XGBoost Nasıl Çalışır? Neden İyi Performans Gösterir?

Emre Rıdvan Muratlar 22 Mart 2020 Genel bir bakış, Makine Öğrenmesi, Python, Sınıflandırma, Teori, Uygulama 3 23500 2391 kelime - 16 dakika 127

XGBoost(eXtreme Gradient Boosting), Gradient Boosting algoritmasının çeşitli düzenlemeler ile optimize edilmiş yüksek performanslı halidir. Tianqi Chen ve Carlos Guestrin'in 2016 yılında yayınladıkları "XGBoost: A Scalable Tree Boosting System" adlı makale ile hayatımıza dahil olmuştur. Algortimanın en önemli özellikleri yüksek tahmin gücü elde edebilmesi, aşırı öğrenmenin önüne geçebilmesi, boş verileri yönetebilmesi ve bunları hızlı yapabilmesidir. Tianqi'ye göre XGBoost diğer popüler algoritmalardan 10 kat daha hızlı çalışmaktadır.

Daha az kaynak kullanarak üstün sonuçlar elde etmek için yazılım ve donanım optimizasyon tekniklerini uygulanmıştır. Karar ağacı tabanlı algoritmaların en iyisi olarak gösterilir.

İlerleyen başlıklarda XGBoost algoritmasının nasıl çalıştığı, neden daha iyi performans göstediği ve Python uygulaması ile diğer algoritmalardan ne kadar ayrıştığı incelenecektir.

NASIL ÇALIŞIR?

Çalışma mantığı Gradient Boosting ile oldukça benzerdir. Bu yazıdan önce Gradient Boosting'i yazımı okumanızı öneririm. Yazıya ulaşmak için linke tıklayabilisiniz.

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.

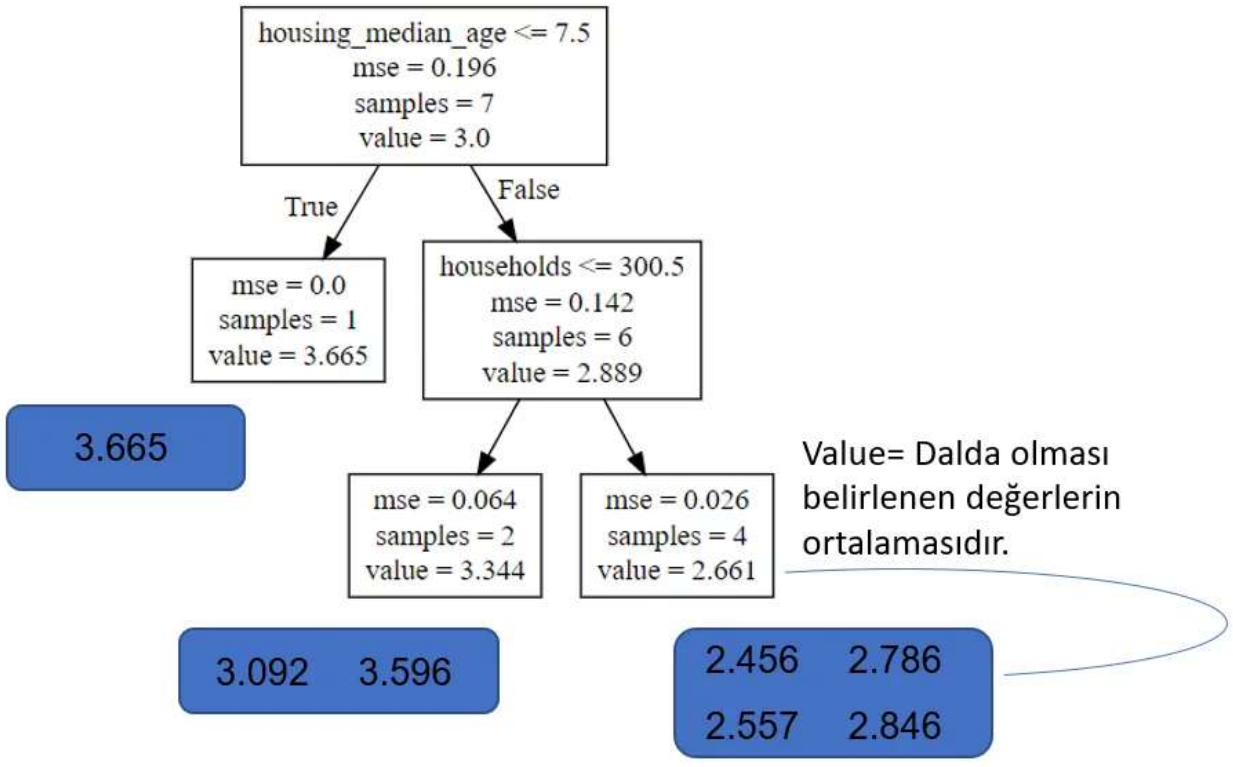
Yapılan bu tahminin ne kadar iyi olduğu modelin hatalı tahminleri(residual) ile incelenir. Hatalar, gözlemlenen değerden tahmin edilen değerin çıkarılması ile bulunmaktadır.

