



**KASTAMONU ÜNİVERSİTESİ**  
**MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Bitirme Projesi**

**Leap Motion Sensör İle Türk İşaret Dilini Tanıma**

DANIŞMAN: Dr. Öğr. Üyesi Abdulkadir KARACI

134410010	Bedirhan SAĞLAM
134410003	Emre YARAR
144410027	Gülsüm BALCİ

Kastamonu, 2018

# İÇİNDEKİLER

Sayfa

## ÖZET

### Hata! Yer işaretü tanımlanmamış.

1.GİRİŞ .....	6
1.1 UYGULAMA KONUSU .....	6
1.2 UYGULAMA AMACI .....	6
1.3 UYGULAMA KULLANICI KİTLESİ .....	6
1.4 UYGULAMA PLANI.....	7
2. MAKİNE ÖĞRENMESİ VE LEAP MOTION.....	8
2.1 LEAP MOTION NEDİR? .....	8
2.1.1 LEAP MOTION NASIL ÇALIŞIR?.....	8
2.1.2 LEAP MOTION NASIL ÇALIŞIR? .....	8
2.1.3 LEAP MOTION TEKNİK ÖZELLİKLER .....	8
2.2. MAKİNE ÖĞRENMESİ ALGORİTMALARI .....	9
2.2.1 RASSAL ORMAN (RANDOM FOREST ) .....	9
2.2.2 KARAR AĞAÇLARI (DECISION TREE) .....	10
2.2.3 NAVİE BAYES .....	11
2.2.4 DESTEK VEKTÖR MAKİNELERİ (SUPPORT VECTOR MACHINE) .....	12
2.2.5 YAPAY SİNİR AĞLARI (NEURAL NETWORK) .....	13
3. YÖNTEM .....	14
3.1 VERİ SETİNİN OLUŞTURULMASI.....	14
3.1.1 VERİLERİN KAYIT EDİLMESİ DOSYA YAPISI VE KODLAR.....	14
3.1.2 KODLAR.....	16
3.2 AĞIN EĞİTİLMESİ VE OFFLINE TEST.....	20
3.2.1 391 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ .....	20
3.2.1.1 DOSYA YAPISI .....	20
3.2.1.2 KODLAR VE AÇIKLAMALAR.....	21
3.2.1.3 ARAYÜZ TASARIMI .....	26
3.2.1.4 OFFLINE TEST SONUÇLARI.....	31
3.2.2 121 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ .....	32
3.2.2.1 DOSYA YAPISI .....	32
3.2.2.2 KODLAR VE AÇIKLAMALAR.....	33
3.2.2.3 ARAYÜZ TASARIMI.....	39
3.2.2.4 OFFLINE TEST SONUÇLARI.....	44
3.3 ONLINE TEST.....	45

3.3.1 DOSYA YAPISI .....	45
3.3.2 KODLAR VE AÇIKLAMALAR.....	45
3.3.3 ÇALIŞMA ORTAMI VE ARAYÜZ .....	52
<b>4. BULGULAR VE SONUÇLAR .....</b>	<b>53</b>
<b>4.1 391 ÖZNİTELİKLİ VERİ SETİNİN BAŞARIM ORANLARI VE KARMAŞIKLIK MATRİSİ .....</b>	<b>53</b>
<b>4.1.1 BAŞARIM ORANLARI .....</b>	<b>53</b>
<b>4.1.2 RASSAL ORMAN KARMAŞIKLIK MATRİSİ .....</b>	<b>54</b>
<b>4.1.3 DESTEK VECTÖR MAKİNESİ KARMAŞIKLIK MATRİSİ .....</b>	<b>55</b>
<b>4.1.4 YAPAY SİNİR AĞLARI KARMAŞIKLIK MATRİSİ.....</b>	<b>56</b>
<b>4.1.5 KARAR AĞAÇLARI KARMAŞIKLIK MATRİSİ .....</b>	<b>57</b>
<b>4.1.6 NAVİE BAYES KARMAŞIKLIK MATRİSİ .....</b>	<b>58</b>
<b>4.2 121 ÖZNİTELİKLİ VERİ SETİNİN BAŞARIM ORANLARI VE KARMAŞIKLIK MATRİSİ .....</b>	<b>59</b>
<b>4.2.1 BAŞARIM ORANLARI .....</b>	<b>59</b>
<b>4.2.2 RASSAL ORMAN KARMAŞIKLIK MATRİSİ .....</b>	<b>60</b>
<b>4.2.3 DESTEK VECTÖR MAKİNESİ KARMAŞIKLIK MATRİSİ .....</b>	<b>61</b>
<b>4.2.4 YAPAY SİNİR AĞLARI KARMAŞIKLIK MATRİSİ.....</b>	<b>62</b>
<b>4.2.5 KARAR AĞAÇLARI KARMAŞIKLIK MATRİSİ .....</b>	<b>63</b>
<b>4.2.6 NAVİE BAYES KARMAŞIKLIK MATRİSİ .....</b>	<b>64</b>
<b>4.3 DEĞERLENDİRME.....</b>	<b>65</b>
<b>KAYNAKÇA .....</b>	<b>66</b>

<b>ŞEKİLLER</b>	<b>Sayfa</b>
-----------------	--------------

Şekil 1 Leap Motion (Üstten Altta Karşıdan Sağ ve Soldan görünüm) .....	8
Şekil 2 Leap Motion ve Temsili Algıladığı Eksenler .....	8
Şekil 3 Karar Ağacı Örneği.....	10
Şekil 4 VERİ KAYIT ÇALIŞMA ORTAMI .....	14
Şekil 5 VERİLERİN KAYIT EDİLMESİ DOSYAYAPISI .....	14
Şekil 6 LeapCode.py Çalışması ve Verilerin Kayıt Edilmesi .....	15
Şekil 7 Verilerin Kayıt Edilmesi .....	16
Şekil 8. MODELS KLASÖRÜ EĞİTİM SONRASI OLUŞAN MODELLER .....	20
Şekil 9 391 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ DOSYA YAPISI .....	20
Şekil 10 TASARIM KLASÖRÜ İÇİNDEKİLER.....	20
Şekil 11 Coding.py __init__ fonksiyonu.....	21
Şekil 12 Coding.py VerileriYukle Fonksiyonu.....	22
Şekil 13 Coding.py verileri_yaz fonksiyonu .....	22
Şekil 14 Coding.py verileri_dok Foksiyonu .....	22
Şekil 15 Coding.py test_verilerini_dok fonksiyonu .....	22
Şekil 16 Coding.py cb_classificier_changed .....	23
Şekil 17 Coding.py train_test_clik Fonksiyonu .....	23
Şekil 18 Coding.py classification Fonksiyonu.....	24
Şekil 19 Coding.py Kullanılan Kütüphaneler.....	24
Şekil 20 DataBase.py Tüm Fonksiyonlar.....	25
Şekil 21 Arayüz Tasarımı Sınıflandırıcı Seç.....	26
Şekil 22 Arayüz Tasarımı Verileri Yükle.....	26
Şekil 234 Arayüz Tasarımı Başarım Oranı .....	27
Şekil 243 Arayüz Tasarımı Eğitim ve Test Verisi Belirle, Test Et .....	27
Şekil 25 Arayüz Tasarımı Eğitim Verileri .....	28
Şekil 26 Arayüz Tasarımı Eğitim Verileri Etiket Bilgileri .....	28
Şekil 27 Arayüz Tasarımı Test Verisi Etiket Bilgileri .....	29
Şekil 28 Arayüz Tasarımı Test Verisi .....	29
Şekil 29 Arayüz Tasarımı Karmaşıklık Matrisi Büyük .....	30
Şekil 30 Arayüz Tasarımı Karmaşıklık Matrisi .....	30
Şekil 31 121 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ DOSYA YAPISI .....	32
Şekil 32 121 Öznitelikli Veri Seti Models Klasörü .....	32
Şekil 33 121 Öznitelikli Veri Seti tasarım klasörü .....	32
Şekil 34 Coding.py __init__ fonksiyonu.....	33
Şekil 35 Coding.py VerileriYukle Fonksiyonu.....	34
Şekil 36 Coding.py verileri_yaz fonksiyonu .....	34
Şekil 37 Coding.py verileri_dok Foksiyonu .....	34
Şekil 38 Coding.py test_verilerini_dok fonksiyonu .....	34
Şekil 39 Coding.py cb_classificier_changed .....	35
Şekil 40 Coding.py train_test_clik Fonksiyonu .....	35
Şekil 41 Coding.py classification Fonksiyonu.....	36
Şekil 42 Coding.py Kullanılan Kütüphaneler.....	36
Şekil 43 Coding .py Verileri İşle Fonksiyonu .....	37

Şekil 44 DataBase.py Bütün Fonksiyonlar .....	38
Şekil 45 Arayüz Tasarımı Sınıflandırıcı Seç .....	39
Şekil 46 Arayüz Tasarımı Verileri Yükle.....	39
Şekil 47 Arayüz Tasarımı Eğitim ve Test Verisi Belirle, Test Et .....	40
Şekil 48 Arayüz Tasarımı Başarım Oranı .....	40
Şekil 49 Arayüz Tasarımı Eğitim Verileri Etiket Bilgileri .....	41
Şekil 50 Arayüz Tasarımı Eğitim Verileri .....	41
Şekil 51 Arayüz Tasarımı Test Verisi Etiket Bilgileri .....	42
Şekil 52 Arayüz Tasarımı Test Verisi .....	42
Şekil 53 Arayüz Tasarımı Karmaşıklık Matrisi Büyük .....	43
Şekil 54 Arayüz Tasarımı Karmaşıklık Matrisi .....	43
Şekil 55 Online Test Dosya Yapısı .....	45
Şekil 56 Online Test LeapCode.py Eklenen Kütüphaneler .....	45
Şekil 57 Online Test LeapCode.py convert_distortion_maps Fonksiyonu .....	46
Şekil 58 Online Test LeapCode.py undistort ve run fonksiyonu .....	47
Şekil 59 Online Test LeapCode.py LeapMotionListener Sınıfı .....	48
Şekil 60 Online Test LeapCode.py Leap Motion Listener on_frame fonksiyonu birinci bölüm.....	49
Şekil 61 Online Test LeapCode.py Leap Motion Listener on_frame fonksiyonu ikinci bölüm .....	50
Şekil 62 Online Test LeapCode.py Main Fonksiyonu .....	51
Şekil 63 Online Test Çalışma Ortamı.....	52
Şekil 64 Online Test Arayüz Tasarımı.....	52

## ÖZET

Toplumumuzda işaret dili bilmeyen insanların sayısı fazladır. Bu durum işitme engelli insanlar ile toplumun arasındaki iletişimini azaltmaktadır hatta işitme engelli insanların toplumdan soyutlanmasımasına neden olmaktadır. Bu çalışma ile işaret dili bilen insanlar ile işaret dili bilmeyen insanlar arasındaki iletişimini sağlamak ve bu engelli insanları topluma kazandırmak amaçlanmaktadır. Bu amaçla Türk Dili Kurumunun (TDK) hazırladığı Türk İşaret Dili sözlüğü referans alınarak işaret dili alfabesi Leap Motion cihazı ile tanıtılmıştır. Tasarlanan bu sistem ile hem sağ hem de sol el verileri üzerinde çalışılmıştır. İki bireyden ilk önce 10 harf için toplam 1000 örnek ve daha sonra 28 harf için toplam 2800 örnek alınarak verilerin %70'i eğitim verisi %30'u test verisi olarak ayrılmıştır. Bu veri kümelerine makine öğrenmesi algoritmalarından Rastgele Orman (Random Forest), Yapay Sinir Ağları (Neural Network), Naive Bayes, Karar Ağaçları (Decision Tree) ve Destek Vektör Makineleri (Support Vector Machine) algoritmaları uygulanıp test edilerek offline ve online başarım sonuçları elde edilmiştir. Başarım sonuçları karşılaştırılarak detaylı analiz yapılmıştır. Offline teste her iki örnek verisi için de en iyi sonucu %100 doğru başarım ile Rastgele Orman (Random Forest) algoritması vermektedir.

**Anahtar Kelime:** işaret dili sözlüğü, leap motion, makine öğrenmesi metotları, çevrimdışı test, gerçek zamanlı test

## 1.GİRİŞ

### 1.1 UYGULAMA KONUSU

Leap Motion Sensör ile Türk işaret dilini tanımlamak ve günlük hayatı kullanıma uygun olup olmadığına testi.

### 1.2 UYGULAMA AMACI

Bu çalışma ile işaret dili bilen insanlar ile işaret dili bilmeyen insanların arasındaki iletişimini sağlamak ve bu engelli insanları topluma kazandırmak amaçlanmaktadır.

### 1.3 UYGULAMA KULLANICI KİTLESİ

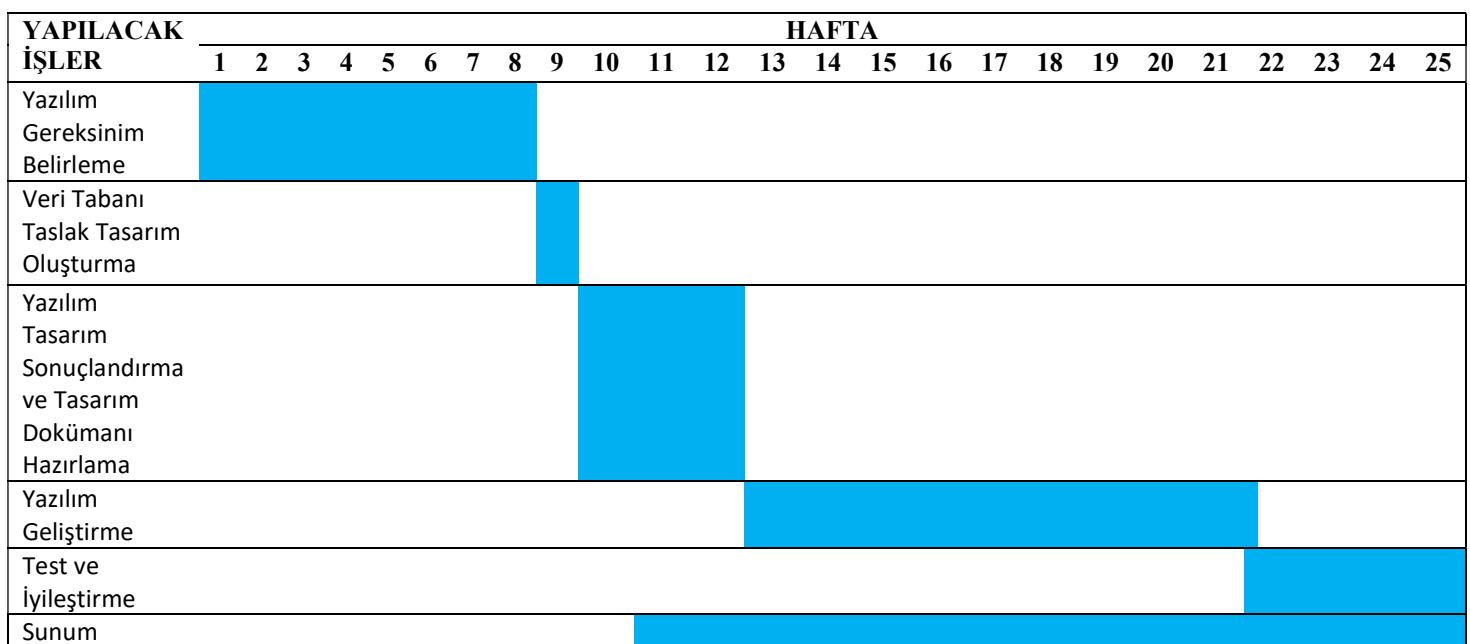
Toplumumuzda işaret dili bilmeyen insanlar ve işitme engelli bireyler.

## 1.4. UYGULAMA PLANI

### 1.4.1 GELİŞİM PLANI

Proje Aşamaları ve Önerilen Takvim Bilgisi		
Çalışma	Planlanan Başlangıç Tarihi	Planlanan Bitiş Tarihi
Analiz Çalışması	20.11.2017	14.01.2018
Tasarım Çalışması	15.01.2018	14.02.2018
Gerçekleştirim Çalışması	15.02.2018	14.04.2018
Test Çalışması	15.04.2018	14.05.2018
Danışman Teslimi	<b>15.05.2018</b>	<b>30.05.2018</b>
Projenin Sunumu (***)	<b>31.05.2018</b>	<b>14.06.2018</b>

### 1.4.2 GELİŞİM SÜRECİ AŞAMALARI



## 2. MAKİNE ÖĞRENMESİ VE LEAP MOTİON

### 2.1 LEAP MOTION

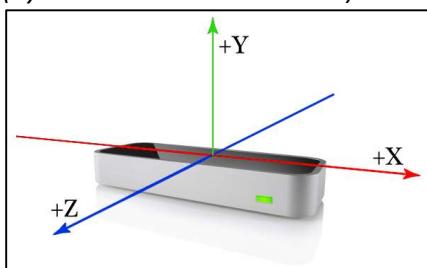
#### 2.1.1 LEAP MOTION NEDİR?

Leap Motion, el hareketlerini algılayabilen bir cihazdır. Bilgisayarınıza USB yoluyla bağlanan cihaz, el hareketlerini algılayarak hiçbir şeye dokunmadan bilgisayarınızı kullanmaya yardımcı olur.

#### 2.1.2 LEAP MOTION NASIL ÇALIŞIR?

Leap Motion, aslında boyut olarak küçük bir cihazdır. Boyutları 10,6x4x1,5 cm'dir. İçinde yerlesik bulunan iki IR kamera ve üç Infrared LED ile yaklaşık bir metre yüksekliğe kadar olan hareketleri algılayabilmektedir. Saniyede 200 kare çekim yapabilen kameralar, görüntülerini USB portu ile bilgisayara gönderir. Bilgisayara gelen iki boyutlu görüntüler, işlemcide işlenerek üç boyutlu görüntüye çevrilir. Bu sayede el hareketlerini sadece 0,7 mm hata payı ile algılamaktadır.

Alıntı: (*Python SDK Documentation.*)



Şekil 2 Leap Motion ve Temsili Algılandığı Eksenler



Şekil 1 Leap Motion (Üstten Altta Karşıdan Sağ ve Soldan görünüm)

#### 2.1.3 LEAP MOTION TEKNİK ÖZELLİKLER

Cihaz Tipi	Hareket Sensörü
Üretici	Leap Motion
Bağlantı Teknolojisi	Kablolu
Arayüz	USB
Doğruluk	± 0.00039 in
Performans	Saniyede 200 kare çekim
Renk	Siyah , Gümüş
Minimum Sistem Gereksinimleri	
Bağlantı Tipi	4 pin USB Type A
Garanti	1 yıl
Genişlik	3.1 in
Yükseklik	0.5 in
Derinlik	1.2 in
Ağırlık	1.6 oz

Alıntı: (*Leap Motion Controller Specs. (2013, Temmuz 22) , Camera Images.*)

## **2.2. MAKİNE ÖĞRENMESİ ALGORİTMALARI**

### **2.2.1 RASSAL ORMAN (RANDOM FOREST )**

Random Forest Algoritması Topluluk öğrenme yöntemi olan Random Forests (Rassal Orman) algoritması ,sınıflandırma işlemi esnasında birden fazla karar ağaçları üreterek sınıflandırma değerini yükseltmeyi hedefleyen bir algoritmadır.Bireysel olarak oluşturulan karar ağaçları bir araya gelerek karar ormanı oluşturur.Buradaki karar ağaçları bağlı olduğu veri setinden rastgele seçilmiş birer alt kümedir.

Random Forests 2001 yılında Leo Breiman tarafından geliştirilmiştir.RF algoritması yine Brieman'in 1996 yılında geliştirdiği Bagging yöntemi ve Kim Ho tarafından geliştirilen Random Subspace yöntemlerinin birleşimidir. Amit ve Geman tarafından 1997'de tanımlanan, her düğüm için en iyi ayrimın rastgele bir seçim üzerinden belirlendiği belirtilen bir çalışmadan da etkilenmiştir.

#### **Özellikleri**

- Mükemmel bir geçerlilik sunar. Pek çok veri seti için Adaboost ve Destek Vektör Makinelerinden (Support Vector Machines) daha kesin sonuçlara sahiptir.
- Oldukça kısa sürede sonuç verir. 100 değişkenli 100 ağaçlık bir karar ormanı, arka arkaya kurulan 3 tekil CART ile aynı sürede oluşturulur.
- Binlerce değişkene ve fazla sayıda sınıf etiketine sahip kategorik değişken içeren, kayıp verili veya dengesiz bir dağılım sergileyen veri setlerini kullanarak sonuçlar verir.
- Topluluğa ağaçlar eklendikçe, test setine ait hata tahmini için yanlılığı düşük sonuçlar vermeye başlar.
- Gürültülü verilerden arındırır.

