YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

INTRODUCTION TO DATA MINING

COURSE PROJECT REPORT

Öğrenci Numarası: 20011626

İsim: Yusuf Yemliha

Soy İsim: ÇELİK

E-Posta Adresi: yemliha.celik@std.yildiz.edu.tr

Doç.Dr. Ayşe Betül OKTAY

3 Oca, 2023

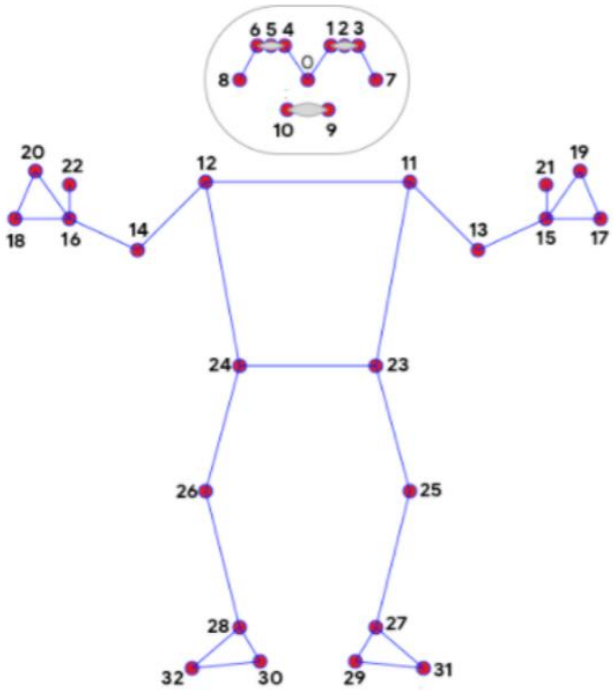
GİRİŞ

Veri kümesi, egzersiz yapan 500 insandan elde edilmiştir.

Bu veri kümesinde 5 tane hareket mevcuttur. Her hareketin yukarı ve aşağı olmak üzere iki faklı durumu vardır.

Hareketler sırasıyla `'squats_up' 'squats_down' 'jumping_jacks_up' 'jumping_jacks_down' 'pushups_up' 'pushups_down' 'situp_up' 'situp_down' 'pullups_up' 'pullups_down'`.dır.

Mediapipe kütüphanesi yordamıyla pose algılama işlemleri gerçekleştirilir.Hareketlerin tespiti vücuttaki 33 farklı dokunun koordinatları yardımıyla gerçeklenir.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

3)KAGGLE SONUÇLARI

Pose tahmini için farklı sınıflandırma algoritmaları kullandım.

Bunlar Knn,Random Forest ve Xgboost algoritmalarıdır.

Bunlardan elde ettiğim score değerlerine göre tercihim

Xgboost algoritması oldu.

Xgboost algoritması

XGBoost(eXtreme Gradient Boosting), Gradient Boosting algoritmasının çeşitli düzenlemeler ile optimize edilmiş yüksek performanslı halidir. Algortimanın en önemli özellikleri yüksek tahmin gücü elde edebilmesi, aşırı öğrenmenin önüne geçebilmesi, boş verileri yönetebilmesi ve bunları hızlı yapabilmesidir. Daha az kaynak kullanarak üstün sonuçlar elde etmek için yazılım ve donanım optimizasyon tekniklerini uygulanmıştır. Karar ağacı tabanlı algoritmaların en iyisi olarak gösterilir.

3)KAGGLE SONUÇLARI

XGBoost'ta ilk adım ilk tahmini (base score) yapmaktır. Bu tahmin, bundan sonraki adımlarda yapılacak işlemler ile yakınsayarak doğru sonuca ulaşılacağı için herhangi bir sayı olabilir. Bu sayı varsayılan olarak 0,5'tir.

Yapılan bu tahminin ne kadar iyi olduğu modelin hatalı tahminleri(residual) ile incelenir. Hatalar, gözlemlenen değerden tahmin edilen değerin çıkarılması ile bulunmaktadır.

Bir sonraki aşamada Gradient Boosting'de olduğu gibi hataları tahminleyen karar ağacı kurulur. Burada amaç hataları öğrenip doğru tahmine yaklaşmaktır.

Oluşturulan ağacın her bir dalı için benzerlik skoru(similarity score) hesaplanır. Benzerlik skoru verilerin dallarda ne kadar iyi gruplandığını gösterir.

$$\text{Similarity Score} = \frac{\text{Sum of Residuals, Squared}}{\text{Number of Residuals} + \lambda}$$

3)KAGGLE SONUÇLARI

Hiperparametreler

XGBoost'u uygularken seçilecek en önemli hiperparametreler hangileridir ve bunlar nasıl ayarlanır?

