Futbolcu Bonservis Bedel Aralığı Tahmini

Metin Binbir
Bilgisayar Mühendisliği
Yıldız Teknik Üniversitesi
İstanbul, Türkiye
19011613

Hüma Bilgin
Bilgisayar Mühendisliği
Yıldız Teknik Üniversitesi
İstanbul, Türkiye
18011087

***Özet*— Bu projede, FIFA22 adlı futbol oyunu veritabanında bulunan futbolcuların, oyun içi istatistiklerinden yola çıkarak bonservis değerleri, belli aralıklara indirgenerek sınıflara ayrılmıştır ve sınıflandırma işlemi uygulanmıştır. Ön işleme (preprocess) uygulanmış veri, orijinal hâlde, PCA uygulanmış hâlde ve normalize edilmiş hâlde çeşitli makine öğrenmesi sınıflandırma algoritmalarıyla eğitilmiştir. Modellerin başarıları accuracy ve F1 skorlarıyla kıyaslanmıştır. En başarılı model %94 accuracy değeri ile orijinal veri setiyle eğitilmiş Random Forest olmuştur.**

Anahtar Kelimeler—KNN, Sınıflandırma, Random Forest, Support Vector Machines, Makine Öğrenmesi, Decision Tree, Veri Madenciliği

I. INTRODUCTION

Futbol takımlarının transferlerini doğru şekilde gerçekleştirebilmeleri için bonservis değerlerinin hesaplanması oldukça önemlidir. Takımlara en çok ekonomik yük oluşturan kalemlerden biri olan bonservis değerlerinin ideal bir şekilde belirlenmesiyle birlikte kulüplerin diğer harcamaları da bu şekle göre planlanacaktır. Bu hesaplamaların yapılması için futbolcuların önceki maçlarına dayalı veriler kullanılmaktadır.

Bu projede de, KNN, Random Forest, Desicion Tree ve SVM olacak şekilde dört makine öğrenmesi sınıflandırma yöntemi ile model eğitimi yapılmaktadır. Modellerin sonucunda hesaplanan accuracy değerleri kıyaslandırılmakta ve en başarılı algoritma bulunmaktadır.

II. VERI KEŞFI

A. *Kullanılan Veri Seti*

Projede kullanılan veri seti Kaggle'da bulunan FIFA22 [1] veri setidir. Veri setindeki 110 attribute'dan 16 tanesi, sonuç üzerindeki anlamlarına bakılarak seçilmiştir. Seçilen attributelar bağımsız değişkenler olarak sırasıyla overall, potential, boy, kilo, mevki, oyuncunun tercih ettiği ayak, oyuncunun zayıf ayak gücü, yetenek hareketleri, hız, şut gücü, pas gücü, dribling gücü, defans gücü, fiziksel, bitiricilik iken hedef değişken olarak oyuncu bonservis değeridir. Veri setine hiçbir işlem uygulanmamış halde iken başlangıçta 19239 adet örnek barındıran veri seti, proprocess işlemleri sonrasında 17041 örneğe inmektedir.

B. *Preprocessing Adımları*

Veri setinin kullanıma hazır hale gelmesi, eğitimde çıkacak pürüzlerin önüne geçilmesi ve daha doğru sonuçlar üretilebilmesi için veri üzerinde ön hazırlık yapılması gerekmektedir. Proje Python programlama diliyle gerçeklenmiştir, yapılan tüm preprocessing adımları Python'daki pandas [8] kütüphanesi ve temel algoritmik yaklaşımlar ile sağlanmıştır. Bu proje kapsamında yapılan preprocess aşamaları şunlardır:

1. Bazı yabancı karakterler verinin okunmasını bozabileceği için pandas kütüphanesi ile okunan veri utf-8 formatı aracılığıyla okunmuştur.
2. 110 attribute içerisinden 16 adet attribute seçilen data frame yapısı, yeni veri seti haline gelmiştir.
3. Kaleci mevkisinde forma giyen oyuncuların hız, dribling, şut, defans vb. performans değerleri diğer oyuncuların değerlerinden farklılık gösterdiğinden veri setindeki tüm kaleciler silinmiştir.
4. 16 adet attribute içerisinde eşsiz olmayı sağlayan isim gibi bir attribute olmadığı için aynı değere sahip verilerin olabilme ihtimaline karşı drop_duplicates fonksiyonu veri setine uygulanmıştır fakat bu ihtimal gerçekleşmemiştir.
5. Oyuncu mevkisi kısmında ham veride bir oyuncu birden fazla mevkide forma giyebilmektedir. Bu mevkiler toplamda 673 tane eşsiz kombinasyondan oluşmaktadır, birden fazla mevkide forma giyen oyuncuların mevkileri virgül (",") karakteriyle ayrılmaktadır. Çok sayıda mevki sayısı algoritmaların başarısını olumsuz derecede etkileyeceğinden dolayı bu değerler öncelikle bir liste yapısına atılmıştır. Daha sonrasında eğer oyuncu tek mevkide oynuyorsa listedeki mevkisinde herhangi bir değişiklik yapılmamıştır, birden fazla mevkide oynuyorsa Python'da yer alan split(",") fonksiyonu ile ilk mevki oyuncunun ana mevkisi olarak atanmıştır. Daha sonrasında bu liste tekrardan veri setinin oyuncu mevki sütunu olarak güncellenmiştir.

```
df['player_positions'].unique()
```

```
array(['RW, ST, CF', 'ST', 'ST, LW', 'LW, CAM', 'CM, CAM', 'CDM, CM',
       'CF, ST', 'LM, CF, LW', 'CDM', 'CB', 'LW', 'RW', 'CDM, RB', 'CM',
       'LW, RW', 'CAM', 'RW, LW', 'CAM, RM, RW', 'CM, LM', 'CB, CDM',
       'CM, CDM', 'CF, CAM', 'LB', 'CM, CDM, CB', 'RB', 'RM, CF, LM',
       'LB, LM', 'LW, CF', 'RW, RM', 'ST, RM, RW', 'CDM, CB', 'RB, LB',
```

