ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4061 PROJE RAPORU

İNSAN YÜRÜYÜŞ STİLLERİNE GÖRE KİMLİK DOĞRULAMA

ERENCAN TEKİN

19290273

DOÇ. DR. BÜLENT TUĞRUL

15.01.2024

ÖZET

Her insanın kendine özgü bir yürüyüş stili vardır. Bazı insanlar kollarını çok az açar, ayaklarını sürüyerek yürürler, dengesiz bir şekilde yürürler veya sakin bir şekilde yürürken bazı insanlar ise tam tersidir, kollarını büyük bir açıyla açar, adımlarını büyük atarlar veya sinirli bir şekilde boyunlarını eğerek yürürler atarlar. Bunu 60 yaşında olgun bir kişinin yürüme şekli ile 25 yaşındaki bir atletin yürüme stillerindeki arasındaki farklılıkta görebiliriz. Bitirme projemde ise geleneksel yollarla, yani yüz tanıma, retina tanıma gibi tespitler, değil daha yenilikçi ve farklı bir şekilde kimlik doğrulaması yapmaya çalıştım. Her insana özgü olan yürüyüş stillerindeki farklılıkları kullanarak ve görüntü işleme (computer vision) teknikleriyle kombine ederek insan kimlik doğrulaması işlemlerinde kullandım. Bu süreçte geniş kütüphanesi, makine öğrenmesinde yaygın olarak kullanılması ve kolay sözdizimine (syntax) sahip olmasıyla bilinen programlama dili olan “ Python “ programlama dilini kullandım. Projemde temel nokta insanların yürürken ki baş, el, kol, bacak, gövde, postür duruş ve hareket pozisyon koordinatlarını alarak eğitiyorum ve daha sonra gerekli işlemlerden geçirerek teste uygun hale getiriyorum. Bu koordinatlar aynı zamanda birbiri arasındaki açısal ilişkiyi de barındırdığı için örneğin yürürken adımını büyük atan ya da küçük atan kişileri rahatlıkla ayırt ederek başarıyla kimlik doğrulama gerçekleştiriyorum.

İÇİNDEKİLER

ÖZET……..……..i
İÇİNDEKİLER……..……...ii
1. GİRİŞ……...1
2. PROJEYE HAZIRLIK……..……..……2
2.1. Jupyter Notebook……..……2
2.2. Python Programlama Dili……...2
2.3. İnsan Hareket Analizleri……3
2.4. Kimlik Analizi (Identity Analysis)……3
2.5. Kimlik Doğrulama (Authentication)……...3
3. PROJE ÇALIŞMALARI……..……5
3.1. Gerekli Kütüphanelerin Seçimi ve Kullanım Alanları……5
3.1.1. OpenCV (Açık Kaynak Bilgisayar Görüntüsü Kütüphanesi)……..5
3.1.2. Mediapipe……...5
3.1.3. Pandas……...6
3.1.4. Scikit-learn……6
3.1.5. Make_pipeline ve StandardScaler……6
3.1.6. Pickle……...6
3.2. Görüntünün Yakalanması ve İşlenmesi……..6
3.3. Modelin Eğitilmesi (Training)……9
3.4. Performans Değerlendirmesi ve Doğruluk (Accuracy) Hesabı……..12
3.5. Modelin Test Edilmesi (Testing)……13
4. SONUÇ……15
5. KAYNAKÇA…..…… .17

1. GİRİŞ

Görüntü işleme, diğer bir adıyla “ Computer Vision “, dijital görüntüye çeşitli algoritmalar ve tekniklerin uygulanarak faydalı bilgilerin çıkarılması ve işlenmesi sürecidir.

Görüntü işleme hayatın her alanında kullanılabilir, örneğin nesne tespiti, nesne takibi, postür tespiti/analizi diğer adlarıyla “ object detection “, “ object tracing ”, “ pose detection “. Her insanın kendine has yürüyüş stili olduğu için bu özelliği kullanarak insanları tespit etmek istedim. Yapmış olduğum bu bitirme projemde de hareket halindeki insanların postür analizinden yararlanarak kimliklerini tespit ettim. Genel olarak anlatmam gerekirse insanları hareket halindeyken her an pozisyonlarını bir CSV uzantılı dosyaya kaydettim. CSV uzantılı dosya kullanmamın nedeni boyut olarak düşük ve esnek bir yapıda olmasıdır. Daha sonra bu kaydettiğim verilerin pozisyonlarını CSV uzantılı dosyamda belirlediğim sınıflara (class) atadım ve eğittim (training). En sonunda ise test videolarım ile projemi test ederek sonucu gördüm. Python programlama dilini kullandım çünkü Python, geniş bir kütüphaneye sahip, yardımsever bir topluluğu var, taşınabilir yani sadece bir platforma özgü değildir, birden fazla platform üzerinde çalışabilmektedir.

