XGBoost Algoritması

Ali Kerem Şimşek

XGBoost Algoritması

XGBoost (eXtreme Gradient Boosting)

- A Scalable Tree Boosting System (Ölçeklenebilir Ağaç Bossting Sistemi)
- XGBoost, GBM 'in hız ve tahmin performansını arttırmak üzere optimize edilmiş; ölçeklenebilir ve farklı platformlara entegre edilebilir versiyonudur. (Tianqi Chen – 2014)

Peki GBM nedir?

GBM (Gradient Boosting Machines)

- Artık optimizasyonuna dayalı çalışan bir ağaç yöntemidir.
- Ağaç yöntemlerine boosting ve gradient descent 'in uygulanmasıdır.

Boosting nedir? Gradient Descent nedir?

1. **Boosting**, zayıf öğrenicileri(Weak Learner) güçlü öğreniciye(Strong Learner) dönüştürme yöntemidir
2. **Gradient Descent**, en popüler optimizasyon algoritmalarından birisidir.
3. **Optimizasyon**, bir fonksiyonu minimize ya da maksimize etmek amacı ile gerçek veya tamsayı değerlerini bir fonksiyona yerleştirerek sistematik olarak bir problemi incelemek veya çözmek anlamına gelmektedir.

- GBM 'in temelleri AdaBoost (Adaptive Boosting) 'a dayalıdır.

Peki ya AdaBoost nedir?