Şekil 2.1: Birden Fazla Mevkiye Sahip Oyuncu Örnekleri

```
for i in range(len(df_positions)):
  if len(df_positions[i]) < 2:
    print(i)
  currentPosition = df_positions[i].split(",")
  if  len(currentPosition) > 1:
    df_positions[i] = currentPosition[0]


listToDF = pd.DataFrame(df_positions,columns=['pos']).squeeze()

df['player_positions'] = listToDF
```

Şekil 2.2: Tek Mevkiye İndirgeme Algoritması

6. Güncellenen oyuncu mevki sütunu incelendiğinde sol kanat, sağ kanat, sol orta saha, sağ orta saha, sol kanat bek, sağ kanat bek gibi ayrımların yine çok sayıda

sınıf yarattığı düşünülmüştür. Sonuç olarak pozisyon mevkisi sol-sağ kanat, orta saha, sol-sağ bek, defansif orta saha, ofansif orta saha, stoper ve santrafor olmak üzere 7 sınıfa indirgenmiştir. Bu indirgeme işlemi için pandas ve replace() fonksiyonu kullanışmıştır. İndirgeme sonucunda mevcut veri yine metinsel olduğu için sklearn kütüphanesinde yer alan LabelEncooder objesi kullanılarak bu mevkiler 0-6 aralığında nümerik verilere dönüştürülmüştür.

```
# CB / LB-RB / CDM / CAM / LW-RW / ST / CM
df['player_positions'] = df['player_positions'].replace(['LW'],"LW-RW")
df['player_positions'] = df['player_positions'].replace(['RW'],"LW-RW")
df['player_positions'] = df['player_positions'].replace(['LM'],"CM")
df['player_positions'] = df['player_positions'].replace(['RM'],"CM")
df['player_positions'] = df['player_positions'].replace(['LB'],"LB-RB")
df['player_positions'] = df['player_positions'].replace(['RB'],"LB-RB")
df['player_positions'] = df['player_positions'].replace(['LWB'],"LB-RB")
df['player_positions'] = df['player_positions'].replace(['RWB'],"LB-RB")
df['player_positions'] = df['player_positions'].replace(['CF'],"ST")
```

Şekil 2.3: 7 Ana Mevkiye İndirgeme Algoritması

```
labelencoder = LabelEncoder()

df['player_positions'] = labelencoder.fit_transform(df['player_positions'])

position_dict = dict(zip(labelencoder.classes_,range(len(labelencoder.classes_))))

position_dict

{'CAM': 0, 'CB': 1, 'CDM': 2, 'CM': 3, 'LB-RB': 4, 'LW-RW': 5, 'ST': 6}
```

Şekil 2.4: 7 Ana Mevkinin Nümerik Veriye Çevrilmesi

7. Oyuncunun tercih ettiği ayak değerleri "Left" ve "Right" olarak metinsel veridir, yine LabelEncoder objesi yardımıyla bu sütuna ait veriler nümerik veriye çevrilmiştir.

```
df['preferred_foot'] = labelencoder.fit_transform(df['preferred_foot'])

pref_foot_dict = dict(zip(labelencoder.classes_,range(len(labelencoder.classes_))))

pref_foot_dict

{'Left': 0, 'Right': 1}
```

Şekil 2.5: Oyuncunun Ayağının Nümerik Veriye Çevrilmesi

8. Oyuncunun "weak_foot" (zayıf ayak) ve "skill_moves" (yetenek hareketleri) değerleri 1-5 arasında değişkenlik göstermektedir, bu değerler pandas ve multiply(0.2) fonksiyonuna tabi tutularak 0-1 arasına indirgenmiştir.

```
df['weak_foot'] = df['weak_foot'].multiply(0.2)
df['skill_moves'] = df['skill_moves'].multiply(0.2)
```

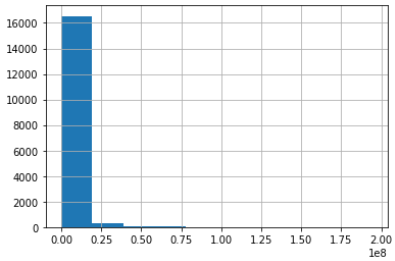
Şekil 2.6: Zayıf Ayak Ve Yetenek Hareketi İndirgemesi

