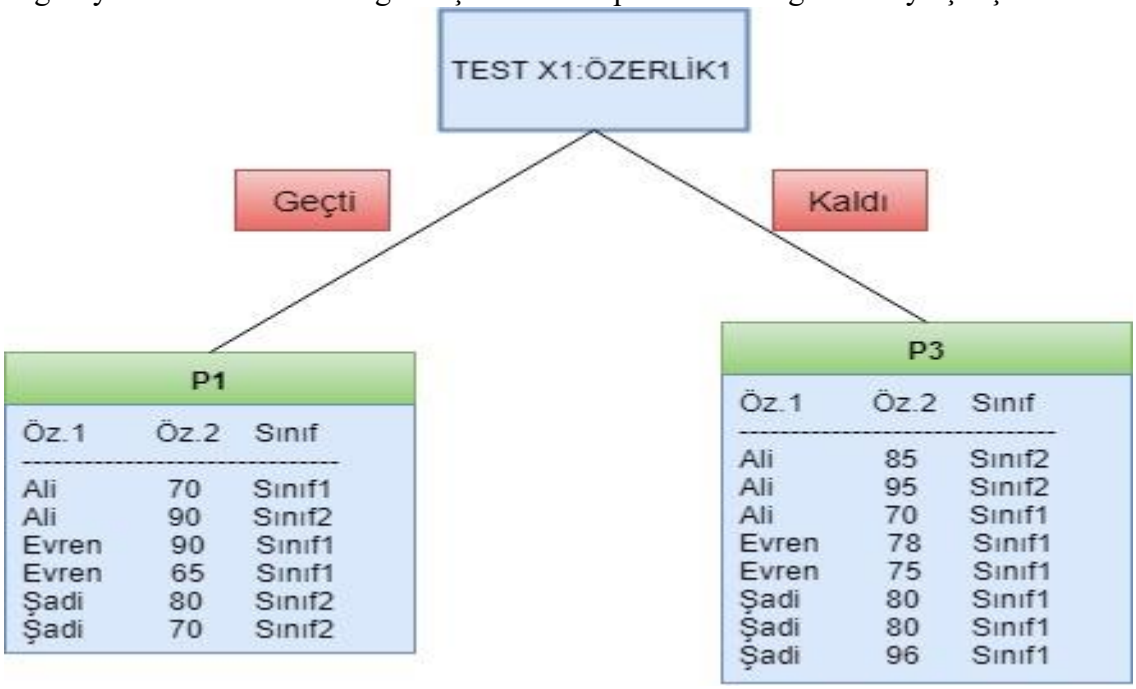
$$+8/14(-6/8*\log_2(6/8)-2/8*\log_2(2/8))$$

$$=0.892\ bit$$

$$Kazanım(Ö_3)=0.940-0.892=0.048\ bit$$

Şimdi aynı hesabı, Özellik 3 için yapalım. Burada geçti / kaldı ihtimalleri bulunuyor dolayısıyla ağaç 2 dala ayrılabilir. Ancak biz önce hesabımızı yapalım.

Özellik 3 için 3/6 ve 6/8 olmak üzere iki parça bulunuyor ve her ikisi için de hesap yapılıp toplam bilgiden çıkarıldığında kazanım değeri olarak 0.048 bit bulunuyor. Bu değeri yorumlamadan önce ağacın şu anda hesaplanan halini göstermeye çalışalım:



Figür 13 ilk karar noktasını Geçti/ Kaldı ihtimalleri üzerine kuracak olursak bilgi kazanımı

Yukarıdaki şekilde ağacın ilk karar noktasını Geçti/ Kaldı ihtimalleri üzerine kuracak olursak bilgi kazanımı olarak 0.048 beklenmektedir.
Bu durumda C4.5 ağacı en yüksek kazanıma sahip olan değeri alacaktır. Bu değer Özellik 1 için isimler olduğundan karar ağacının bu adımda Özellik 1'e göre karar noktası eklemesi yerinde olacaktır.
Arıdından diğer adımlar için benzer şekilde hesaplamalar yapılarak ağacın karar noktaları oluşturulmaya devam edecektir.
C4.5 Ağacının önemli bir diğer özelliği ise budama işlemidir. Esas olarak ağaçlarda iki tip budama yapılabilir. Birisi ön budama (preprunning) diğeri ise son budama (post prunning). C4.5 ağacı son budama (postprunning) yöntemini tercih etmektedir.
Hemen burada karar ağaçlarında (decision trees) ön budama ve son budamanın nasıl yapıldığından bahsedelim. Ön budama, genelde ağaç oluşturulurken bazı dalların oluşturulmaması yönündedir. Örneğin bazı dallar anlamsız olacağından veya hiç eleman içermeyeceğinden oluşturulmaz. Son budama ise ağacın bütün dallarını oluşturur ve sonra

bazı şartlara göre budama yapar. Burada da eleman içermeyen dallar budanabileceği gibi, bazı durumlarda istatistiksel yaklaşımlar da kullanılabilir. Örneğin yukarıdaki veri kümemizi Özellik 2'yi kullanarak 10luk dilimlere bölmek isteyelim. 100-90 arasında 90-80 ve 80-70 arasında verilerimiz olacak ancak 70'in altında verilerimiz olmayacak. Bunu verilere bakmadan anlayamayız. Şayet illaki ağaç oluşturulacaksa ve kural 0 ile 100 arasındaki notların 10'arlık dilimler halinde bölünmesi ise bu dalların da ağaçta yer alması gerekir ancak hiç veri içermeyeceği için bu dallanmalar anlamsız olacak ve budanacaktır.

C4.5 ağacı için son olarak WEKA programında J48 olarak açık kaynak kodlu bir versiyonunun yazılmış olduğunu söylemekte yarar vardır.

2.6 WEKA

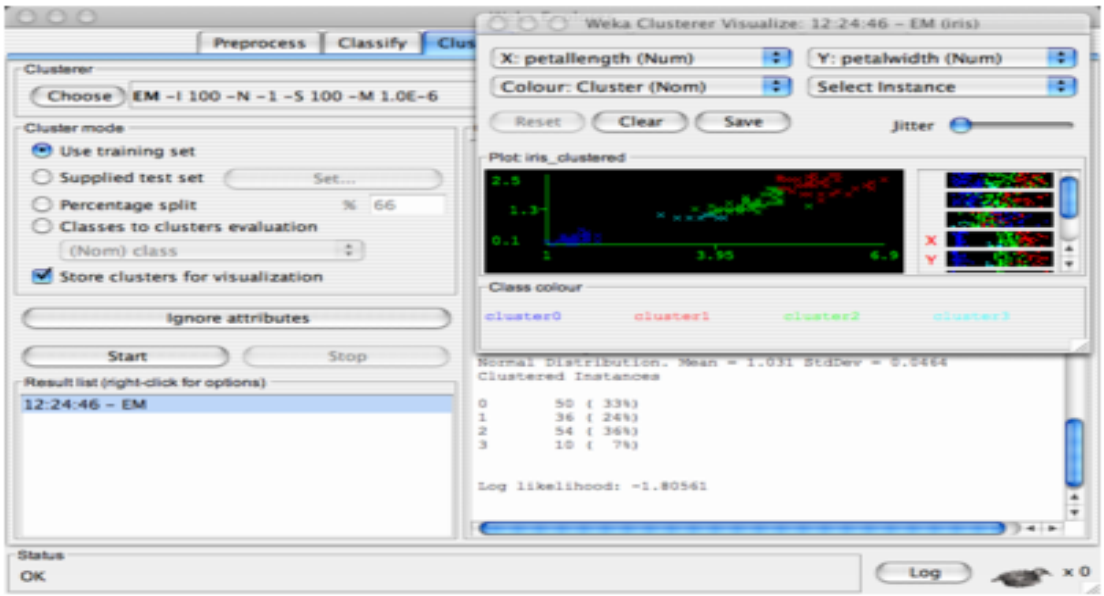
WEKA, bilgisayar bilimlerinin önemli konularından birisi olan makine öğrenmesi (machine language) konusunda kullanılan paketlerden birisinin ismidir. Waikato üniversitesinde açık kaynak kodlu olarak JAVA dili üzerinde geliştirilmiştir ve GPL lisansı ile dağıtılmaktadır. İsmi de buradan gelir ve Waikato Environment for Knowledge Analysis kelimelerinin baş harflerinden oluşur.

WEKA verileri basit bir dosyadan okur ve veriler üzerindeki stokastik değişkenlerin sayısal veya nominal değerler olduğunu kabul eder. Aynı zamanda veritabanı (database) üzerinden de veri çekebilir ancak bu durumda verilerin bir dosya verisi şeklinde olması beklenir.