Booster

booster, 3 seçeneğiniz olan artırma algoritmasıdır: gbtree, gblinear veya dart. Varsayılan seçenek, gbtree'dir. Dart, aşırı öğrenmeyi (over-fitting) önlemek için bırakma (dropout) tekniklerini kullanan benzer bir sürümdür. Gblinear ise, karar ağacı yerine genelleştirilmiş doğrusal regresyon kullanır.

reg_alpha and reg_lambda

reg_alpha ve reg_lambda, sırasıyla L1 ve L2 regülasyon terimleridir. Bu sayılar ne kadar büyükse, model o kadar tutucu (aşırı öğrenmeye daha az eğilimli) olur. Her iki regülasyon terimi için önerilen değerler 0-1000 arasındadır.

max_depth

max_depth hiperparametresi, karar ağaçlarının maksimum derinliğini ayarlar. Bu sayı ne kadar büyük olursa model o kadar az tutucu hale gelir. 0 olarak ayarlanır ise, ağaçların derinliği için bir sınır söz konusu olmaz.

3)KAGGLE SONUÇLARI

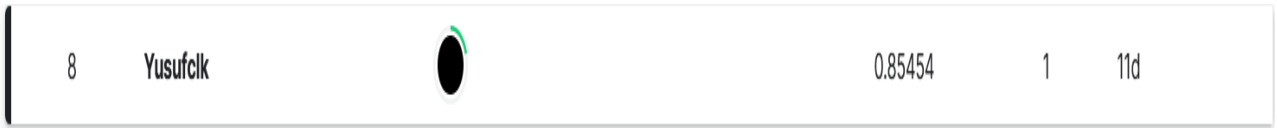
Hiperparametreler

subsample

subsample, tahminleyicileri eğitirken kullanılacak örnek oranının boyutudur. Varsayılan değeri 1'dir, yani örnekleme yoktur ve tüm veriler kullanılır. Örneğin bu parametre, 0.7 olarak ayarlanırsa, gözlemlerin %70'i her artırma yinelemesinde kullanılmak üzere rastgele örneklenir. Aşırı öğrenmeyi önlemeye yardımcı olan bir parametredir.

num_estimators

num_estimators, kullanılacak olan artırılan ağaçların tur sayısını ayarlar. Bu sayı ne kadar büyükse, aşırı öğrenme riski de o kadar artar. Ancak düşük sayılar aynı zamanda düşük performansa da yol açabilir.



VERİ ANALİZİ

Veri setimiz toplam 101 sütun ve 1097 satırdan oluşmaktadır.

Bunlardan 33 tanesi vücuttaki organlardır.Bunların 3 boyutlu

Koordinat verileri sonucu 99 adet sütunumuz olmuştur diğer sütunlar pose_id ve pose bilgisini içerir.

```
data=pd.read_csv("train.csv")
data
```

✓ 1.3s

	pose_id	pose	x_nose	y_nose	z_nose	x
0	0	squats_up	-0.382815	-48.231250	-54.405792	
1	1	situp_down	54.146880	-12.822491	5.564175	
2	2	situp_down	9.891440	-54.147266	85.344970	
3	3	jumping_jacks_up	0.904673	-51.350130	-33.606970	
4	4	jumping_jacks_down	-3.153129	-55.255062	-17.745928	
...	...	...	...	...	...	
1092	1092	situp_up	-25.679585	-47.380875	-5.901453	
1093	1093	jumping_jacks_up	-1.185803	-51.386070	-31.526268	
1094	1094	pullups_down	-4.307419	-49.337822	7.097422	
1095	1095	situp_down	-41.915108	-1.429882	-64.905620	
1096	1096	jumping_jacks_up	-0.599986	-52.802720	-33.876865	

1097 rows × 101 columns

dataframe üzerinde boş değer olup olmadığını konrol ettik. Eksik verinin olmadığı görüldü.

```
df=pd.read_csv("train.csv")

X=df.drop(['pose_id', 'pose'], axis=1)
y=df['pose']
y.value_counts()
```

[16] ✓ 0.1s

```
jumping_jacks_down    151
jumping_jacks_up      145
pullups_down          123
pushups_up            115
squats_up             111
pullups_up            108
squats_down           101
situp_down             82
pushups_down           82
situp_up               79
Name: pose, dtype: int64
```

Dataframe olarak ayırdığımız pose sınıflandırma verilerinin toplam sayılarını gösterdik.