9. Attributelar içerisinde yer alan oyunculara ait bonservis değeri, zayıf ayak, yetenek hareketleri, boy ve kilo hariç tüm attributelar 0-100 aralığında yer almaktadır. Bu yüzden hesaplamaların daha hızlı yapılabilmesi adına boy ve kilo da dahil olmak üzere 0-100 aralığında yer alan tüm attributelara ait değerler 100'e bölünmüştür.

```
df[['overall','potential',"pace", "shooting",
    "passing", "dribbling", "defending", "physic",
    "attacking_finishing", "height_cm", 'weight_kg']] /= 100
```

Şekil 2.7: Veri Setindeki Bazı Sütunların Indirgenmesi

10. Problem bir sınıflandırma problemi olduğu için bonservis değerleri sınıflara ayrılmalıdır. Bonservis değerleri pandas kütüphanesinde yer alan histt() fonksiyonu ile görselleştirildiği zaman bu sütuna ait verinin skewed olduğu anlaşılmıştır. Bu yüzden çeşitli aralıklara ait kaç farklı örnek olduğunu tespit etmek için manuel olarak denemeler yapılmıştır ve 0-750.000 €, 750.000 – 2.500.000€, 2.500.000€ - 10.000.000€, 10.000.000€ - 50.000.000€ ve 50.000.000€ üzeri olarak 5 sınıfa ayrılmıştır. Buna rağmen veri yine skewed halde kalmıştır. Pandas kütüphanesinde yer alan cut işlemi ile bu değer aralıklarına label etiketleri atanarak 5 farklı sınıf 1, 2, 3, 4, 5 olarak sınıflandırılmıştır.



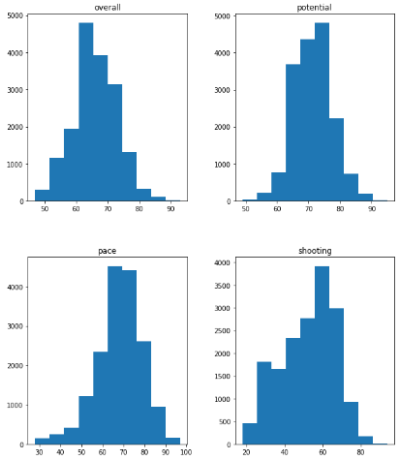
Şekil 2.8: Oyunculara Ait Bonservis Bedeli Dağılımına Ait Histogram

```
((15000 <= df['value_eur']) & (df['value_eur'] < 750000)).sum()
6359

((750000 <= df['value_eur']) & (df['value_eur'] < 2500000)).sum()
6966

((2500000 <= df['value_eur']) & (df['value_eur'] < 10000000)).sum()
2688

((10000000 <= df['value_eur']) & (df['value_eur'] < 50000000)).sum()
938

((50000000 <= df['value_eur'])).sum()
90

df['value_eur'] = pd.cut(df['value_eur'], bins=[0, 749999, 2499999,
                  9999999, 49999999, 200000000], labels=[1,2,3,4,5])
```

Şekil 2.9: Bonservislerin Sınıflara Ayrılması

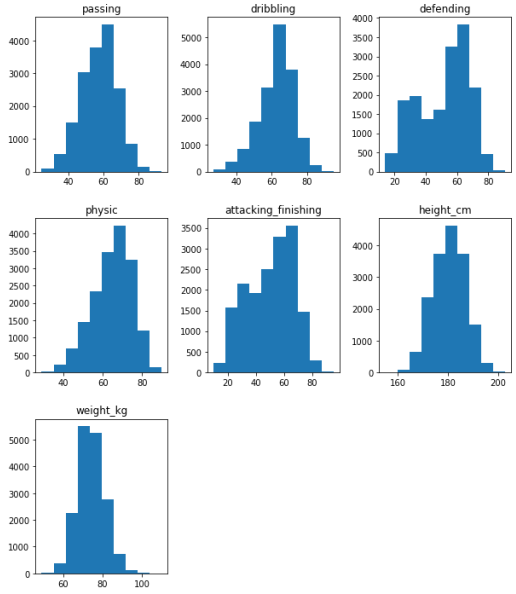
III. VERİ GÖRSELLEŞTİRME

Veri setinde "overall", "potential", "pace", "shooting", "passing", "dribbling", "defending", "physic", "attacking_finishing", "height_cm" ve "weight_kg" sütunundaki veriler 0-100 aralığında değişkenlik göstermektedir. Bu veriler histogramla ifade edilmektedir.