WEKA üzerinde makine öğrenmesi ve istatistik ile ilgili pekçok kütüphane hazır olarak gelmektedir. Örneğin veri ön işlemesi (data preprocessing), ilkelleme (regression), sınıflandırma (classification), gruplandırma (clustering), özellik seçimi veya özellik çıkarımı (feature extraction) bunlardan bazılarıdır. Ayrıca bu işlemler sonucunda çıkan neticelerinde görsel olarak gösterilmesini sağlayan görüntüleme (visualization) araçları bulunmaktadır. Aşağıda bu amaçla yazılan ekranlardan bazıları gösterilecektir

2.6.1 Ön işleme paneli (Preprocess Panel)

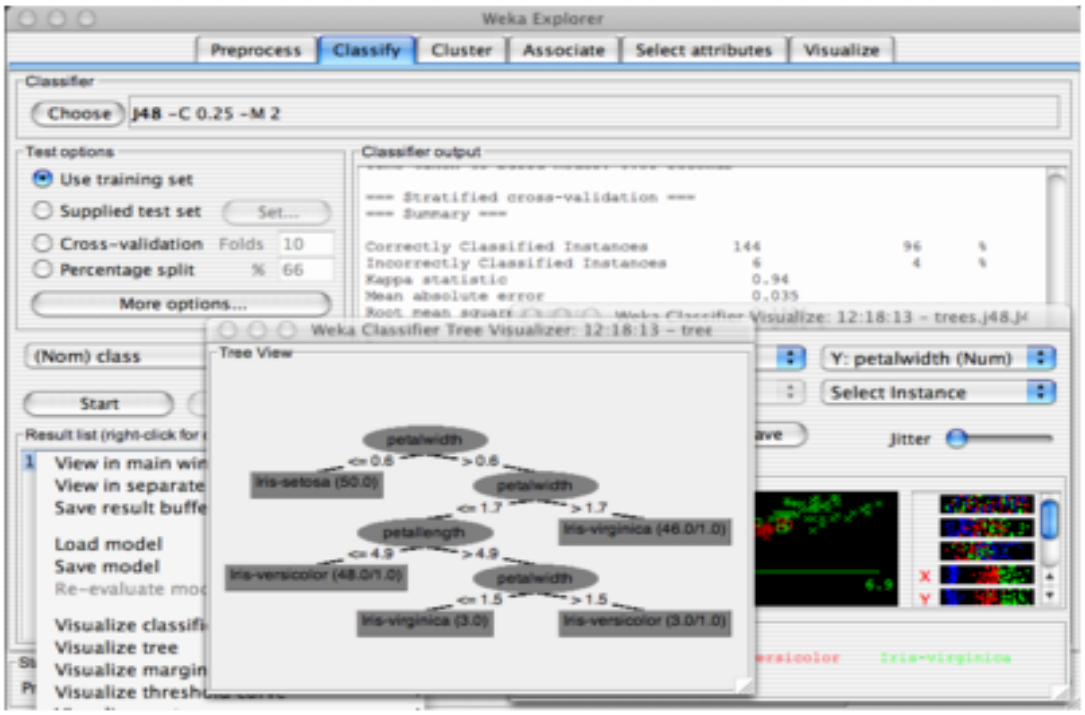
Bilgi dolaşıcısının (knowledge explorer) başlangıç noktası önişleme panelidir. Bu panelde veri kümleri (dataset) yüklenebilir veya WEKA içerisinde bulunan filitreler ile veriler üzerinde işlemler yapılarak veri işlenebilir.



Figür 14 Weka Preprocess Panel

2.6.2 Sınıflandırıcı Paneli (Classifier Panel)

WEKA içerisinde yüklü olan sınıflandırma algoritmalarından herhangi birisini kullanarak mevcut veri kümesi üzerinde bu ekran marifetiyle sınıflandırma yapılabilir. Ayrıca bu ekranda test ve sağlama (validation) için ayrı kümeler kullanmak da mümkündür. Sınıflandırma hataları ayrı bir ekranda açılır ve şayet sınıflandırma algoritması bir karar ağacı (decision tree) oluşturursa bu da ayrıca bir ekranda görüntülenir.



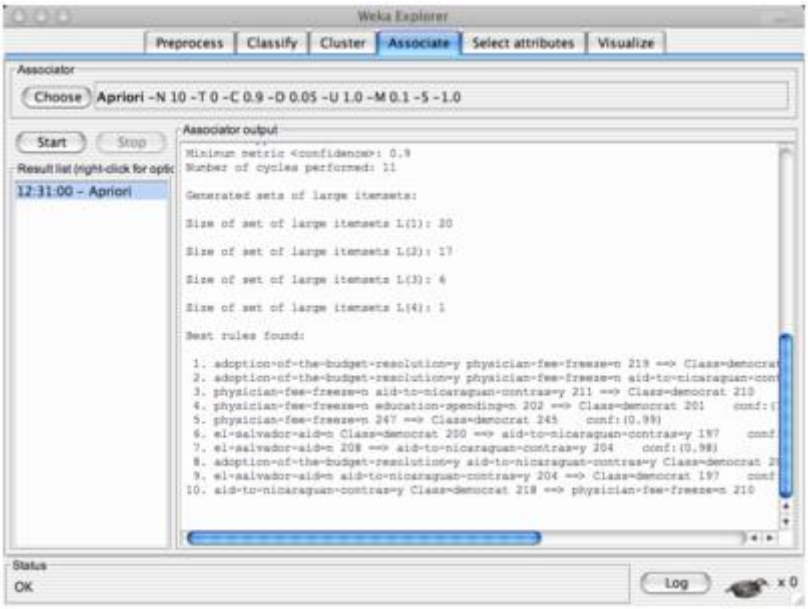
Figür 15 Weka Classifier Panel

2.6.3 Klasör Paneli (Cluster Panel)

Sınıflamaya benzer şekilde klasösrleme (gruplama) için kullanılan ekrandır ve benzer şekilde görselleştirme arayüzü bulunmaktadır.

2.6.4 Birleştirme Paneli (Associate Panel)

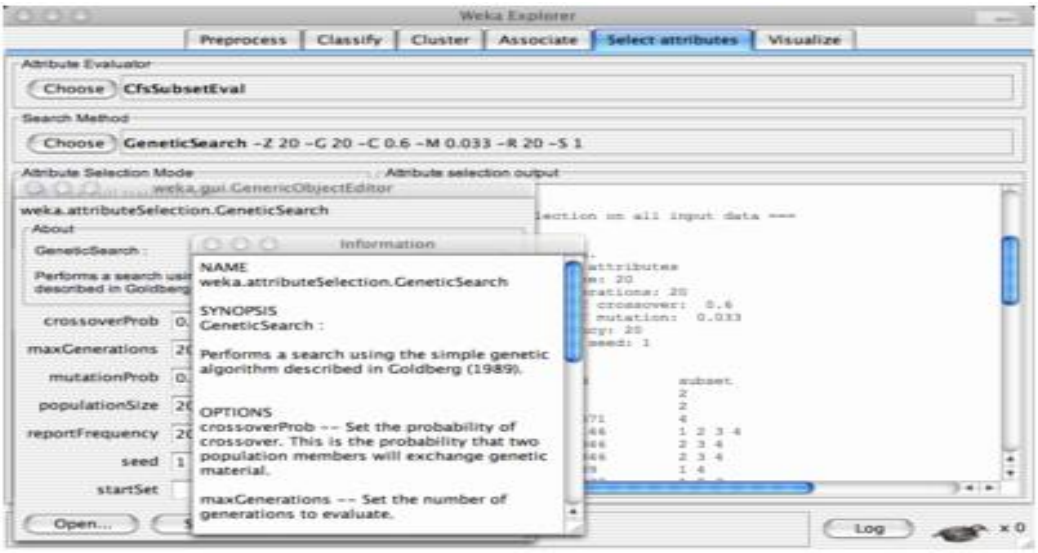
Weka içerisinde tanımlı birleştirme algoritmaları kullanılarak, mevcut veri kümesi üzerinde veri madenciliği (data mining) işlemi yapılmasını sağlar.