4)Verilen test setindeki sonuçlar

Gridsearch yaparak tüm komşukları test edip uygun değer elde ediyoruz

```
from sklearn.model_selection import GridSearchCV#create new a knn model
knn2 = KNeighborsClassifier()#create a dictionary of all values we want to test for n_neighbors
param_grid = {"n_neighbors": np.arange(1,25)}#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=50)#fit model to data
knn_gscv.fit(X, y)
```

[10] ✓ 2m 36.2s

```
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier
```

```
#check top performing n_neighbors value
knn_gscv.best_params_
```

[11] ✓ 0.9s

```
{'n_neighbors': 1}
```

```
#check mean score for the top performing value of n_neighbors
knn_gscv.best_score_
```

[12] ✓ 0.1s

```
0.7986147186147184
```

Knn ile elde edilen en yüksek score değeri

```
from sklearn.neighbors import KNeighborsClassifier# Create KNN classifier
knn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier to the data
knn.fit(X_train,y_train)
knn.score(X_test, y_test)
```

[13] ✓ 0.1s

```
0.8318181818181818
```

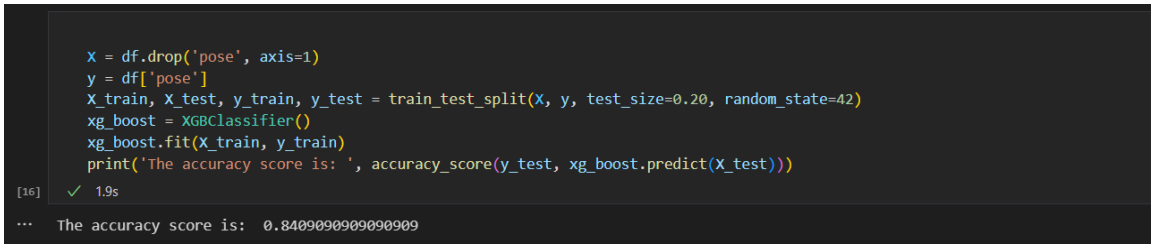
Rff ile elde edilen score değeri

```
rf_model = RandomForestClassifier(criterion='log_loss', n_estimators=100)
rf_model.fit(X_train, y_train)
rf_model.score(X_test, y_test)
```

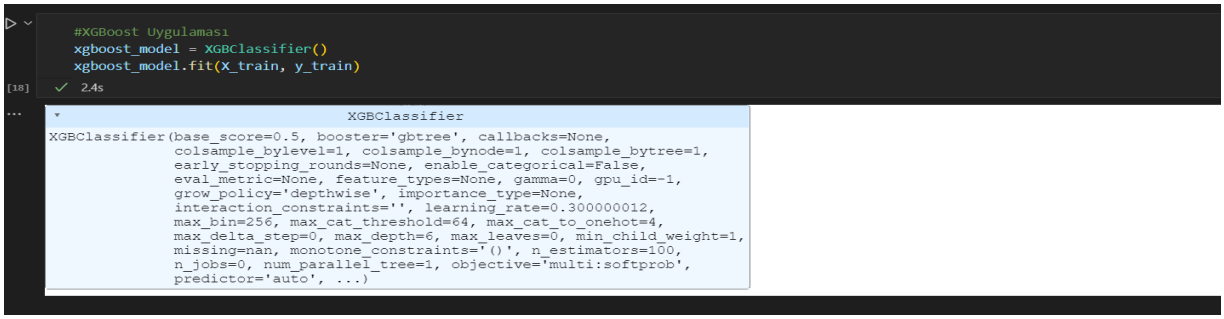
```
0.8272727272727273
```

4)Verilen test setindeki sonuçlar

Xgboost ile tranin seti test ettik ve 84 score elde ettik



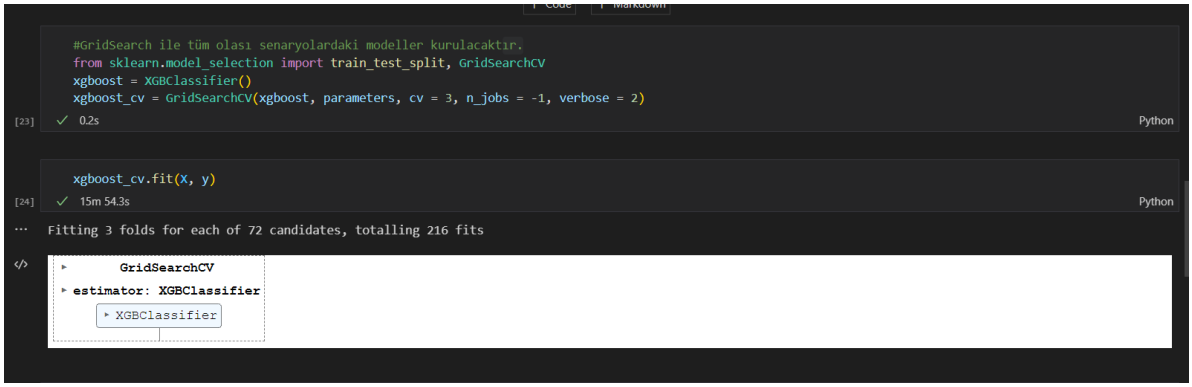
Ardından xgboost için en uygun parametreleri seçip bize verilen test setini gerçek solution dosyamızla test edeceğiz.