Aşağıdaki örnekte bulunan veri setinde tahminlenmek istenen değişken "median_income"dır. İlk tahmin "initial leaf" kolonunda, tahmin hatası ise "residual1" kolonunda bulunmaktadır.

	housing_median_age	households	median_house_value	median_income	initial_leaf	residual1
9491	22.0	285.0	118800.0	3.5917	0.5	3.0917
11843	15.0	316.0	118800.0	2.9559	0.5	2.4559
11271	21.0	683.0	213300.0	3.2857	0.5	2.7857
19219	19.0	433.0	190300.0	3.0568	0.5	2.5568
14356	11.0	1105.0	159800.0	3.3456	0.5	2.8456

Tablo-1: İlk Tahmin Sonrası Veri Setinin Görünümü

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



Şekil-1: 1. Hata Tahmin Ağaçı

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

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

Formül-
1:
Benzerlik
Skoru
Formülü

Benzerlik skorları hesaplandıktan sonraki soru daha iyi bir tahmin yapılıp yapılamayacağıdır. (Formüldeki lamda(λ), regülarizasyon parametresidir. Bu konuya daha sonra değinilecektir.)

Bu soruyu cevaplamak için olabilecek tüm olasılıklardaki ağaçlar kurulur. Hepsi için benzerlik skorları hesaplanır.

Hangi ağacın daha iyi olduğuna karar vermek için kazanç(gain) hesaplanacaktır.

“Benzerlik skoru” ile dallar değerlendirilirken, “kazanç” ile bütün ağaç değerlendirilmektedir.

Kazanç = Sol Benzerlik Skoru+ Sağ Benzerlik Skoru – Önceki Ağacın Benzerlik Skoru ‘dur.

Hangi ağacın daha yüksek kazanç değerine sahip olduğu belirlenip, ağacın kullanılmasına karar verildikten sonra budama(prune) işlemi başlayacaktır. Budama için “gamma” adı verilen değer seçilir. Gamma, kazanç skoruna getirilen bir değerlendirmedir. Kazanç skoru, gamma skorundan düşük olan dallar budanacaktır. Dolayısıyla gamma’yı arttırmak sadece değerli dalların ağaçta kalmasına ve aşırı öğrenmenin önüne geçilmesine yardımcı olur. Budama işlemi en son daldan yukarı doğru devam eder. Eğer en alttaki dalı budamama kararı alınmışsa üstündeki dallar için inceleme yapmaya gerek yoktur.

Varsayılan gamma değeri 0’dır. Gamma değerinin 0 olması kazanç skorunun eksi değer alması durumunda dalların budanacağı anlamına gelmektedir. Eğer dalların hepsi budanmışsa en iyi tahmin en başta yapılan 0,5’tir.

Bu tanımlar ve hesaplamalardan sonra benzerlik skorunda bahsedilen lambda değeri daha detaylı incelenebilir.