Platform olarak “ Jupyter Notebook ” kullanmayı seçtim. Hücreler halinde çalışmayı sağladığı için oldukça verimli bir platformdur. Daha da önemlisi NumPy, Pandas, MatplotLib, Sklearn, OpenCV gibi çok çeşitli kütüphaneleri rahatça ve kolaylıkla uygun versiyonda kullanmamızı sağlamaktadır. Bu sayede hem kod kolaylığı sağlarken hem de alan tasarrufu sağlamaktadır.

2. PROJEYE HAZIRLIK

2.1. Jupyter Notebook

Jupyter Notebook kod ve alan tasarrufu sağlayan bir Project Jupyter platformunun parçasıdır. Hücre tabanlı yapısı sayesinde kullanıcıların kodu parçalar halinde inceleyerek çalıştırabilmesine olanak sağlamaktadır. Bu durum ise okunabilirliği ve kod esnekliği arttırmaktadır. Hata ayıklamayı kolaylaştırır ve hızlı bir şekilde ilerlemeye katkıda bulunur. Aynı zamanda kullanmak istemediğimiz kod hücrelerini atlamamıza olanak sağlamaktadır. Diğer bir artısı ise NumPy, Pandas, Scikit gibi kütüphaneleri uygun versiyonlarla kolaylıkla kullanabilmemize (“import“ edebilmemize) olanak sağlamaktadır. Aynı zamanda taşınabilir (portable) bir uygulamadır, diğer bir deyişle bilgisayara herhangi bir kurulum gerektirmeyen çalıştırılabilen bir uygulamadır.

2.2. Python Programlama Dili

Kodumu yazarken “ Python “ programlama dilini tercih ettim çünkü geniş bir kütüphaneye sahiptir. Özellikle görüntü işleme ve makine öğrenmesinde kullanılabilecek OpenCV, Scikit gibi önemli kütüphanelere sahiptir.

Kolay bir sözdizimine (“ syntax “) sahiptir. Dinamik değişkenlere izin verir bu da kod esnekliğini sağlamaktadır. Oldukça açık ve sade yapısı sayesinde okunabilirliği arttırmakta ve ileride yapılacak modifikasyon işlemleri için daha rahat bir yapıdadır. Bu şekilde model oluşturma ve CSV dosyalarının modifiyesi çok daha rahat bir şekilde gerçekleşmektedir.

Oldukça büyük yardımsever bir topluluğa sahiptir bu sayede herhangi bir sorun ile karşılaşıldığında kolaylıkla bu sorun halledilebilmektedir.

Diğer bir özelliği ise interpreter (“ yorumlayıcı “) bir dildir. Kod zaman kaybı olmaksızın anında çalıştırılır, ve bu sayede hatalar daha hızlı bir şekilde tespit edilebilir. Kodlar platformdan bağımsız olarak yürütülebilir.

Python, makine öğrenmesi ve özellikle de görüntü işleme alanlarında oldukça güçlü kütüphaneler sunar ve interpreter bir dildir. Bu nedenle projemde kullanmayı tercih ettim.

2.3. İnsan Hareket Analizleri

İnsan hareket analizi, bir görüntüden elde edilen verilerin işlenerek insanların fiziksel hareket durumlarının anlaşılması ve çözümlenmesidir. Her insan hız olarak, yürürken ki el-kol hareketi olarak veya yürürken ki bacak hareketi olarak birbirinden ayrılabilir. Her insanın yürüyüş stili farklı olduğu ve kişiye özgü karakteristik bir özellik olduğu için bu özellikler kullanılarak kişiler birbirlerinden ayırt edilebilir.