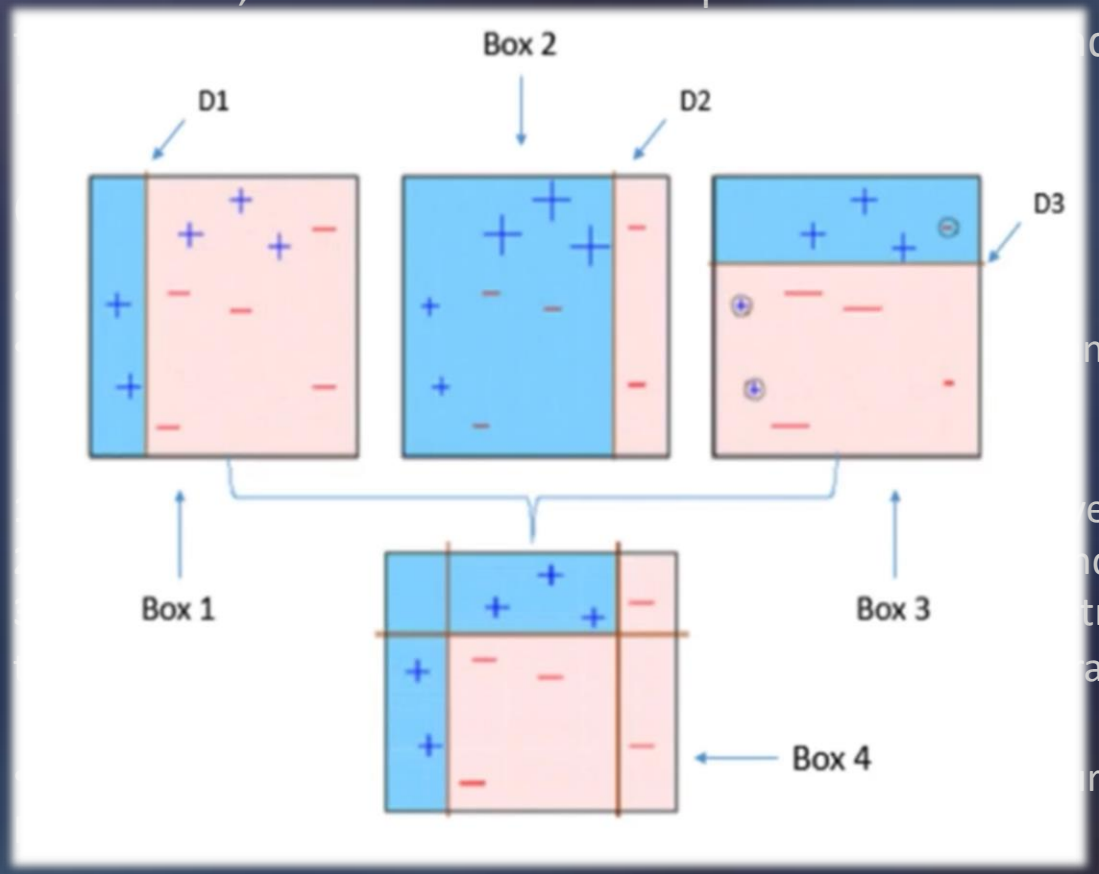
XGBoost Algoritması

XGBoost (eXtreme Gradient Boosting)

AdaBoost (Adaptive Boosting)

- A Scalable Tree Boosting System (Ölçeklenebilir Ağaç Bossting Sistemi)
- Zayıf sınıflandırıcıların bir araya gelerek güçlü bir sınıflandırıcı oluşturması fikrine dayanır.
- XGBoost, GBM 'in hız ve tahmin performansını arttırmak üzere optimize edilmiş; ölçeklenebilir ve

qi Chen – 2014)



nmasıdır.

e(Strong Learner) dönüştürme yöntemidir.

dan birisidir.

tmek amacı ile gerçek veya

ak bir problemi incelemek veya çözmek anlamına gelmektedir.

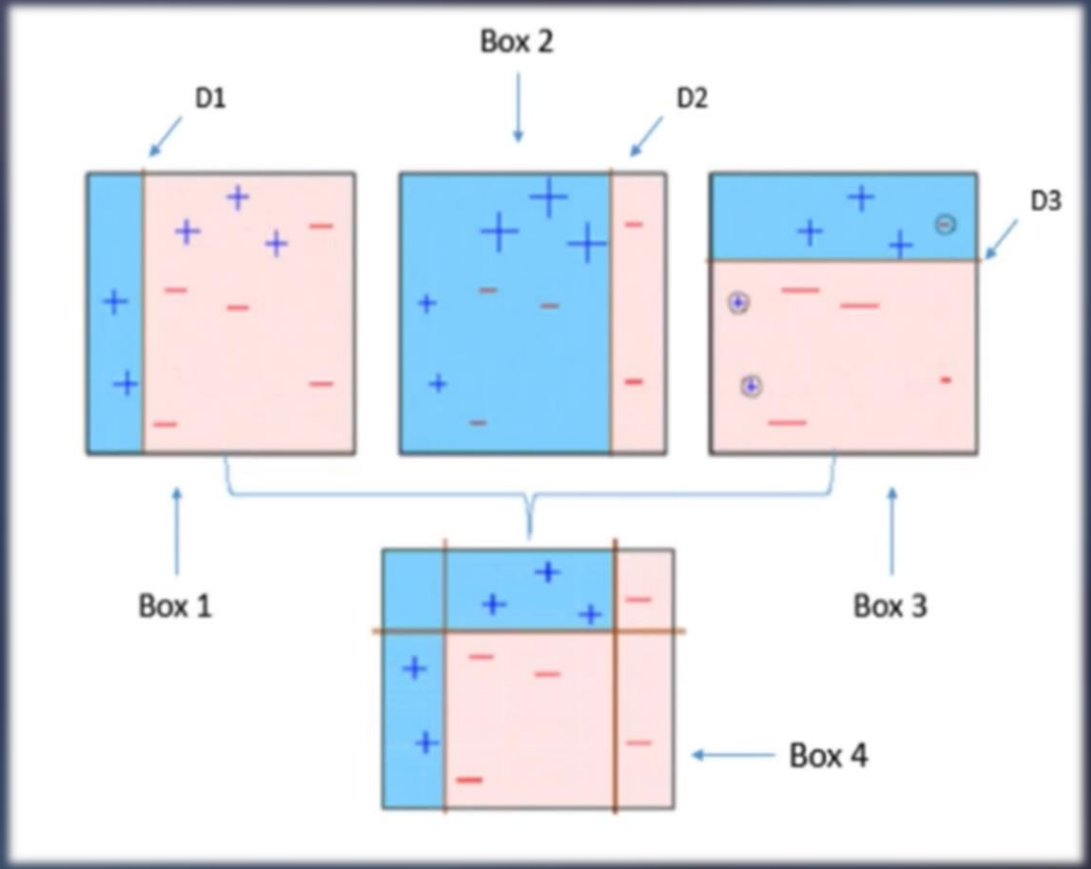
r.