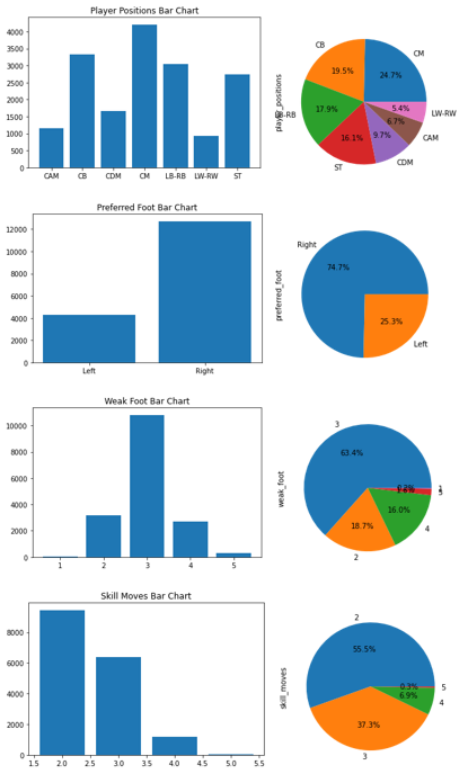
Şekil 3.1: Bağımsız ve Sürekli Değişkenlerin Histogramla Görselleştirilmesi - 1

Histogramı yansıtılan veriler incelendiğinde boy ve kilo hariç 0-100 arasında değişkenlik gösteren değerlerde genelde normal bir dağılım gözlemlenmektedir fakat genel yetenek reytingleri en üst düzeyde olan örnek sayısının azlığı sınıflandırma problemi için risk teşkil etmektedir.



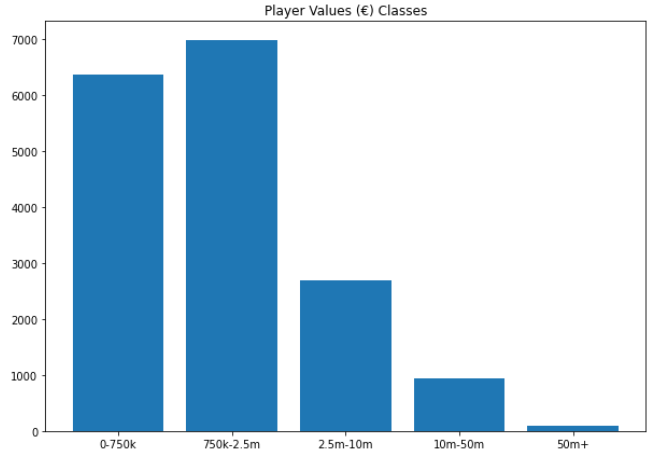
Şekil 3.2: Bağımsız ve Sürekli Değişkenlerin Histogramla Görselleştirilmesi - 2

Oyuncuların "position", "preferred_foot", "weak_foot", "skill_moves" sütunlarına ait verileri sınıflandırma içerdiğinden bu verilerin görselleştirilmesi bar chart ve pie chartlar ile yapılmıştır.



Şekil 3.3: Bağımsız ve Kategorik Değişkenlerin Bar Char ve Pie Chart ile Görselleştirilmesi

Oyuncu bonservis bedellerinin sınıflara ayrılmasından sonra bu sınıflar bar chart ile görselleştirilmiştir.



Şekil 3.4: Oyuncu Bonservis Bedellerine Ait Sınıflar

IV. METODOLOJI

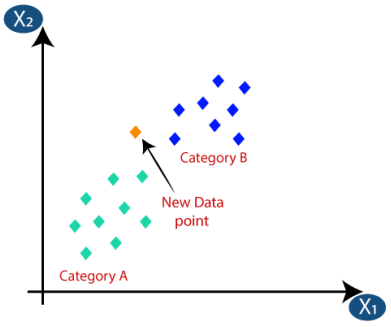
Proje kapsamında dört adet makine öğrenmesi sınıflandırma algoritması kullanılmıştır. Her algoritmaya da veri seti orijinal haliyle, PCA uygulanmış haliyle ve normalizasyon uygulanmış olmak üzere üç farklı formatta verilip sonuçları kıyaslanmıştır. Tüm yöntemlerde veri setini en iyi şekilde bölmeyi sağlayan cross validation işlemii uygulanmıştır. Tüm bu yöntem ve işlemlere ait detaylar ve sonuçlar aşağıda tek tek ele alınmıştır. Öncesinde veri seti %80 eğitim ve %20 test olacak şekilde bölünmüştür. Daha sonrasında PCA işleminin daha sağlıklı sonuç vermesini sağlayan standardizasyon işlemi StandarScaler() fonksiyonu ile uygulanmıştır. PCA objesiyle transform edilerek yeni bir data frame'ye çevrilmiştir. Normalize edilmiş veri seti de ilk başta bölünen veri setini sklearn kütüphanesinde yer alan normalize fonksiyonuyla elde edilmiştir.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()
X_train_pca = scaler.fit_transform(X_train)
X_test_pca = scaler.transform(X_test)
pca = PCA(n_components=5)

X_train_pca = pca.fit_transform(X_train_pca)
X_test_pca = pca.fit_transform(X_test_pca)