Bu değer modeli hassaslaştırır. Benzerlik Skoru formülüne göre, lambda arttıkça hesaplanan benzerlik skoru, dolayısıyla kazanç skoru düşer. Bunun sonucunda model dal budamada daha cömert davranır. Sadece yüksek skorlu dallar ha

kalabilir, budamadan kurtulabilir.

Lamda değeri varsayılan olarak 1 gelir. Lambda arttıkça öğrenme zorlaşacak ancak aşırı öğrenme(overfitting) azalacaktır.

Lambda, yine formüldeki konumu nedeniyle tek değeri olan dalları daha çok etkiler. Diğer bir deyişle dalda ne kadar fazla değer varsa, lambda benzerlik skorunu o kadar az düşürür. Yani dalda az sayıda değer olmasını istemez. Bu özelliği de aşırı öğrenmenin önüne geçilmesine yardımcı olur.

Dalların çıktısı(output value) = (Daldaki Değerlerin Toplamı)/(Daldaki Değer Sayısı + lambda) olarak hesaplanır.

Lambda arttıkça çıktı değeri olması gerekenden az gelecektir. Yani lambda ne kadar yüksekse doğru tahmine o kadar fazla iterasyon ile ulaşılacaktır.

Yukarıda bahsedilen işlemler (hataları tahminleyen ağaçların kurulması, ağaçlar için benzerlik ve kazanç skorlarının hesaplanması, budama ve model çıktılarının elde edilmesi) ile ilk ağaç tamamlanır.

Gradient Boosting'de olduğu gibi tahmin yapılırken, (ilk tahmin + öğrenim oranı(learning rate) * 1. Ağaç) hesabı ile 2. tahminleme yapılır.

Öğrenim oranı varsayılan olarak 0,3 gelir. XGBoost'da öğrenme oranına eta adı da verilebilmektedir.

2. tahmin ilkine göre daha yakın bir sonuç verecektir.

Aşağıda ikinci tahmin "tree1_result", tahmin hataları ise "residual2" kolonunda bulunmaktadır. Görüldüğü gibi 2. hata değerleri 1. hata değerlerinden oldukça düşüktür. Yani doğru tahmine yaklaşılmıştır.

Tablo-2: İkinci Tahmin Sonrası Veri Setinin Görünümü

Bu şekilde ağaçlar hesaplanarak tahminler düzeltilmeye devam edilir.

Hatalar çok düşük olana ya da belirlenen ağaç sayısına ulaşıncaya kadar bu işlemler devam edecektir.

Varsayılan olarak dallar 6 kez aşağı doğru bölünür.

NEDEN DAHA İYİ PERFORMANS GÖSTERİR?

Gradient Boosting ve XGBoost aynı prensiple çalışmaktadır. Aralarındaki farklar detaylardadır. XGBoost, farklı teknikler kullanarak daha yüksek tahmin başarısı gösterir ve büyük veri setlerinde çalışmak için optimize edilmiştir.

Gradient Boosting'den ayrıştığı başlıca konular aşağıdadır.

- Regülarizasyon
- Budama
- Boş Değerler ile Çalışılabilesi
- Sistem Optimizasyonu

Bu maddelerden regülarizasyon ve budama yukarıda detaylıca anlatılmıştır.

Boş Değerlerle Çalışılabilmesi

Veri setlerinin en büyük problemlerinden biri boş değerlerin olmasıdır. Kimi zaman teknik sorunlarla kimi zaman veri akışının yapısı gereği boş veriler olabilmektedir. Her ne sebeple olursa olsun birçok algortimayı kullanabilmek için bu veriler doldurulmalı ya da ilgili satır veri setinden kaldırılmalıdır. XGBoost'un farklılıklarından biri boş değerler ile çalışılabilmesidir.

XGBoost'un arka planında ilk tahmin varsayılan 0,5 olarak atandığı için tahmin sonucunda hata(residual) değerleri boş verilerin olduğu satırlar için de oluşturulacaktır. İkinci tahmin için oluşturulan karar ağacında, boş verilerin olduğu satırlardaki hata değerleri olabilecek tüm olasılıklar için farklı dallara yerleştirilir ve her durum için kazanç skoru hesaplanır. Skor hangi durumda fazla ise boş değerler o dallara atanacaktır.