#### **Uygulama Alanları**

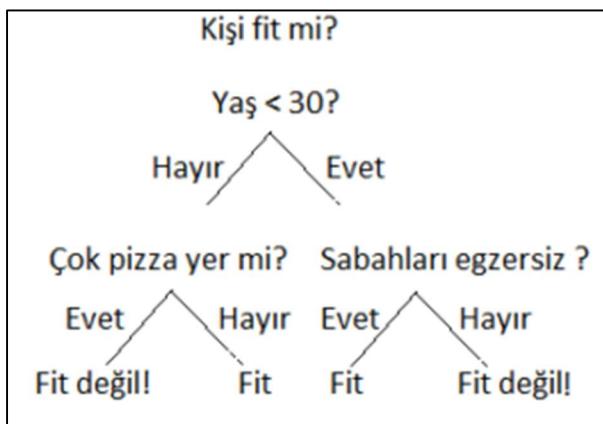
- Astronomi ,Biyomedikal
- Kontrol sistemleri , Finans Analizi
- Sağlık ,Moleküler Biyoloji
- Fizik ,Yazılım geliştirme

#### **Algoritma Olarak Random Forests**

- RF yöntemi bilinen makine öğrenme yöntemleri içerisinde eşsiz bir tahmin geçerliliği ve model yorumlanabilirliği sağlar. Rastgele örneklemeyi ve topluluk yöntemlerdeki tekniklerin iyileştirilmiş özelliklerini içermesi nedeniyle RF yöntemi daha iyi genellemeler sunar ve geçerli tahminlerde bulunur.
- RF yönteminin tahminlerinin kesin olması nedenleri sapması düşük sonuçlar vermesi ve ağaçlar arasındaki düşük korelasyondur. Düşük sapma miktarı, oldukça büyük ağaçların oluşturulması sonucu elde edilir. Mümkün olduğunda birbirinden farklı ağaçlar oluşturarak da düşük korelasyon yapısında bir topluluk elde edilir.
- Birbirinden farklı olarak kurulan sınıflama ve regresyon karar ağaçları bizi sonuca götürecek karar ormanı topluluğunu oluşturur. Karar ormanı oluşumu sırasında elde edilen sonuçlar bir araya getirilerek en son tahmin yapılır.
- RF yönteminde ağaçlar, seçilen bootstrap örneklemeleri ve her düğüm ayrimında rastgele seçilen m adet tahlimci ile oluşturulur. m adet tahlimcinin toplam tahlimci sayısından oldukça küçük olmasına dikkat edilir. Oluşturulan her bir karar ağaçının geniş haliyle bırakılır ve budanmaz.
- Sınıflandırma için ağaçlar; her yaprak düğümü sadece bir sınıfın üyelerini içerecek şekilde oluştururlar. Regresyon için ise; yaprak düğümde az sayıda birim kalana kadar ağaçlar bölünmeye devam ederler. Alıntı:( *Fidancı, A. S. (2017, Ocak 23.)*)

## 2.2.2 KARAR AĞAÇLARI (DECISION TREE)

Karar Ağaçları, belirli bir parametreye göre verilerin sürekli olarak bölündüğü bir Denetimli Makine Öğrenmesi türüdür. Ağaç, karar düğümleri ve yapraklar olmak üzere iki varlık tarafından açıklanabilir. Yapraklar kararlar veya nihai sonuçlardır. Ve karar düğümleri verilerin nereye bölündüğünü gösterir.



Şekil 3 Karar Ağacı Örneği

Bir karar ağacı örneği, yukarıdaki ikili ağaç kullanılarak açıklanabilir. Diyelim ki, bir insanın yaş, yeme alışkanlığı ve fiziksel aktivite gibi verilmiş bilgilerle fit olup olmadığını tahmin etmek isteyebilirsiniz. Buradaki karar düğümleri, ‘Yaş durumu’, ‘Sabahları egzersiz yapıyor mu?’, ‘Fazla pizza yer mi?’ ve yapraklar da, ‘fit’ ya da ‘fit değil’ şeklindedir. Bu durumda bu, binary(ikili) bir sınıflandırma problemiydi (evet hayır tip problemi).

### Karar Ağaçları ile Sınıflandırma

- Sınıflandırma problemleri için yaygın kullanılan yöntemdir.
- Sınıflandırma doğruluğu diğer öğrenme metodlarına göre çok etkindir.
- Öğrenmiş sınıflandırma modeli ağaç şeklinde gösterilir ve karar ağacı (decision tree) olarak adlandırılır.
- Karar ağaçları akış şemalarına benzeyen yapılardır. Her bir nitelik bir düğüm tarafından temsil edilir. Dallar ve yapraklar ağaç yapısının elemanlarıdır. En son yapı yaprak en üst yapı kök ve bunların arasında kalan yapılar dal olarak isimlendirilir.

Sınıflandırma Uygulamaları :Kredi başvurusu değerlendirme , Kredi kartı harcamasının sahtekarlık olup olmadığına karar verme, Hastalık teşhisini, Ses tanıma, Karakter tanıma ,Gazete haberlerini konularına göre ayırmaya , Kullanıcı davranışlarını belirlemeye.

Sınıflandırma yöntemleri olarak Karar ağaçları'nın yanı sıra Yapay Sinir Ağları (Artificial Neural Networks), Bayes Sınıflandırıcılar (Bayes Classifier), İlişki Tabanlı Sınıflandırıcılar (Association-Based Classifier), k-En Yakın Komşu Yöntemi (k- Nearest Neighbor Method), Destek Vektör Makineleri (Support Vector Machines) ve Genetik Algoritmalar (Genetic Algorithms) gibi yöntemler bulunmaktadır.

Karar ağaçlarında sınıflandırma yöntemleri 2 çeşittir. Bunlar “Entropiye Dayalı Algoritmalar” ve “Sınıflandırma ve Regresyon Ağaçları(CART)”dır. ID3 Algoritması ve C4.5 Algoritması “Entropiye dayalı algoritmalar” arasındayken Twoing Algoritması ve Gini Algoritması “Sınıflandırma ve regresyon ağaçları” sınıfındadır. Alıntı: (Akarsu, H. B. (2017, Kasım 5))

### 2.2.3 NAVİE BAYES

Navie Bayes sınıflandırma algoritması, adını Matematikçi Thomas Bayes'den alan bir sınıflandırma/ kategorilendirme algoritmasıdır. Navie Bayes sınıflandırması olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını yani kategorisini tespit etmeyi amaçlar.

Navie Bayes sınıflandırmasında sisteme belirli bir oranda öğretilmiş veri sunulur (Örn: 100 adet). Öğretim için sunulan verilerin mutlaka bir sınıfı/kategorisi bulunmalıdır. Öğretilmiş veriler üzerinde yapılan olasılık işlemleri ile, sisteme sunulan yeni test verileri, daha önce elde edilmiş olasılık değerlerine göre işletilir ve verilen test verisinin hangi kategoride olduğu tespit edilmeye çalışılır. Elbette öğretilmiş veri sayısı ne kadar çok ise, test verisinin gerçek kategorisini tespit etmek o kadar kesin olabilmektedir.

Navie Bayes sınıflandırma yönteminin birçok kullanım alanı bulunabilir fakat, burada neyin sınıflandırıldığından çok nasıl sınıflandırıldığı önemli. Yani öğretilecek veriler binary veya text veriler olabilir, burada veri tipinden ve ne olduğundan ziyade, bu veriler arasında nasıl bir oransal ilişki kurduğumuz önem kazanıyor.

#### Örnek:

Aşağıda yer alan tabloda 4 adet döküman, bunların içerikleri ve kategorileri yer almaktadır. Tabloda yer alan bilgiler Multinomial Naive Bayes sınıflandırma yöntemi ile sisteme öğretilecek bilgilerdir. Buradaki amaç ise, sisteme yeni sunulan bir dökümanın hangi kategoriye ait olduğunu bulmaktır.

Döküman numarası	Öğreten içerik/cümle
1	Chinese Beijing Chinese
2	Chinese Chinese Shanghai
3	Chinese Macao
4	Tokyo Japan Chinese

Multinomial Navie Bayes sınıflandırmasına göre ilk olarak bir kategorinin tüm satırlara göre oranı bulunmalıdır.

#### Örneğin:

$P(C) = 3/4 = 0.75$  (Öğretilecek verilerde Ç kategorisindeki satırların tüm satırlara oranı)  
 $P(J) = 1/4 = 0.25$  (Öğretilecek verilerde Japonya kategorisindeki satırların tüm satırlara oranı)  
Sonrasında ise, öğretilecek kelimelerin ait olduğu kategoriye göre koşullu olasılığı bulunur  
 $P(X|Y) = (Y \text{ kategorisindeki satırlarda } "X" \text{ ifadesinin tekrar sayısı} + 1) / (Y \text{ kategorisindeki satırlarda bulunan tüm kelimelerin sayısı} + \text{Öğreten veri sayısı})$

#### Koşullu olasılıklar (6 adet)

$$P(\text{Chinese} | C) = (5 + 1) / (8 + 6) = 0.428$$

$$P(\text{Tokyo} | C) = (0 + 1) / (8 + 6) = 0.071$$

$$P(\text{Japan} | C) = (0 + 1) / (8 + 6) = 0.071$$

$$P(\text{Chinese} | J) = (1 + 1) / (3 + 6) = 0.222$$

$$P(\text{Tokyo} | J) = (1 + 1) / (3 + 6) = 0.222$$

$$P(\text{Japan} | J) = (1 + 1) / (3 + 6) = 0.222$$

### **Yeni bir dökümanın test edilmesi**

Şimdi ise herhangi bir cümlenin öğretilen 6 bilgiye göre kategorisini bulmaya çalışalım.

**Not:** Eğer test için sunulan kelimelerden herhangi biri için olasılık değeri bulunmuyorsa, etkisini indirmek adına çarpma işleminde etkisiz eleman olan “1” oran olarak verilebilir. (Örn: İstanbul, Lüleburgaz..)

Test verisi	Chinese	Chinese	İstanbul	Chinese	Tokyo	Japan
-------------	---------	---------	----------	---------	-------	-------

$$P(\text{Ç} | \text{Test}) = P(\text{Ç}) * P(\text{Chinese} | \text{Ç}) * P(\text{Chinese} | \text{Ç}) * P(\text{İstanbul} | \text{Ç}) * P(\text{Chinese} | \text{Ç}) * P(\text{Tokyo} | \text{Ç}) * P(\text{Japan} | \text{Ç})$$

$$P(\text{Ç} | \text{Test}) = 0.75 * 0.428 * 0.428 * 1 * 0.428 * 0.071 * 0.071 = 0.0003$$

$$P(\text{Japonya} | \text{Test}) = P(\text{J}) * P(\text{Chinese} | \text{J}) * P(\text{Chinese} | \text{J}) * P(\text{İstanbul} | \text{J}) * P(\text{Chinese} | \text{J}) * P(\text{Tokyo} | \text{J}) * P(\text{Japan} | \text{J})$$

$$P(\text{Japonya} | \text{Test}) = 0.25 * 0.222 * 0.222 * 1 * 0.222 * 0.022 * 0.022 = 0.0001$$

#### **Sonuç:**

$P(\text{Ç} | \text{Test}) > P(\text{J} | \text{Test})$  olduğundan dolayı, Test verisinin Ç grubunda olma olasılığı olmama olasılığının yaklaşık 3 katıdır. Bu yüzden kategori olarak Ç denilebilir.

Alıntı: (*Usta, R. (2014, Mayıs 28).*

### **2.2.4 DESTEK VEKTÖR MAKİNELERİ (SUPPORT VECTOR MACHINE)**

1963 yılında Vladimir Vapnik ve Alexey Chervonenkis tarafından temelleri atılan “Destek Vektör Makineleri (DVM)” istatiksel öğrenme teorisine dayalı bir gözetimli öğrenme algoritmasıdır. Her ne kadar temelleri 60’lı yıllara dayansada 1995 yılında Vladir Vapnik, Berhard Boser ve Isabelle Guyon tarafından geliştirilmiştir (Akpinar, H., 2017).

Destek Vektör Makineleri, temel olarak iki sınıfa ait verileri birbirinden en uygun şekilde ayırmak için kullanılır. Bunun için karar sınırları yada diğer bir ifadeyle hiper düzlemler belirlenir.

DVM’ler günümüzde yüz tanıma sistemlerinden, ses analizine kadar birçok sınıflandırma probleminde kullanılmaktadırlar.

Avantajları:

- Yüksek boyutlu uzaylarda etkilidirler.
- Boyut sayısının, örneklem sayısından fazla olduğu durumlarda etkilidirler.
- Karar fonksiyonunda bir takım eğitim noktaları kullanılır (“support vectors”). Dolayısıyla bellek verimli bir şekilde kullanılmış olur.
- Çok yönlü: Karar fonksiyonu için çok farklı çekirdek fonksiyonları (“kernel functions”) kullanılabilir.

Alıntı: (*Makine Öğrenimi Bölüm-4 (Destek Vektör Makineleri). (2017, Kasım 30.)*)

## 2.2.5 YAPAY SINİR AĞLARI (NEURAL NETWORK)

Yapay zeka uygulamalarından biri olan yapay sinir ağları, insan beyninin çalışma yapısını taklit ederek mevcut verileri analiz edip, bu verilerden farklı öğrenme algoritmaları ile yeni bilgiler oluşturan bilgi işlem teknolojisidir.

Yapay sinir ağlarının tarihsel gelişimindeki kritik noktalarının bazlarına kısaca değinelim. Yapay sinir ağlarının temelleri 1940’ların başında araştırmalara başlayan Mc Cullogh ve Pitts’in 1943 yılında yayınladıkları bir makaleyle atılmış. Fakat yapay sinir ağı literatüründe XOR problemi olarak bilinen problemdeki başarısızlığı nedeniyle belli bir süre yapay sinir ağlarına olan ilgi azalmıştır.

1980’li yıllarda Hopfield tarafından yayınlanan çalışmalar ile yapay sinir ağını genelleştirileceği ve özellikle geleneksel bilgisayar programlama ile çözülmesi zor olan problemlere çözüm üretilebileceği gösterilmiştir.

1988 yılında Rumelhart, “Paralel Distributed Processing” adlı çalışmasında ileri beslemeli modellerde yeni öğrenme modeli olan hatanın geriye yayılma algoritmasını (Back Propagation *Algorithm*) geliştirerek bu konuda daha önce iddia edilen (XOR problemi gibi) aksaklılıkların aşılabilceğini göstermiştir.

Yapay sinir ağları biyolojik nöronlardan (sinir hücresi) esinlenerek, beynin çalışma sistemine yapay olarak benzetim çalışmaları sonucunda ortaya çıkmıştır. Genel anlamda insan beynindeki birçok biyolojik nöronun birbirine bağlanması gibi, yapay sinir ağlar; biyolojik nöronun girdi, işlem, çıktı karakteristğini taklit eden birçok basit, genellikle adaptif işlem birimlerinin (yapay nöron) değişik etki seviyelerinde, belirli bir bütün işlem yapısını gerçekleştirmek üzere birbirine bağlanması ile oluşturulmuştur

Yapay sinir ağlarının genel avantajları aşağıdaki gibi sıralanabilir (Haykin, 1999).

- Lineer olmayan yapıda olması
- Girdi ve çıktı eşleştirmeleri ile tasarlanabilmesi
- Adapte olabilmesi
- Hataya karşı toleranslı olması

Yapay sinir ağlarının gerçek hayatı yaygın uygulama alanlarına şu örnekler verilebilir.

- Kalite Kontrol.
- Finansal Öngörü.
- Ekonomik Öngörü.
- Kredi Derecelendirme.Konuşma ve Yapı Tanımlama.
- Anormal durum öngörü (siber güvenlik, log analizi), İflas Tahmini.

Alıntı: (*Filiz, F. (2017, Temmuz 23).*)

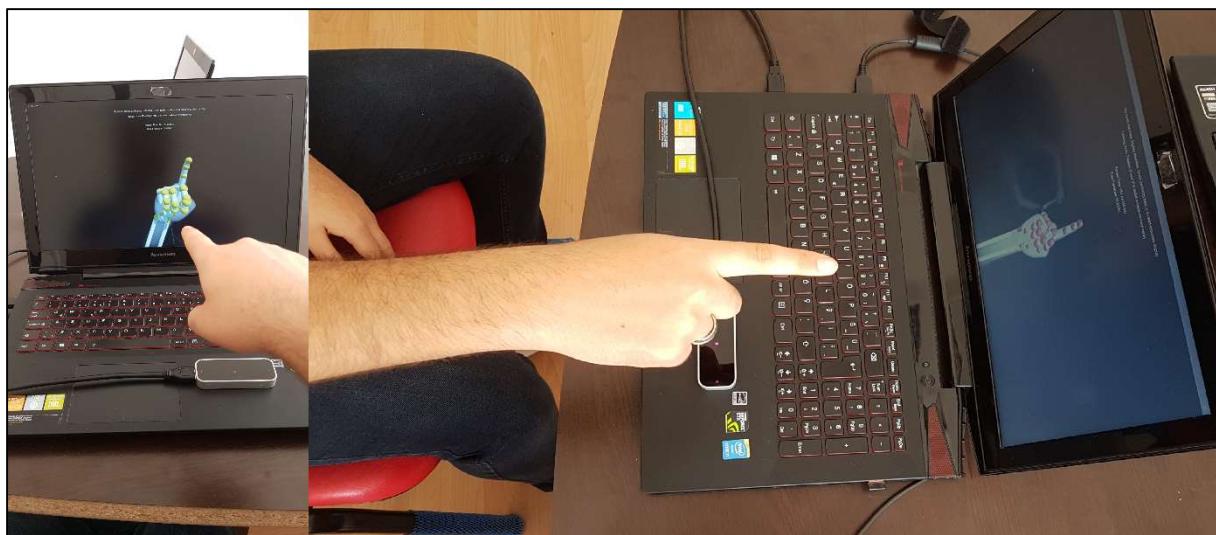
### 3. YÖNTEM

#### 3.1 VERİ SETİNİN OLUŞTURULMASI

Bu çalışmada sol ve sağ el üzerinde çalışılmıştır. Bu şekilde iki elinde kullanıldığı harflerin öğretilmesi sağlanmıştır. İki farklı örnek veri seti oluşturulmuştur. Örnek veri setlerini oluşturmak için 2 bireyden veriler alınmıştır. İlk önce 10 harf için toplam 1000 adet (her harf için 100 örnek) veri alınmıştır. Daha sonra 28 harf için toplam 2800 adet (her harf için 100 örnek) veri alınmıştır.

Elde edilen veri kümelerinden iki farklı şekilde öznitelik çıkarımı yapılmaktadır:

1. Oluşturulan veri setinde kol pozisyonu, bilek pozisyonu, dirsek pozisyonu, avuç içi pozisyonu ve parmak kemiklerinin (metacarpal, proximal, intermediate, distal) pozisyonları öznitelik olarak kullanılmıştır. Etiket bilgisiyle birlikte toplam 391 öznitelik elde edilmiştir.
2. Var olan veri setinden avuç içi pozisyonundan parmak kemiklerinin bitiş noktasındaki değerlerin farkına göre yeni öznitelikler oluşturulmuştur. Toplamda etiket bilgisiyle birlikte 121 özniteligi sahip veri seti oluşturulmuştur.



Şekil 4 VERİ KAYIT ÇALIŞMA ORTAMI

#### VERİLERİN KAYIT EDİLMESİ DOSYA YAPISI VE KODLAR

BITİRME PROJESİ > Proje > SaveData			
Ad	Değiştirme tarihi	Tür	Boyut
DataBase.py	28.05.2018 09:37	PY Dosyası	6 KB
DataBase.pyc	28.05.2018 09:39	PYC Dosyası	9 KB
Leap.dll	14.11.2017 02:48	Uygulama uzantısı	3.777 KB
Leap.lib	14.11.2017 02:48	VisualStudio.lib.32...	126 KB
Leap.py	14.11.2017 02:48	PY Dosyası	84 KB
Leap.pyc	4.12.2017 11:44	PYC Dosyası	85 KB
LeapCode.py	28.05.2018 09:41	PY Dosyası	4 KB
LeapPython.pyd	14.11.2017 02:48	PYD Dosyası	431 KB

Şekil 5 VERİLERİN KAYIT EDİLMESİ DOSYAYAPISI

**DataBase.py** : Veritabanı bağlantılarının gerçekleştiği , veri ekleme ve verileri seçme işlemlerinin yapıldığı python kodları.