Figür 16 Weka Associate Panel

2.6.5 Seçim Özellikleri Paneli (Select Attributes Panel)

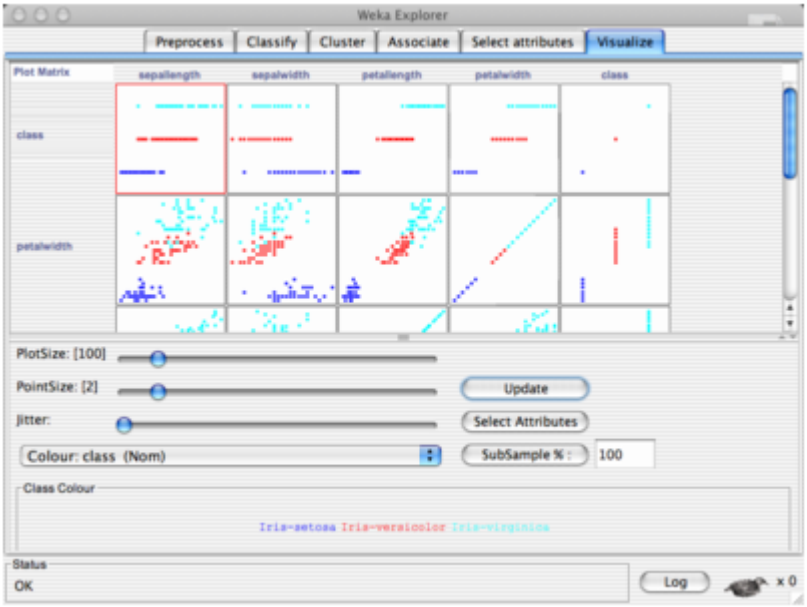
Veri kümesi üzerinde yapılan seçme ve işleme özelliklerini ayarlamaya yarar. Şayet seçme şemalarından birisi veriyi dönüştürüyorsa, dönüşmüş veri görselleştirme ekranında görülebilir.



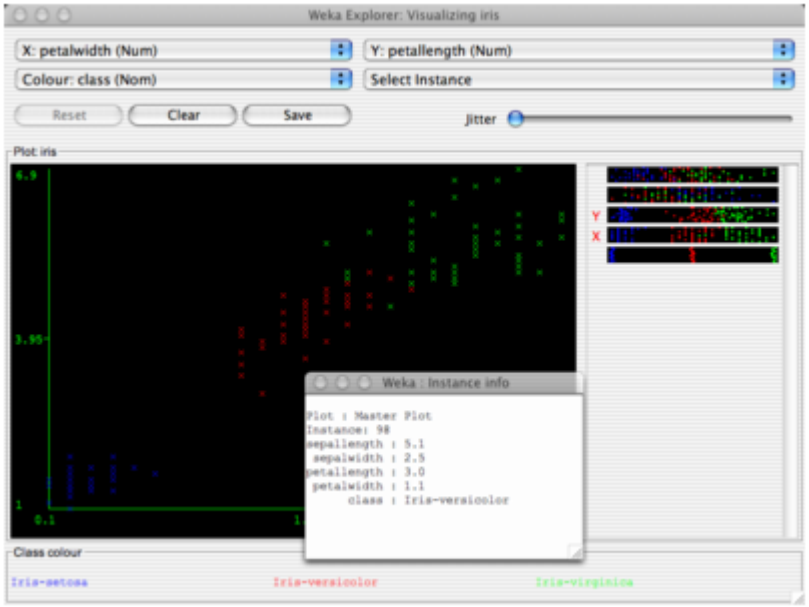
Figür 17 Weka Select Attibutes Panel

2.6.6 Visualize Panel

Bu panelde veri kümesi üzerinden bir çizim gösterilebilmektedir. Hücrelerin ve noktaların boyutları, ekranın alt tarafındaki panelden ayarlanabilir. Seçim özellikleri erkranından, matris üzerindeki hücre sayısı değiştirilebilir. Ayrıca çok büyük veri kümeleri ile çalışılırken, işlem kolaylığı olması açısından sadece alt örneklem uzayının kullanılması da mümkündür.



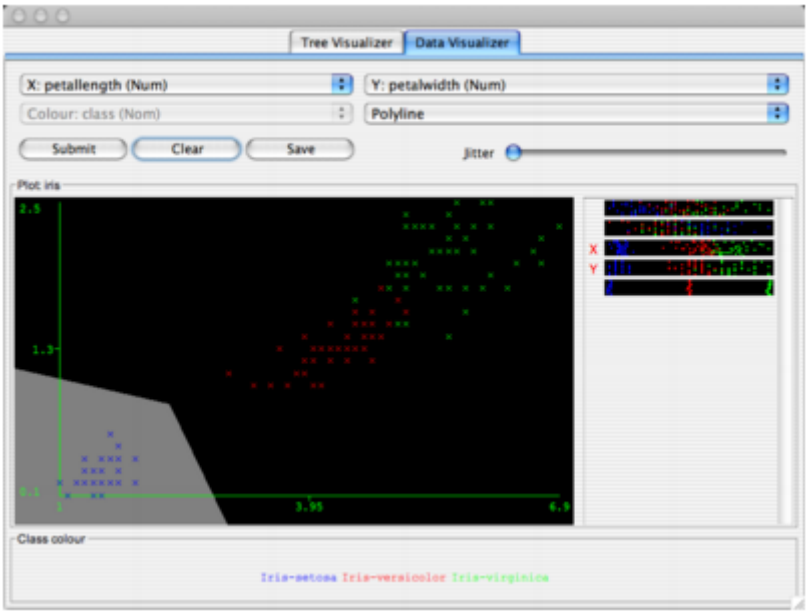
Figür 18 Weka Visulize Panel



Figür 19 Weka Visulize Panel-2

2.6.7Etkileşimli Karar Ağacı İnşası (Interactive decision tree construction)

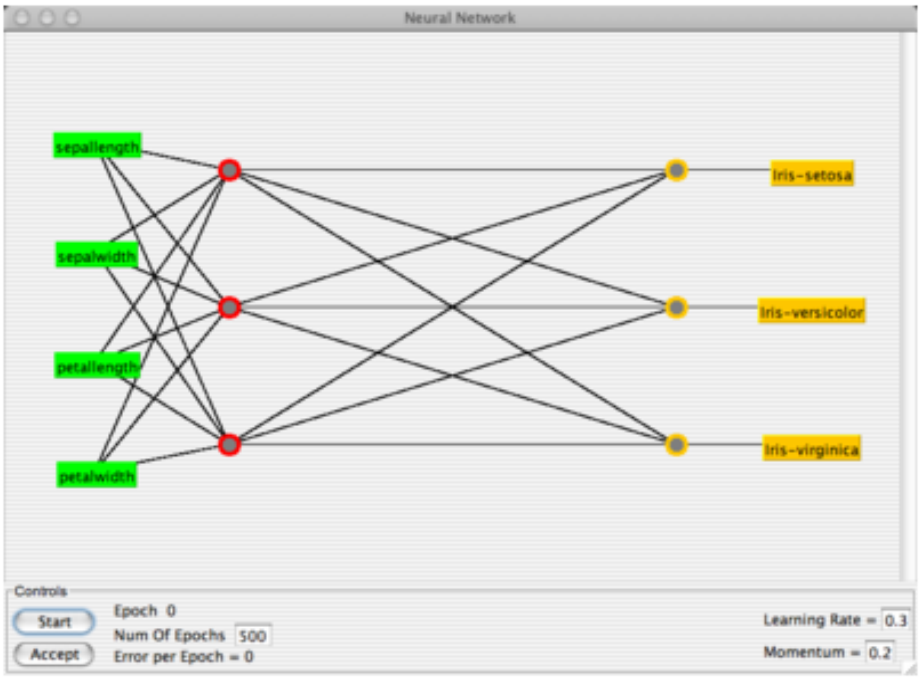
WEKA içerisindeki bu araç ile çift alternatifli (bi-variate) bölünmeleri ve bu bölünmeler üzerinde bir ağaç yapsını etkileşimli olarak inşa etmek mümkündür. Ayrıca inşa edilen bu ağacın yeniden değerlendirilmesi veya değiştirilmesi de mümkündür



Figür 20 Weka Etkileşimli Karar Ağacı İnşası

2.6.8Yapay Sinir Ağı (Neural Network GUI)

WEKA içerisinde bulunan yapay sinir ağı (neural network) arayüzüdür. Bu arayüz marifetiyle çok seviyeli perseptron (multi layer perceptron)ve eğitimi kontrol eden parametrelerin girilmesi mümkündür.



Figür 21 Weka Yapay Sinir Ağ

2.7 Web Uygulamasının Geliştirilme Ortamı Python (v3.5.2)

Python, Guido Van rossum adlı hollandalı bir programcı tarafından yazılmış bir programlama dilidir. Geliştirilmesine 1990 yılında başlayan Python; C ve C++ gibi programlama dillerine kıyaslarsak şöyle sonuçlar elde edebiliriz.

- Daha kolay öğrenilir.
- Program geliştirme sürecini kısaltır yani hızlı yazılır.
- Yukaridaki verilen programlama dillerine aksine ayrı bir derleyici ihtiyaci duymaz.
- Hem daha okunaklı, hem daha temiz kodsal söz dizimine sahiptir.