X_train_norm = preprocessing.normalize(X_train)
X_test_norm = preprocessing.normalize(X_test)
```

Şekil 4.1: Veri setinin orijinal, PCA ve normalize şekilde ayrılması

A. *K-Nearest Neighbor (KNN)*

KNN algoritması temel olarak tahmin edilecek değerin komşularının hangi sınıfta yoğun olduğuna bakarak sınıflandırma yapar. Bu bağlamda algoritma uzaklık ve komşuluk sayısı değerlerine göre tahmin üzertmektedir.



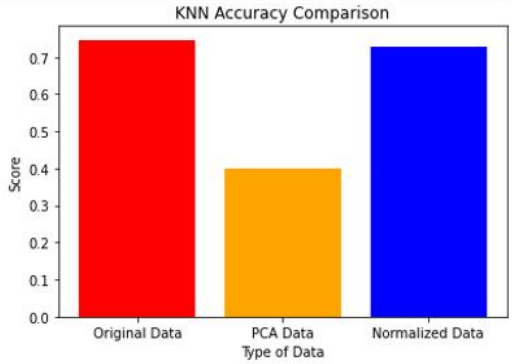
Şekil 4.2: KNN

En iyi sonucu verecek model seçimi için KNN parametrelerinden biri olan komşu sayısı ve cross validation uygulama maksatlı KNN modeli GridSearchCV fonksiyonu ile gerekli parametrelerle çağrılmıştır. Model öncelikle orijinal veri setiyle eğitilmiştir. Orijinal veri setiyle eğitilen KNN modeli %74 accuracy değerine sahip olmuştur. Eğer veri setinde 0–100 aralığında yer alan sütunların verileri 0-1 aralığına indirgenmeseydi uzayda yer alan veriler arasındaki mesafe fazla olacağından bu algoritma daha kötü sonuç verecekti.

Daha sonra, yeni oluşturulan KNN modeli, 5 component parametresiyle PCA işlemi uygulanmış veri setiyle eğitilmiştir. PCA fonksiyonu 5 component parametresi ile çağrılmıştır, yani 15 attribute, 5 attribute'a indirgenmiştir. Bu işlem sonucunda PCA veri setiyle eğitilmiş KNN modeli %39 accuracy değerine sahip olmuştur. 5 attribute'a indirgeme işleminde karar vermeyi sağlayan önemli sütunların etkilerinin azaldığı söylenebilir.

Son olarak tekrardan yeni oluşturulmuş bir KNN modeli, orijinal veri setinin normalizasyon uygulanmış haliyle eğitilmiştir. Normalize edilmiş veri setiyle eğitilen KNN modeli %72 accuracy değeri elde etmiştir.

```
Accuracy With KNN (original): 0.7468465825755354
Accuracy With KNN (original): 0.7444439759102224

Accuracy With KNN (PCA): 0.39894397183924907
F1 With KNN (PCA Data): 0.39591349741263426

Accuracy With KNN (Normalized): 0.7268993839835729
F1 With KNN (Normalized): 0.7220299507533903
```

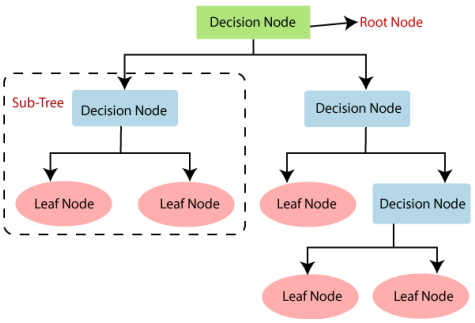


Şekil 4.3: KNN ile Eğitilen Veri Setlerinin Accuracy ve F1 Kıyaslaması

KNN için en başarılı sonucu veren veri seti orijinal veri seti olarak belirlenmiştir. Fakat veri üzerindeki dengesizlik yüzünden bu sonuca varıldığı yorumu da yapılabilir.

B. Karar Ağacı (DT)

Karar ağaçları, bazı kurallar uygulayarak veri setini birçok kez bölerek küçük parçalara ayırır. Bu bölünmelerin gerçekleştirilme şekilleri değiştirilerek ağacın daha doğru sonuçlar üretmesi sağlanmaktadır.



Şekil 4.4: Decision Tree Sınıflandırıcısı

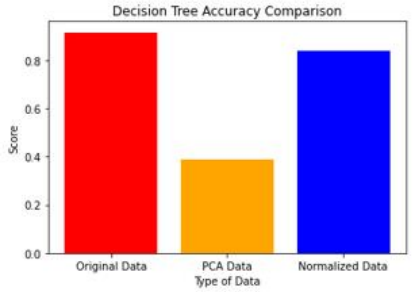
Karar ağacı algoritması yalnızca sınıflandırmayla sınırlı kalmayıp regresyon için de kullanılmaktadır. Proje kapsamında konu sınıflandırma olduğu için sklearn kütüphanesinde yer alan DecisionTreeClassifier() fonksiyonundan yararlanılmıştır. Karar ağacının dallanırken karar almasına etki eden faktörlerden olan gini ve entropi değerleri ve cross validation için gerekli parametreler GridSearchCV fonksiyonuyla fit edilmiştir.

İlk olarak orijinal veri seti ile eğitilen karar ağacı sınıflandırma modeli %91 accuracy değerine sahip olmuştur.

Daha sonrasında yeni KNN modeli yine criterion ve cross validation bayraklarına gerekli parametreler verilerek oluşturulmuştur ve PCA işlemi uygulanmış veri setiyle eğitilmiştir. PCA uygulanan veri setiyle eğitilen model %39 accuracy değeri elde etmiştir.

Son olarak bir KNN modeli daha önceki işlemlerde verilen parametrelerle oluşturulmuştur ve normalize edilmiş veri setiyle eğitilmiştir. Bu işlem sonucunda %83 accuracy değerine sahip olduğu gözlemlenmiştir.