Ağırlıklandırılmış Quantile Sketch

XGBoost her değişkene göre kazanç skorunu en yüksek yapacak şekilde olası tüm senaryolarda karar ağaçları kurar. Bu tür algoritmalara "Greedy Algorithm" denir. Enine boyuna büyük veri setlerinde bu işlem çok uzun sürebilir.

XGBoost, verideki her değeri incelemek yerine veriyi parçalara(quantile) böler ve bu parçalara göre çalışır. Parça miktarı arttırıldıkça algoritma daha küçük aralıklara bakacak ve daha iyi tahminleme yapacaktır. Tabi ki bu durum modelin öğrenme süresi de artacaktır.

Parça sayısı varsayılan olarak 33 tanedir.

Bu yaklaşımın problemi tabii ki performans sorunudur. Parçaları belirlemek için her bir kolon sıralanmalı, parçaların sınırları belirlenmeli ve ağaçlar kurulmalıdır. Bu durum yavaşlığa sebep olur.

Sorunu aşmak için "Sketches" adı verilen algoritma kullanılır. Amacı parçaları bulmak için yakınsama yapmasıdır.

XGboost Ağırlıklandırılmış Sketches Algoritması ile parçaların sınırlarını belirler.

Ağırlık = Önceki Değer * (1 – Önceki Değer) olarak belirlenir. Ağırlık ne kadar fazlaysa tahmin o kadar kararsızdır. Parçalar bu ağırlıklara göre belirlenir. Ağırlıklar yaklaşık olacak şekilde parçalara bölünür. Bu sayede kararsız tahmin değerlerinin olduğu dallar daha küçük aralıklara bölünmüş olur. Bu durum daha doğru tahminlemeye yardımcı olacaktır.

Sistem Optimizasyonu

Bilgisayarlarımızda hard disk, RAM ve önbellek gibi farklı bellek türleri bulunmaktadır. Ön bellek, bellekler arasında en hızlı kullanılan ancak en küçük alana sahip olandır. Bir programın hızlı çalışması isteniyorsa ön bellek maksimum seviyede kullanılmalıdır.

XGBoost benzerlik skoru ve ağaç çıktılarını(output value) ön bellekte hesaplatır. Bu sebeple hızlı hesaplamalar yapılabilmektedir.

UYGULAMA

Uygulamada Kaggle web sitesinde bulunan “Human Activity Recognition” veri seti kullanılmıştır. 30 gönüllü denek ile oluşturulan bu veri setinde, gönüllülere akıllı telefonlar üzerlerindeyken yürüme, merdiven çıkma, merdiven inme, oturma, kalkma ve uzanma hareketleri yaptırılmıştır. Veri setinin amacı elde edilen verilerle insan davranışlarının tahminlenmesidir.

Uygulama Windows 10 işletim sisteminde, Python ile Jupyter Notebook kullanılarak yapılmıştır.

```
#Uygulamada kullanılacak kütüphaneler tanımlanmıştır.
import os
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier


from warnings import filterwarnings
filterwarnings('ignore')
```

```
df = pd.read_csv('Simplified_Human_Activity_Recognition.csv')
df = df.drop(labels=['rn'], axis=1)
```

```
df.head()
```

```
df.shape
```

```
(3609, 562)
```

Datada 3.609 satır ve 562 sütun bulunmaktadır.

```
#Bağımlı değişken y, bağımsız değişkenler X'e tanımlanmıştır.
y = df[['activity']]
X = df.iloc[:,1:]
```

```
#Veri seti %70 train, %30 test olarak bölünmüştür.
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, rand
```

XGBoost model performansı, Gradient Boosting ve Random Forest algoritmaları ile model öğrenme süresi ve doğru tahmin oranı üzerinden karşılaştırılacaktır.

```
#XGBoost Uygulaması
xgboost_model = XGBClassifier()
xgboost_model.fit(X_train, y_train)
```

```
#XGBoost varsayılan ayarları gösterilmiştir.
xgboost_model
```

```
#Gradient Boosting Uygulaması
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)
```

```
#Random Forest Uygulaması
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
```

Model performans karşılaştırması için XGBoost, Gradient Boosting(GBM) ve Random Forest ile karşılaştırılmıştır.