Python'un bu ve buna benzer özelliklerinden dolayı, dünya çapında ün sahibi büyük kuruluşlar (Google, Yahoo! Ve Dropbox gibi) bünyelerinde herzaman Python programcılarına ihtiyaç duyuyor.
Mesela pek çok büyük şirketin Python bilen programcılara iş olanağı sunduğu, Python'un baş geliştiricisi Guido Van Rossum'un 2005 ile 2012 yılları arasında Google'de çaliştığını, 2012 yılının sonlarında doğru ise Dropbox şirketine geçtiğini söylersek, bu programlama dilinin önemi ve geçerliliğini herhalde daha belirgin bir şekilde ortaya çıkaracaktır.
Bu arada Python dilinin, her nekadar Python Programlama dili ile ilgili çoğu görsel malzemeninn üzerine yılan resmi olarak görsek de, Python kelimesi aslında çoğu kişinin zannetiği aksine Piton anlamına gelmiyor. Python programlama dili ismini, Guido Van Rossum'un çok sevdiği Monty Python adlı altı kişilik bir ingiliz komedi grubun Monty Python's Flying Circus adlı gösterisinden gelmektedir.

Neden Python dili ?
Diğer programlama dilleri gibi tahmin edebilceğiniz gibi Python (C,C++,Perl,Ruby ve benzeri gibi) bir programlama dilidir. Tıpkı öteki yazılım dilleri gibi, önünüzde duran kara kutuya yani bilgisayara hükmetmenizi yani yönetmenizi sağlar.

Python dilinin öne çıkaran unsurlardan biri ise bilimsel yöntemlerde kullanılması ve çok hızlı işlem yapmasıdır [23].

2.8 Kullanılan Kütüphaneler

2.8.1Flask(v0.12.2) Web Framework

Flask, Python ortamında web uygulamaları geliştirmek için mikro seviyede bir frameworktür. Flask ile mikro seviyede RESTful web servisler geliştirmek hızlı ve kolaydır, dolayısıyla zaman kazandırır. Flask'ın bizim uygulamamızda kullanılma amacı da bu framework'ün genel amaçlarına bağlı olarak hızlı, kolay, komplikasyon yaratmadan küçük bir web uygulaması geliştirmektir. [24]

2.8.2VirtualEnv (v15.1.0)

VirtualEnv, projelerinizde gerekli olan paketleri sistemden bağımsız bir şekilde kurup, kullanmanızı sağlayacak sanal ortam sağlayan bir yapıdır. Ele alınan temel sorun bağımlılıklar ve versiyonlardan ve dolaylı izinlerden biridir. LibFoo'nun 1. sürümüne ihtiyaç duyan bir uygulamanız olduğunu düşünün, ancak başka bir uygulama sürüm 2'yi gerektirir. Her iki uygulamayı da nasıl kullanabilirsiniz? Her şeyi /usr/lib/python2.7/site-packages içine yüklerseniz (veya platformunuzun standart konumu ne olursa olsun), yeni sürüme geçirilmemesi gereken bir uygulamayı istemeden yükseltmek istediğiniz bir durumda sonlandırmak kolaydır.

Ya da daha genel olarak, bir uygulama yüklemek ve onu bırakmak isterseniz ne olur? Bir uygulama çalışıyorsa, kitaplıklarındaki herhangi bir değişiklik veya bu kitaplıkların sürümleri uygulamayı bozabilir.

Ayrıca, global site-paketleri dizinine paketleri kuramazsanız ne olur? Örneğin, paylaşılan başka bir ana bilgisayarda.

Örneğin, projenizde kullanmak istediğiniz modül sisteminizde yüklü fakat siz projenizde daha düşük veya daha yüksek bir sürümünü kullanmak istediniz ama direk sisteminize kurmak yerine, virtualenv sanal ortamı üzerine kurarak işiniz bittiğinde kaldırabilirsiniz. Bu sayede sisteminizde karışıklık yaratmamış olursunuz. Yüklemeleri tek komut satırıyla kaldırabilirsiniz. [25]

Tüm bu durumlarda, virtualenv size yardımcı olabilir. Kendi sanal dizin ortamlarıyla kütüphaneleri paylaşmayan kendi kurulum dizinlerine sahip bir ortam yaratır (ve isteğe bağlı olarak, global olarak yüklenen kitaplıklara da erişmez).[26]

2.8.3 Pillow (v1.1.7): PIL, Fredrik Lundh ve Katkıda bulunanlar tarafından hazırlanan Python Görüntüleme Kütüphanesidir (Python Imaging Library).

2.8.4 DropBox API : DropBox Api dosyaları DropBox uygulamasının bulut ortamına taşımak için kullanılan bir uygulama programlama arayüzüdür.

2.8.5NumPy(v1.14.2): Python'da bilimsel hesaplamanın temel paketidir. Çok boyutlu bir dizi nesnesi, çeşitli türetilmiş nesneler (maskelenmiş diziler ve matrisler gibi) ve dizilerdeki hızlı işlemler için matematiksel, mantıksal, şekil işleme, sıralama, seçme, g / Ç içeren bir dizi yordam sunan bir Python kütüphanesidir, Ayrık Fourier dönüşümü, temel doğrusal cebir, temel istatistik işlemler, rassal simülasyon ve çok daha fazlasının yapılabilmesi için gerekli olan kütüphanedir.

2.8.6 Imutils(v3.4.0) : OpenCV ve hem Python 2.7 hem de Python 3 ile çeviri, döndürme, yeniden boyutlandırma, iskeletleme ve Matplotlib görüntülerinin görüntülenmesi gibi temel görüntü işleme işlevlerini gerçekleştiren bir python kütüphanesidir.

2.8.7Dlib (v19.10.0): Dlib kütüphanesi C++ programlama dili ile geliştirilmiş ve 2002 yılından buyana geliştirilmeye devam eden içerisinde makine öğrenimi, derin öğrenme ve bilgisayarlı görü algoritmalarını barındıran açık kaynak kodlu bir kütüphanedir. C++ ve Python apisi sayesinde de Python programlama dili ile uygulama geliştirilebilmektedir. Farklı programlama dilleri ve platformlar içinde wrapper'ları geliştirilmiştir. Dlib OS X, MS Windows, Linux, Solaris, BSD ve HP-UX işletim sistemlerini desteklemektedir, ayrıca Raspberry, Tinkerboard gibi gömülü donanımlar veya Android, IOS mobil platformlar üzerinde çalışabilmektedir. [27]

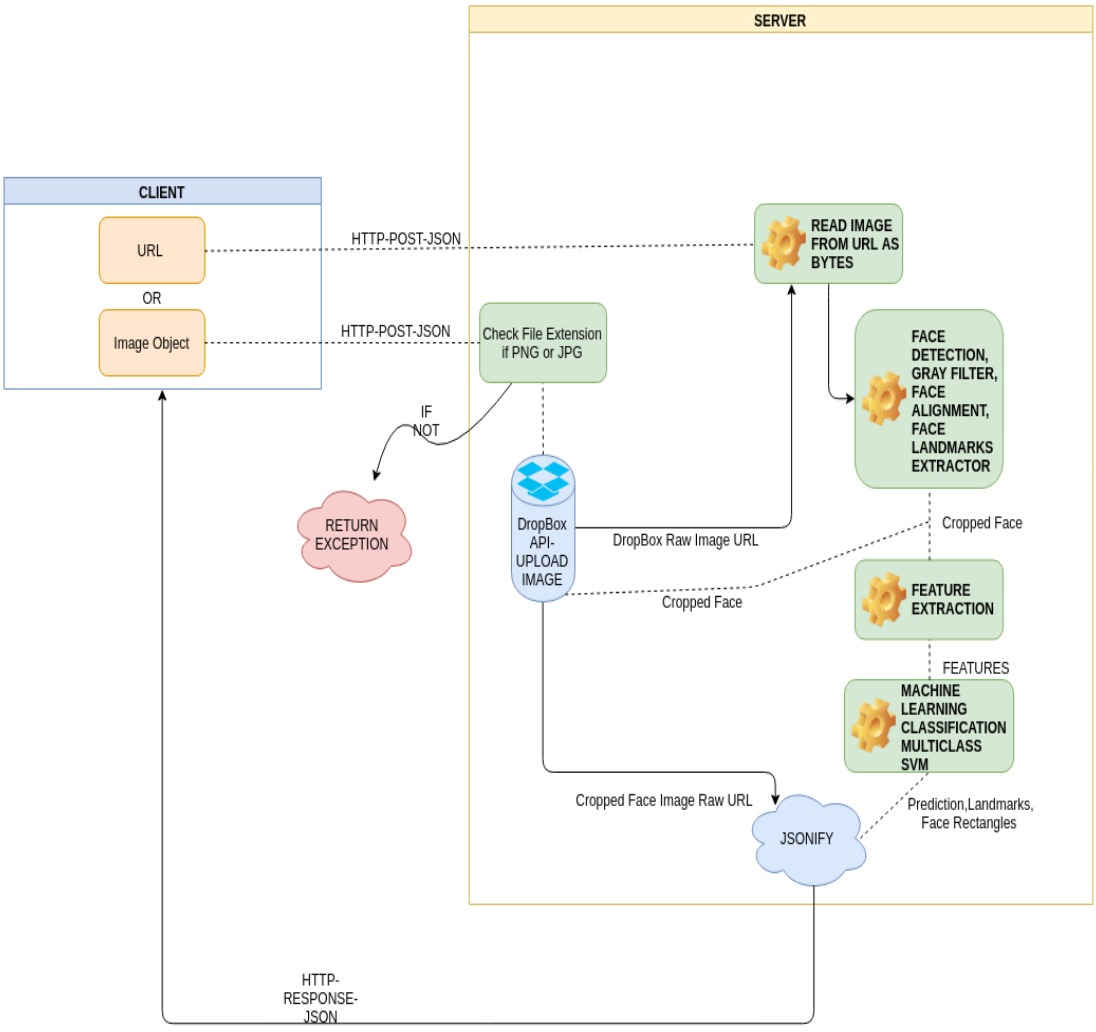
2.8.8Opencv2(cv2-v3.4.0): OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir. [28]