2.4. Kimlik Analizi (Identity Analysis)

Bir cismin veya kişinin tanımlanmasını ve benzersiz özelliklerinin ortaya çıkarılarak incelenmesini içeren bir sistemdir. Kimlik analizi diğer bir ismiyle " Identity Analysis ", cisimlerin veya kişinin özelliklerinin (el izi, ses, göz retinası, fiziksel davranışları vb.) incelenmesine, anlaşılmasına olanak sağlar. Kimlik analizi, elde edilen verilerin önceki var olan verilerle karşılaştırılıp, yönetilip sınıflandırılmaları yoluyla gerçekleşir.

Günümüzde kimlik analizi, görüntü işlemeyle birlikte bireyin biyometrik verileri alındıktan sonra çeşitli aşamalardan (yüz tanıma, retinal tarama vb.) geçirilerek yapılmaktadır.

2.5. Kimlik Doğrulama (Authentication)

Bir kişi veya cismin kimliğinin, önceden tanımlanmış kimlikler göz önünde bulundurularak doğru olup olmadığını test eden bir sistem sürecidir. Bu süreç biyometrik veriler, postür koordinatları, yürüyüş sırasında el, kol, bacak hareket koordinatları, açıları vb. faktörler aracılığıyla gerçekleşmektedir. Bu süreçler sayesinde kişinin kim olduğu bulunmaktadır.

Kimlik analizi ve kimlik doğrulama ne kadar birbirine yakın iki terim olsa da aslında farklı iki terimdir. Kimlik analizi, veri setindeki bireyleri tanımlama amacındayken kimlik doğrulama bireyin hangi veri ile eşleşip eşleşmediğini bulmaktadır.

Projemde de görüntü işlemeyi kullanarak ilk önce yürüyüş halindeki bireyleri tespit ediyor ve yürüyüşleri boyunca koordinatlarını kaydediyor ve daha sonra test aşamasında bu gerçek zamanlı olarak görüntüdeki yürüyüş stilinin hangi sınıfa ait olduğunu olasılıklarıyla beraber bulmuş oluyorum.

3. PROJE ÇALIŞMALARI

3.1. Gerekli Kütüphanelerin Seçimi ve Kullanım Alanları

Görüntü işlemede (Computer Vision) önemli olan görüntünün analiz edilerek üzerlerinde işlem yapılabilinmesinin sağlanılmasıdır. Görüntünün yakalanmasında ve dosya modifikasyon işlemlerinde oldukça kullanışlı ve esnek bir programlama dili olduğu için Python programlama dilini kullanmayı seçtim. Bu esnekliği ve avantajları kullanmak doğru kütüphaneyi seçmek ile olur. Kütüphane (Library), bir alanda özelleşmiş, belirli bir görevi getirmek için bir araya gelen kod parçalarının birleşmesiyle oluşan paketler (modül) demektir.

3.1.1. OpenCV (Açık Kaynak Bilgisayar Görüntüsü Kütüphanesi)

İnsan hareketlerini algılamak ve görüntü üzerinde modifikasyonlar yapmak için kullanılan kullanışlı bir Python kütüphanesidir.

Sunduğu geniş işlevsellik, görüntüde yakalamada cisimlerin tespiti gibi özellikleri sayesinde projemde kullandım, çünkü modelimi eğitmek için ilk önce insanların hareket halindeki duruşlarını hareketlerini algılayıp koordinatlarını görüntüden çekmem gerekti.

3.1.2. Mediapipe

El, yüz, vücut, postür gibi vücut parçalarını algılayabilen ve önceden eğitilmiş modelleri olan bir kütüphanedir. İnsan hareketlerini algılama, bu hareketler sırasında hareketli vücut kısımlarının koordinatların, postürlerin yakalanmasını ve üzerlerinde modifikasyon işlemlerinin uygulanabilinmesini üzerine yoğunlaşılmış bir kütüphanedir.

Kullanımı kolay olan ve geniş kullanım alanı olan çok avantajlı bir kütüphanedir. İnsan hareketleri üzerinde tanımlama ve analiz işlemleri için oldukça başarılı sonuçlar verdiğinden projemde kullandım.