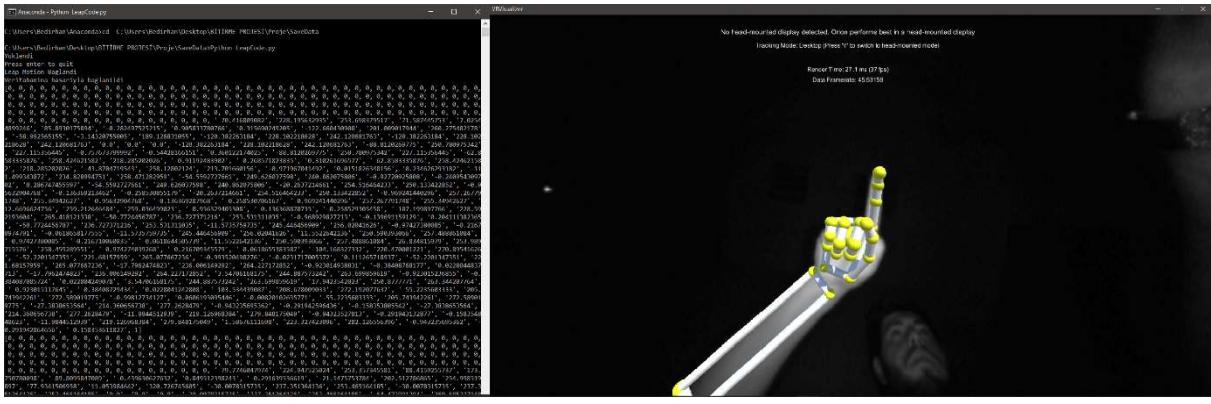
**Leap.dll** : Leap Motion nun projede çalışabilmesi için gerekli sistem dosyası

## **Leap.lib: Leap Motion Kütüphanesi**

**Leap.py** : Leap Motion geliştiricileri tarafından eklenen python kodları

**LeapCode.py** : Leap Motion dan verileri alıp , işleyip , veritabanına kayıt etmek için gerekli python kodları.

*Şekil 6 LeapCode.py Çalışması ve Verilerin Kayıt Edilmesi*



**Sekil 7 Verilerin Kayit Edilmesi**

### 3.1.2 KODLAR

#### Database.py:

```
import mysql.connector
from mysql.connector import errorcode

def connect(username,password,host,databasename):
    config = {
        'user': username,
        'password': password,
        'host': host,
        'database': databasename,
        'raise_on_warnings': True }
    try:
        cnx=mysql.connector.connect(**config)
        print("Veritabanina basariyla baglanildi")
    except mysql.connector.Error as err:
        if err.errno==errorcode.ER_ACCESS_DENIED_ERROR:
            print("Kullanici adi veya parola hatali")
        elif err.errno ==errorcode.ER_BAD_DB_ERROR:
            print ("Database adi hatali veya yanlis yazilmis")
        else:
            print(err)
    return cnx

def text_seperate(tablerow):
    t_row=""
    for i,row in enumerate (tablerow):
        if i!=len(tablerow)-1:
            t_row+=str(row)+", "
        else:
            t_row+=str(row)
    return t_row

def selectTable(cursor,tablename):
    crs=cursor
    query=("SELECT * FROM "+str(tablename))
    data=crs.execute(query)
    return data
```

```

def addTable(cursor,tabledata):
    tablerow="pp_x_0","pp_y_0","pp_z_0","pitch_0","roll_0","yaw_0","arm_d_x_0","arm_d_y_0"
    "elb_p_x_0","elb_p_y_0","elb_p_z_0","m_s_x_00","m_s_y_00","m_s_z_00","m_e_x_00",
    "p_s_x_00","p_s_y_00","p_s_z_00","p_e_x_00","p_e_y_00","p_e_z_00","p_d_x_00","p_d_y_00"
    "i_s_y_00","i_s_z_00","i_e_x_00","i_e_y_00","i_e_z_00","i_d_x_00","i_d_y_00","i_d_z_00"
    "d_s_z_00","d_e_x_00","d_e_y_00","d_e_z_00","d_d_x_00","d_d_y_00","d_d_z_00","m_s_x_01"
    "m_e_x_01","m_e_y_01","m_e_z_01","m_d_x_01","m_d_y_01","m_d_z_01","p_s_x_01","p_s_y_01"
    "p_e_y_01","p_e_z_01","p_d_x_01","p_d_y_01","p_d_z_01","i_s_x_01","i_s_y_01","i_s_z_01"
    "i_e_z_01","i_d_x_01","i_d_y_01","i_d_z_01","d_s_x_01","d_s_y_01","d_s_z_01","d_e_x_01"
    "d_d_x_01","d_d_y_01","d_d_z_01","m_s_x_02","m_s_y_02","m_s_z_02","m_e_x_02","m_e_y_02"
    "m_d_y_02","m_d_z_02","p_s_x_02","p_s_y_02","p_s_z_02","p_e_x_02","p_e_y_02","p_e_z_02"
    "p_d_z_02","i_s_x_02","i_s_y_02","i_s_z_02","i_e_x_02","i_e_y_02","i_e_z_02","i_d_x_02"
    "d_s_x_02","d_s_y_02","d_s_z_02","d_e_x_02","d_e_y_02","d_e_z_02","d_d_x_02","d_d_y_02"
    "m_s_y_03","m_s_z_03","m_e_x_03","m_e_y_03","m_e_z_03","m_d_x_03","m_d_y_03","m_d_z_03"
    "p_s_z_03","p_e_x_03","p_e_y_03","p_e_z_03","p_d_x_03","p_d_y_03","p_d_z_03","i_s_x_03"
    "i_e_x_03","i_e_y_03","i_e_z_03","i_d_x_03","i_d_y_03","i_d_z_03","d_s_x_03","d_s_y_03"
    "d_e_y_03","d_e_z_03","d_d_x_03","d_d_y_03","d_d_z_03","m_s_x_04","m_s_y_04","m_s_z_04"
    "m_e_z_04","m_d_x_04","m_d_y_04","m_d_z_04","p_s_x_04","p_s_y_04","p_s_z_04","p_e_x_04"
    "p_d_x_04","p_d_y_04","p_d_z_04","i_s_x_04","i_s_y_04","i_s_z_04","i_e_x_04","i_e_y_04"
    "i_d_y_04","i_d_z_04","d_s_x_04","d_s_y_04","d_s_z_04","d_e_x_04","d_e_y_04",
    d_d_z_04","pp_x_1","pp_y_1","pp_z_1","pitch_1","roll_1","yaw_1","arm_d_x_1","arm_d_y_1"
    "elb_p_x_1","elb_p_y_1","elb_p_z_1","m_s_x_10","m_s_y_10","m_s_z_10","m_e_x_10",
    "p_s_x_10","p_s_y_10","p_s_z_10","p_e_x_10","p_e_y_10","p_e_z_10","p_d_x_10","p_d_y_10"
    "i_s_y_10","i_s_z_10","i_e_x_10","i_e_y_10","i_e_z_10","i_d_x_10","i_d_y_10","i_d_z_10"
    "d_s_z_10","d_e_x_10","d_e_y_10","d_e_z_10","d_d_x_10","d_d_y_10","d_d_z_10","m_s_x_11"
    "m_e_x_11","m_e_y_11","m_e_z_11","m_d_x_11","m_d_y_11","m_d_z_11","p_s_x_11","p_s_y_11"
    "p_e_y_11","p_e_z_11","p_d_x_11","p_d_y_11","p_d_z_11","i_s_x_11","i_s_y_11","i_s_z_11"
    "i_e_z_11","i_d_x_11","i_d_y_11","i_d_z_11","d_s_x_11","d_s_y_11","d_s_z_11","d_e_x_11"
    "d_d_x_11","d_d_y_11","d_d_z_11","m_s_x_12","m_s_y_12","m_s_z_12","m_e_x_12","m_e_y_12"
    "m_d_y_12","m_d_z_12","p_s_x_12","p_s_y_12","p_s_z_12","p_e_x_12","p_e_y_12","p_e_z_12"
    "p_d_z_12","i_s_x_12","i_s_y_12","i_s_z_12","i_e_x_12","i_e_y_12","i_e_z_12","i_d_x_12"
    "d_s_x_12","d_s_y_12","d_s_z_12","d_e_x_12","d_e_y_12","d_e_z_12","d_d_x_12","d_d_y_12"
    "m_s_y_13","m_s_z_13","m_e_x_13","m_e_y_13","m_e_z_13","m_d_x_13","m_d_y_13","m_d_z_13"
    "p_s_z_13","p_e_x_13","p_e_y_13","p_e_z_13","p_d_x_13","p_d_y_13","p_d_z_13","i_s_x_13"
    "i_e_x_13","i_e_y_13","i_e_z_13","i_d_x_13","i_d_y_13","i_d_z_13","d_s_x_13","d_s_y_13"
    "d_e_y_13","d_e_z_13","d_d_x_13","d_d_y_13","d_d_z_13","m_s_x_14","m_s_y_14","m_s_z_14"
    "m_e_z_14","m_d_x_14","m_d_y_14","m_d_z_14","p_s_x_14","p_s_y_14","p_s_z_14","p_e_x_14"
    "p_d_x_14","p_d_y_14","p_d_z_14","i_s_x_14","i_s_y_14","i_s_z_14","i_e_x_14","i_e_y_14"
    "i_d_y_14","i_d_z_14","d_s_x_14","d_s_y_14","d_s_z_14","d_e_x_14","d_e_y_14","d_e_z_14"
    "d_d_z_14","etiket"
        tablename="leap_data_test"
        crs=cursor
        t_row=text_seperate(tablerow)
        data=text_seperate(tabledata)
        query=( "INSERT INTO "+str(tablename)+" ("+t_row+") VALUES ("+data+")")
        crs.execute(query)

```

## LeapCode.py

```

import Leap,sys/thread,time
from Leap import CircleGesture,KeyTapGesture,ScreenTapGesture,SwipeGesture
import time
import DataBase
from DataBase import connect
from DataBase import addTable

```

```

class LeapMotionListener(Leap.Listener):
    finger_names=['Thumbs','Index','Middle','Ring','Pinky']
    bone_names=['Metacarpal','Proximal','Intermediate','Distal']
    state_names=['STATE_INVALID','STATE_START','STATE_UPDATE','STATE_END']

    def on_init(self,controller):
        print "Yuklendi"

    def on_connect(self,controller):
        self.i=0
        print "Leap Motion Baglandi"
        self.cnx=connect("root","","127.0.0.1","leap_data_new")
        controller.enable_gesture(Leap.Gesture.TYPE_CIRCLE);
        controller.enable_gesture(Leap.Gesture.TYPE_KEY_TAP);
        controller.enable_gesture(Leap.Gesture.TYPE_SCREEN_TAP);
        controller.enable_gesture(Leap.Gesture.TYPE_SWIPE);

    def on_disconnect(self,controller):
        print "Leap Motion baglantisi kesildi"

    def on_frame(self,controller):
        self.i+=1
        frame=controller.frame()
        cursor=self.cnx.cursor()
        data=[]
        for i in range (0,391):
            data.append(0)
        etiket=1
        now_data=[]
        for i,hand in enumerate(frame.hands):
            handTypedata=0 if hand.is_left else 1
            normal= hand.palm_normal
            direction=hand.direction
            arm=hand.arm
            now_data=str(hand.palm_position.x),str(hand.palm_position.y),\
            str(hand.palm_position.z),str(direction.pitch* Leap.RAD_TO_DEG),\
            str(normal.roll*Leap.RAD_TO_DEG),str(direction.yaw*Leap.RAD_TO_DEG),\
            str(arm.direction.x),str(arm.direction.y),str(arm.direction.z),\
            str(arm.wrist_position.x),\
            str(arm.wrist_position.y),str(arm.wrist_position.z),\
            str(arm.elbow_position.x),str(arm.elbow_position.y),\
            str(arm.elbow_position.z),
            for finger in hand.fingers:
                for b in range(0,4):
                    bone=finger.bone(b)
                    now_data+=str(bone.prev_joint.x),str(bone.prev_joint.y),\
                    str(bone.prev_joint.z),\
                    str(bone.next_joint.x) , str(bone.next_joint.y),\
                    str(bone.next_joint.z) , \
                    str(bone.direction.x) , str(bone.direction.y) , \
                    str(bone.direction.z),

```

```

if(handTypedata==0 and i==0):#"1.el ve sol" b
    for x in range(0,195):
        data[x]=now_data[x]
    for y in range (195,390):
        data[y]=0
elif (handTypedata==1 and i==0): #1.el ve sag a
    for x in range(0,195):
        data[x]=0
    for y in range (195,390):
        data[y]=now_data[y-195]
elif (handTypedata==0 and i==1): # 2.el ve sol ccc
    for x in range(0,195):
        data[x]=now_data[x]
elif (handTypedata==1 and i==1): #2.el ve sag ccc
    for y in range (195,390):
        data[y]=now_data[y-195]

data[390]=etiket
if data[389]==0 and data[0]==0: #el verisi yoksa kayıt etme
    pass
else:
    if self.i<11:
        print (str(data))
        addTable(cursor,data)

def main():
    listener=LeapMotionListener()
    controller=Leap.Controller()
    controller.add_listener(listener)

    print "Press enter to quit"

    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        controller.remove_listener(listener)
if __name__ == "__main__":
    main()

```

## 3.2-AĞIN EĞİTİLMESİ VE OFFLINE TEST

### 3.2.1 391 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ

Veritabanından çekilen veriler kullanıcının istediği oran doğrultusunda eğitim ve test verisi olarak ayrıldıktan sonra kullanıcının uygulamak istediği makine öğrenmesi algoritmalarından Rastgele Orman (*Random Forest*), Yapay Sinir Ağları (*Neural Network*), *Naive Bayes*, Karar Ağacıları (*Decision Tree*) veya Destek Vektör Makineleri (*Support Vector Machine*) seçeneklerinden birini seçerek modelleme yapılmıştır

#### 3.2.1.1 DOSYA YAPISI

models
tasarım
Coding.py
Coding.pyc
confusion.py
confusion.pyc
DataBase.py
DataBase.pyc
DecisionTreeModel Confusion Matrix.xlsx
GaussianNavieBayesModel Confusion Ma...
Main_Dialog.py
NeuralNetworkModel Confusion Matrix.x
Random Forest Confusion Matrix.xlsx
SupportVectorMachineModel Confusion ...
tasarim.py
tasarim.pyc
x_test.py
x_test.pyc
x_train.py
x_train.pyc
y_test_table.py
y_test_table.pyc
y_train_table.py
y_train_table.pyc

Şekil 9 391 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ DOSYA YAPISI

DecisionTreeModel.pkl
GaussianNavieBayesModel.pkl
NeuralNetworkModel.pkl
RandomForestModel.pkl
SupportVectorMachineModel.pkl

Şekil 8. MODELS KLASÖRÜ EĞİTİM SONRASI OLUŞAN MODELLER

confusion.ui
Tasarim.ui
x_test.ui
x_train.ui
y_test.ui
y_train.ui

Şekil 10 TASARIM KLASÖRÜ İÇİNDEKİLER

**Coding.py :** Model oluşturabilmek ve oluşturulan modelin başarım oranını test edebilmek için yazılan bütün kodlar burada yer alır.

**DataBase.py :** Veritabanı bağlantılarının yapıldığı ve veri getirme işlemlerinin uygulandığı kodlar burada yer alır.

**MainDialog.py :** Bütün python kodlarının birleştirilip çalıştırıldığı ana pencere kodları burada yer alır.

**confusion.py , x\_test.py,x\_train.py,y\_test\_table.py,**

**y\_train\_table.py :** Veri ayırmaya işleminden sonra ayrılan veri değerlerini görebilmek için oluşturulmuş tablolardır.

**.ui uzantılı Dosyalar:** Tasarım dosyalarıdır form görüntüleri bu dosyalar sayesinde oluşturulur.

**.pyc uzantılı Dosyalar:** Derlenmiş Python dosyaları. Modül olarak kullanılırlar. Bir Python programı içinden çalıştırılabilirler ama açılmazlar. Yani okunaksız dosyalar.

### 3.2.1.2 KODLAR VE AÇIKLAMALAR

Coding.py `__init__` fonksiyonu: Bu fonksiyon form ilk yükleniği anda yapılacak işlemleri belirler.

```
def __init__(self):
    QtGui.QMainWindow.__init__(self)
    self.setupUi(self)
    self.statusBar().showMessage(unicode("Hazır\n"))
    self.btn_dosyaYukle3.clicked.connect(self.VerileriYukle)
    self.hs_test_value.valueChanged.connect(self.hs_valueChanged)
    self.cb_classificier.currentIndexChanged.connect(self.cb_classificier_changed)
    self.btn_split.clicked.connect(self.train_and_test_click)
    self.liste=[]
    self.cnx=connect("root","","127.0.0.1","leap_data_new") #Veritabanı bağlantı ayarları
    self.secim=0
    self.modelName='Random Forest'
    self.confusionMatrix=confusion.Ui_Dialog() #confusionMatrix örneği alınıyor
    self.confusionWidget = QtGui.QWidget()
    self.confusionMatrix.setupUi(self.confusionWidget) # veriler widgeta set ediliyor.

    #Eğitim verilerini göstermek için form örneği oluşturuluyor.
    self.formXtrain=x_train.Ui_Dialog()
    self.formXtrainWidget=QtGui.QWidget()
    self.formXtrain.setupUi(self.formXtrainWidget)
    #Eğitim verilerinin etiketleri gösterilmek için form örneği oluşturuluyor
    self.formYtrain=y_train_table.Ui_Dialog()
    self.formYtrainWidget=QtGui.QWidget()
    self.formYtrain.setupUi(self.formYtrainWidget)

    #Test verilerini göstermek için form örneği oluşturuluyor
    self.formXtest=x_test.Ui_Dialog()
    self.formXtestWidget=QtGui.QWidget()
    self.formXtest.setupUi(self.formXtestWidget)

    #Test Verilerinin etiketleri gösterilmek için form örneği oluşturuluyor
    self.formYtest=y_test_table.Ui_Dialog()
    self.formYtestWidget=QtGui.QWidget()
    self.formYtest.setupUi(self.formYtestWidget)
    #Butonlar aktif ediliyor
    self.btn_confusion.clicked.connect(self.confusion_show)
    self.pb_x_train.clicked.connect(self.x_train_show)
    self.pb_x_test.clicked.connect(self.x_test_show)
    self.pb_y_test.clicked.connect(self.y_test_show)
    self.pb_y_train.clicked.connect(self.y_train_show)
```

Şekil 11 Coding.py `__init__` fonksiyonu

Coding.py VerileriYukle:

```
def VerileriYukle(self): #veritabanına bağlanıp tablodan veri çekme işlemeleri
    cursor=self.cnx.cursor(buffered=True)
    self.alldata=[]
    select=selectTable(cursor)

    for a in select:
        self.alldata.append(a)
    self.alldata=np.array(self.alldata)
    self.verileri_dok(self.alldata,self.table_all_data)
    veri_sayisi=self.alldata.shape[0]
    oznitelik_sayisi=self.alldata.shape[1]
    self.statusBar().showMessage(unicode("Veriler Yükleniyor...Toplam "+str(veri_sayisi)+
    " veri ve "+str(oznitelik_sayisi)+" öznitelik bulunuyor.",'utf-8'))
```

Şekil 12 Coding.py VerileriYukle Fonksiyonu

Coding.py verileri\_yaz: Confusion matrix sonuçlarını excel tablosuna dökmek için yazılmıştır.

```
def verileri_yaz(self,veri,ismi):
    harfler=['A','B','C',unicode('Ç','utf-8'),'D','E','F','G','H','I',
             unicode('İ','utf-8'),'J','K','L','M','N','O',unicode('Ö','utf-8'),
             'P','R','S',unicode('Ş','utf-8'),'T','U',unicode('Ü','utf-8'),'V','Y','Z']
    excel=Workbook()
    excelws=excel['Sheet']

    for i,harf in enumerate(harfler):
        excelws.cell(row=0,column=i+1).value=harf
        excelws.cell(row=i+1,column=0).value=harf

    for rowNumber,row in enumerate(veri): #Veri satır satır okunuyor
        for columnNumber in range(0,len(veri[0])): # Her bir satırdaki veri sütun sütun okunuyor
            excelws.cell(row=rowNumber+1, column=columnNumber+1).value = str(row[columnNumber]) #tablonun rowNumber'inci satır
            #columnNumber'inci sütünuna veri ekleniyor.
    excel.save(u"./"+ismi+".xlsx")
```

Şekil 13 Coding.py verileri\_yaz fonksiyonu

Coding.py verileri\_dok: Bu fonksiyon gelen tabloya , gelen verileri dökmek için kullanılır.

```
def verileri_dok(self,X,tablo):
    num_rows=len(X) #gönderilen veriden satır sayısı alınıyor
    tablo.clear() # tablo temizleniyor
    tablo.setRowCount(len(X[0])) # gönderilen veriden sütun sayısı alınıyor ve tabloya set ediliyor.
    tablo.setColumnCount(num_rows) #alınan satır sayısı tabloya set ediliyor.
    for rowNumber, row in enumerate(X): #Veri satır satır okunuyor
        for columnNumber in range(0,len(X[0])): # Her bir satırdaki veri sütun sütun okunuyor
            tablo.setItem(rowNumber,columnNumber,QtGui.QTableWidgetItem(str(row[columnNumber]))) #tablonun rowNumber'inci
            #satır columnNumber'inci sütünuna veri ekleniyor.
```