```
Accuracy With Decision Tree (Original Data): 0.9172777940745086
F1 With Decision Tree (Original Data) 0.9171831824184806

Accuracy With Decision Tree (PCA Data): 0.39014373716632444
F1 With Decision Tree (PCA Data): 0.39189744658266096

Accuracy With Decision Tree (Normalized Data): 0.8386623643297154
F1 With Decision Tree (Normalized Data):  0.8386226862270582
```

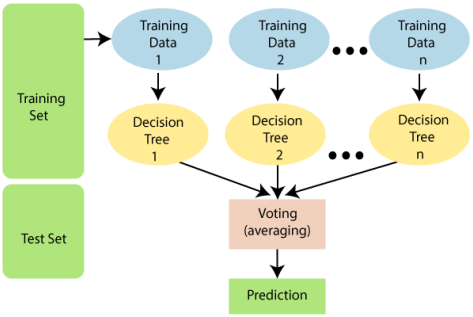


Şekil 4.5: DT ile Eğitilen Veri Setlerinin Accuracy ve F1 Kıyaslaması

Karar ağacı sınıflandırma algoritması için de en başarılı sonuç orijinal veri setiyle eğitilen modelden elde edilmiştir. Diğer algoritmalara göre yüksek olma sebebi olarak, sınıfların limit değerlerine yakın olduğu yerlerde uygulanan normalizasyon işlemlerinin, orijinal veriye göre daha hatalı ayırması olarak yorumlanabilir. Yine veri seti üzerindeki skewness durumu yüzünden başarının yüksek olması yorumu yapılabilir.

C. *Random Forest Classification (RFC)*

Random Forest Algoritması, birden çok karar ağacı üreterek aralarından en iyi sonucu üreten ağacı seçme işlemini gerçekleştirir. Üretilen ağaç sayısı arttıkça başarı elde etme oranı artmaktadır.



Şekil 4.6 : Random Forest Sınıflandırıcısı

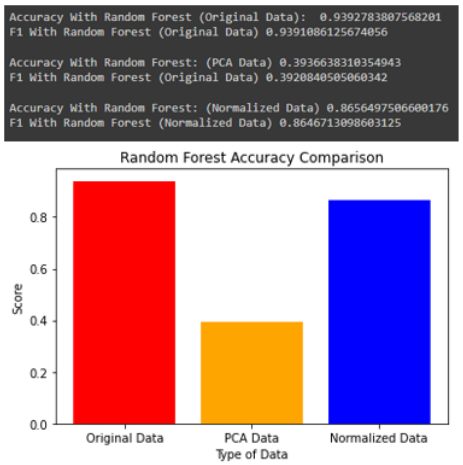
Kümülatif bir karar ağacı yöntemi olan random forest algoritması da hem sınıflandırma hem de regresyon için kullanılabilmektedir. Proje kapsamında sklearn kütüphanesinde yer alan RandomForestClassifier() fonksiyonundan faydalanılmıştır. GridSearchCV() fonksiyonuyla cross validation ile karar ağacındaki gini ve entropiye ek olarak kaç kez hesaplama yapılacağını bildiren n_estimators parametresine 100 ve 200 değerleri verilerek en çok sayıda elde edilen karar ağacı modelinin elde edilmesi sağlanmıştır.

İlk olarak orijinal veri setiyle eğitilmiş random forest sınıflandırma modeli %93 accuracy değerine sahip olmuştur.

Ardından, PCA uygulanmış veri setiyle eğitilen random forest modeli %39 accuracy skoru elde etmiştir.

Son olarak da normalizasyon uygulanmış veri setiyle eğitilmiş random forest modeli %86 accuracy değeri elde etmiştir.

Sonuçların değerlendirilmesi, bir önceki başlıkta yer alan karar ağacı algoritmasıyla aynıdır.

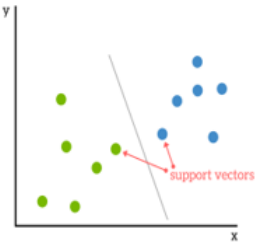


Şekil 4.7: RFC ile Eğitilen Veri Setlerinin Accuracy ve F1 Kıyaslaması

Identify applicable funding agency here. If none, delete this text box.

D. *Support Vector Machines (SVM)*

SVM algoritması bir düzlem üzerinde doğru çizerek sınıflandırma yapan bir algoritmadır. Çizilen bu çizgi iki sınıfı en iyi şekilde ayıran doğruyu ifade etmektedir. Proje kapsamında da olduğu gibi, SVM çoklu sınıf sınıflandırmada da kullanılabilmektedir. Çoklu sınıflandırma, çok sayıda binary classificationa bölünerek sınıfların belli olması one-to-one yaklaşımıdır fakat sklearn kütüphanesi varsayılan olarak bu yaklaşım yerine one-to-rest yaklaşımını izler. One-to-rest yaklaşımında her sınıfa ait veriler, kalan veri setinin tamamına karşı yine ardışıl olarak binary classification işlemine tabi tutulur. One-to-one yaklaşımı özelinde bakılacak olursa örneğin 5 farklı sınıf için 5*(5-1)/2 = 10 farklı binary classification izlenirken one-to-rest yaklaşımı özelinde sınıf sayısı yani 5 farklı binary classification izlenir.