- Box1 : Mavinin içindeki maviler doğru sınıflandırılmış, kırmızının içindeki kırmızılar da doğru sınıflandırılmış fakat kırmızının içindeki maviler yanlış sınıflandırılmış verilerdir.
- Yanlış sınıflandırılan noktalara ağırlık verildiğinde sınıflandırma değişti, Box2 'deki gibi oldu. Fakat hala yanlışlarımız var.
- Bir kez daha denediğimizde Box3 'de yine yanlışlar olduğunu görüyoruz. Giderek yanlışlarımız azaldı.
- Üç tane sınıflandırıcı ağırlık olarak bir araya getirildiğinde ise Box4 karşımıza çıkıyor. Topluluk öğrenme yaklaşımı var. Fakat asıl olayımız hatalardan öğrenme.

XGBoost Algoritması

AdaBoost (Adaptive Boosting)

- Zayıf sınıflandırıcıların bir araya gelerek güçlü bir sınıflandırıcı oluşturması fikrine dayanır.



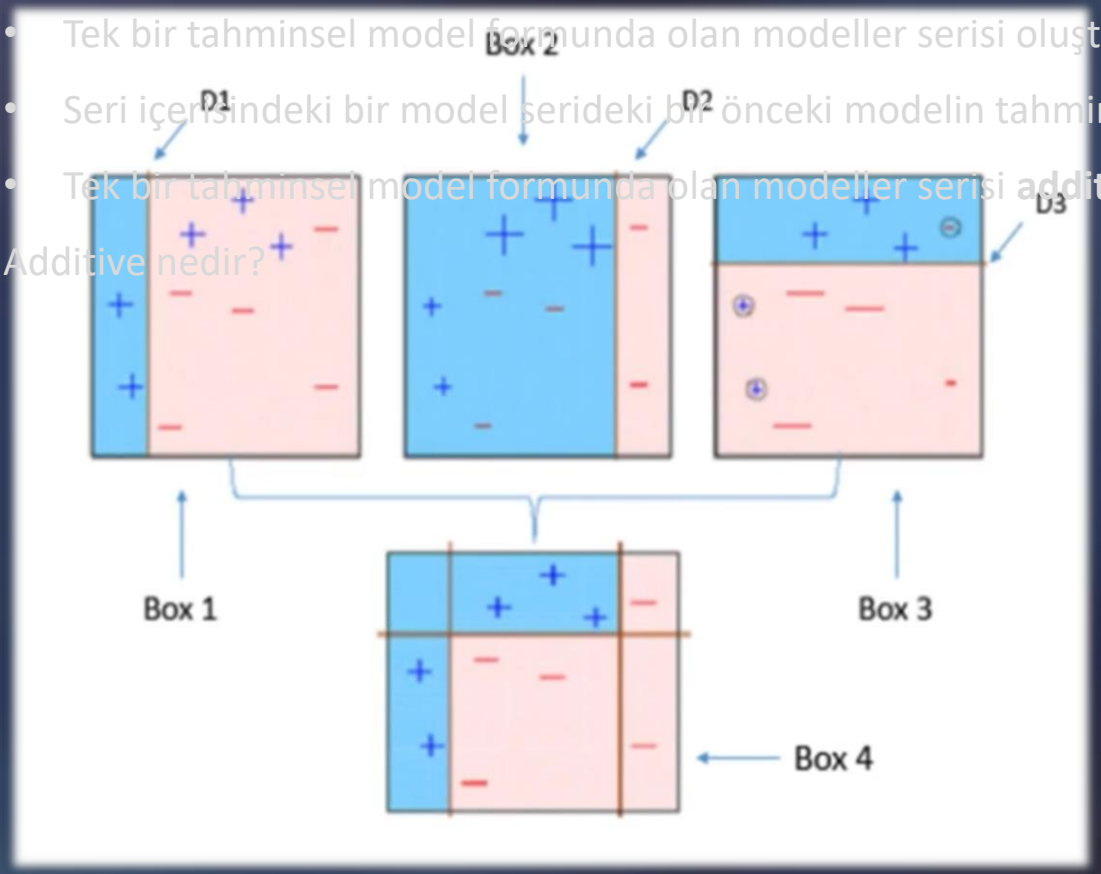
- Box1 : Mavinin içindeki maviler doğru sınıflandırılmış, kırmızının içindeki kırmızılar da doğru sınıflandırılmış fakat kırmızının içindeki maviler yanlış sınıflandırılmış verilerdir.
- Yanlış sınıflandırılan noktalara ağırlık verildiğinde sınıflandırma değişti, Box2 'deki gibi oldu. Fakat hala yanlışlarımız var.
- Bir kez daha denediğimizde Box3 'de yine yanlışlar olduğunu görüyoruz. Giderek yanlışlarımız azaldı.
- Üç tane sınıflandırıcı ağırlık olarak bir araya getirildiğinde ise Box4 karşımıza çıkıyor. Topluluk öğrenme yaklaşımı var. Fakat asıl olayımız hatalardan öğrenme.