Şekil 14 Coding.py verileri\_dok Foksiyonu

Coding.py test\_verilerini\_dok: verileri dök fonksiyonundan farkı sütun sayısının sabit olması yani bu fonksiyon etiket değerlerini göstermek için kullanılıyor.

```
def test_verilerini_dok(self,X,tablo):
    num_rows=len(X) #gönderilen veriden satır sayısı alınıyor
    tablo.clear() # tablo temizleniyor
    tablo.setRowCount(1) # gönderilen veriden sütun sayısı alınıyor ve tabloya set ediliyor.
    tablo.setColumnCount(num_rows) #alınan satır sayısı tabloya set ediliyor.
    for rowNumber, row in enumerate(X): #Veri satır satır okunuyor
        tablo.setItem(rowNumber,0,QtGui.QTableWidgetItem(str(row))) #tablonun rowNumber'inci satır columnNumber'inci
        #sütünuna veri ekleniyor
```

Şekil 15 Coding.py test\_verilerini\_dok fonksiyonu

Coding.py cb\_classificier\_changed: Kullanıcının sınıflandırıcı türünü seçebilmesi için forma eklenmiş olan combobox nesnesinin changeIndex Metodu

```
def cb_classificier_changed(self,value): # sınıflandırıcı türünü seçmek için kullanılan combobox 'ın changeIndex Metodu.
    if (value==0):
        self.secim=0
        self.modelName='RandomForestModel'
    elif(value==1):
        self.secim=1
        self.modelName='NeuralNetworkModel'
    elif(value==2):
        self.secim=2
        self.modelName='GaussianNaiveBayesModel'
    elif(value==3):
        self.secim=3
        self.modelName='SupportVectorMachineModel'
    elif(value==4):
        self.secim=4
        self.modelName='DecisionTreeModel'
```

Şekil 16 Coding.py cb\_classificier\_changed

Coding.py hs\_ValueChanged: Eğitim ve test verileri ayrılırken test verisinin yüzdesini belirlemek amacıyla kullanılan horizontal slider in ValueChanged metodu.

```
def hs_ValueChanged(self): # Eğitim ve test verisini ayırmak test verisinin yüzdesini
#belirlemek amacıyla kullanılan horizontal slider valueChanged olayı
    val=% " +str(self.hs_test_value.value()) #Value nun görsel olarak kullanıcılaraya aktarılması
    self.lbl_hs_value.setText(val)
```

Coding.py train\_test\_click: Veritabanından çekilen veriler eğitim ve test verisi olmak üzere ayrılıyor ve sınıflandırma fonksiyonuna gönderilip model oluşturuluyor.

```
def train_and_test_click(self):
    X=self.alldata[:, :390]
    y=self.alldata[:, 390]
    value=float(self.hs_test_value.value())
    value=value/100
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=value)
    self.classification(X_train,y_train,X_test,y_test)
```

Şekil 17 Coding.py train\_test\_clik Fonksiyonu

Coding.py classification: Burada yapılan seçime göre sınıflandırıcı oluşturuluyor. Oluşturulan sınıflandırıcı eğitim verileri ile eğitiliyor. Eğitilen model kayıt ediliyor daha sonra kayıt edilen model ile test verileri karşılaştırılıyor. Oluşan sonuçlar ile beklenen sonuçlar karşılaştırılıyor ve karmaşıklık matrisi ve başarı oranı oluşturuluyor.

```

def classification(self,X_train,y_train,X_test,y_test):
    if(self.secim==0): #randomForest
        clf=RandomForestClassifier(max_depth=None,random_state=42)
    elif(self.secim==1): #neuralNetwork
        clf=MLPClassifier(activation='tanh', solver='adam', alpha=1e-5,hidden_layer_sizes=(5,), random_state=1)
    elif(self.secim==2):#decisionTree
        clf=DecisionTreeClassifier()
    elif(self.secim==3):#supportVectorMachine
        clf=SVC()
    elif(self.secim==4):#gauisanNavieBayes
        clf=GaussianNB()

    clf.fit(X_train,y_train) #sınıflandırıcı veriler ile eğitiliyor
    mname='./models/'+str(self.modelName)+'.pkl' #eğitilen modele isim veriliyor
    joblib.dump(clf,mname ) #eğitilen model kayıt ediliyor
    result=clf.predict(X_test) # Model üzerinde test verileri test ediliyor
    self.verileri_dok(confusion_matrix(result,y_test,labels=[1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29]))
    self.confusionMatrix.table_confusion_matrix #doğru sonuçlar ile elde edilen sonuçlar karşılaştırılıyor
    #ve karmaşıklık matrisi oluşturuluyor.
    self.confusionMatrix.table_confusion_matrix.setHorizontalHeaderLabels(['A','B','C',
        unicode('Ç','utf-8'),unicode('İ','utf-8'),unicode('Ö','utf-8'),'D','E','F','G','H','I',
        unicode('Ü','utf-8'),'J','K','L','M','N','O',unicode('Ş','utf-8'),'P','R','S',unicode('Ş','utf-8'),'T','U',
        unicode('Ü','utf-8'),'V','Y','Z'])
    self.confusionMatrix.table_confusion_matrix.setVerticalHeaderLabels(['A','B','C',unicode('Ç','utf-8'),'D','E','F','G','H','I',
        unicode('Ü','utf-8'),'J','K','L','M','N','O',unicode('Ş','utf-8'),'P','R','S',unicode('Ş','utf-8'),'T','U',
        unicode('Ü','utf-8'),'V','Y','Z'])
    lbl_text=(("Başarı oranı: %"+ str ((round(accuracy_score(y_test,result),2)*100))) #doğru sonuçlar ile elde edilen
    #sonuçlar karşılaştırılıyor ve başarı oranı ölçülüyor. Kullanıcıya gösteriliyor.
    self.lbl_basari.setText(lbl_text) #kullanıcıya gösterilmek için label set ediliyor.

    self.verileri_dok(X_train,self.formXtrain.table_x_train) #Eğitim Verileri gösterilmek için set ediliyor
    self.verileri_dok(X_test,self.formXtest.table_x_test) # Test Verileri gösterilmek için set ediliyor
    self.test_verileri_dok(y_train,self.formYtrain.table_y_train) #Eğitim verilerinin etikleri gösteriliyor
    self.test_verileri_dok(y_test,self.formYtest.table_y_test) #Test verileri etiketleri gösteriliyor

    self.verileri_yaz(confusion_matrix(result,y_test,labels=[1,2,3,4,5,6,7,8,10,11,12,13,14,
        15,16,17,18,19,20,21,22,23,24,25,26,27,28,29]),
        self.modelName+" Confusion Matrix")
    #verileri yaz ile modele ait karmaşıklık matrisi excel tablosuna bastırılıyor

```

Şekil 18 Coding.py classification Fonksiyonu

Coding.py eklenen kütüphaneler:

```

@author: Bedirhan
"""

from PyQt4 import QtGui
from PyQt4 import QtCore
import numpy as np
from tasarim import Ui_Dialog
import confusion
import x_train
import x_test
import y_test_table
import y_train_table
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from DataBase import connect
from DataBase import selectTable
from sklearn.neural_network import MLPClassifier
from sklearn.externals import joblib
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from openpyxl import Workbook

```

Şekil 19 Coding.py Kullanılan Kütüphaneler

Database.py Fonksiyonlar:

Connect: Veritabanı bağlantılarının yapıldığı fonksiyon

Text\_seperate: Verileri eklemek için gönderilen string deki sütun isimlerini ayırmak için kullanılıyor

addTable: Verileri Tabloya ekler SelectTable: Eklenmiş verileri tablodan getirir.

```
@author: Bedirhan
"""

import mysql.connector
from mysql.connector import errorcode

def connect(username,password,host,databasename):
    config = {
        'user': username,
        'password': password,
        'host': host,
        'database': databasename,
        'raise_on_warnings': True }

    try:
        cnx=mysql.connector.connect(**config)
        print("Veritabanina basariyla baglanildi")
    except mysql.connector.Error as err:
        if err.errno==errorcode.ER_ACCESS_DENIED_ERROR:
            print("Kullanici adi veya parola hatali")
        elif err.errno ==errorcode.ER_BAD_DB_ERROR:
            print ("Database adi hatali veya yanlis yazilmis")
        else:
            print(err)

    return cnx

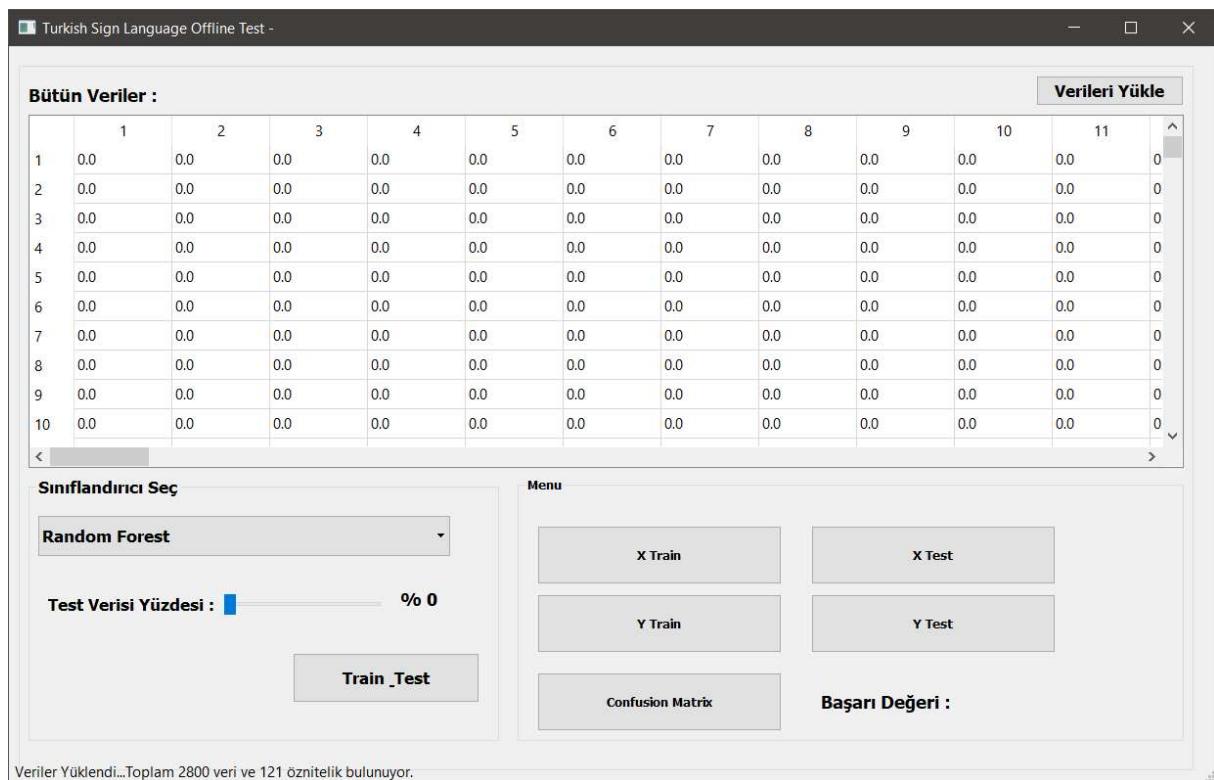
def text_seperate(tablerow):
    t_row=""
    for i,row in enumerate (tablerow):
        if i!=len(tablerow)-1:
            t_row+=str(row)+", "
        else:
            t_row+=str(row)
    return t_row

def addTable(cursor,tablename,tablerow,tabledata):
    crs=cursor
    t_row=text_seperate(tablerow)
    data=text_seperate(tabledata)
    query=( "INSERT INTO "+str(tablename)+" ("+t_row+") VALUES ("+data+")")
    crs.execute(query)
#    crs.close()

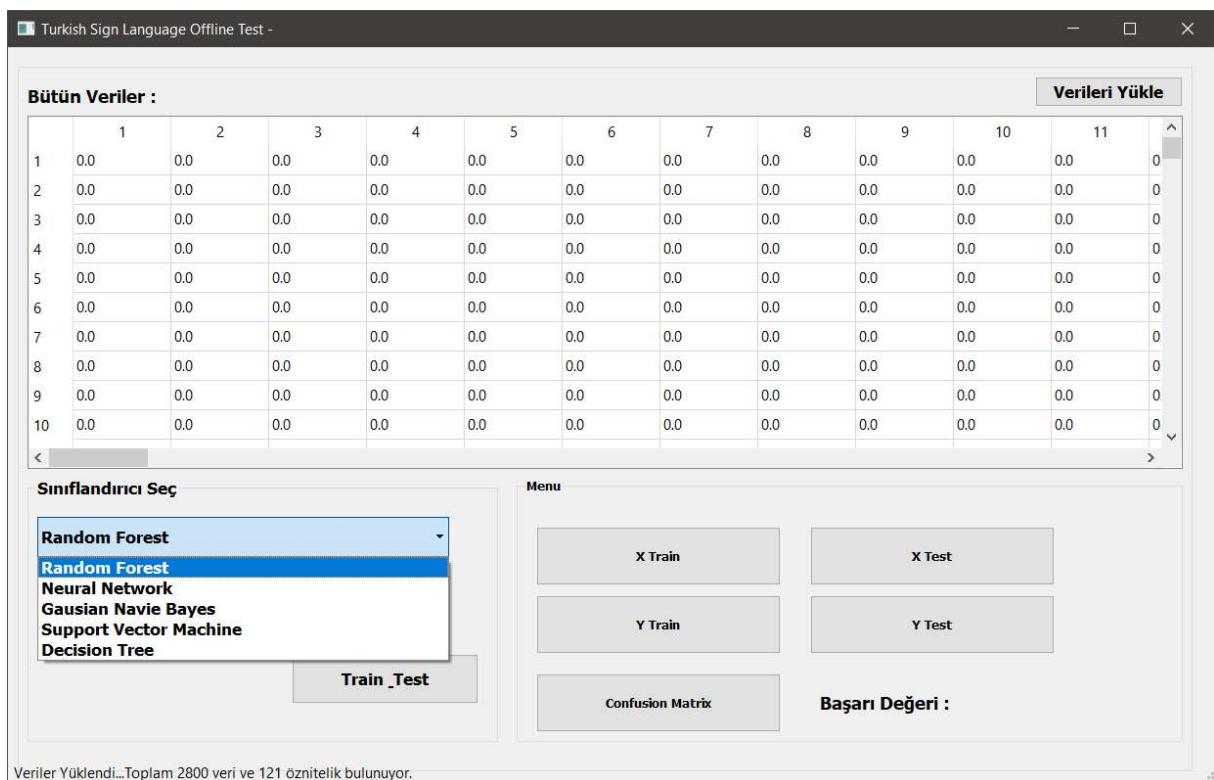
def selectTable(cursor):
    crs=cursor
    query=("SELECT * FROM leap_data")
    crs.execute(query)
    for i in crs:
        print(str(len(i)))
        break
    return crs
```