2.8.9 Sklearn(v0.19.1): Scikit-learn (eski scikits.learn), Python programlama dili için ücretsiz bir yazılım makinesi öğrenme kütüphanesidir. Destek vektör makineleri, rasgele ormanlar, gradyan artırma, k-araçları ve DBSCAN dahil olmak üzere çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarına sahiptir ve Python sayısal ve bilimsel kütüphaneler NumPy ve SciPy ile birlikte çalışmak üzere tasarlanmıştır.

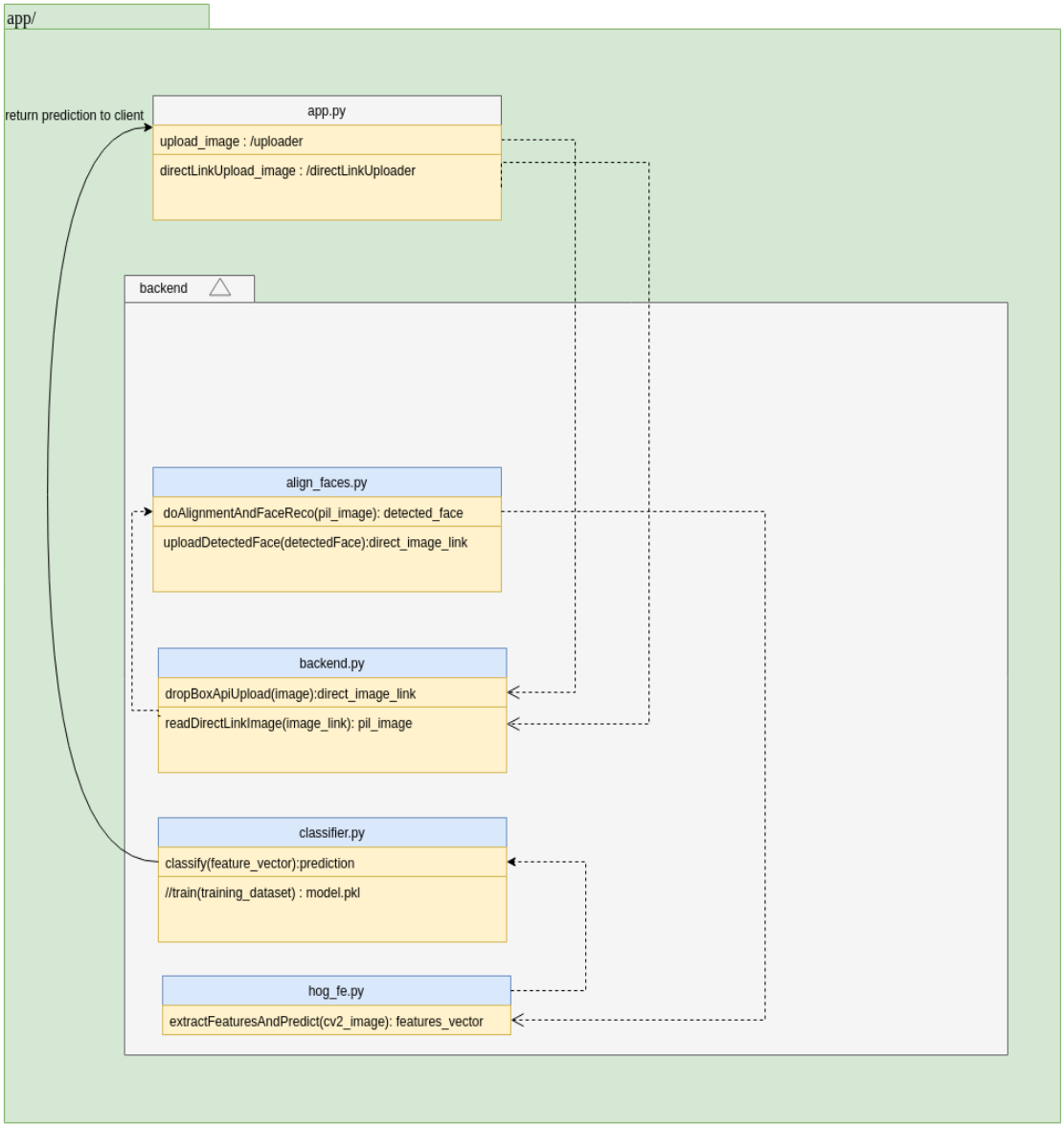
2.8.10 Pandas(v0.22.0): Python programlama dili için yüksek performanslı, kullanımı kolay veri yapıları ve veri analiz araçları sağlayan açık kaynaklı bir BSD lisanslı kütüphanedir [29]