XGBoost Algoritması

GBM (Adaptif Güçlendirme);

- Zayıf sınıflandırıcıların bir araya gelerek güçlü bir sınıflandırıcı oluşturması fikrine dayanır.
- Tek bir tahminsel model formunda olan modeller serisi oluşturur.
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının üzerine kurularak (fit) oluşturulur.
- Tek bir tahminsel model formunda olan modeller serisi **additive** şeklinde kurulur.

Additive nedir?



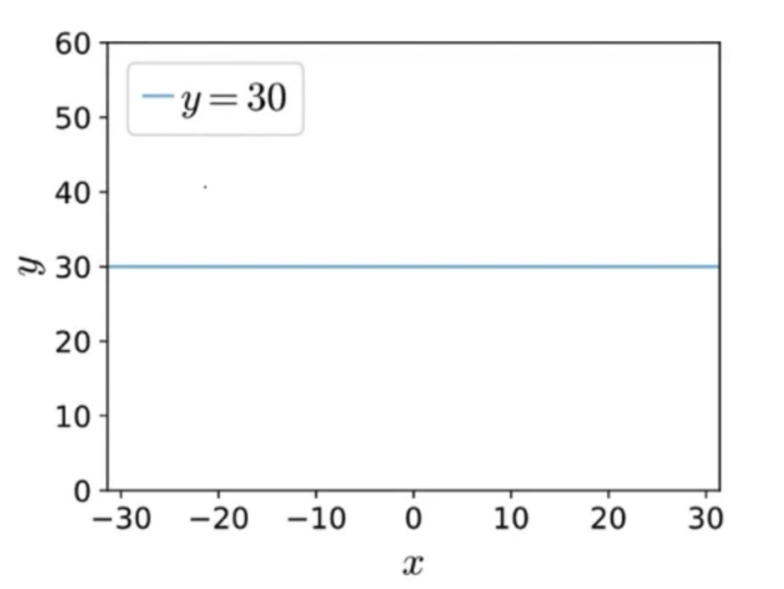
- Box1 : Mavinin içindeki maviler doğru sınıflandırılmış, kırmızının içindeki kırmızılar da doğru sınıflandırılmış fakat kırmızının içindeki maviler yanlış sınıflandırılmış verilerdir.
- Yanlış sınıflandırılan noktalara ağırlık verildiğinde sınıflandırma değişti, Box2 'deki gibi oldu. Fakat hala yanlışlarımız var.
- Bir kez daha denediğimizde Box3 'de yine yanlışlar olduğunu görüyoruz. Giderek yanlışlarımız azaldı.
- Üç tane sınıflandırıcı ağırlık olarak bir araya getirildiğinde ise Box4 karşımıza çıkıyor. Topluluk öğrenme yaklaşımı var. Fakat asıl olayımız hatalardan öğrenme.