Şekil 20 DataBase.py Tüm Fonksiyonlar

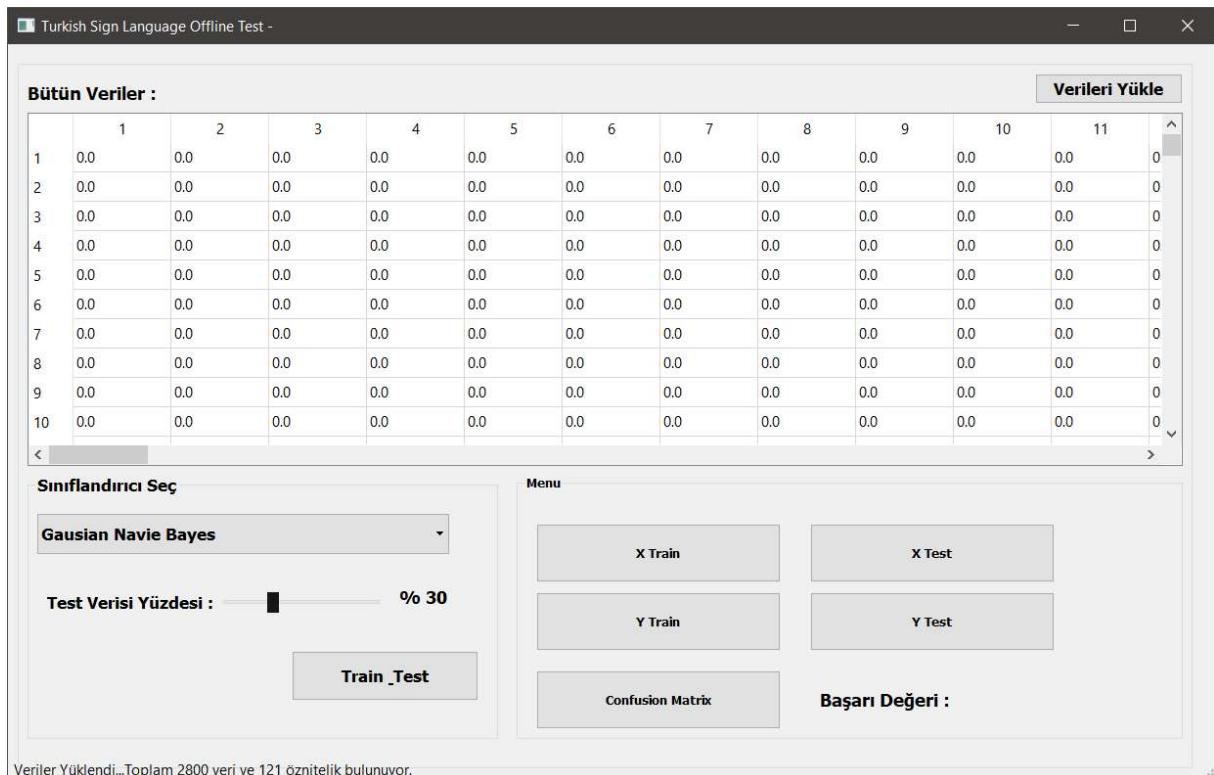
### 3.2.1.3 ARAYÜZ TASARIMI



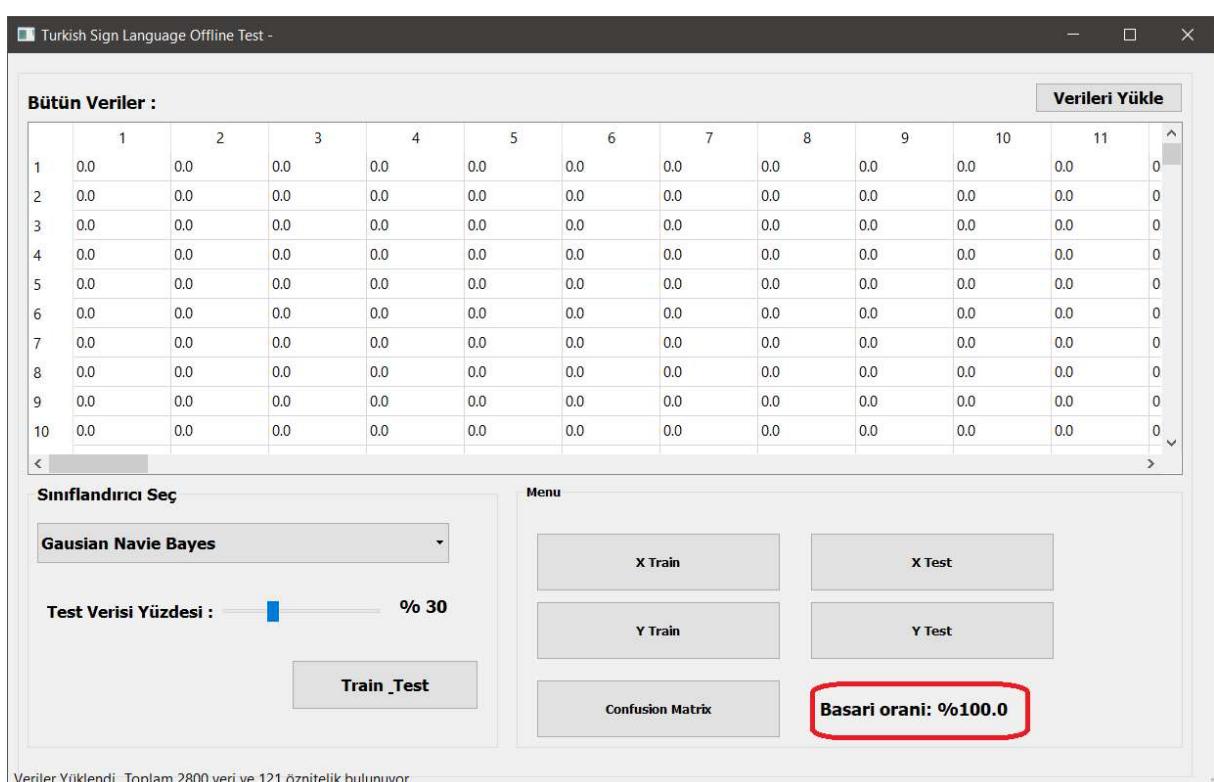
Şekil 22 Arayüz Tasarımı Verileri Yükle



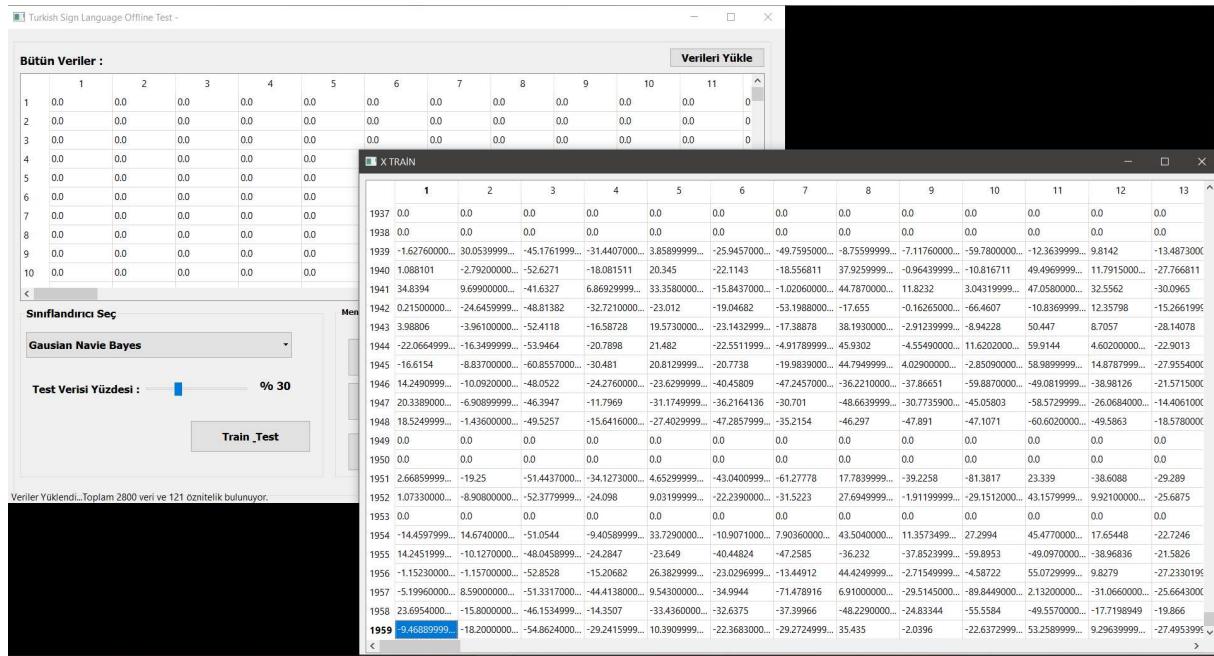
Şekil 21 Arayüz Tasarımı Sınıflandırıcı Seç



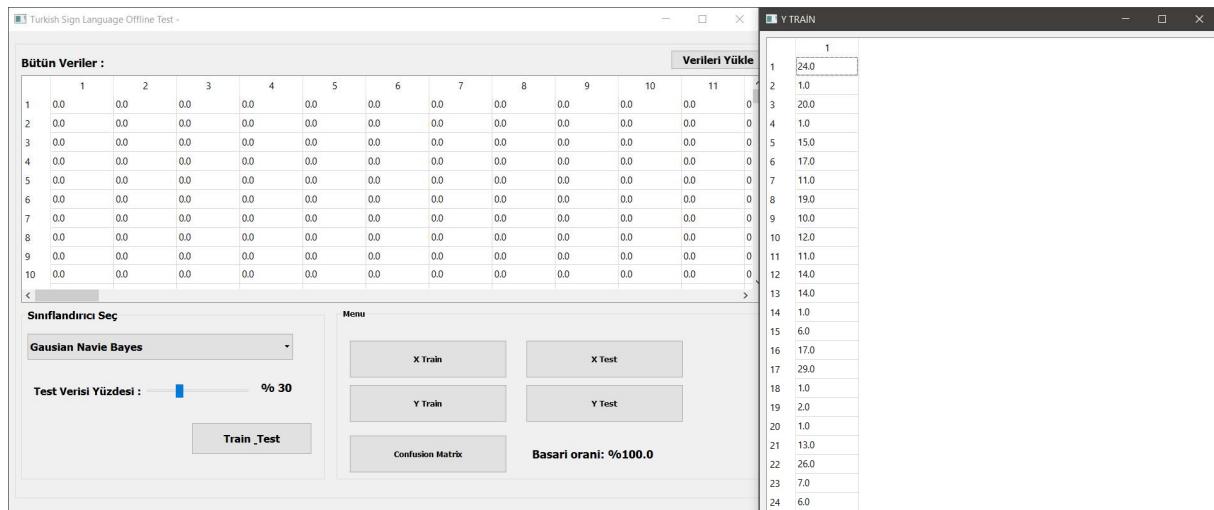
Şekil 243 Arayüz Tasarımı Eğitim ve Test Verisi Belirle, Test Et



Şekil 234 Arayüz Tasarımı Başarım Oranı



Şekil 25 Arayüz Tasarımı Eğitim Verileri



Şekil 26 Arayüz Tasarımı Eğitim Verileri Etiket Bilgileri

**Bütün Veriler :**

	1	2	3	4	5	6	7	8	9	10	11
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

**Verileri Yükle**

**Smiflendirici Seç :** Gausian Navie Bayes

**Test Verisi Yüzdesi :** 0% 30

**Train \_Test**

**Menu**

X Train X Test  
Y Train Y Test  
Confusion Matrix Basarı orani: %100.0

**X TEST**

1	2	3	4	5	6	7	8	9	10	11
818	16.6593999...	-11.0769999...	-47.0494000...	-14.9065000...	-32.2220000...	-30.2677000...	-36.2131			
819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
820	-15.1824999...	-5.4550000...	-52.57208	-24.7057	-38.875	-24.49548	-34.0314			
821	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
822	4.13449999...	-11.3359999...	-54.0971999...	-34.2599	-40.7499999...	-31.7205999...	-62.889400			
824	-9.44599999...	-18.5209999...	-54.7589000...	-29.1738999...	10.3000000...	-22.4411	-29.181999			
825	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
826	-21.9237	-16.0400000...	-54.0968	-20.8393	21.6479999...	-22.5232000...	-4.9011			
827	-3.24899999...	-14.4219999...	-55.1481	-35.44269	-27.0389999...	-24.2312	-55.6597			
829	17.7000000...	-7.88900000...	-51.3103	-15.375	14.8549999...	-31.9671000...	-27.185900			
830	45.4831	16.6800000...	-20.2029	7.25379999...	32.9130000...	-10.5413999...	-13.881700			
831	-15.6855000...	-5.78600000...	-52.6350000...	-25.3010000...	21.774	-18.5630000...	-19.1459			
832	-13.6323999...	14.2420000...	-51.4032	-9.15140000...	33.4160000...	-11.2452	7.8277			
833	40.4886000...	3.86199999...	-33.1794	7.46010000...	26.9259999...	-19.1961	-7.3373999			
834	-8.7369999...	-6.20599999...	-53.1391999...	-37.3835	-14.3859999...	-26.5093000...	-60.5293			
835	16.07879999...	-20.0100000...	-48.23124	-17.4286000...	4.09600000...	-31.90504	-32.5435			
836	30.3310000...	-14.7170000...	-44.4824	-2.08199999...	14.2839999...	-31.661	-14.888999			
837	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
838	-15.03474	3.89600000...	-52.1588	-6.05673	28.155	-16.3028000...	14.0261			
839	18.2971	-15.9120000...	-45.0102	-21.8429000...	-26.2150000...	-42.2249	-45.992000			
840	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Şekil 28 Arayüz Tasarımı Test Verisi

**Bütün Veriler :**

	1	2	3	4	5	6	7	8	9	10	11
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

**Verileri Yükle**

**Smiflendirici Seç :** Gausian Navie Bayes

**Test Verisi Yüzdesi :** 0% 30

**Train \_Test**

**Menu**

X Train X Test  
Y Train Y Test  
Confusion Matrix Basarı orani: %100.0

**Y TEST**

1
20.0
29.0
23.0
29.0
10.0
29.0
5.0
1.0
18.0
6.0
29.0
28.0
19.0
11.0
25.0
10.0
2.0
25.0
20.0
3.0
26.0
12.0
27.0
5.0

Şekil 27 Arayüz Tasarımı Test Verisi Etiket Bilgileri

The screenshot shows two windows side-by-side. The left window is titled 'Turkish Sign Language Offline Test -' and contains a table titled 'Bütün Veriler :'. It has 11 columns labeled 1 through 11. Below this is a section titled 'Sınıflandırıcı Seç' with a dropdown menu set to 'Gaussian Navie Bayes'. A slider bar indicates 'Test Verisi Yüzdesi : % 30'. At the bottom are buttons for 'Train\_Test' and 'Confusion Matris'. A button 'Verileri Yükle' is located at the top right of this window. The right window is titled 'Karmaşıklık Matrisi' and displays a 'Confusion Matrix' table with rows and columns labeled with letters A through T. The matrix values are as follows:

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K
A	28	0	0	0	0	0	0	0	0	0	0	0	0
B	0	32	0	0	0	0	0	0	0	0	0	0	0
C	0	0	29	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	35	0	0	0	0	0	0	0	0	0
D	0	0	0	0	36	0	0	0	0	0	0	0	0
E	0	0	0	0	0	35	0	0	0	0	0	0	0
F	0	0	0	0	0	0	27	0	0	0	0	0	0
G	0	0	0	0	0	0	0	33	0	0	0	0	0
H	0	0	0	0	0	0	0	0	29	0	0	0	0
I	0	0	0	0	0	0	0	0	0	27	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	23	0	0
J	0	0	0	0	0	0	0	0	0	0	30	0	0
K	0	0	0	0	0	0	0	0	0	0	0	35	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0

Şekil 30 Arayüz Tasarımı Karmaşıklık Matrisi

The screenshot shows a single window titled 'Karmaşıklık Matrisi' containing a large 'Confusion Matrix' table. The table has 26 rows and 15 columns, labeled with letters A through T and characters İ, Ö, and Ş. The matrix values are identical to the one shown in Figure 30, with the first row (A) having a value of 28 and the second row (B) having a value of 32.

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K
A	28	0	0	0	0	0	0	0	0	0	0	0	0
B	0	32	0	0	0	0	0	0	0	0	0	0	0
C	0	0	29	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	35	0	0	0	0	0	0	0	0	0
D	0	0	0	0	36	0	0	0	0	0	0	0	0
E	0	0	0	0	0	35	0	0	0	0	0	0	0
F	0	0	0	0	0	0	27	0	0	0	0	0	0
G	0	0	0	0	0	0	0	33	0	0	0	0	0
H	0	0	0	0	0	0	0	0	29	0	0	0	0
I	0	0	0	0	0	0	0	0	0	27	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	23	0	0
J	0	0	0	0	0	0	0	0	0	0	30	0	0
K	0	0	0	0	0	0	0	0	0	0	0	35	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0

Şekil 29 Arayüz Tasarımı Karmaşıklık Matrisi Büyüğ

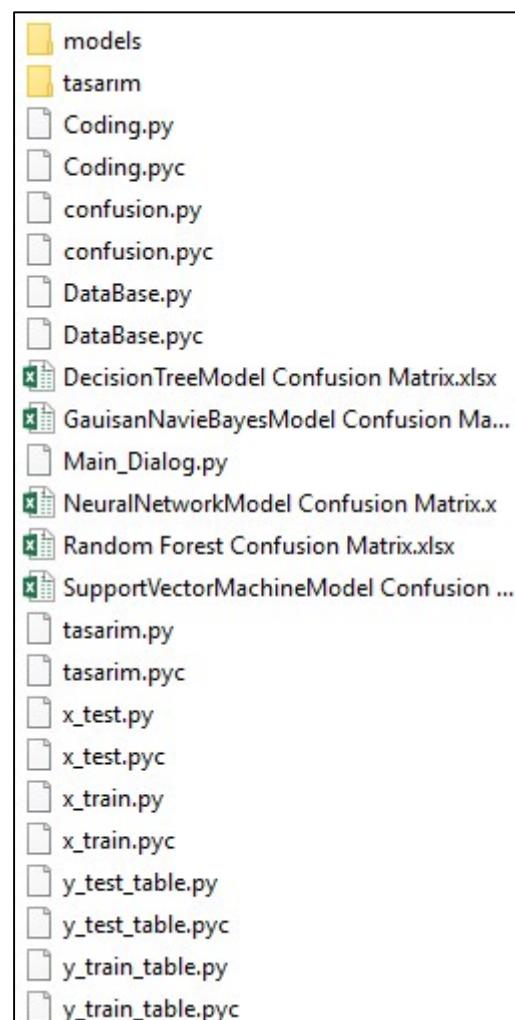
### **3.2.1.4 OFFLINE TEST SONUÇLARI**

<b>Algoritmalar</b>	<b>Öznitelik Sayısı</b>	<b>391 Öznitelik</b>
Random Forest	100%	
Navie Bayes	100%	
Neural Network	23%	
Support Vector Machine	100%	
Decision Tree	99%	

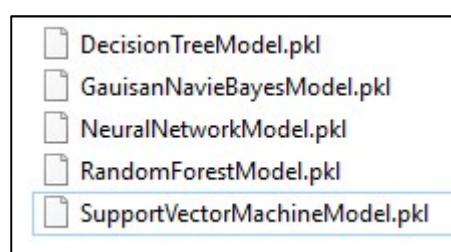
### 3.2.2 121 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK AĞ EĞİTİMİ

Veritabanından çekilen veriler kullanıcının istediği oran doğrultusunda eğitim ve test verisi olarak ayırdıktan sonra kullanıcının uygulamak istediği makine öğrenmesi algoritmalarından Rastgele Orman (*Random Forest*), Yapay Sinir Ağları (*Neural Network*), *Naive Bayes*, Karar Ağaçları (*Decision Tree*) veya Destek Vektör Makineleri (*Support Vector Machine*) seçeneklerinden birini seçerek modelleme yapılmıştır

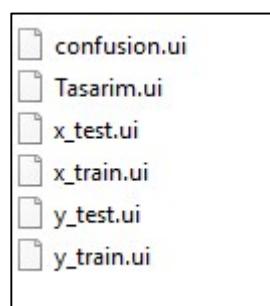
#### 3.2.2.1 DOSYA YAPISI



Sekil 31 121 ÖZNİTELİKLİ VERİ SETİ KULLANILARAK  
AĞ EĞİTİMİ DOSYA YAPISI



Sekil 32 121 Öznitelikli Veri Seti Models  
Klasörü



Sekil 33 121 Öznitelikli Veri  
Seti tasarım klasörü

**Coding.py :** Model oluşturabilmek ve oluşturululan modelin başarım oranını test edebilmek için yazılan bütün kodlar burada yer alır.

**DataBase.py :** Veritabanı bağlantılarının yapıldığı ve veri getirme işlemlerinin uygulandığı kodlar burada yer alır.

**MainDialog.py :** Bütün python kodlarının birleştirilip çalıştırıldığı ana pencere kodları burada yer alır.

**confusion.py , x\_test.py,x\_train.py,y\_test\_table.py,  
y\_train\_table.py :** Veri ayırma işleminden sonra ayrılan veri değerlerini görebilmek için oluşturulmuş tablolardır.

**.ui uzantılı Dosyalar:** Tasarım dosyalarıdır form görüntüleri bu dosyalar sayesinde oluşturulur.

**.pyc uzantılı Dosyalar:** Derlenmiş Python dosyaları. Modül olarak kullanılırlar. Bir Python programı içinden çalıştırılabilirler ama açılmazlar. Yani okunaksız dosyalar.

### 3.2.2.2 KODLAR VE AÇIKLAMALAR

Coding.py `__init__` fonksiyonu: Bu fonksiyon form ilk yükleniği anda yapılacak işlemleri belirler.

```
def __init__(self):
    QtGui.QMainWindow.__init__(self)
    self.setupUi(self)
    self.statusBar().showMessage(unicode("Hazır\n"))
    self.btn_dosyaYukle3.clicked.connect(self.VerileriYukle)
    self.hs_test_value.valueChanged.connect(self.hs_valueChanged)
    self.cb_classificier.currentIndexChanged.connect(self.cb_classificier_changed)
    self.btn_split.clicked.connect(self.train_and_test_click)
    self.liste=[]
    self.cnx=connect("root","","127.0.0.1","leap_data_new") #Veritabanı bağlantı ayarları
    self.secim=0
    self.modelName='Random Forest'
    self.confusionMatrix=confusion.Ui_Dialog() #confusionMatrix örneği alınıyor
    self.confusionWidget = QtGui.QWidget()
    self.confusionMatrix.setupUi(self.confusionWidget) # veriler widgeta set ediliyor.

    #Eğitim verilerini göstermek için form örneği oluşturuluyor.
    self.formXtrain=x_train.Ui_Dialog()
    self.formXtrainWidget=QtGui.QWidget()
    self.formXtrain.setupUi(self.formXtrainWidget)
    #Eğitim verilerinin etiketleri gösterilmek için form örneği oluşturuluyor
    self.formYtrain=y_train_table.Ui_Dialog()
    self.formYtrainWidget=QtGui.QWidget()
    self.formYtrain.setupUi(self.formYtrainWidget)

    #Test verilerini göstermek için form örneği oluşturuluyor
    self.formXtest=x_test.Ui_Dialog()
    self.formXtestWidget=QtGui.QWidget()
    self.formXtest.setupUi(self.formXtestWidget)

    #Test Verilerinin etiketleri gösterilmek için form örneği oluşturuluyor
    self.formYtest=y_test_table.Ui_Dialog()
    self.formYtestWidget=QtGui.QWidget()
    self.formYtest.setupUi(self.formYtestWidget)
    #Butonlar aktif ediliyor
    self.btn_confusion.clicked.connect(self.confusion_show)
    self.pb_x_train.clicked.connect(self.x_train_show)
    self.pb_x_test.clicked.connect(self.x_test_show)
    self.pb_y_test.clicked.connect(self.y_test_show)
    self.pb_y_train.clicked.connect(self.y_train_show)
```

Şekil 34 Coding.py `__init__` fonksiyonu

Coding.py VerileriYukle:

```
def VerileriYukle(self): #veritabanına bağlanıp tablodan veri çekme işlemeleri
    cursor=self.cnx.cursor(buffered=True)
    self.alldata=[]
    select=selectTable(cursor)

    for a in select:
        self.alldata.append(a)
    self.alldata=np.array(self.alldata)
    self.verileri_dok(self.alldata,self.table_all_data)
    veri_sayisi=self.alldata.shape[0]
    oznitelik_sayisi=self.alldata.shape[1]
    self.statusBar().showMessage(unicode("Veriler Yükleniyor...Toplam "+str(veri_sayisi+1)
    +" veri ve "+str(oznitelik_sayisi)+" öznitelik bulunuyor.",'utf-8'))
```

Şekil 35 Coding.py VerileriYukle Fonksiyonu

Coding.py verileri\_yaz: Confusion matrix sonuçlarını excel tablosuna dökmek için yazılmıştır.

```
def verileri_yaz(self,veri,ismi):
    harfler=['A','B','C',unicode('Ç','utf-8'),'D','E','F','G','H','I',
             unicode('İ','utf-8'),'J','K','L','M','N','O',unicode('Ö','utf-8'),
             'P','R','S',unicode('Ş','utf-8'),'T','U',unicode('Ü','utf-8'),'V','Y','Z']
    excel=Workbook()
    excelws=excel['Sheet']

    for i,harf in enumerate(harfler):
        excelws.cell(row=0,column=i+1).value=harf
        excelws.cell(row=i+1,column=0).value=harf

    for rowNumber,row in enumerate(veri): #Veri satır satır okunuyor
        for columnNumber in range(0,len(veri[0])): # Her bir satırdaki veri sütun sütun okunuyor
            excelws.cell(row=rowNumber+1, column=columnNumber+1).value = str(row[columnNumber]) #tablonun rowNumber'inci satır
            #columnNumber'inci sütünuna veri ekleniyor.
    excel.save(u"./"+ismi+".xlsx")
```