Şekil 4.8: Support Vector Machines

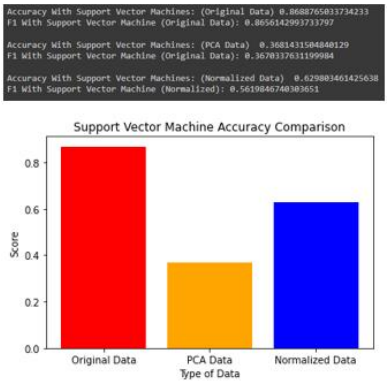
Proje özelinde diğer algoritmalarda olduğu gibi öncelikle SVC modeli orijinal veri setiyle eğitilmiştir. Eğitim öncesinde yine GridSearchCV() ile cross validation ve SVC başarısı için önemli olan kernel parametresi lineer olarak ayarlanmıştır.

Orijinal veri setiyle eğitilmiş SVM modeli %86 accuracy skoruna sahiptir.

Sonrasında PCA uygulanmış eğitim veri setiyle eğitilen SVM modeli %36 accuracy değerine sahip olmuştur.

Normalize edilmiş veri setiyle eğitilen SVM modeli ise %62 accuracy skoru elde etmiştir.

Sonuçlara bakıldığında veri setindeki dengesizlik sonuçları doğrudan etkiliyor yorumu yapılabilir, ayrıca sistemin daha iyi sonuçlar vermesi için farklı kernel türleri de model parametresi olarak kullanılabilir.

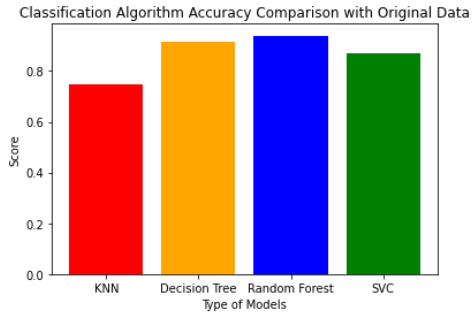


Şekil 4.9: SVM ile Eğitilen Veri Setlerinin Accuracy ve F1 Kıyaslaması

V. SONUÇLAR

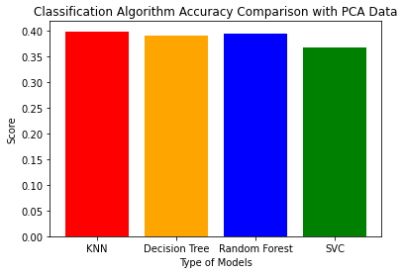
Bir önceki başlıkta kullanılan algoritmalar ve her algoritmada kullanılan veri seti özelinde accuracy kıyaslamaları yapılmıştır. Bu kısımda veri seti temel alınarak kullanılan algoritmaların başarıları kıyaslanacaktır. Ardından accuracy değeri en yüksek modellere ait confusion matrixler verilecektir.

Orijinal veri setiyle eğitilen modeller arasında en fazla accuracy skoru elde eden model random forest algoritmasıdır. En düşük skoru elde eden model KNN'dir.



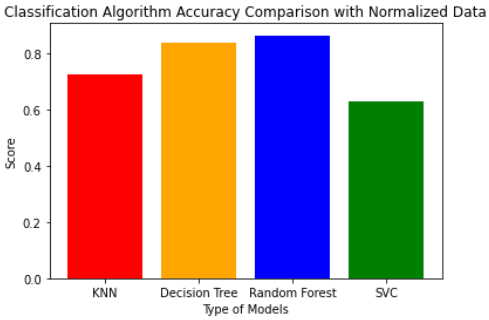
Şekil 5.1: Orijinal Veri Setinin Algoritmalara Göre Accuracy Değerleri

PCA uygulanmış veri setiyle eğitilen modeller arasında tüm modeller yüksek başarı elde edememiştir. KNN, Decision Tree ve Random Forest algoritmaları birbirine çok yakın sonuçlar elde etmişken SVC modelinin başarısı daha düşüktür.



Şekil 5.2: PCA Uygulanmış Veri Setinin Algoritmalara Göre Accuracy Değerleri

Normalize edilmiş veri setiyle eğitilen 4 farklı model arasından da en başarılı sonucu veren model random forest algoritmasıdır. SVC algoritması eğitildiği veri setiyle beraber diğer modellere göre daha düşük accuracy skoruna sahip olmuştur.



Şekil 5.3: Normalize Edilmiş Veri Setinin Algoritmalara Göre Accuracy Değerleri

	precision	recall	f1-score	support
0-750000	0.84	0.78	0.81	1245
750k - 2.5 m	0.71	0.81	0.76	1421
2.5 m - 10 m	0.62	0.61	0.62	515
10m - 50m	0.77	0.50	0.61	204
50m - 200m	1.00	0.17	0.29	24
accuracy			0.75	3409
macro avg	0.79	0.58	0.62	3409
weighted avg	0.75	0.75	0.74	3409

Confusion Matrix for Best KNN Model

Şekil 5.4: En Başarılı KNN Modeline Ait Confusion Matrix

	precision	recall	f1-score	support
0-750000	0.94	0.93	0.93	1245
750k - 2.5 m	0.91	0.92	0.91	1421
2.5 m - 10 m	0.89	0.91	0.90	515
10m - 50m	0.92	0.88	0.90	204
50m - 200m	0.80	0.67	0.73	24
accuracy			0.92	3409
macro avg	0.89	0.86	0.87	3409
weighted avg	0.92	0.92	0.92	3409

Confusion Matrix for Best Decision Tree Model

Şekil 5.5: En Başarılı DT Modeline Ait Confusion Matrix