XGBoost Algoritması

GBM ‘e tekrar döndüğümüzde;

- Hatalar/artıklar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur.
- Tek bir tahminsel model formunda olan modeller serisi oluşturur.
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının üzerine kurularak (fit) oluşturulur.
- Tek bir tahminsel model formunda olan modeller serisi **additive** şekilde kurulur.

Additive nedir?



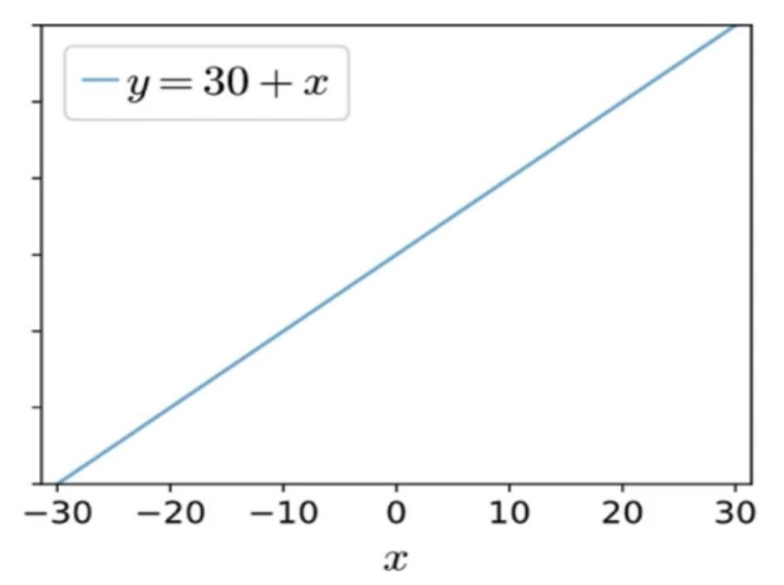
- y = 30 şeklinde sabit bir fonksiyonumuz var. X ‘ e ne girersek girelim, sonucu değiştiremiyoruz. Değiştirmek istersek ne yapmamız gerekiyor? y = 30 + x yapabiliriz.

XGBoost Algoritması

GBM ‘e tekrar döndüğümüzde;

- Hatalar/artıklar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur.
- Tek bir tahminsel model formunda olan modeller serisi oluşturur.
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının üzerine kurularak (fit) oluşturulur.
- Tek bir tahminsel model formunda olan modeller serisi **additive** şekilde kurulur.

Additive nedir?



- Artık x kaç olursa, y ona göre şekillenecektir.
- Şimdi amacım x ‘in etkisini daha da arttırmak/detaylandırmak.

Ne yapabilirim?

- X ‘in karesini alabiliriz mesela, ya da sinx, cosx gibi değerler ekleyebiliriz.