3. YÖNTEMLER

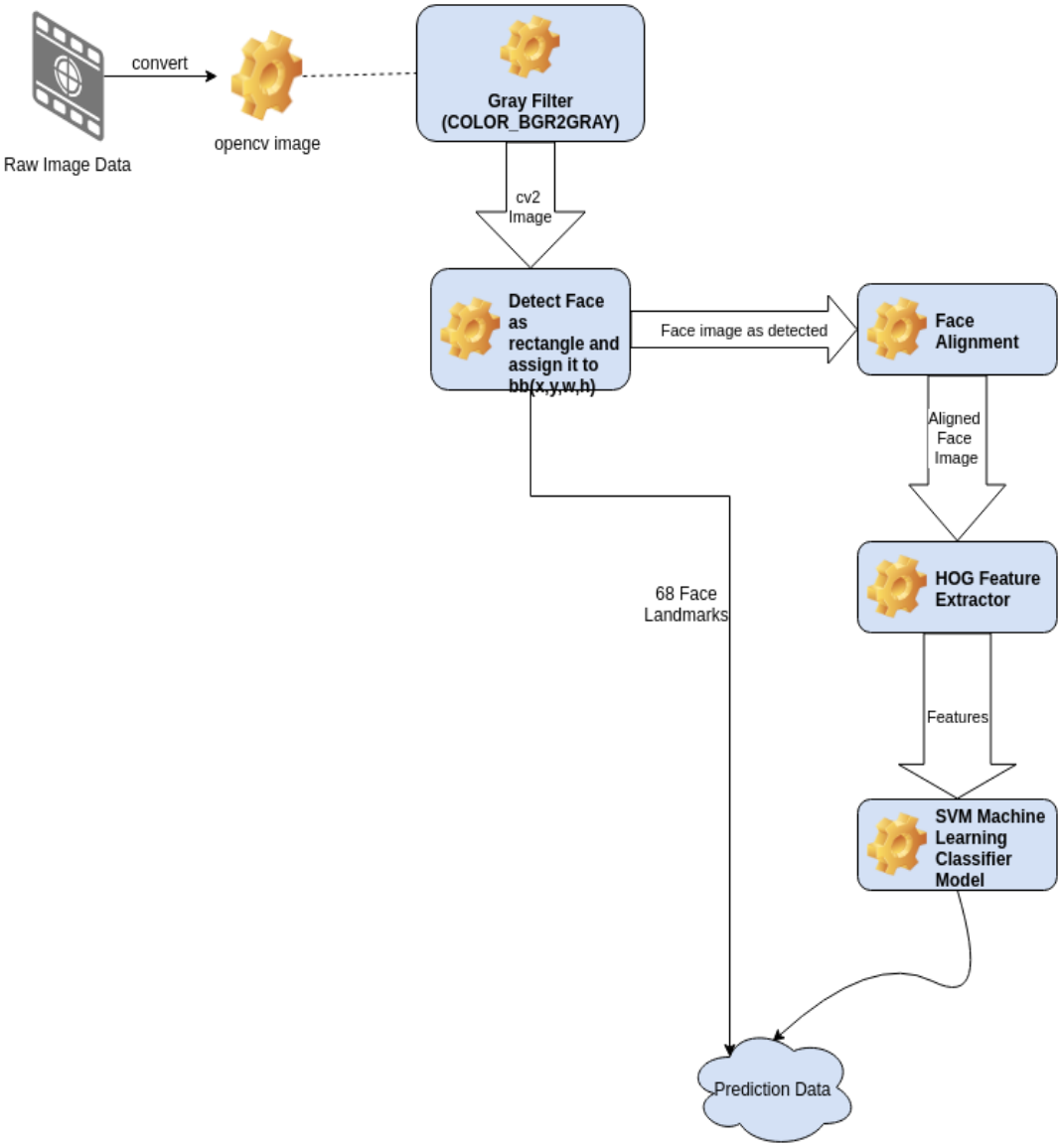
Figür 22 Uçtan Uca Web Uygulaması Akışı



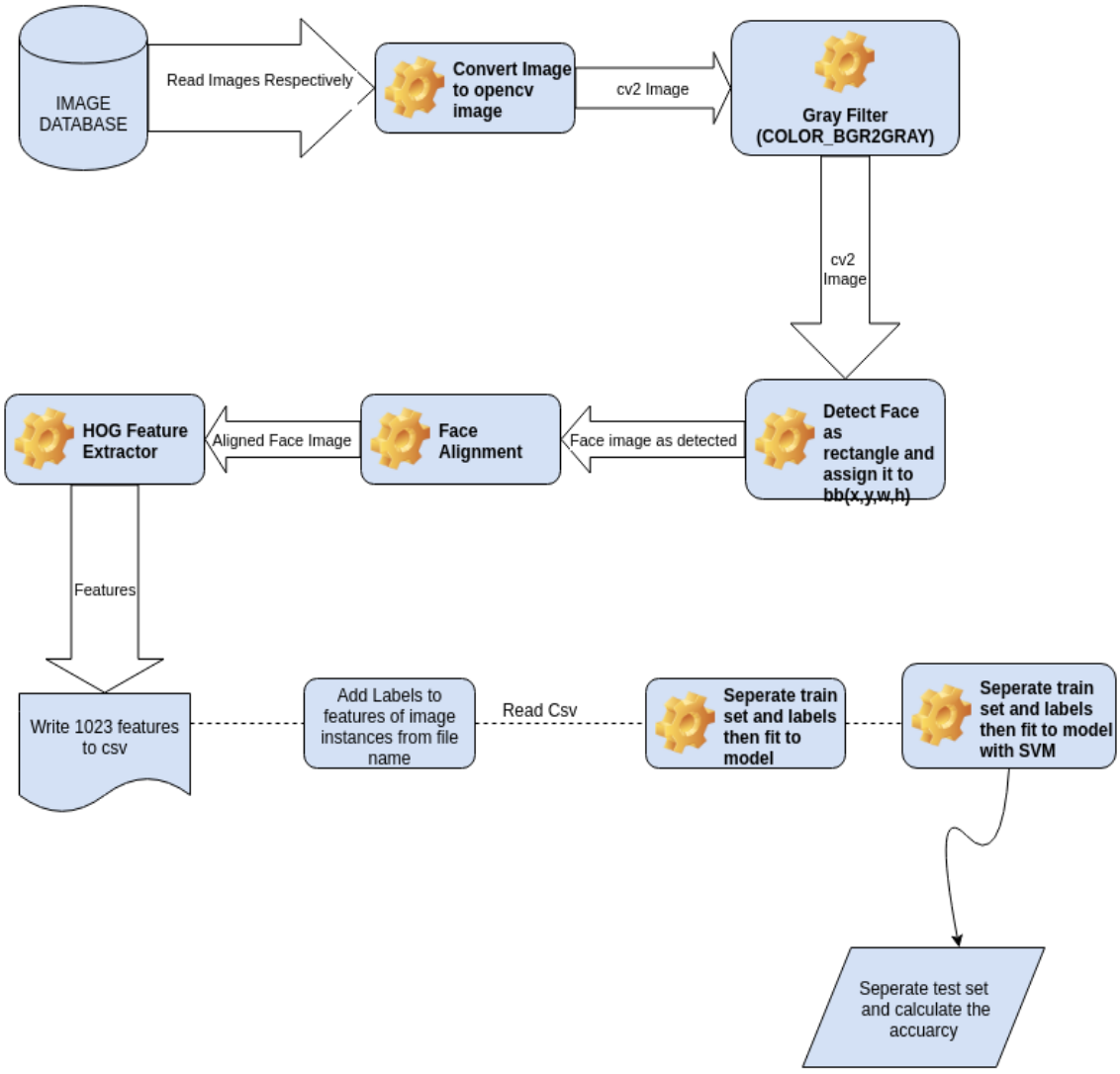
Figür 23 Server Client Akış Diagramı



Figür 24 Paket ve Fonksiyon Mimarisi



Figür 25 Makine Öğrenmesi Tahminleme Akış Diagramı



Figür 26 Makine Öğrenmesi Model Eğitimi Akış Şeması

4. SONUÇLAR

4.1 LBP ile Çıkarılmış Eğitim Verilerinin J48 Makine Öğrenmesi Algoritması ile Sınıflandırılması

a=korkmuş, b=öfkeli, c=tiksinme, d=mutlu, e=natural, f=üzgün, g=şaşkın,

```
=== Summary ===

Correctly Classified Instances         119               38.6364 %
Incorrectly Classified Instances       189               61.3636 %
Kappa statistic                          0.2841
Mean absolute error                      0.1757
Root mean squared error                  0.4059
Relative absolute error                 71.7367 %
Root relative squared error            115.999  %
Total Number of Instances              308

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 11  5  6  4  6  6  6 |  a = 1
  8  9  7  2 10  6  2 |  b = 2
  7  6 15  8  1  5  2 |  c = 3
  1  5  1 34  0  0  3 |  d = 4
  8  5  1  0 17 12  1 |  e = 5
  6  6  6  2  7 13  4 |  f = 6
 12  4  0  3  3  2 20 |  g = 7
```

Tablo 3 LBP Test veri seti ile

```
Correctly Classified Instances         298               44.3452 %
Incorrectly Classified Instances       374               55.6548 %
Kappa statistic                          0.3507
Mean absolute error                      0.1602
Root mean squared error                  0.3858
Relative absolute error                 65.4175 %
Root relative squared error            110.2503 %
Total Number of Instances              672

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 20 15 16  5 11 15 14 |  a = 1
 16 36 14  3  8 10  9 |  b = 2
 10 15 47  9  5  7  3 |  c = 3
  4  8  9 68  2  1  4 |  d = 4
 13 10  2  1 42 23  5 |  e = 5
 12 17  7  1 21 33  5 |  f = 6
 15  7  3  7  7  5 52 |  g = 7
```

Tablo 4 LBP Çapraz Doğrulama K-Katlama 8 Tutularak

4.2 LBP ile Çıkarılmış Eğitim Verilerinin Rassal Ormanlar Makine Öğrenmesi Algoritması ile Sınıflandırılması

```
Correctly Classified Instances         207               67.2078 %
Incorrectly Classified Instances       101               32.7922 %
Kappa statistic                          0.6174
Mean absolute error                      0.2044
Root mean squared error                  0.3003
Relative absolute error                 83.4508 %
Root relative squared error             85.8306 %
Total Number of Instances              308

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 14  5  2  6  4  6  7 |  a = 1
  3 26  3  1  9  2  0 |  b = 2
  2  3 31  4  1  3  0 |  c = 3
  0  1  0 42  0  0  1 |  d = 4
  2  2  0  0 34  6  0 |  e = 5
  3  3  5  1  6 26  0 |  f = 6
  1  3  0  1  5  0 34 |  g = 7
```

Tablo 5 LBP –Rassal Orman Test Veri Seti İle

Correctly Classified Instances	459	68.3036 %
Incorrectly Classified Instances	213	31.6964 %
Kappa statistic	0.6302	
Mean absolute error	0.2041	
Root mean squared error	0.3002	
Relative absolute error	83.3316 %	
Root relative squared error	85.7777 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
29	8	3	9	9	17	21	a = 1
6	53	14	1	12	10	0	b = 2
2	8	75	3	2	6	0	c = 3
1	0	0	94	0	1	0	d = 4
3	5	0	0	76	8	4	e = 5
11	7	4	3	15	52	4	f = 6
10	0	0	1	4	1	80	g = 7