3.1.3. Pandas

Veri analizi ve veri üzerinde modifikasyon uygulamaya elverişli bir kütüphanedir. Görüntü işlemeden elde ettiğim verileri uygun bir şekilde bir dizi (array) içerisinde tutmamı sağladığı ve bu diziler içerisinde rahatça modifiye işlemleri uygulayarak dizileri eğitime (training) hazır, istediğim biçime getirebildiğim için kullandım. Aynı zamanda bu düzenlediğim dizilerin verilerini, yani koordinatlarını, CSV dosyalarına da eklememi ve gerektiği zaman CSV uzantılı dosyalardan çekmemi sağladığı için kullanmayı tercih ettiğim bir kütüphanedir.

3.1.4. Scikit-learn

Makine öğrenmesi için özelleşmiş algoritmalar içeren bir kütüphanedir. Modelimi eğitmek ve veriler üzerinde düzenlemeler yapmak için kullanacağım fonksiyonları bu kütüphaneden aldım.

3.1.5. Make_pipeline ve StandardScaler

Bu modüllerin sağladığı fonksiyonlar aracılığıyla eğittiğim modele özgü bir hat (pipeline) oluşturabildim ve “ StandarScaler “ modülüm sayesinde veri setlerimi (data set) standart hale getirdim (standardization).

3.1.6. Pickle

Nesneleri daha düzenli hale getirmek için, yani seri hale, ya da tam tersi seri haldeki veriyi açmak için kullandığım bir kütüphanedir. Bu sayede eğittim modeli daha sonra tekrar tekrar istediğim şekilde kullanabilmekteyim.

3.2. Görüntünün Yakalanması ve İşlenmesi

Görüntünün yakalanması projenin yapıtaşlarından biridir. Görüntünün yakalanmasını OpenCV kütüphanesi kullanarak videodaki hareketleri ,örneğin yürüyen, koşan insanların el, kol, bacak, kafa hareketleri ve postürleri, yakaladım. Görüntüyü yakalamak için kullandığım kod şekil 3.1.’de verilmiştir.

```
import cv2
cap = cv2.VideoCapture(0)
```

Şekil 3.1. OpenCV kütüphanesiyle video akışı (Görüntü) alma

0, kişinin kendi kamerasının indeksidir. Tek kamerası varsa genellikle 0'dır. Eğer kamera değil de video oynatılmak istenirse 0 yerine videonun path'i verilmelidir.

Daha Sonra bu yakaladığım bu görüntüleri işlemek için Python programlama dilinin sağlamış olduğu bir diğer kütüphane olan " Mediapipe "ı kullandım. Daha detaylı bir analiz için Mediapipe kütüphanesinin " mholistic.Holistic " sınıfının fonksiyonlarını kullanmayı seçtim (şekil 3.2.). Bu sınıfın fonksiyonları sayesinde vücut, kol, el, bacak ve postür landmark'larını (bir cisim üzerinde analiz yapılmak istenen karakteristik noktalar) net bir şekilde analiz edebildim. Fonksiyonun aldığı 2 adet önemli parametre vardır. Bunlar " min_detection_confidence " ve " min_tracking_confidence "'dır.

İlk parametre olan " min_detection_confidence " parametresi bir landmark'ın (bir cisim üzerindeki karakteristik nokta) algılanabilinmesi için gereken mininmum eşik değeridir. Bu eşik değeri 0 ile 1 arasında bir değer alır. Ben kodumda bu eşik değerini daha iyi sonuçlar almak için 0.5 (%50) olarak seçtim. Yani burada eğer bir landmark'ın confidence değeri (güvenilirlik değeri) %50 altındaysa bu landmark algılanmayacaktır, ama eğer landmark değeri minimum %50 ve üzeriyse bu sefer algılanacaktır.

Fonksiyonumun aldığı ikinci parametre ise " min_tracking_confidence " parametresidir. Bu parametrenin birinci parametre olan " min_detection_confidence " parametresinden olan farkı bu parametresinin algılanan bir landmark'ın takip edilebilirlik eşik değerini belirtmesidir. Alabileceği değerler 0 ile 1 arasındadır. Kodumda bu eşik değerini 0.5 (% 50) olarak seçtim. Yani algılanan bir karakteristik noktanın takip edilebilinmesi için minimum %50 ve üzerinde bir güvenilirliğe sahip olması gerekmektedir.