Şekil 36 Coding.py verileri\_yaz fonksiyonu

Coding.py verileri\_dok: Bu fonksiyon gelen tabloya , gelen verileri dökmek için kullanılır.

```
def verileri_dok(self,X,tablo):
    num_rows=len(X) #gönderilen veriden satır sayısı alınıyor
    tablo.clear() # tablo temizleniyor
    tablo.setRowCount(len(X[0])) # gönderilen veriden sütun sayısı alınıyor ve tabloya set ediliyor.
    tablo.setColumnCount(num_rows) #alınan satır sayısı tabloya set ediliyor.
    for rowNumber, row in enumerate(X): #Veri satır satır okunuyor
        for columnNumber in range(0,len(X[0])): # Her bir satırdaki veri sütun sütun okunuyor
            tablo.setItem(rowNumber,columnNumber,QtGui.QTableWidgetItem(str(row[columnNumber]))) #tablonun rowNumber'inci
            #satır columnNumber'inci sütünuna veri ekleniyor.
```

Şekil 37 Coding.py verileri\_dok Foksiyonu

Coding.py test\_verilerini\_dok: verileri dök fonksiyonundan farkı sütun sayısının sabit olması yani bu fonksiyon etiket değerlerini göstermek için kullanılıyor.

```
def test_verilerini_dok(self,X,tablo):
    num_rows=len(X) #gönderilen veriden satır sayısı alınıyor
    tablo.clear() # tablo temizleniyor
    tablo.setRowCount(1) # gönderilen veriden sütun sayısı alınıyor ve tabloya set ediliyor.
    tablo.setColumnCount(num_rows) #alınan satır sayısı tabloya set ediliyor.
    for rowNumber, row in enumerate(X): #Veri satır satır okunuyor
        tablo.setItem(rowNumber,0,QtGui.QTableWidgetItem(str(row))) #tablonun rowNumber'inci satır columnNumber'inci
        #sütünuna veri ekleniyor
```

Şekil 38 Coding.py test\_verilerini\_dok fonksiyonu

Coding.py cb\_classificier\_changed: Kullanıcının sınıflandırıcı türünü seçebilmesi için forma eklenmiş olan combobox nesnesinin changeIndex Metodu

```
def cb_classificier_changed(self,value): # sınıflandırıcı türünü seçmek için kullanılan combobox 'ın changeIndex Metodu.
    if (value==0):
        self.secim=0
        self.modelName='RandomForestModel'
    elif(value==1):
        self.secim=1
        self.modelName='NeuralNetworkModel'
    elif(value==2):
        self.secim=2
        self.modelName='GaussianNaiveBayesModel'
    elif(value==3):
        self.secim=3
        self.modelName='SupportVectorMachineModel'
    elif(value==4):
        self.secim=4
        self.modelName='DecisionTreeModel'
```

Şekil 39 Coding.py cb\_classificier\_changed

Coding.py hs\_ValueChanged: Eğitim ve test verileri ayrılrken test verisinin yüzdesini belirlemek amacıyla kullanılan horizontal slider in ValueChanged metodu.

```
def hs_ValueChanged(self): # Eğitim ve test verisini ayırrken test verisinin yüzdesini
#belirlemek amacıyla kullanılan horizontal slider valueChanged olayı
    val=% "+str(self.hs_test_value.value()) #Value nun görsel olarak kullanıcılaraya aktarılması
    self.lbl_hs_value.setText(val)
```

Coding.py train\_test\_click: Veritabanından çekilen veriler eğitim ve test verisi olmak üzere ayrılıyor ve sınıflandırma fonksiyonuna gönderip model oluşturuluyor.

```
def train_and_test_click(self):
    X=self.alldata[:,120]
    y=self.alldata[:,120]
    value=float(self.hs_test_value.value())
    value=value/100
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=value)
    self.classification(X_train,y_train,X_test,y_test)
```

Şekil 40 Coding.py train\_test\_clik Fonksiyonu

Coding.py classification: Burada yapılan seçime göre sınıflandırıcı oluşturuluyor. Oluşturulan sınıflandırıcı eğitim verileri ile eğitiliyor. Eğitilen model kayıt ediliyor daha sonra kayıt edilen model ile test verileri karşılaştırılıyor. Oluşan sonuçlar ile beklenen sonuçlar karşılaştırılıyor ve karmaşıklık matrisi ve başarı oranı oluşturuluyor.

```

def classification(self,X_train,y_train,X_test,y_test):
    if(self.secim==0): #randomForest
        clf=RandomForestClassifier(max_depth=None,random_state=42)
    elif(self.secim==1): #neuralNetwork
        clf=MLPClassifier(activation='tanh', solver='adam', alpha=1e-5,hidden_layer_sizes=(5,), random_state=1)
    elif(self.secim==2):#decisionTree
        clf=DecisionTreeClassifier()
    elif(self.secim==3):#supportVectorMachine
        clf=SVC()
    elif(self.secim==4):#gauisanNavieBayes
        clf=GaussianNB()

    clf.fit(X_train,y_train) #sınıflandırıcı veriler ile eğitiliyor
    mname='./models/'+str(self.modelName)+'.pkl' #eğitilen modele isim veriliyor
    joblib.dump(clf,mname ) #eğitilen model kayıt ediliyor
    result=clf.predict(X_test) # Model üzerinde test verileri test ediliyor
    self.verileri_dok(confusion_matrix(result,y_test,labels=[1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29]))
    self.confusionMatrix.table_confusion_matrix #doğru sonuçlar ile elde edilen sonuçlar karşılaştırılıyor
    #ve karmaşıklık matrisi oluşturuluyor.
    self.confusionMatrix.table_confusion_matrix.setHorizontalHeaderLabels(['A','B','C',
        unicode('Ç','utf-8'),unicode('İ','utf-8'),unicode('Ö','utf-8'),'D','E','F','G','H','I',
        unicode('Ü','utf-8'),'J','K','L','M','N','O',unicode('Ş','utf-8'),'P','R','S',unicode('Ş','utf-8'),'T','U',
        unicode('Ü','utf-8'),'V','Y','Z'])
    self.confusionMatrix.table_confusion_matrix.setVerticalHeaderLabels(['A','B','C',unicode('Ç','utf-8'),'D','E','F','G','H','I',
        unicode('Ü','utf-8'),'J','K','L','M','N','O',unicode('Ş','utf-8'),'P','R','S',unicode('Ş','utf-8'),'T','U',
        unicode('Ü','utf-8'),'V','Y','Z'])
    lbl_text=("Başarı oranı: %"+ str ((round(accuracy_score(y_test,result),2)*100)) #doğru sonuçlar ile elde edilen
    #sonuçlar karşılaştırılıyor ve başarı oranı ölçülüyor. Kullanıcıya gösteriliyor.
    self.lbl_basari.setText(lbl_text) #kullanıcıya gösterilmek için label set ediliyor.

    self.verileri_dok(X_train,self.formXtrain.table_x_train) #Eğitim Verileri gösterilmek için set ediliyor
    self.verileri_dok(X_test,self.formXtest.table_x_test) # Test Verileri gösterilmek için set ediliyor
    self.test_verileri_dok(y_train,self.formYtrain.table_y_train) #Eğitim verilerinin etikleri gösteriliyor
    self.test_verileri_dok(y_test,self.formYtest.table_y_test) #Test verileri etiketleri gösteriliyor

    self.verileri_yaz(confusion_matrix(result,y_test,labels=[1,2,3,4,5,6,7,8,10,11,12,13,14,
        15,16,17,18,19,20,21,22,23,24,25,26,27,28,29]),
        self.modelName+" Confusion Matrix")
    #verileri yaz ile modele ait karmaşıklık matrisi excel tablosuna bastırılıyor

```

Şekil 41 Coding.py classification Fonksiyonu

Coding.py eklenen kütüphaneler:

```

@author: Bedirhan
"""

from PyQt4 import QtGui
from PyQt4 import QtCore
import numpy as np
from tasarim import Ui_Dialog
import confusion
import x_train
import x_test
import y_test_table
import y_train_table
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from DataBase import connect
from DataBase import selectTable
from sklearn.neural_network import MLPClassifier
from sklearn.externals import joblib
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from openpyxl import Workbook

```

Şekil 42 Coding.py Kullanılan Kütüphaneler

Coding.py verileri işle: Veritabanından çekilen veriler işlenmek üzere bu fonksiyona gönderilir. Veriler işlenerek 121 öznitelik oluşturulur.

```

def verileri_isle(self,X,table):
    islenmisveri=[]
    for rowNumber, row in enumerate(X): #Veritabanından çekilen veriler işlenmek üzere satır satır okunuyor
        temp=[] #geçici bir liste oluşturuluyor
        listx=[] #1.el için kemiklerin x koordinatlarını tutabilmek için liste tanımlanıyor.
        listy=[] #1.el için kemiklerin y koordinatlarını tutabilmek için liste tanımlanıyor.
        listz=[] #1.el için kemiklerin z koordinatlarını tutabilmek için liste tanımlanıyor.
        for i in range(0,3): #0-1-2 indisler 1.el için palm_position değerlerini tutmakta
            for j in range(3,63): #3-62 arasındaki indisler 1.el için m_e , p_e,i_e,d_e değerlerini tutmakta
                if(i==0 and j%3==0): # mod 3 olarak kemiklerin x,y,z koordinatlarını elde ediyoruz.
                    a=row[i]-row[j] #(pp_x)-(m_e_x) ,(pp_x)-(p_e_x) , (pp_x)-(i_e_x) , (pp_x)-(d_e_x) işlemleri sırasıyla yapılır
                    listx.append(a) # daha sonra bu değerler listede tutulur.
                elif (i==1 and j%3==1):
                    a=row[i]-row[j] #(pp_y)-(m_e_y) ,(pp_y)-(p_e_y) , (pp_y)-(i_e_y) , (pp_y)-(d_e_y) işlemleri sırasıyla yapılır
                    listy.append(a) # daha sonra bu değerler listede tutulur.
                elif (i==2 and j%3==2):
                    a=row[i]-row[j] #(pp_z)-(m_e_z) ,(pp_z)-(p_e_z) , (pp_z)-(i_e_z) , (pp_z)-(d_e_z) işlemleri sırasıyla yapılır
                    listz.append(a) # daha sonra bu değerler listede tutulur.

        for i in range(0,20): #listelerde tutulan x,y,z değerleri doğru yerlerine yerlestirilmek için farklı
            temp.append(listx[i])#bir dizide doğru sıra ile toplanırlar
            temp.append(listy[i])
            temp.append(listz[i])

        #yukarıda 1.el için yapılan işlemler , 2. el için yapılır.
        listx=[]
        listy=[]
        listz=[]
        for i in range(63,66):
            for j in range(66,126):
                if(i==63 and j%3==0):
                    a=row[i]-row[j]
                    table.setItem(rowNumber,j,QtGui.QTableWidgetItem(str(a)))
                    listx.append(a)
                elif (i==64 and j%3==1):
                    a=row[i]-row[j]
                    table.setItem(rowNumber,j,QtGui.QTableWidgetItem(str(a)))
                    listy.append(a)
                elif (i==65 and j%3==2):
                    a=row[i]-row[j]
                    table.setItem(rowNumber,j,QtGui.QTableWidgetItem(str(a)))
                    listz.append(a)

        for i in range(0,20):
            temp.append(listx[i])
            temp.append(listy[i])
            temp.append(listz[i])

        temp.append(row[126]) #Yapılan işlemlerin sonunda işlenmiş bir satır verİYE en son etiketi eklenir ve bu veri
        islenmisveri.append(temp) #işlenmiş olarak yeni listede yerini alır.
    islenmisveri=np.array(islenmisveri) # işlenmiş veriler dizi ye çevrilir
    self.verileri_dok(islenmisveri,table) # diziye çevrilen veri kullanıcıya gösterilmek üzere tabloya gönderilir.
    self.alldata=islenmisveri #işlenen veriyi sınıflandırma işlemlerinde kullanılması için global bir değişkene aktarılır.

```

Şekil 43 Coding .py Verileri İşle Fonksiyonu

Database.py Fonksiyonlar:

Connect: Veritabanı bağlantılarının yapıldığı fonksiyon

Text\_seperate: Verileri eklemek için gönderilen string deki sütun isimlerini ayırmak için kullanılıyor

addTable: Verileri Tabloya ekler SelectTable: Eklenmiş verileri tablodan getirir.

```
@author: Bedirhan
"""

import mysql.connector
from mysql.connector import errorcode

def connect(username,password,host,databasename):
    config = {
        'user': username,
        'password': password,
        'host': host,
        'database': databasename,
        'raise_on_warnings': True }

    try:
        cnx=mysql.connector.connect(**config)
        print("Veritabanina basariyla baglanildi")
    except mysql.connector.Error as err:
        if err.errno==errorcode.ER_ACCESS_DENIED_ERROR:
            print("Kullanici adi veya parola hatali")
        elif err.errno ==errorcode.ER_BAD_DB_ERROR:
            print ("Database adi hatali veya yanlis yazilmis")
        else:
            print(err)

    return cnx

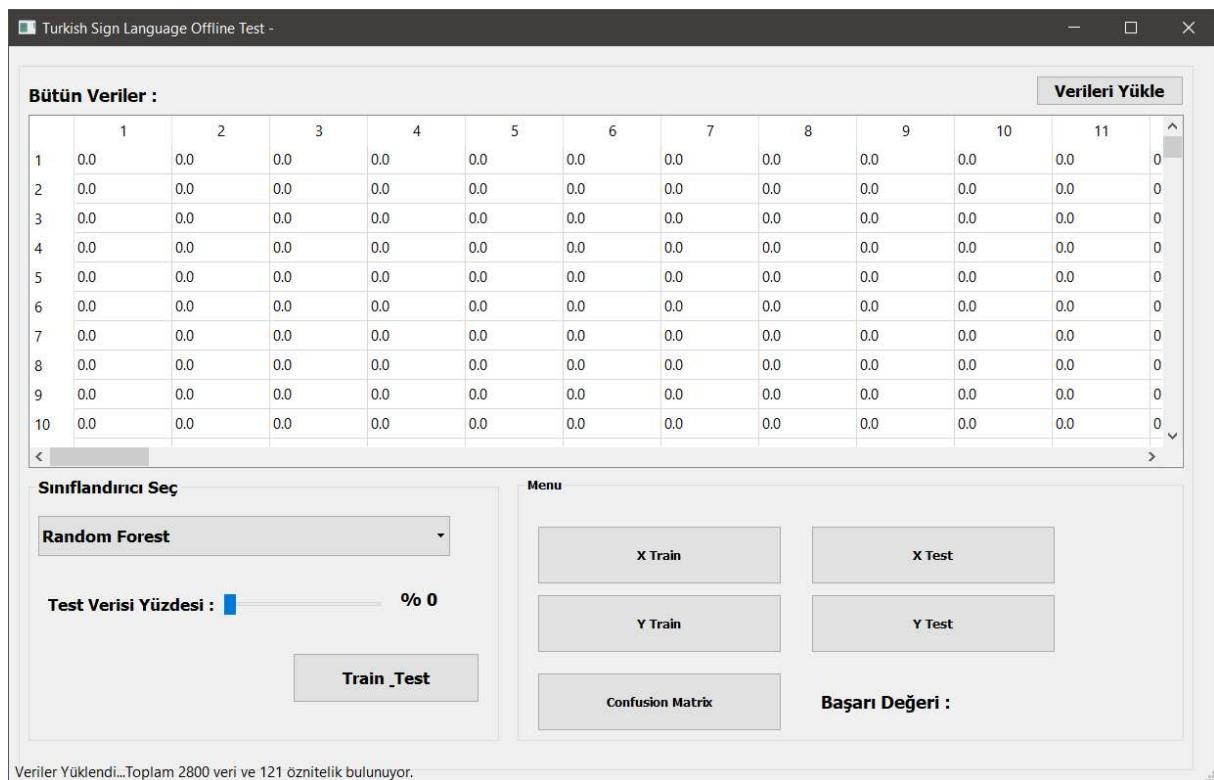
def text_seperate(tablerow):
    t_row=""
    for i,row in enumerate (tablerow):
        if i!=len(tablerow)-1:
            t_row+=str(row)+", "
        else:
            t_row+=str(row)
    return t_row

def addTable(cursor,tablename,tablerow,tabledata):
    crs=cursor
    t_row=text_seperate(tablerow)
    data=text_seperate(tabledata)
    query=( "INSERT INTO "+str(tablename)+" ("+t_row+") VALUES ("+data+")")
    crs.execute(query)
#    crs.close()

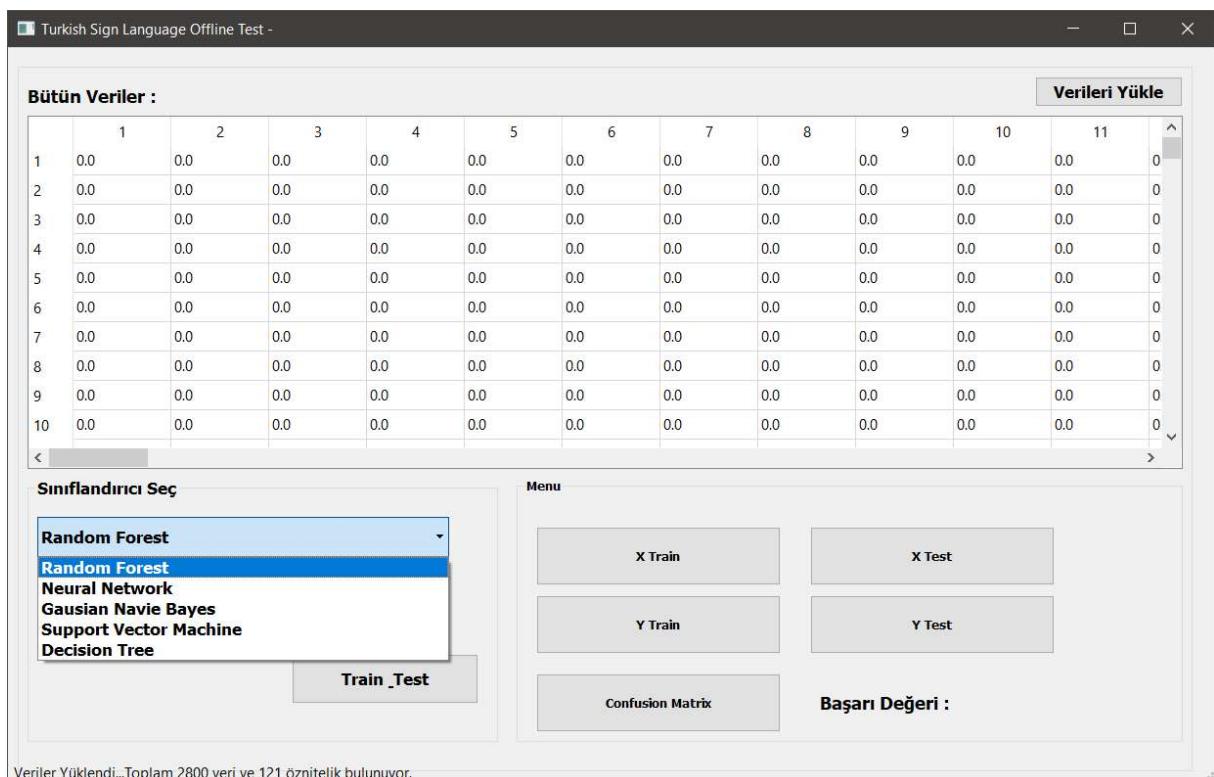
def selectTable(cursor):
    crs=cursor
    query=("SELECT * FROM leap_data")
    crs.execute(query)
    for i in crs:
        print(str(len(i)))
        break
    return crs
```