Tablo 6 LBP –Rassal Orman Çapraz Doğrulama K-Katlama 8 Tutularak

4.3 LBP ile Çıkarılmış Eğitim Verilerinin SMO Makine Öğrenmesi Algoritması İle Sınıflandırılması

Correctly Classified Instances	238	77.2727 %
Incorrectly Classified Instances	70	22.7273 %
Kappa statistic	0.7348	
Mean absolute error	0.2089	
Root mean squared error	0.3087	
Relative absolute error	85.3175 %	
Root relative squared error	88.2209 %	
Total Number of Instances	308	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
67	1	1	1	5	9	12	a = 1
4	75	12	0	2	3	0	b = 2
1	8	76	1	0	10	0	c = 3
1	1	0	93	0	1	0	d = 4
7	0	0	0	86	3	0	e = 5
13	2	4	0	8	69	0	f = 6
13	0	0	1	2	0	80	g = 7

Tablo 7 LBP –SMO Test Veri Seti İle

Correctly Classified Instances	546	81.25 %
Incorrectly Classified Instances	126	18.75 %
Kappa statistic	0.7813	
Mean absolute error	0.208	
Root mean squared error	0.3072	
Relative absolute error	84.9372 %	
Root relative squared error	87.7976 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
67	1	1	1	5	9	12	a = 1
4	75	12	0	2	3	0	b = 2
1	8	76	1	0	10	0	c = 3
1	1	0	93	0	1	0	d = 4
7	0	0	0	86	3	0	e = 5
13	2	4	0	8	69	0	f = 6
13	0	0	1	2	0	80	g = 7

Tablo 8 LBP –SMO Çapraz Doğrulama K-Katlama 8 Tutularak

4.4 HOG ile Çıkarılmış Eğitim Verilerinin J48 Makine Öğrenmesi Algoritması ile Sınıflandırılması

Correctly Classified Instances	169	54.8701 %
Incorrectly Classified Instances	139	45.1299 %
Kappa statistic	0.4735	
Mean absolute error	0.1307	
Root mean squared error	0.3489	
Relative absolute error	53.3511 %	
Root relative squared error	99.7075 %	
Total Number of Instances	308	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
15	7	4	3	1	3	11	a = 1
5	14	8	2	4	7	4	b = 2
3	11	22	4	1	2	1	c = 3
2	1	3	35	0	1	2	d = 4
2	3	0	1	25	12	1	e = 5
8	2	3	1	5	23	2	f = 6
3	2	0	1	2	1	35	g = 7

Tablo 9 HOG J48 Test Veri Seti İle

Correctly Classified Instances	388	57.7381 %
Incorrectly Classified Instances	284	42.2619 %
Kappa statistic	0.5069	
Mean absolute error	0.1219	
Root mean squared error	0.336	
Relative absolute error	49.7757 %	
Root relative squared error	96.0267 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
40	9	7	3	6	11	20	a = 1
12	46	10	8	6	12	2	b = 2
6	19	51	6	5	8	1	c = 3
4	4	9	79	0	0	0	d = 4
5	7	0	0	67	14	3	e = 5
17	11	6	1	14	43	4	f = 6
22	5	1	1	5	0	62	g = 7

Tablo 10 HOG J48 Çapraz Doğrulama K-Katlama 8 Tutularak

4.5 HOG ile Çıkarılmış Eğitim Verilerinin Rassal Ormanlar Makine Öğrenmesi Algoritması ile Sınıflandırılması

Correctly Classified Instances	240	77.9221 %
Incorrectly Classified Instances	68	22.0779 %
Kappa statistic	0.7424	
Mean absolute error	0.1588	
Root mean squared error	0.2508	
Relative absolute error	64.8258 %	
Root relative squared error	71.674 %	
Total Number of Instances	308	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
24	4	4	2	3	1	6	a = 1
2	30	5	2	2	3	0	b = 2
1	3	36	1	0	3	0	c = 3
0	0	2	42	0	0	0	d = 4
1	2	0	0	38	2	1	e = 5
0	3	1	1	7	32	0	f = 6
3	0	0	0	3	0	38	g = 7

Tablo 11 HOG Rassal Orman Test Veri Seti İle

Correctly Classified Instances	552	82.1429 %
Incorrectly Classified Instances	120	17.8571 %
Kappa statistic	0.7917	
Mean absolute error	0.1542	
Root mean squared error	0.2439	
Relative absolute error	62.9809 %	
Root relative squared error	69.6978 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
54	3	1	2	10	11	15	a = 1
8	68	13	0	2	5	0	b = 2
0	5	83	2	0	6	0	c = 3
2	0	0	93	0	1	0	d = 4
1	3	0	0	88	4	0	e = 5
2	1	0	0	14	78	1	f = 6
5	0	0	0	1	2	88	g = 7

Tablo 12 HOG Rassal Orman Çapraz Doğrulama K-Katlama 8 Tutularak

4.6 HOG ile Çıkarılmış Eğitim Verilerinin SMO Makine Öğrenmesi Algoritması ile Sınıflandırılması

Correctly Classified Instances	254	82.4675 %
Incorrectly Classified Instances	54	17.5325 %
Kappa statistic	0.7955	
Mean absolute error	0.2077	
Root mean squared error	0.3067	
Relative absolute error	84.8124 %	
Root relative squared error	87.6583 %	
Total Number of Instances	308	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
31	4	4	0	0	2	3	a = 1
3	32	5	1	2	1	0	b = 2
1	3	38	1	0	1	0	c = 3
0	0	1	43	0	0	0	d = 4
1	0	0	0	40	3	0	e = 5
2	2	3	1	4	32	0	f = 6
4	0	0	1	1	0	38	g = 7

Tablo 13 HOG-SMO Test Veri Seti İle

Correctly Classified Instances	607	90.3274 %
Incorrectly Classified Instances	65	9.6726 %
Kappa statistic	0.8872	
Mean absolute error	0.2061	
Root mean squared error	0.304	
Relative absolute error	84.1435 %	
Root relative squared error	86.8758 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
76	2	0	2	2	7	7	a = 1
1	87	6	0	0	2	0	b = 2
2	3	86	0	0	5	0	c = 3
1	0	0	94	1	0	0	d = 4
0	0	0	0	93	2	1	e = 5
6	1	1	0	3	85	0	f = 6
9	0	0	0	0	1	86	g = 7

Tablo 14 HOG-SMO Çapraz Doğrulama K-Katlama 8 Tutularak

4.7 HOG ve LBP Çıkarılmış Eğitim Verilerinin Kombin Edilerek K-Katlama 8 Tutularak J48, Rassal Ormanlar ve SMO Makine Öğrenmesi Algoritmalarında Sınıflandırma Başarımlarının Deneylenmesi

Correctly Classified Instances	365	54.3155 %
Incorrectly Classified Instances	307	45.6845 %
Kappa statistic	0.467	
Mean absolute error	0.1312	
Root mean squared error	0.3509	
Relative absolute error	53.559 %	
Root relative squared error	100.2753 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
39	9	5	4	8	13	18	a = 1
8	40	14	8	8	12	6	b = 2
4	24	51	8	1	7	1	c = 3
1	4	10	79	0	1	1	d = 4
12	7	0	0	59	15	3	e = 5
12	15	7	1	11	44	6	f = 6
28	5	5	1	1	3	53	g = 7

Tablo 15 HOG-LBP K-Katlama 8 Tutularak J48

Correctly Classified Instances	321	47.7679 %
Incorrectly Classified Instances	351	52.2321 %
Kappa statistic	0.3906	
Mean absolute error	0.1492	
Root mean squared error	0.3863	
Relative absolute error	60.9375 %	
Root relative squared error	110.397 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
31	12	9	12	7	8	17	a = 1
13	36	11	4	16	11	5	b = 2
7	13	52	5	3	15	1	c = 3
13	1	7	65	2	5	3	d = 4
9	13	3	1	48	15	7	e = 5
20	8	10	1	8	42	7	f = 6
26	6	4	4	6	3	47	g = 7

Tablo 16 HOG-LBP K-Katlama 8 Tutularak Rassal Ormanlar İle

Correctly Classified Instances	606	90.1786 %
Incorrectly Classified Instances	66	9.8214 %
Kappa statistic	0.8854	
Mean absolute error	0.206	
Root mean squared error	0.304	
Relative absolute error	84.1104 %	
Root relative squared error	86.8785 %	
Total Number of Instances	672	

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
74	1	0	2	3	8	8	a = 1
1	86	6	0	0	3	0	b = 2
2	3	84	1	0	6	0	c = 3
1	0	0	94	1	0	0	d = 4
1	0	0	0	93	2	0	e = 5
5	0	1	0	4	86	0	f = 6
5	0	0	1	1	0	89	g = 7