Örnek kısım şekil 3.2.'de verilmiştir.

```
import mediapipe as mpipe

mholistic = mpipe.solutions.holistic
mdrawing = mpipe.solutions.drawing_utils

with mholistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    # Code
```

Şekil 3.2. Holistic modeli oluşumu

Okunan karakteristik karelerin tespit edildikten sonra görselleştirilmesi için " Mediapipe " kütüphanesinden yararlandım. Bu kütüphane sayesinde görüntüdeki yakalanan karakteristik noktalar (landmarks) birleştirilerek görselleştirilmektedir. Çizim için " mdrawing.draw_landmarks " özelleşmiş fonksiyonunu kullandım. Bu fonksiyon parametre olarak görüntü, landmark (karakteristik nokta) kümesi, bu landmark'lar arası çizgilerin çizilmesini sağlayan çizgiler, bağlantılar, ve noktaların çizilmesi için gerekli olan özellikler.

Örnek verilmek istenilirse şekil 3.3. gösterilebilir.

```
mdrawing.draw_landmarks(img, holisticImg.pose_landmarks, mholistic.POSE_CONNECTIONS,
                        mdrawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                        mdrawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                        )

mdrawing.draw_landmarks(img, holisticImg.right_hand_landmarks, mholistic.HAND_CONNECTIONS,
                        mdrawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                        mdrawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                        )
```

Şekil 3.3. Görüntüdeki "Pose Detection" ve "Right Hand Detection" tespitleri

Şekil 3.3.'de verilen " img " parametresi görüntü üzerinde çizim yapılabilinmesi için gereken görüntüyü temsil eder.

" holisticImg.right_hand_landmarks " parametresi sağ elin karakteristik noktalarını içeren veri kümesidir.

“ mholistic.HAND_CONNECTIONS ”, sol elin algılanmış karakteristik noktaları arasında çizim yapılabilmesi için bağlantıların tanımlandığı yerdir.

“ mdrawing.DrawingSpec() “, çizimi yapılacak noktaların özelliklerinin (noktaların rengi, kalınlığı ve çapı) verildiği yerdir.

Bu işlemler yapıldıktan sonra cv2.imshow() ile işlenmiş görüntü ekrana verilir. Eğer video ise tamamını değil istenilen yerde videonun sonlandırılması, görüntünün kesilmesi istenilirse cv2.waitKey() fonksiyonu kullanılır. Bu sayede istenilen tuşa basıldıktan sonra görüntü sonlanır (şekil 3.4.).

```
cv2.imshow('Webcam', image)

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

Şekil 3.4. Görüntünün ekrana verilmesi ve komut ile kapatılması

Bütün bu işlemler yapılıp görüntüdeki landmark’lar çıkarıldıktan sonra bir döngü içerisinde bu işlemler görüntü sonlanana kadar tekrar ve tekrar yapılır. Görüntüde her landmark yakalandığı an işlenerek ileriki aşamalarda yapılacak olan eğitim (training) ve test (testing) aşamaları için bir CSV dosyası içerisinde depolanır.

3.3. Modelin Eğitilmesi (Training)

Bu kısımda daha önceden verileri saklamış olduğum CSV dosyasından tekrar okuyorum ve verilerimi “ feature “ ve “ target value “ olmak üzere iki kısıma ayırıyorum. Feature veri setimde CSV dosyasında saklamış olduğum verilerimden “ class “ sütununu atarak geriye kalan tüm kısımları , yani koordinatların tutulduğu kısımlar, kullanıyorum. Targer veri setimde ise sadece “ class “ sütununun verilerini tutmaktayım. Daha sonra ise bu ayırmış olduğum feature ve target veri setlerimi kullanarak veri setimi eğitim ve test setleri olmak üzere parçalıyorum (şekil 3.5.).