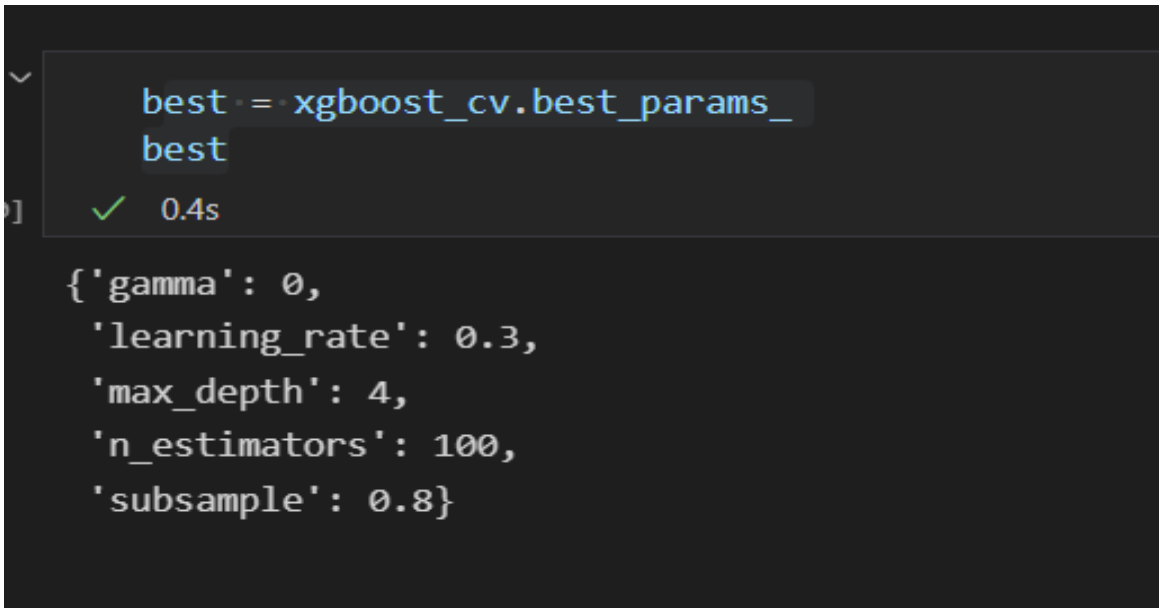
Burdaki parametreler varsayılan parametre değerlerimizdir

4)Verilen test setindeki sonuçlar

Grid search bize en uygun parametreleri seçerek modelimizi Daha güçlü hale getirecektir.



Olası tüm senaryoları denedikten sonra en yüksek tahmin oranını veren parametreler gösterilmiştir.



4)Verilen test setindeki sonuçlar

Güçlendirmiş olduğumuz modelimizi solution dosyasıyla karşılaştırdığımızda başarı oranımız 85.4545 çıkmıştır.

```
xgboost = XGBClassifier(best)
xgboost.fit(X,y)
print('The accuracy score is: ', accuracy_score(solution_df['pose'], xg_boost.predict(test_df.drop('pose_id', axis=1))))
print('The classification report is: ', classification_report(solution_df['pose'], xg_boost.predict(test_df.drop('pose_id', axis=1))))
```

[32] ✓ 2.1s

```
The accuracy score is:  0.8545454545454545
The classification report is:               precision    recall  f1-score   support

           0       0.83      0.89      0.86        38
           1       0.78      0.89      0.83        36
           2       0.83      0.81      0.82        31
           3       0.86      0.70      0.78        27
           4       1.00      0.85      0.92        20
           5       0.88      0.97      0.92        29
           6       0.95      1.00      0.98        20
           7       0.82      0.90      0.86        20
           8       0.89      0.92      0.91        26
           9       0.82      0.64      0.72        28

    accuracy                           0.85       275
   macro avg       0.87      0.86      0.86       275
weighted avg       0.86      0.85      0.85       275
```

Xgboost algoritmasının başka algoritmalarla karşılaştırılması