Tablo 17 HOG-LBP K-Katlama 8 Tutularak SMO ile

Sınıflandırma Algoritması /Görüntü Betimleyici	J48 Ağaç	Random Forest	Support Vector Machine (SVM)
LBP	%44.34	%68.30	%81.25
HOG	%57.73	%82.14	%90.32
HOG+LBP	%54.31	%47.76	%90.17

Tablo 18 Test olarak k-kat çaprazlama yöntemi k=8 tutularak uygulandığında başarı oranları

Sınıflandırma Algoritması /Görüntü Betimleyici	J48 Ağaç	Random Forest	Support Vector Machine (SVM)
LBP	%38.63	%67.20	%77.27
HOG	%54.87	%77.92	%82.46
HOG+LBP	%51.29	%78.24	%84.41

Tablo 19 Test olarak test veri seti kullanıldığında başarı oranları

Weka makine öğrenmesi programında, makine öğrenmesi yöntemleri uygulandıktan sonra bulgular üzerinde analiz yapılmıştır. En yüksek sınıflandırma başarısı sonuçları içinde HOG ve LBP+HOG öznitelikleri barındıran veri setleri olmuştur. Bunlardan HOG ile çıkarılmış öznitelikleri bulunduran veri seti SMO ile k-kat çaprazlamalı test güdümlü sınıflandırma algoritmasıyla %90.3274 başarı oranına ulaşarak en yüksek başarımı elde etmiştir. Genel olarak en başarılı sınıflandırma algoritması SMO olarak saptanmıştır. Deneylerde aynı zamanda HOG ile çıkarılmış özniteliklerin LBP ile çıkarılmış özniteliklerin ile birleştirilmesinden sonra sınıflandırma sonuçlarınının yine de sadece HOG ile çıkarılmış özniteliklerin sınıflandırma başarısına yakın bir oran sergilediği görülmüştür.

Veritabanımızı kullanarak çıkarttığımız öznitelikler ile modelimizi kütüphaneler yardımıyla programatik olarak eğittikten sonra web uygulamasına entegre ettik. Bir

kullanıcı olarak yaptığımız testler sonucunda önden çekilmiş ve efekt verilmemiş yüzlerin modelimiz tarafından başarılı bir şekilde tahmin edildiğini gördük. Yandan çekilmiş yüzlerin duygu durumlarının veritabanımızda mevcut olmaması sebebiyle modelimiz tarafından tahmin edilemediğini gözlemledik. Ayrıca veritabanımızdaki yüzlerin gürültüsüz(sakal,yara izi, makyaj) olması sebebiyle modelin eğitim durumu etkilenmiş ve kullanıcı testlerinde bu durum olumsuz olarak saptanmıştır.

Yaptığımız araştırmalar sonucunda derin sinir ağlarının kullanılması, veritabanının yeterli büyüklükte olması modelin başarılı tahmin yapmasında doğrudan etkili olduğu gözlemlenmiştir.

Eğitim sürecimiz her ne kadar kişi bağımsız olsa insan yüzlerinin aynı tepkiyi ifade etmesine rağmen belli nirengi noktalarının(kaşlar,alt-üst dudak, göz çizgileri) farklı açılarda olması modelin tahminine etki etmiştir.

İğrenme, Üzgün ve Kızgın duygu durumları negatif duygu durumları olduğu ve birbirine yakın yüz ifadelerini ifade ettiği için modelimizin bu duygu durumlarını tahmin ederken yanlış tahminler verdiği saptanmıştır

5.KAYNAKLAR

[1]A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery Juliano E. C. Cruza1 , Elcio H. Shiguemorib2 and Lamartine N. F. Guimar˜aes 2015

[2] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 886-893, 2005. doi: 10.1109/CVPR.2005.177

[3] T. Ojala, M. Pietikainen, and D. Harwood, Performance evaluation of texture measures with classification based on kullback discrimination of distributions, International Conference on Pattern Recognition,1:582-585, 1994.doi:10.1109/ICPR.1994.576366

[4] C. Chen, Handbook of pattern recognition and computer vision, World Scientific Publishing Company,Incorporated, 2009.

[5] YEREL İKİLİ ÖRÜNTÜ (LBP) VE ÖNSEL ŞEKİL BİLGİSİ TABANLI BİR DESEN BÖLÜTLEME METODU Erkin Tekeli, Müjdat Çetin, Aytül Erçil

[6] T. Ahonen, A. Hadid and M. Pietik¨ainen, Face recognition with local binary patterns, ECCV 2004, LNCS 3021 (2004), 469–481.

[7] A. Hadid, M. Pietikainen and T. Ahonen, A discriminative feature space for detecting and recognizing faces, 2004 IEEE Computer Society Conference on Computer Vision and PatternRecognition (CVPR'04) 2 (2004), 797–804.

[8] H-C Lian and B-L Lu, Multi-view gender classification using local binary patterns and support vector machines, ISNN (2)'06, LNCS 3972 (2006), 202–209.

[9] L. He, C. Zou, L. Zhao and D. Hu, An enhanced LBP feature based on facial expression

recognition, Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual

Conference (2006), 3300–3303

[10] T. Ojala, M. Pietikäinen, T. Mäenpää, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns", IEEE Trans. on Pattern

Anal. Mac. Intel., vol. 24, pp. 971 – 987,2002.

[11] B. Kurt, V.V. Nabiyev, "Down syndrome recognition using local binary patterns and statistical evaluation of the system", Expert Systems with Applications, vol. 38, pp. 8690–

8695, 2011.

[12] Kaya, Y., Kayci, L., Tekin, R., & Faruk Ertuğrul, Ö, "Evaluation of texture features for automatic detecting butterfly species using extreme learning machine", Journal of

Experimental & Theoretical Artificial Intelligence, Vol. 26(2), pp. 267-281, 2014.

[13] Shashua A., Gdalyahu Y., and Hayon G., "Pedestrian detection for driving assistance systems: Single-frame classification and system level performance", In Proceedings of IEEE Intelligent Vehicles Symposium, 2004.

[14] N. Dalal and B. Triggs., "Histograms of oriented gradients for human detection", In C. Schmid, S. Soatto, and C. Tomasi, editors, International Conference on Computer

[15] http://bilgisayarkavramlari.sadievrenseker.com/2008/12/01/svm-support-vector-machine-destekci-vektor-makinesi/

[16] Quinlan R.J C4.5: Programs for Machine Learning Q325.5.Q56 1993

[17]https://medium.com/@Emreyz/y%C3%B6ntemler-4-1-c4-5-algoritmas%C4%B1 7382de92584e 17

[18] Contrasting and Combining Least Squares Based Learnersfor Emotion Recognition in the Wild 18

[19] T. Ojala, M. Pietikainen, and T. Maenpaa.Multiresolution gray-scale and rotation invarianttexture classification with local binary patterns. Pattern Analysis and Machine Intelligence, IEEETransactions on, 24(7):971–987, 2002. 19

[20] http://bilgisayarkavramlari.sadievrenseker.com/2012/11/13/c4-5-agaci-c4-5-tree/

[21] http://bilgisayarkavramlari.sadievrenseker.com/2009/06/01/weka/

[22] http://uzalcbs.org/wp-content/uploads/2016/11/2010_18.pdf

[23] https://www.python.tc/python-nedir/

[24]https://virtualenv.pypa.io/en/stable/

[25]https://medium.com/gokhanyavas/virtualenv-nedir-nas%C4%B1l-kurulur-205f1fd57f05

[26]http://mesutpiskin.com/blog/dlib-ile-makine-ogrenimi-ve-goruntu-isleme-1-giris.html

[27]http://mesutpiskin.com/blog/opencv-nedir.html

[28]https://en.wikipedia.org/wiki/Scikit-learn

[29] https://pandas.pydata.org/

[30] https://azure.microsoft.com/tr-tr/services/cognitive-services/emotion/

[31] Kaya H, Gürpınar F., Salah A. Combining Deep Facial and Ambient Features for First Impression Estimation ECCV 2016: Computer Vision – ECCV 2016 Workshops pp 372-385

[32] Kaya H, Gürpınar F., Salah A. Video-based emotion recognition in the wild using deep transfer learning and score fusion jimavis.2017.01.012

TEŞEKKÜR

İlk olarak proje çalışmamızda bilgi ve deneyimlerini esirgemeyen hocamız Dr. Öğr.Üyesi Heysem KAYA'a teşekkür ederiz.

Lisans eğitimiz boyunda emeklerinden dolayı tüm hocalarımıza teşekkürü borç biliriz.

Son olarak bugünlere gelmemizde bizim kadar emeği olan, bizden maddi ve manevi desteklerinden dolayı ailelerimize teşekkür ederiz.