```
featureVals = df.drop('class', axis=1) # features
targetVals = df['class'] # target value

train_featureVals, test_featureVals, train_targetVals, test_targetVals = train_test_split(featureVals, targetVals, test_size=0.3, random_state=1000)
```

Şekil 3.5. Verilerin eğitim (training) ve test (testing) setleri olarak ikiye bölünmesi

Bölme işlemi için train_test_split() fonksiyonunu kullandım. Burada “ test_size “ parametresini 0.3 belirleyerek verilerimin %30’unu test seti olarak ayıracağımı belirtmiş oldum. Geriye kalan %70 ise eğitim (training) veri seti olacaktır. En sonda bulunan random_state parametresi ise işlemin tekrarlanabilirliğini sağlamaktadır.

Bu kısımda verilerimi başarıyla makine öğrenmesi (machine learning) kullanarak eğitim (training) ve test olmak üzere ayırmış oldum. Eğitim setinden modelerin parametrelerini öğrenirker, test setinden modelin performansını deneyip tespit etmeye hazır hale getirmiş oldum.

Bir sonraki adımda ise artık veri setlerimi standartlaştırıp sınıflandırmaya (classification) başladım. Standartlaştırma için “ StandardScaler “ Python kütüphanesini kullandım. Sınıflandırma yapmak için ise oluşturduğum boru hatlarını (pipeline) kullandım.

Bunlar;

- LogisticRegression
- RidgeClassifier
- RandomForestClassifier
- GradientBoostingClassifier

olmak üzere 4 adettir.

Logistic Regression, hızlı ve özellikel çok sınıflı (multiclass) modellerde oldukça hızlı işlem yapabilmektedir.

Ridge Classifier, oldukça fazla sayıda özellik vektörü içeren modül üzerinde çalışılıyorsa modeli overfitting'den (aşırı öğrenme) korumak için kullanılır.

Random Forest Classifier, çok karmaşık veri setlerinde veya çok özellik vektörü olan veri setlerinde karmaşık ilişkileri kolaylıkla öğrenebilir. Aynı zamanda boş gelen veriyi (data) programın hata vermemesi için yok sayabilir, görüntüdeki gürültüyü minimalize edebilir.

Gradient Boosting Classifier, gürültüye dayanıklı ve çok verili modellerde yüksek performans gösterebilir. Aynı zamanda az veri setli modellerde de yüksek performans sergileyebilir.

```
pipelines = {
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),
    'lr': make_pipeline(StandardScaler(), LogisticRegression(solver='saga', max_iter=5000)),
```

Şekil 3.6. Boru Hatları (Pipelines)

Şekil 3.6. 'da boru hatları verilmiştir. Burada Logistic Regression içerisinde "solver" ve "max_iter" parametreleriyle çağırılmıştır. Veri seti çok büyük olduğunda Logistic Regression hiçbir parametre kullanılmadan çağırıldığında hata verdiği için "solver" ve "max_iter" parametreleri belirlenerek çağırılmıştır. "saga" çözücüsü (solver) büyük data setleriyle çalışabilmek ve "max_iter" ile de maksimum iterasyon sayısı belirtilmiştir. Çözümleyici olarak "saga" kullanmamın nedeni çok fazla parametre ve sınıf olduğunda diğer çözümleyicilere kıyasla daha hızlı çözümlemesi ve konveks olmayan problemlerde daha iyi optimizasyon yapmasıdır. Bu şekilde büyük veri setleri hatasız bir şekilde modifiye edilebilir.

Boru hatlarımı tanımladıktan sonra veri setlerimi, bir döngü içerisinden her bir boru hattından (pipeline) geçirerek modellerimi eğitmiş oldum.

3.4. Performans Değerlendirmesi ve Doğruluk (Accuracy) Hesabı

Bu kısımda eğitilmiş modellerin performansını değerlendirdim ve “ accuracy_score “ fonksiyonuyla da modelin yaptığı tahminlerle gerçek etiketlerin (label) eşleşip eşleşmediğini hesaplayarak doğruluk oranı hesabı yaptım.

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

Doğruluk oranları sonucunda tuttuğum modellerden “ Random Forest “ modelini bir sonraki test aşamasında kullanmak amacıyla “ PKL “ uzantılı bir dosyaya kaydettim.

Random Forest Modelini seçmemdeki nedenler arasında;

- Çok Sayıda Özellik ile Çalışma Esnekliği
  Modelimdeki çok fazla sayıda koordinat yakaladığım ve sakladığım için bu koordinatları göz önünde bulundurarak eğitirken (training) ya da test ederken iyi sonuçlar çıkarıyor.
- Robustluk ve Yüksek Performans
  Güçlü genelleme yeteneği sayesinde büyük ve karmaşık veri setleri üzerinde çalışırken doğru tahminler yapabiliyor.
- Aşırı Uyuma Karşı Dirençlilik
  Modelin daha sonra eklenecek olan yeni veri setlerine daha iyi uyum sağlaması ve genelleme yapabilmesi için modeli ezberlemeyi, overfitting, engeller.

sayılabilir.

Modelimi PKL uzantılı dosyaya saklamamın nedeni PKL uzantılı dosya Python programlama dilinin “ Pickle ” modeli tarafından kullanılan bir dosya uzantısıdır. Bu modül sayesinde modülümü serileştirebilir ve daha sonra okumak istediğimde deserileştirebilirim, test aşamasında (testing) bunu yapacağım.

3.5. Modelin Test Edilmesi (Testing)

Bu bölüm daha önce eğitilmiş (training) veri setlerimi kullanarak test işlemlerini gerçekleştirdiğim bölümdür.

İlk olarak " pickle " Python modülü sayesinde bir önceki adımda modelimi kaydettiğim .pkl uzantılı dosyadan modelimi çekiyorum. Daha sonra " OpenCV " kütüphanesini kullanarak video akışını başlatıyorum.

" cap.read() " fonksiyonuyla görüntüdeki her bir kareyi döngü içerisinde okuyorum. Daha sonra ise kareyi BGR renk uzayından RGB renk uzayına dönüştürüyorum. Bunu yapmamın nedeni MediaPipe kütüphanesinin Holistic modülünün RGB formatındaki görüntüleri daha iyi algılayıp daha iyi sonuçlar üretmesidir. Ardından görüntü üzerindeki değişikliklerin yapılabilirliğini kapatıyorum. (şekil 3.7.)