Karşılaştırmanın grafiği aşağıdadır. Bar grafik modelin öğrenim(training) süresini, çizgi grafik ise doğru tahmin oranını göstermektedir.

Sonuçlar incelendiğinde XGBoost'un en yüksek tahmin oranını elde ettiği ve Gradient Boosting'den yaklaşık 20 kat data hızlı çalıştığı görülmektedir.

Model Hiperparametre Ayarı

Algoritmalardan en yüksek verimi almak için hiperparametrelerin çalışmaya özgün şekilde ayarlanması gerekmektedir. Bu durum model tahmin gücünü arttırmak, modelin öğrenme süresini optimize etmek ve/veya aşırı öğrenmenin önüne geçmek için yapılır.

```
#Modelde kullanılabilecek parametreler tanımlanmıştır.
parameters = {
        'n_estimators': [100, 500],
        'subsample': [0.8, 1.0],
        'gamma' : [0,1,5],
        'max_depth': [3, 4, 5],
```

```
            'learning_rate': [0.1, 0.3]}
```

```
#GridSearch ile tüm olası senaryolardaki modeller kurulacaktır.
xgboost = XGBClassifier()
xgboost_cv = GridSearchCV(xgboost, parameters, cv = 3, n_jobs = -1, verbose = 2)
```

```
xgboost_cv.fit(X_train, y_train)
```

```
#En yüksek tahmin oranını veren parametreler gösterilmiştir.
best = xgboost_cv.best_params_
best
```

```
{'gamma': 0,
 'learning_rate': 0.3,
 'max_depth': 4,
 'n_estimators': 500,
 'subsample': 0.8}
```

```
#Model tekrar en iyi tahmin oranını veren parametreler ile öğrenim sürecine gire
xgboost = XGBClassifier(best)
xgb_tuned =  xgboost.fit(X_train,y_train)
```

Model ayarları için birçok parametre mevcuttur. Yazıda daha önce bahsedilmeyen ve modelde kullanılan paramatreler; n_estimators, subsample ve max_depth'dir. n_estimators; modelde kurulacak ağaç sayısı, subsample; herbir ağacı oluşturmak için alınan satır oranı, max_depth ağacın derinliğini ifade etmektedir. XGBoost ile ilgili tüm parametrelere link'ten ulaşabilirsiniz.

SONUÇ

Yukarıda anlatılan sebepler ve uygulama göz önüne alındığında XGBoost hem tahmin gücü hem de algoritma çalışma hızı açısında mevcuttaki en iyi algoritmalardan biridir. Ancak her ne kadar güçlü bir algoritma olsa da işin sırrı hiperparametreleri optimize etmektedir. Düşük öğrenim oranı ve gamma ile başlayıp, yavaş yavaş bu parametreler güncellenerek optimum sayılar bulunabilir.

Bu denemeler için "GridSearchCV" çok büyük kolaylık sağlamaktadır. Eklenen hiperparametrelerin tüm kombinasyonları kadar model kurulup test edilece[illegible] önüne alınmalıdır. Aksi taktirde uzun saatler sürebilecek bir çalışma başlatıl[illegible].

XGBoost çok iyi bir algortima olsa da her zaman en iyi sonucu verecek diye bir kaide yoktur. Mümkün olduğunca diğer algoritmalar da denenmelidir.