	precision	recall	f1-score	support
0-750000	0.98	0.93	0.95	1245
750k - 2.5 m	0.92	0.96	0.94	1421
2.5 m - 10 m	0.90	0.94	0.92	515
10m - 50m	0.92	0.90	0.91	204
50m - 200m	0.93	0.54	0.68	24
accuracy			0.94	3409
macro avg	0.93	0.85	0.88	3409
weighted avg	0.94	0.94	0.94	3409

Confusion Matrix for Best Random Forest Model

Şekil 5.6: En Başarılı RF Modeline Ait Confusion Matrix

	precision	recall	f1-score	support
0-750000	0.93	0.89	0.91	1245
750k - 2.5 m	0.85	0.90	0.88	1421
2.5 m - 10 m	0.79	0.83	0.81	515
10m - 50m	0.83	0.69	0.75	204
50m - 200m	0.00	0.00	0.00	24
accuracy			0.87	3409
macro avg	0.68	0.66	0.67	3409
weighted avg	0.86	0.87	0.87	3409

Şekil 5.7: En Başarılı SVC Modeline Ait Confusion Matrix

VI. BENZER ÇALIŞMALAR

Projede kullanılan veri seti ile yapılan diğer çalışmalar incelendiğinde, sınıflandırma yerine regresyon içeren modellerin oyuncu maaşlarına olan etkisi örnek gösterilebilir [6]. Fakat yapılan birkaç çalışmaya bakıldığında yalnızca bir algoritma ile çalışma yapıldığı görülmektedir. Aksine, bu projede birden çok algoritma test edilerek, algoritmalar test edilmiş ve doğru algoritmaya bu şekilde karar verilmiştir.

VII. ÇIKARIM

Bu proje kapsamında internetten alınan FIFA22 oyunundaki futbolcuların sahip olduğu özelliklerden yola çıkılarak oyuncunun oyun içerisindeki bonservis bedelinin sınıflandırma yoluyla tahmin edilmesi işlemi yapılmıştır.

Toplanan veri seti, analiz ve sunum aşaması için Python programlama dilindeki araçlarla görselleştirilmiş, çeşitli preprocess işlemlerinden geçirildikten sonra orijinal, PCA

uygulanmış ve normalize edilmiş olarak üç farklı kategoriye ayrılmıştır. Ardından da her veri seti kategorisi; KNN, Decision Tree, Random Forest ve SVC algoritmalarıyla eğitilerek accuracy ve F1 skorları elde edilmiştir ve kıyaslanmıştır.

Çoğu yapay zeka probleminin çözümünde karşılaşılan veri setindeki dengesizliğin giderildikten sonra tekrardan sınıflandırma algoritmalarına verilerek eğitilmesi, daha sağlıklı sonuçlar alınması konusunda yardımcı olacaktır.

VIII. REFERANSLAR

[1] Stefano Leone, 01.10.2021, 'FIFA 22 complete player dataset', https://www.kaggle.com/datasets/stefanoleone992/fifa-22-complete-player-dataset

[2] Evren Arslan, May 19, 2020, 'Makine Öğrenmesi — KNN (K-Nearest Neighbors) Algoritması Nedir?', https://arslanev.medium.com/makine-%C3%B6%C4%9Frenmesi-knn-k-nearest-neighbors-algoritmas%C4%B1-bdfb688d7c5f

[3] E. Kaan Ulgen, Nov 12, 2017, 'Makine Öğrenimi Bölüm-5 (Karar Ağaçları)', https://medium.com/@k.ulgen90/makine

[4] Ece Akdağlı, Mar 4, 2021, 'Makine Öğrenmesinde Random Forest Algoritması', https://ece-akdagli.medium.com/makine-%C3%B6%C4%9Frenmesinde-random-forest-algoritmas%C4%B1-a79b044bbb31

[5] Sait Alay, 22 Haziran 2020, 'AlgoRithm:Destek Vektör Makineleri' https://www.datasciencearth.com/algorithmdestek-vektor-makinelerisupport-vector-machinesr-kod-ornekli/

[6] Muhammet Ali Kula, 19.01.2022, 'fifa_wage_prediction', https://www.kaggle.com/code/muhammetalikula/fifa-wage-prediction

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[8] McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).