```
img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
img.flags.writeable = False
```

Şekil 3.7. Görüntüdeki her bir karenin BGR renk uzayından RGB renk uzayına dönüşümü

holistic.process() fonksiyonuyla MediaPipe kütüphanesinin Holistic modeline gönderiyorum. Burada RGB formundaki görüntüden el, kol, gövde, bacak, postür hareket tespiti yapıyorum.

Görüntü üzerindeki değişikliklerin yapılabilirliğini açarak eski haline alıyorum, ve geri RGB formatından BGR formatına alıyorum, çünkü OpenCV kütüphanesi BGR formatında işlem yapmaktadır. (şekil 3.8.)

```
img.flags.writeable = True
img = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

Şekil 3.8. Görüntüdeki her bir karenin RGB renk uzayından BGR renk uzayına dönüşümü

Elde ettiğim bu verileri önceden eğittiğim Random Forest modeli ile sınıflandırma (classification) yapıyorum. Tespit ettiğim bu noktaları da “ mdrawing.draw_landmarks() ” fonksiyonuyla test videosundaki yakalanan landmark’lar (karekteristik noktalar) kullanılarak gerçek zamanlı olarak sınıflandırma sonuçlarını her kare harekette ekrana yansıtıyorum. Aynı zamanda model her karede belirlediği bu veriler üzerinde işlem yaparak sınıf ve olasılık değerlerini hesaplar ve görüntü üzerinde eş zamanlı olarak gösterir.

4. SONUÇ

Görüntü işleme diğer adıyla “ Computer Vision “ hızla gelişen dünyada oldukça popüler olan, görüntünün analizi ve istenilen amaç uğuruna görüntüden elde edilen verileri çeşitli işlemlere tabi tutan bir teknoloji dalıdır. Görüntüdeki kişiye özel ayırıcı noktalar yakalanarak kimlik analizleri ve kimlik doğrulama işlemleri uygulanabilir.

Her insanın yürüyüş stili farklıdır ve bu yürüyüş stilleri her kişiye özgü bir biyometrik özellik olarak kabul edilir. Vücut yapısı, kas-iskelet yapısı, yaş, genetik faktörler ve duygusal durumlar kişinin yürüyüş stilini etkileyen özelliklerdir. Kişileri birbirlerinden ayırt etmek için bu farklılıklar kullanılabilir. Projemde de görüntü işleme sayesinde bu yürüyüş farklılıklarından yararlanarak kimlik doğrulama işlemi yaptım.