Şekil 44 DataBase.py Bütün Fonksiyonlar

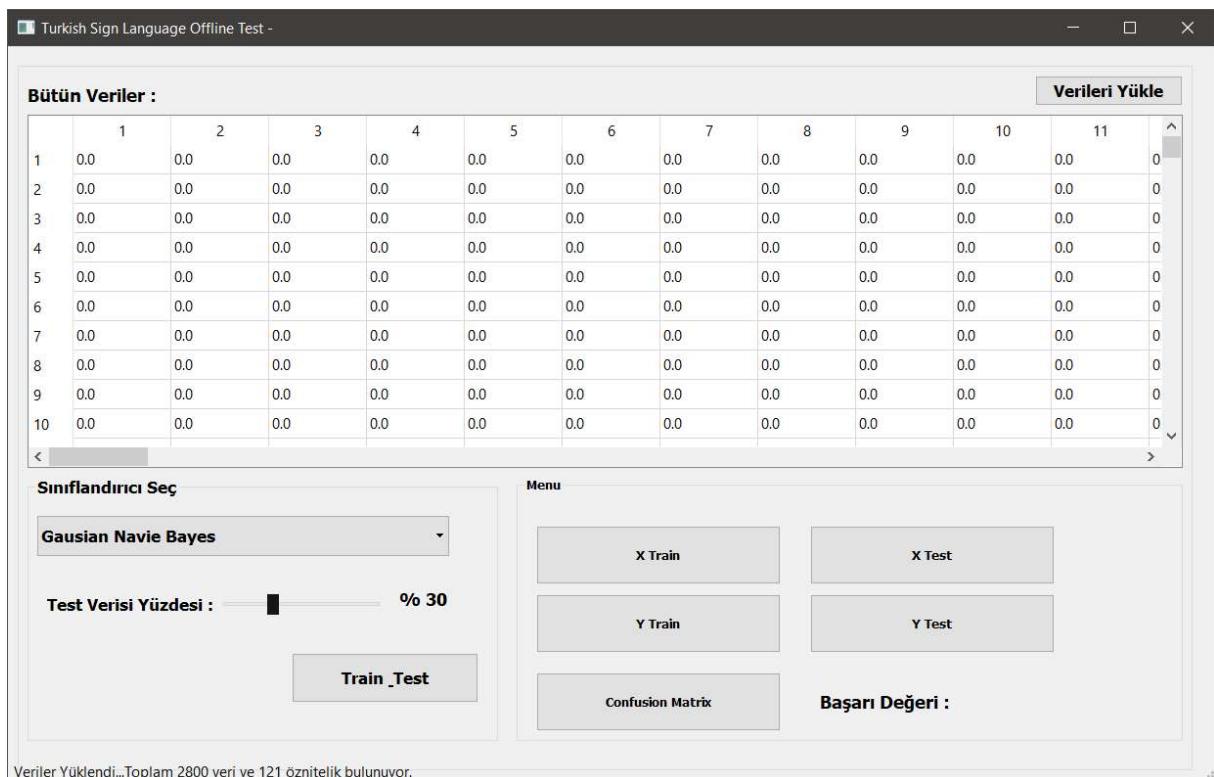
### 3.2.2.3 ARAYÜZ TASARIMI



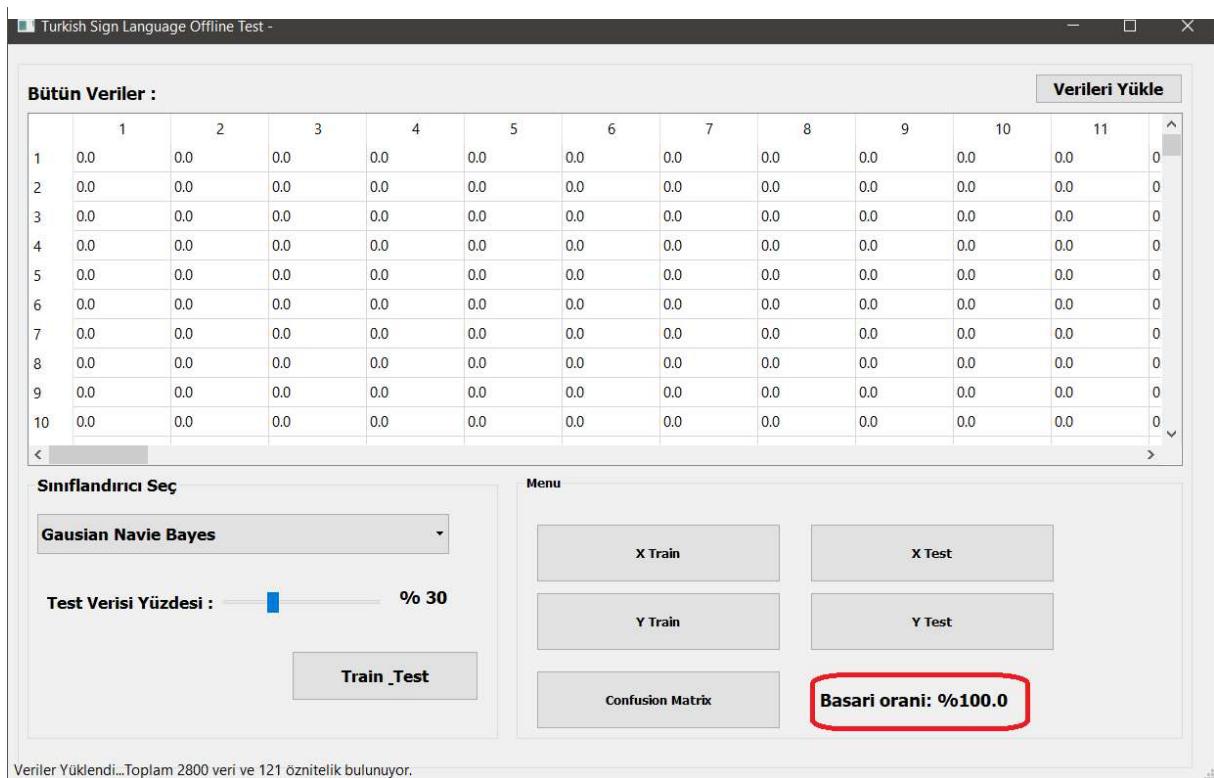
Şekil 46 Arayüz Tasarımı Verileri Yükle



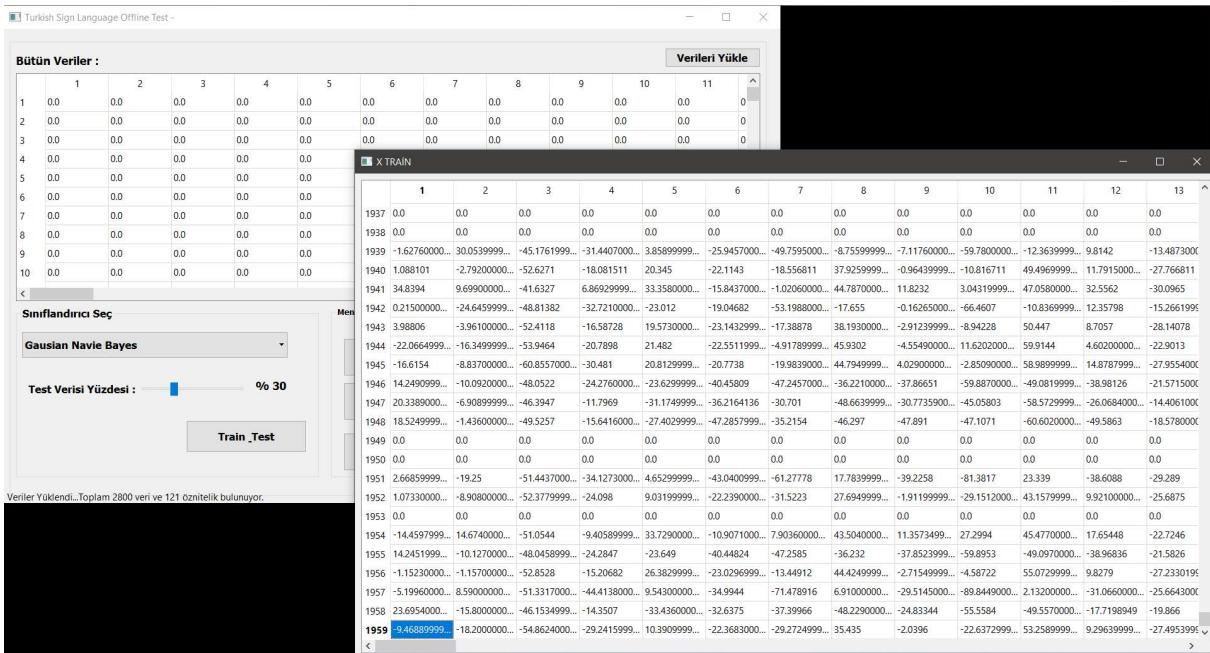
Şekil 45 Arayüz Tasarımı Sınıflandırıcı Seç



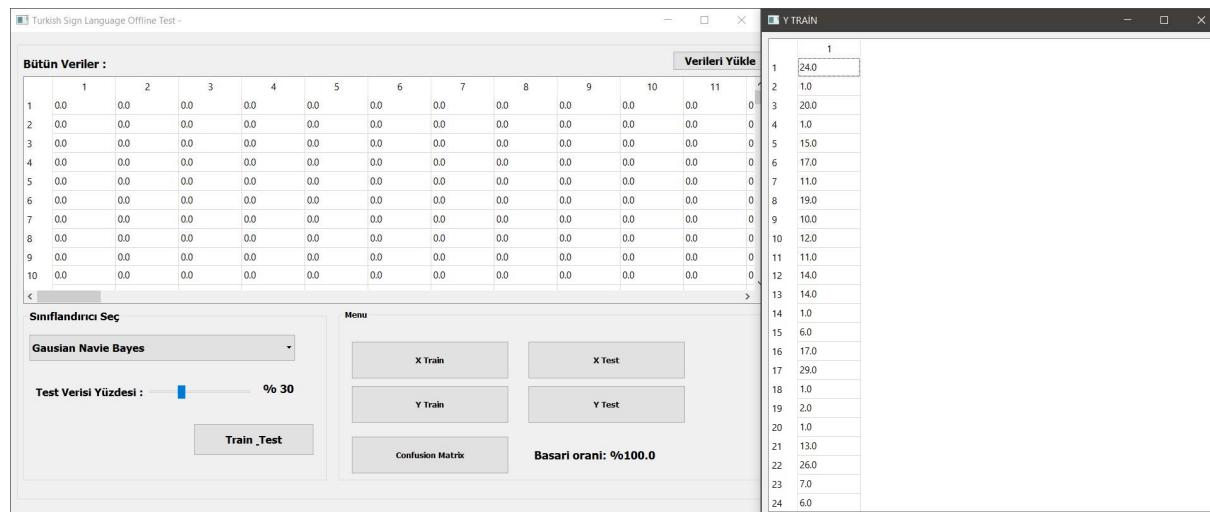
*Şekil 47 Arayüz Tasarımı Eğitim ve Test Verisi Belirle, Test Et*



*Şekil 48 Arayüz Tasarımı Başarım Oranı*



Şekil 50 Arayüz Tasarımı Eğitim Verileri



Şekil 49 Arayüz Tasarımı Eğitim Verileri Etiket Bilgileri

**Bütün Veriler :**

	1	2	3	4	5	6	7	8	9	10	11
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

**Verileri Yükle**

1	2	3	4	5	6	7	8	9	10	11
818	16.6593999...	-11.0769999...	-47.0494000...	-14.9065000...	-32.2220000...	-30.2677000...	-36.2131			
819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
820	-15.1824999...	-5.4550000...	-52.57208	-24.7057	-38.875	-24.49548	-34.0314			
821	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
822	4.13449999...	-11.3359999...	-54.0971999...	-34.2599	-4.07499999...	-31.7205999...	-62.889400			
824	-9.44599999...	-18.5209999...	-54.7589000...	-29.1738999...	10.3000000...	-22.4411	-29.181999			
825	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
826	-21.9237	-16.0400000...	-54.0968	-20.8393	21.6479999...	-22.5232000...	-4.9011			
827	-3.24899999...	-14.4219999...	-55.1481	-35.44269	-27.0389999...	-24.2312	-55.6597			
829	17.7000000...	-7.88900000...	-51.3103	-15.375	14.8549999...	-31.9671000...	-27.185900			
830	45.4831	16.6800000...	-20.2029	7.25379999...	32.9130000...	-10.5413999...	-13.881700			
831	-15.6855000...	-5.78600000...	-52.6350000...	-25.3010000...	21.774	-18.5630000...	-19.1459			
832	-13.6323999...	14.2420000...	-51.14032	-9.15140000...	33.4160000...	-11.2452	7.8277			
833	40.4886000...	3.86199999...	-33.1794	7.46010000...	26.9259999...	-19.1961	-7.3373999			
834	-8.7369999...	-6.20599999...	-53.1391999...	-37.3835	-14.3859999...	-26.5093000...	-60.5293			
835	16.07879999...	-20.0100000...	-48.23124	-17.4286000...	4.09600000...	-31.90504	-32.5435			
836	30.3310000...	-14.7170000...	-44.4824	-2.08199999...	14.2839999...	-31.661	-14.888999			
837	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
838	-15.03474	3.89600000...	-52.1588	-6.05673	28.155	-16.3028000...	14.0261			
839	18.2971	-15.9120000...	-45.0102	-21.8429000...	-26.2150000...	-42.2249	-45.992000			
840	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Şekil 52 Arayüz Tasarımı Test Verisi

**Bütün Veriler :**

	1	2	3	4	5	6	7	8	9	10	11
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

**Verileri Yükle**

1	
1	20.0
2	29.0
3	23.0
4	29.0
5	10.0
6	29.0
7	5.0
8	1.0
9	18.0
10	6.0
11	29.0
12	28.0
13	19.0
14	11.0
15	25.0
16	10.0
17	2.0
18	25.0
19	20.0
20	3.0
21	26.0
22	12.0
23	27.0
24	5.0

Şekil 51 Arayüz Tasarımı Test Verisi Etiket Bilgileri

The screenshot shows two windows side-by-side. The left window is titled 'Turkish Sign Language Offline Test -' and contains a table of data with 10 rows and 12 columns, labeled 1 through 11. A progress bar at the bottom indicates 'Verileri Yükle' (Loading Data) is at 0%. The right window is titled 'Karmaşıklık Matrisi' and shows a confusion matrix table with 26 rows and 17 columns, labeled A through T and Ç through K. The matrix values are as follows:

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K
A	28	0	0	0	0	0	0	0	0	0	0	0	0
B	0	32	0	0	0	0	0	0	0	0	0	0	0
C	0	0	29	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	35	0	0	0	0	0	0	0	0	0
D	0	0	0	0	36	0	0	0	0	0	0	0	0
E	0	0	0	0	0	35	0	0	0	0	0	0	0
F	0	0	0	0	0	0	27	0	0	0	0	0	0
G	0	0	0	0	0	0	0	33	0	0	0	0	0
H	0	0	0	0	0	0	0	0	29	0	0	0	0
I	0	0	0	0	0	0	0	0	0	27	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	23	0	0
J	0	0	0	0	0	0	0	0	0	0	0	30	0
K	0	0	0	0	0	0	0	0	0	0	0	0	35
L	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0

Şekil 54 Arayüz Tasarımı Karmaşıklık Matrisi

The screenshot shows a single window titled 'Karmaşıklık Matrisi' displaying a confusion matrix table with 26 rows and 17 columns, labeled A through T and Ç through K. The matrix values are identical to the one in Figure 54.

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K
A	28	0	0	0	0	0	0	0	0	0	0	0	0
B	0	32	0	0	0	0	0	0	0	0	0	0	0
C	0	0	29	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	35	0	0	0	0	0	0	0	0	0
D	0	0	0	0	36	0	0	0	0	0	0	0	0
E	0	0	0	0	0	35	0	0	0	0	0	0	0
F	0	0	0	0	0	0	27	0	0	0	0	0	0
G	0	0	0	0	0	0	0	33	0	0	0	0	0
H	0	0	0	0	0	0	0	0	29	0	0	0	0
I	0	0	0	0	0	0	0	0	0	27	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	23	0	0
J	0	0	0	0	0	0	0	0	0	0	0	30	0
K	0	0	0	0	0	0	0	0	0	0	0	0	35
L	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0

Şekil 53 Arayüz Tasarımı Karmaşıklık Matrisi Büyüğ

### **3.2.2.4 OFFLINE TEST SONUÇLARI**

<b>Algoritmalar</b>	<b>Öznitelik Sayısı</b>	<b>121 Öznitelik</b>
Random Forest	100%	
Navie Bayes	100%	
Neural Network	63%	
Support Vector Machine	100%	
Decision Tree	100%	

### 3.3. ONLINE TEST

#### 3.3.1 DOSYA YAPISI

	models	9.03.2018 14:47	Dosya klasörü
	Leap.dll	14.11.2017 02:48	Uygulama uzantısı 3.777 KB
	Leap.lib	14.11.2017 02:48	VisualStudio.lib.32... 126 KB
	Leap.py	14.11.2017 02:48	PY Dosyası 84 KB
	Leap.pyc	4.12.2017 11:44	PYC Dosyası 85 KB
	LeapCode.py	8.03.2018 20:10	PY Dosyası 9 KB
	LeapPython.pyd	14.11.2017 02:48	PYD Dosyası 431 KB

Şekil 55 Online Test Dosya Yapısı

**Leap.dll** : Leap Motion nun projede çalışabilmesi için gerekli sistem dosyası

**Leap.lib**: Leap Motion Kütüphanesi

**Leap.py** : Leap Motion çalışabilmesi için geliştiricileri tarafından eklenen python kodları

**LeapCode.py** : Leap Motion dan verileri alıp , işleyip , online olarak test ettiğimiz python kodları

**Models klasörü** : Önceden eğitilmiş ağ dosyalarının bulunduğu klasör.

#### 3.3.2 KODLAR VE AÇIKLAMALAR

LeapCode.py Eklenen kütüphaneler:

```
@author: Bedirhan
"""

import Leap,sys,thread,time
import time
from sklearn.externals import joblib
import numpy as np

import cv2, Leap, math, ctypes
```

Şekil 56 Online Test LeapCode.py Eklenen  
Kütüphaneler

LeapCode.py Online Test aşamasında Leap Motion kamerasının aktif olması için eklenen hazır fonksiyonlar:

- convert\_distortion\_maps(image)
- undistort(image, coordinate\_map, coefficient\_map, width, height)
- run(controller)

```
def convert_distortion_maps(image):  
  
    distortion_length = image.distortion_width * image.distortion_height  
    xmap = np.zeros(distortion_length/2, dtype=np.float32)  
    ymap = np.zeros(distortion_length/2, dtype=np.float32)  
  
    for i in range(0, distortion_length, 2):  
        xmap[distortion_length/2 - i/2 - 1] = image.distortion[i] * image.width  
        ymap[distortion_length/2 - i/2 - 1] = image.distortion[i + 1] * image.height  
  
    xmap = np.reshape(xmap, (image.distortion_height, image.distortion_width/2))  
    ymap = np.reshape(ymap, (image.distortion_height, image.distortion_width/2))  
  
    #resize the distortion map to equal desired destination image size  
    resized_xmap = cv2.resize(xmap,  
                               (image.width, image.height),  
                               0, 0,  
                               cv2.INTER_LINEAR)  
    resized_ymap = cv2.resize(ymap,  
                               (image.width, image.height),  
                               0, 0,  
                               cv2.INTER_LINEAR)  
  
    #Use faster fixed point maps  
    coordinate_map, interpolation_coefficients = cv2.convertMaps(resized_xmap,  
                                                               resized_ymap,  
                                                               cv2.CV_32FC1,  
                                                               nninterpolation = False)  
  
    return coordinate_map, interpolation_coefficients
```

Şekil 57 Online Test LeapCode.py convert\_distortion\_maps Fonksiyonu

```

def undistort(image, coordinate_map, coefficient_map, width, height):
    destination = np.empty((width, height), dtype = np.ubyte)

    #wrap image data in numpy array
    i_address = int(image.data_pointer)
    ctype_array_def = ctypes.c_ubyte * image.height * image.width
    # as ctypes array
    as_ctype_array = ctype_array_def.from_address(i_address)
    # as numpy array
    as_numpy_array = np.ctypeslib.as_array(as_ctype_array)
    img = np.reshape(as_numpy_array, (image.height, image.width))

    #remap image to destination
    destination = cv2.remap(img,
                           coordinate_map,
                           coefficient_map,
                           interpolation = cv2.INTER_LINEAR)

    #resize output to desired destination size
    destination = cv2.resize(destination,
                           (width, height),
                           0, 0,
                           cv2.INTER_LINEAR)
    return destination

def run(controller):
    maps_initialized = False
    while(True):
        frame = controller.frame()
        image = frame.images[0]
        if image.is_valid:
            if not maps_initialized:
                left_coordinates, left_coefficients = convert_distortion_maps(frame.images[0])
                right_coordinates, right_coefficients = convert_distortion_maps(frame.images[1])
                maps_initialized = True

            undistorted_left = undistort(image, left_coordinates, left_coefficients, 400, 400)
            undistorted_right = undistort(image, right_coordinates, right_coefficients, 400, 400)

            #display images
            cv2.imshow('Left Camera', undistorted_left)
            cv2.imshow('Right Camera', undistorted_right)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

```

Sekil 58 Online Test LeapCode.py undistort ve run fonksiyonu

### LeapCode.py LeapMotionListener Sınıfı:

- finger\_names: Parmak isimleri
- bona\_names : Parmaklardaki kemiklerin isimleri
- state\_names : Durumların isimleri
- on\_init : Leap Motion Yüklendiğinde yapılacak işlemler
- on\_connect: Leap Motion Bağlandığında yapılacak işlemler. Leap motion bağlandığında önceden eğitmiş olduğumuz models klasöründeki modelimizi yükliyoruz. Leap motionun hareketlerini set ediyoruz
- on\_disconnect: Leap Motion Bağlantısı kesildiğinde yapılacak işlemler.

```
class LeapMotionListener(Leap.Listener):
    finger_names=['Thumbs','Index','Middle','Ring','Pinky']
    bone_names=['Metacarpal','Proximal','Intermediate','Distal']
    state_names=['STATE_INVALID','STATE_START','STATE_UPDATE','STATE_END']

    def on_init(self,controller):
        print "Yuklendi"

    def on_connect(self,controller):
        print "Leap Motion Baglandi"
        self.clf=joblib.load('./models/RandomForestModel.pkl')
        controller.enable_gesture(Leap.Gesture.TYPE_CIRCLE);
        controller.enable_gesture(Leap.Gesture.TYPE_KEY_TAP);
        controller.enable_gesture(Leap.Gesture.TYPE_SCREEN_TAP);
        controller.enable_gesture(Leap.Gesture.TYPE_SWIPE);

    def on_disconnect(self,controller):
        print "Leap Motion baglantisi kesildi"
```