Ayrıca, XGBoost'tan sonra geliştirilen Catboost ve Light GBM algoritmalarının daha yüksek performansta çalıştıklarını iddia etmektedir. Bu algoritmalar bundan sonraki yazılarda incelenecek ve performans karşılaştırmaları yapılacaktır. Takipte kalın. 😊

KAYNAKÇA

XGBoost: A Scalable Tree Boosting System – Tianqi Chen, Carlos Guestrin

A Learning Based Demand Classification Service with Using XGBoost in Institutional Area – Çağrı Gürakın

XGBoost ve Karar Ağaçları Tabanlı Algoritmaların Diyabet Veri Setleri Üzerindeki Uygulaması – Gülçin Yangın

https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

https://statquest.org/

Veri seti: https://www.kaggle.com/mboaglio/simplifiedhuarus

Emre Rıdvan Muratlar

Related Posts:

- MS SQL Sorgularının Söz Dizimi ve Performans Ölçümü

boosting extreme gradient boosting machine learning makine öğrenmesi Python tuning xgboost xgboosting

Emre Rıdvan Muratlar

2016 yılından bu yana finans sektöründe veri bilimi üzerine çalışmaktadır. Yıldız Teknik Üniversitesi İstatistik bölümü doktora öğrencisidir.

Tüm yazılarını gör

Şunlar da ilginizi çekebilir

İlgili içerik

dbt (DataBuildTool) ile Veri Analitiği Yolculuğunda Yeni Bir Dönem

İlgili içerik

Oppenheimer Filmi ve Nükleer Bombaların Gölgesinde Veri Biliminin Önemi

Önceki yazı

R ile Veri Bilimi ve Machine Learning (35 Saat)

Sonraki yazı ▸

Bokeh ile İnteraktif Veri Görselleştirme -1

Yorumlar (3 yorum)

sezgin ●

Yanıtla

27 Mart 2020 - 04:53

İlginç bir algoritma. Denemeye değer.

kml ●

Yanıtla

9 Eylül 2022 - 09:22

Denemeye değerden de öte, “Hiç bir şey bilmiyorsan XGBoost kullan.” demişler.

Elif Derya ●

12 Mayıs 2021 - 00:09

Yanıtla

Denemek için sabırsızlanıyorum, emeğinize sağlık 🙂

Bir yanıt yazın

E-posta adresiniz yayınlanmayacak. Gerekli alanlar * ile işaretlenmişlerdir

Yorumunuzu yazın *

Adınız *

E-posta adresiniz *

Website

Daha sonraki yorumlarımda kullanılması için adım, e-posta adresim ve site adresim bu tarayıcıya kaydedilsin.

Yorum gönder

Son Yorumlar

Burçin Minitab’de X-R Kontrol Grafiği Oluşturma

Nihat Tensorflow Lite Modeli ile Colab Üzerinden Görüntü Sınıflandırma: Derin Öğrenme Uygulaması

melih Microsoft Excel’ de VBA Yardımıyla Otomatik Olarak E-posta Gönderimi

Hakan NAKİŞ Microsoft Excel’ de VBA Yardımıyla Çalışma Sayfalarını Ayrı Ayrı PDF Olarak Kaydetme

kenan R ile Makine Öğrenmesi Uygulamaları: Doğrusal Regresyon

Emre Rıdvan Muratlar

2016 yılından bu yana finans sektöründe veri bilimi üzerine çalışmaktadır. Yıldız Teknik Üniversitesi İstatistik bölümü doktora öğrencisidir.

Son Yorumlar

Burçin Minitab'de X-R Kontrol Grafiği Oluşturma

Nihat Tensorflow Lite Modeli ile Colab Üzerinden Görüntü Sınıflandırma: Derin Öğrenme Uygulaması

melih Microsoft Excel' de VBA Yardımıyla Otomatik Olarak E-posta Gönderimi

Hakan NAKIŞ Microsoft Excel' de VBA Yardımıyla Çalışma Sayfalarını Ayrı Ayrı PDF Olarak Kaydetme

kenan R ile Makine Öğrenmesi Uygulamaları: Doğrusal Regresyon

Bunları Da Oku

1 **3 Adımda CRM (Müşteri İlişkileri Yönetimi)**

2 **AB Testi Nedir ? İstatistiksel A/B Testleri Nasıl Yapılır?**

3 **Aşırı Öğrenme (Overfitting)**

VBO Ekibine Özel

VBO Dosya Merkezi

VBO Ekip Eğitimi