Kimlik doğrulama işlemi yaparken makine öğrenmesi (machine learning), derin öğrenme (deep learning) alanlarından yararlandım. Bireylerin yürüyüş stillerini görüntü işleme ile her pikselde pozisyonlarını algılayıp (pose detection) koordinatlarını bir CSV uzantılı dosyaya kaydettim. Bu pozisyonları kaydetmemin amacı yürürken kol, bacak, vücut, postür analizleri içindir. Örneğin, yürürken bacağını çok açarak mı yürüyor yoksa bacağını sürüyerek mi yürüyor ya da yürürken kollarını çok açarak, sallayarak mı yürüyor yoksa kolları hareketsiz bir şekilde mi yürüyor. Bu soruların cevaplarını bulmak için bu verileri kaydettim. Daha sonra ise modeli eğitim (training) ve test (testing) aşamalarından çeşitli boru hatları (pipeline) kullanarak geçireceğim ve bu sayede modelimi değerlendirmiş olacağım.

Bu projenin avantajlar bahsedilirse, yüzü gözükmeyen insanlar üzerinde kimlik doğrulama işlemlerinin yapılması gerektiği zaman yürüyüş stillerine göre bir model eğitilip bu kişi aranabilir, veya güvenlik sistemlerinde bireylerin tanınması ve izleme amacıyla kullanılabilir. Kapı girişlerinde ve güvenli alanlara giriş-çıkışlarda yürüyüş analizlerinden yararlanılarak kimlik doğrulaması yapılabilir.

Sonuç olarak, her kişiye özel olan yürüyüş stillerinin farklılıklarını görüntü işleme sayesinde kullanarak başarılı bir şekilde kişi kimlik doğrulaması yaptım. Bu proje sayesinde uygun veri seti bulmanın en az veriyi işlemek kadar uğraştırıcı olduğu bir kez daha ortaya çıkmış oldu. Eğitim (training) aşamasında kullanılan boru hatlarının

(pipeline) modelin overfitting (aşırı öğrenme) veya underfitting (az öğrenme) olmamasında büyük pay sahibidir. Aynı zamanda bu proje etik ve gizlilik konuları ihlal edilmediği sürece özellikle güvenlik alanlarında kullanılabilecek oldukça etkili bir teknoloji haline gelmiştir.

5. KAYNAKÇA

Jarrett, Christian. (2016, Mayıs 20). “DERGİ – Yürüme Tarzı Kişilikle İlgili Ne Anlatıyor?”

https://www.bbc.com/turkce/haberler/2016/05/160520_vert_fut_yurume_tarzi_kisilik

Bilgierdemdir.com. (2018). “Yürüme Şekline Göre Karakter Analizi”

https://www.bilgierdemdir.com/2018/05/yurume-sekline-gore-karakter-analizi.html

Mahesh. (2023, Mayıs 6). “Overfitting and Underfitting:Simple Explanations and Python Examples”

https://notebook.community/karlstroetmann/Artificial-Intelligence/Python/4%20Linear%20Regression/Underfitting-Overfitting

Li, David. (2023, Haziran 1). “Using OpenCV in Python:A Comprehensive Guide”

https://dlcoder.medium.com/using-opencv-in-python-a-comprehensive-guide-2b38f186f597

developers.google.com. (2023, Aralık 2). “Poz Algılama”

https://developers.google.com/ml-kit/vision/pose-detection?hl=tr

scikit-learn.org. (2023). “sklearn.linear_model.LogisticRegression”

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

scikit-learn.org. (2023). “sklearn.ensamble.RandomForestClassifier”

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Jain, Mayank. (2024, Ocak 05). “Top 90+ Python Libraries”

https://flexiple.com/python/python-libraries

gabormelli.com. (2023, June 19). “sklearn.ensemble.GradientBoostingClassifier”

https://www.gabormelli.com/RKB/index.php?title=sklearn.ensemble.GradientBoostingClassifier&mobileaction=toggle_view_desktop

scikit-learn.org. (2023). “sklearn.linear_model.RidgeClassifier”

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html

forecastegy.com. (2023, July 1). “How To Solve Logistic Regression Not Converging in Scikit-Learn”

https://forecastegy.com/posts/how-to-solve-logistic-regression-not-converging-in-scikit-learn/

Tirendaz AI. (2022, Ocak 29). “7 Steps to Build a Machine Learning Model with Python”

https://medium.com/mlearning-ai/machine-learning-project-with-linear-regression-algorithm-b433d770fefd