Şekil 59 Online Test LeapCode.py LeapMotionListener Sınıfı

### LeapCode.py LeapMotionListener on\_frame Fonksiyonu:

Burada Leap Motion cihazından çerçeve bilgileri alınır. Alınan bilgiler analiz edilerek oluşturulan dizi de doğru yerlere eklenir. Oluşturulan dizi de Leap Motiona gösterdiğimiz el hareketinin pozisyonları bulunur. On connect fonksiyonunda önceden eğitilmiş olan ağ ile oluşturulan dizi test edilir. Test sonuçları beklenen sonuçlarla karşılaştırılır ve karşılaştırma sonunda tahmin edilen etiket numarası ekrana bastırılır.

```
def on_frame(self, controller):
    frame=controller.frame()
    data=[]
    for i in range (0,390):
        data.append(0)

    now_data=[]

    for i,hand in enumerate(frame.hands):
        handTypedata=0 if hand.is_left else 1
        normal= hand.palm_normal
        direction=hand.direction
        arm=hand.arm

        now_data=str(hand.palm_position.x),str(hand.palm_position.y) , str(hand.palm_position.z),\
        str(direction.pitch* Leap.RAD_TO_DEG),str(normal.roll*Leap.RAD_TO_DEG),str(direction.yaw*Leap.RAD_TO_DEG) , \
        str(arm.direction.x),str(arm.direction.y),str(arm.direction.z) ,\
        str(arm.wrist_position.x),str(arm.wrist_position.y),str(arm.wrist_position.z), \
        str(arm.elbow_position.x),str(arm.elbow_position.y),str(arm.elbow_position.z),

        for finger in hand.fingers:
            for b in range(0,4):
                bone=finger.bone(b)
                now_data+=str(bone.prev_joint.x),str(bone.prev_joint.y),str(bone.prev_joint.z), \
                str(bone.next_joint.x) , str(bone.next_joint.y), str(bone.next_joint.z) , \
                str(bone.direction.x) , str(bone.direction.y) , str(bone.direction.z),

        if(handTypedata==0 and i==0):#"1.el ve sol"
            for x in range(0,195):
                data[x]=now_data[x]
            for y in range (195,390):
                data[y]=0

        elif (handTypedata==1 and i==0): #1.el ve sağ a
            for x in range(0,195):
                data[x]=0
            for y in range (195,390):
                data[y]=now_data[y-195]

        elif (handTypedata==0 and i==1): # 2.el ve sol ccc
            for x in range(0,195):
                data[x]=now_data[x]

        elif (handTypedata==1 and i==1): #2.el ve sağ ccc
            for y in range (195,390):
                data[y]=now_data[y-195]
```

Şekil 60 Online Test LeapCode.py Leap Motion Listener on\_frame fonksiyonu birinci bölüm

```

if data[389]==0 and data[0]==0:
    pass
else:
    data=np.array(data).reshape(-1,1).T
    result=self.clf.predict(data)
    lbl_text=""
    if result==1:
        lbl_text+="A"
    elif result==2:
        lbl_text+="B"
    elif result==3:
        lbl_text+="C"
    elif result==4:
        lbl_text+="Ç"
    elif result==5:
        lbl_text+="D"
    elif result==6:
        lbl_text+="E"
    elif result==7:
        lbl_text+="F"
    elif result==8:
        lbl_text+="G"
    elif result==9:
        lbl_text+="Ğ"
    elif result==10:
        lbl_text+="H"
    elif result==11:
        lbl_text+="I"
    elif result==12:
        lbl_text+="İ"
    elif result==13:
        lbl_text+="J"
    elif result==14:
        lbl_text+="K"
    elif result==15:
        lbl_text+="L"
    elif result==16:
        lbl_text+="M"
    elif result==17:
        lbl_text+="N"
    elif result==18:
        lbl_text+="O"
    elif result==19:
        lbl_text+="Ö"
    elif result==20:
        lbl_text+="P"
    elif result==21:
        lbl_text+="R"
    elif result==22:
        lbl_text+="S"
    elif result==23:
        lbl_text+="Ş"
    elif result==24:
        lbl_text+="T"
    elif result==25:
        lbl_text+="U"
    elif result==26:
        lbl_text+="Ü"
    elif result==27:
        lbl_text+="V"
    elif result==28:
        lbl_text+="Y"
    elif result==29:
        lbl_text+="Z"
    print(lbl_text)
time.sleep(1)

```

*Sekil 61 Online Test LeapCode.py Leap Motion Listener on\_frame fonksiyonu ikinci bölüm*

LeapCode.py Main Fonksiyonu:

Oluşturduğumu Listener sınıfından bir örnek alınır. Geliştiricilerin oluşturduğu Leap Kütüphanesinden Controller nesnesi oluşturulur. Leap Motionun çalışabilmesi için controller set edilir. Leap Motion bizim kodladığımız gibi çalışın diye controllera yazdığımız listener sınıfı eklenir.

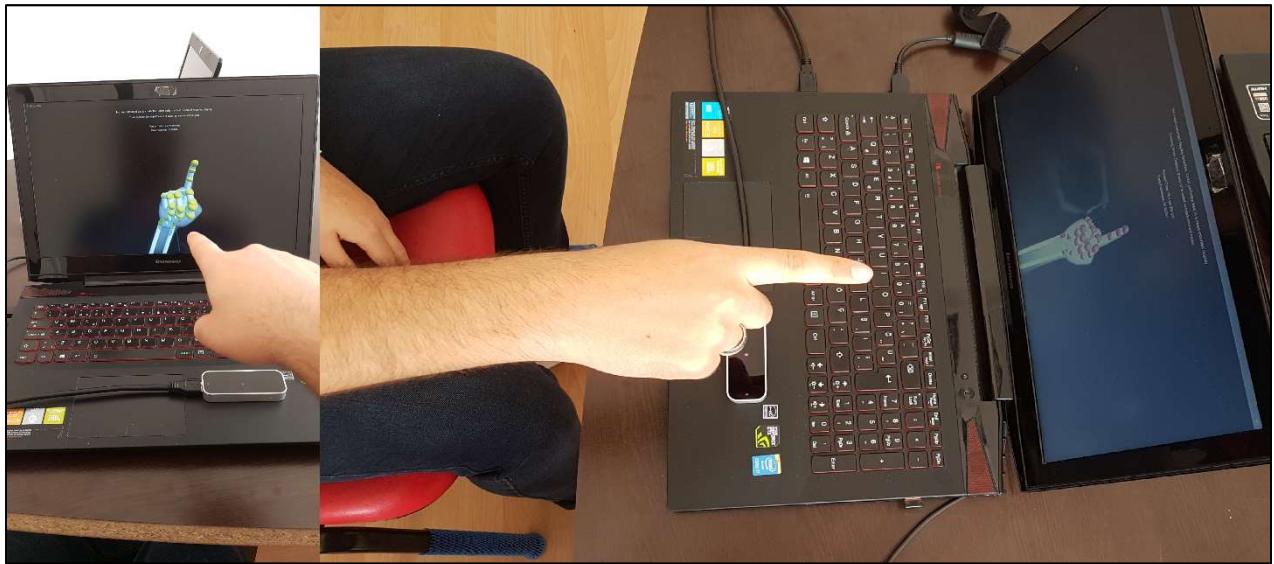
```
def main():
    listener=LeapMotionListener()
    controller=Leap.Controller()
    controller.set_policy_flags(Leap.Controller.POLICY_IMAGES)
    controller.add_listener(listener)
    print "Press enter to quit"
    try:
        run(controller)
    except KeyboardInterrupt:
        sys.exit(0)

    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        controller.remove_listener(listener)

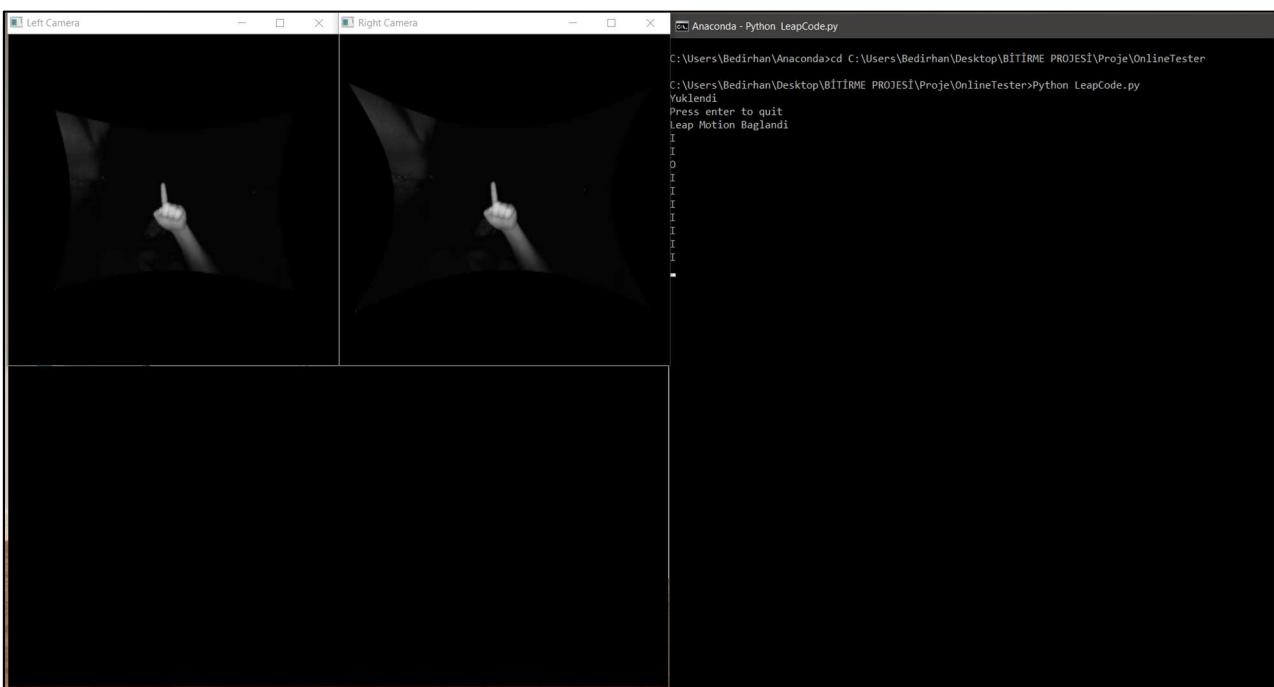
if __name__=="__main__":
    main()
```

Şekil 62 Online Test LeapCode.py Main Fonksiyonu

### 3.3.3 ÇALIŞMA ORTAMI VE ARAYÜZ



Şekil 63 Online Test Çalışma Ortamı



Şekil 64 Online Test Arayüz Tasarımı

## **4. BULGULAR VE SONUÇLAR**

### **4.1 391 ÖZNİTELİKLİ VERİ SETİNİN BAŞARIM ORANLARI VE KARMAŞIKLIK MATRİSİ**

#### **4.1.1 BAŞARIM ORANLARI**

Algoritmalar	Öznitelik Sayısı	391 Öznitelik
Random Forest		100%
Navie Bayes		100%
Neural Network		23%
Support Vector Machine		100%
Decision Tree		99%

#### 4.1.2 RASSAL ORMAN KARMAŞIKLIK MATRİSİ

	<b>A</b>	<b>B</b>	<b>C</b>	<b>Ç</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>İ</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>	<b>N</b>	<b>O</b>	<b>Ö</b>	<b>P</b>	<b>R</b>	<b>S</b>	<b>Ş</b>	<b>T</b>	<b>U</b>	<b>Ü</b>	<b>V</b>	<b>Y</b>	<b>Z</b>
<b>A</b>	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>B</b>	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>C</b>	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>Ç</b>	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>D</b>	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>E</b>	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>F</b>	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>G</b>	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>H</b>	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>I</b>	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>İ</b>	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>J</b>	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>K</b>	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>L</b>	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>M</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>N</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>O</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
<b>Ö</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0
<b>P</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0
<b>R</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0
<b>S</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0
<b>Ş</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0
<b>T</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0
<b>U</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0
<b>Ü</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0
<b>V</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0
<b>Y</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0
<b>Z</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31

### 4.1.3 DESTEK VEKTÖR MAKİNESİ KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Ç	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
E	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
İ	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
J	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
K	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
L	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	

#### 4.1.4 YAPAY SİNİR AĞLARI KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	9	30	0	0	0	0	20	31	0	0	0	0	0	0	0	0	0	20	0	31	18	0	14	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16
F	0	0	0	0	0	0	4	0	0	0	31	0	0	0	0	0	0	1	0	0	8	0	11	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	15	0	27	0	0	0	0	0	0	24	0	0	27	42	0	0	28	0	0	0	0	0	0	0	0	34	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	12	0	0	8	0	18	0	0	0	0	0	0	0	0	0	34	27	0	0	0	0	0	0	0	0	0	23	14
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	24	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ü	0	0	0	28	33	10	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	31	0	11	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### 4.1.5 KARAR AĞAÇLARI KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	
A	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Ç	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
E	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
İ	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
J	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
K	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
L	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	
Ş	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	

#### 4.1.6 NAVİE BAYES KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Ç	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
E	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
İ	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
J	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
K	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
L	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
M	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	

## **4.2 121 ÖZNİTELİKLİ VERİ SETİNİN BAŞARIM ORANLARI VE KARMAŞIKLIK MATRİSİ**

### **4.1.1 BAŞARIM ORANLARI**

<b>Algoritmalar</b>	<b>Öznitelik Sayısı</b>	<b>121 Öznitelik</b>
Random Forest		100%
Navie Bayes		100%
Neural Network		63%
Support Vector Machine		100%
Decision Tree		100%

#### 4.2.2 RASSAL ORMAN KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Ç	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
E	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
F	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
H	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
İ	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
J	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
K	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
L	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
M	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	

### 4.2.3 DESTEK VEKTÖR MAKİNESİ KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27

#### 4.2.4 YAPAY SİNİR AĞLARI KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	24	0	0	0	0	8	0	0	0	0	0	0	29	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0
Ç	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	36	0	0	0
D	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
E	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	18	0	0	0	0	0	0	0	0	0	30	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	30	0	0	6	0	0	0	0	0	0	0	0	24	26	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24

#### 4.2.5 KARAR AĞAÇLARI KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
İ	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28

#### 4.2.6 NAVİE BAYES KARMAŞIKLIK MATRİSİ

	A	B	C	Ç	D	E	F	G	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
C	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ç	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
İ	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
Ö	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0
Ş	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0
Ü	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33

### **4.3 DEĞERLENDİRME**

Türk dil alfabetesinden ‘Ğ’ harfi hariç toplam 28 harf ve 5 farklı makine öğrenmesi algoritmalarından oluşturulan modeller ile gerçek zamanlı ve çevrimdışı testler gerçekleştirılmıştır. Bu testler için oluşturulan veri setlerinin; %70’ i eğitim(train) ve %30’ u test verisi olarak ayrılmıştır.

Cevrimdışı testler için karmaşıklık matrisleri ve başarım oranları hesaplanmıştır. 28 harf için 121 ve 391 öznitelik sayısına göre oluşturulan veri setlerine 5 farklı makine öğrenmesi algoritmaları uygulandığından her bir algoritma için bir karmaşıklık matrisi ve başarım oranı elde edilmektedir. 121 öznitelik sayısı, 5 farklı algoritma için 5 tane karmaşıklık matrisi ve başarım oranı elde edilmişdir. 391 öznitelik sayısı için de 5 tane karmaşıklık matrisi ve başarım oranı elde edilir. 28 harf için de hesaplamalar yapıldığında 5 algoritma için 121 ve 391 öznitelik sayısına göre toplam 10 tane karmaşıklık matrisi ve başarım oranı elde edilir.

Bu test ortamında 391 ve 121 olmak üzere iki farklı şekilde elde edilen özniteliklerin sayısına göre oluşturulan veri setlerinin karmaşıklık matrisleri ve başarım oranları hesaplanmıştır. En kötü başarımı sırasıyla %23 ve %63 olmak üzere Yapay Sinir Ağları (Neural Network) makine öğrenmesi algoritması vermektedir. En iyi başarımı veren üç farklı algoritma bulunuyor bunlar sırasıyla %100 ve %100 ile Rassal Orman , %100 ve %100 ile Destek Vektör Makineleri ve %100 ve %100 ile Navie Bayes Algoritmaları vermiştir. Bunları takip eden en başarılı algoritma ise %99 ve %100 ile Karar Ağaçları olmuştur.

Gerçek zamanlı testlerde tek el ile yapılan statik hareketlerde gösterdiği başarayı çift el ile yapılan dinamik ve statik hareketlerde gösteremediği için gerçek zamanlı testlerin sonuçları çevrimdışı testlerdeki kadar etkili olmamıştır. Bu nedenle Türk işaret dilinin tek bir Leap Motion kullanılarak tanımlanması mümkün olmamaktadır. Gerçek zamanlı testlerin başarılı olabilmesi için iki adet leap motiondan veriler alınarak işlenmesi veya leap motion sensörün başka bir kamera sistemi ile desteklenmesi gerekmektedir.

Sonuç olarak Leap Motion ile Türk İşaret Dili Tanımlama projesi çevrimdışı testlerde gösterdiği başarı sayesinde teorik olarak mümkündür fakat gerçek zamanlı testlerde tek bir leap motion kullanılarak mümkün gözükmemektedir. Sistem tasarıımı değiştirilerek gerçek zamanlı testlerde büyük başarım oranlı elde etmek mümkündür.

## KAYNAKÇA

- Supervised learning.* (tarih yok). <http://scikit-learn.org>: [http://scikit-learn.org/stable/supervised\\_learning.html#supervised-learning](http://scikit-learn.org/stable/supervised_learning.html#supervised-learning) adresinden alındı
- Akarsu, H. B. (2017, Kasım 5). *Karar Ağacı Öğrenmesi(Decision Tree Learning)*. [yazilimagiris.com](http://yazilimagiris.com): <http://yazilimagiris.com/2017/11/karar-agaci/> adresinden alındı
- Basma Hisham, D. A. (2017). Arabic Static and Dynamic Gestures. *Journal of Computer Science*, 338-354.
- Camera Images.* (tarih yok). [developer.leapmotion.com](http://developer.leapmotion.com): [https://developer.leapmotion.com/documentation/python/devguide/Leap\\_Images.html](https://developer.leapmotion.com/documentation/python/devguide/Leap_Images.html) adresinden alındı
- Fazlur Rahman Khan, H. F. (2016). A Sign Language to Text Converter Using Leap Motion. *International Journal on Advance Science Engineering Information Technology*, 1089-1095.
- Fidancı, A. S. (2017, Ocak 23). *Random Forest Algoritması*. [www.slideshare.net](http://www.slideshare.net): <https://www.slideshare.net/SezerFidanc/random-forest-algoritmas> adresinden alındı
- Filiz, F. (2017, Temmuz 23). *4.1.1 Yapay Sinir Ağları*. Medium.com: <https://medium.com/@fahrettinf/4-1-1-yapay-sinir-a%C4%9Flar%C4%B1-86f0440f85c2> adresinden alındı
- Leap Motion Controller Specs.* (2013, Temmuz 22). cnet.com: <https://www.cnet.com/products/leap-motion-controller/specs/> adresinden alındı
- Makine Öğrenimi Bölüm-4 (Destek Vektör Makineleri).* (2017, Kasım 30). Medium.com: <https://medium.com/@k.ulgen90/makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-4-destek-vekt%C3%B6r-makineleri-2f8010824054> adresinden alındı
- Miada A. Almasre, H. A.-N. (2016). A Real-Time Letter Recognition Model for. *International Journal of Advanced Engineering, Management and Science (IJAEMS)*, 514-523.
- Python SDK Documentation.* (tarih yok). [developer.leapmotion.com](http://developer.leapmotion.com): <https://developer.leapmotion.com/documentation/python/index.html> adresinden alındı
- Usta, R. (2014, Mayıs 28). *Naïve Bayes Sınıflandırma Algoritması*. [kodedu.com](http://kodedu.com): <https://kodedu.com/2014/05/naive-bayes-siniflandirma-algoritmasi/> adresinden alındı