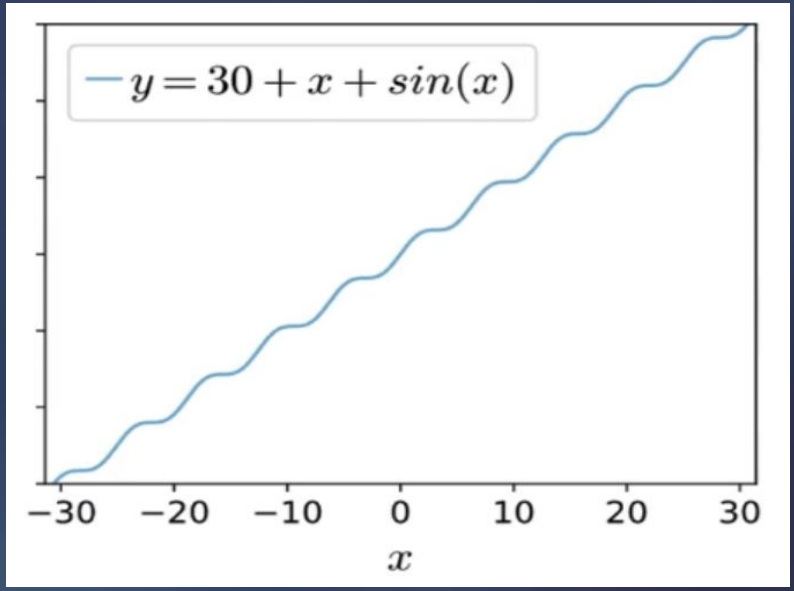
Yani özetle x ‘e bağlı terim ekleyebiliriz.

XGBoost Algoritması

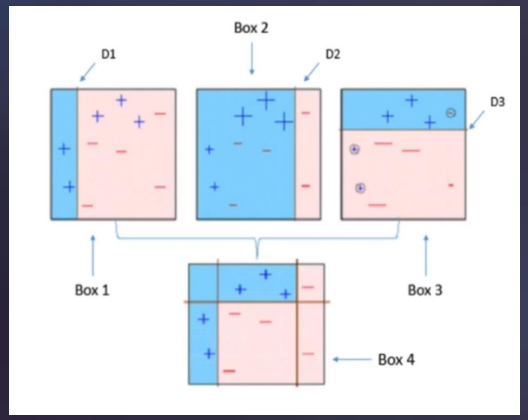
GBM ‘e tekrar döndüğümüzde;

- Hatalar/artıklar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur.
- Tek bir tahminsel model formunda olan modeller serisi oluşturur.
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının üzerine kurularak (fit) oluşturulur.
- Tek bir tahminsel model formunda olan modeller serisi **additive** şekilde kurulur.

Additive nedir?



- Artık gözlemleyeceğiniz üzere fonksiyon daha da hassaslaştı.
- Peki neden bunları öğrendik?
- Az önce de gördüğümüz gibi ağaç yönetimi, modeli belirli bölgelere ayırıyordu;



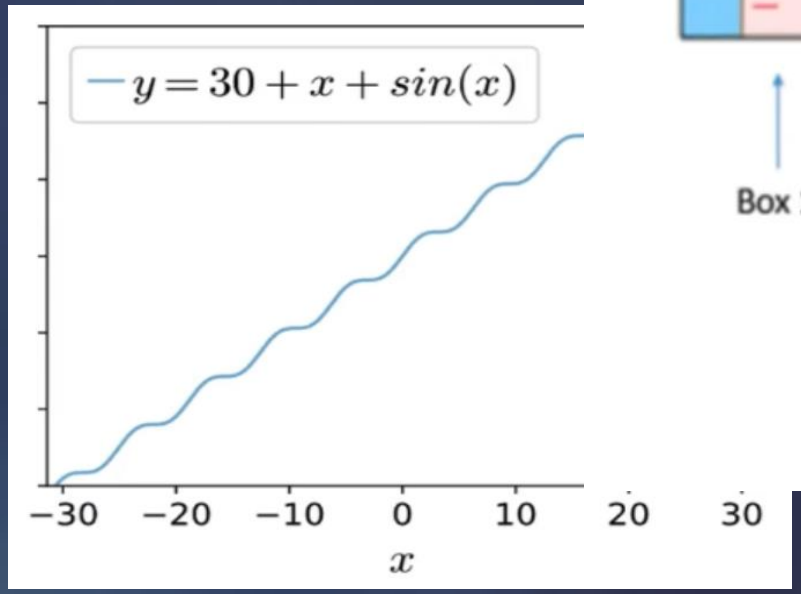
XGBoost Algoritması

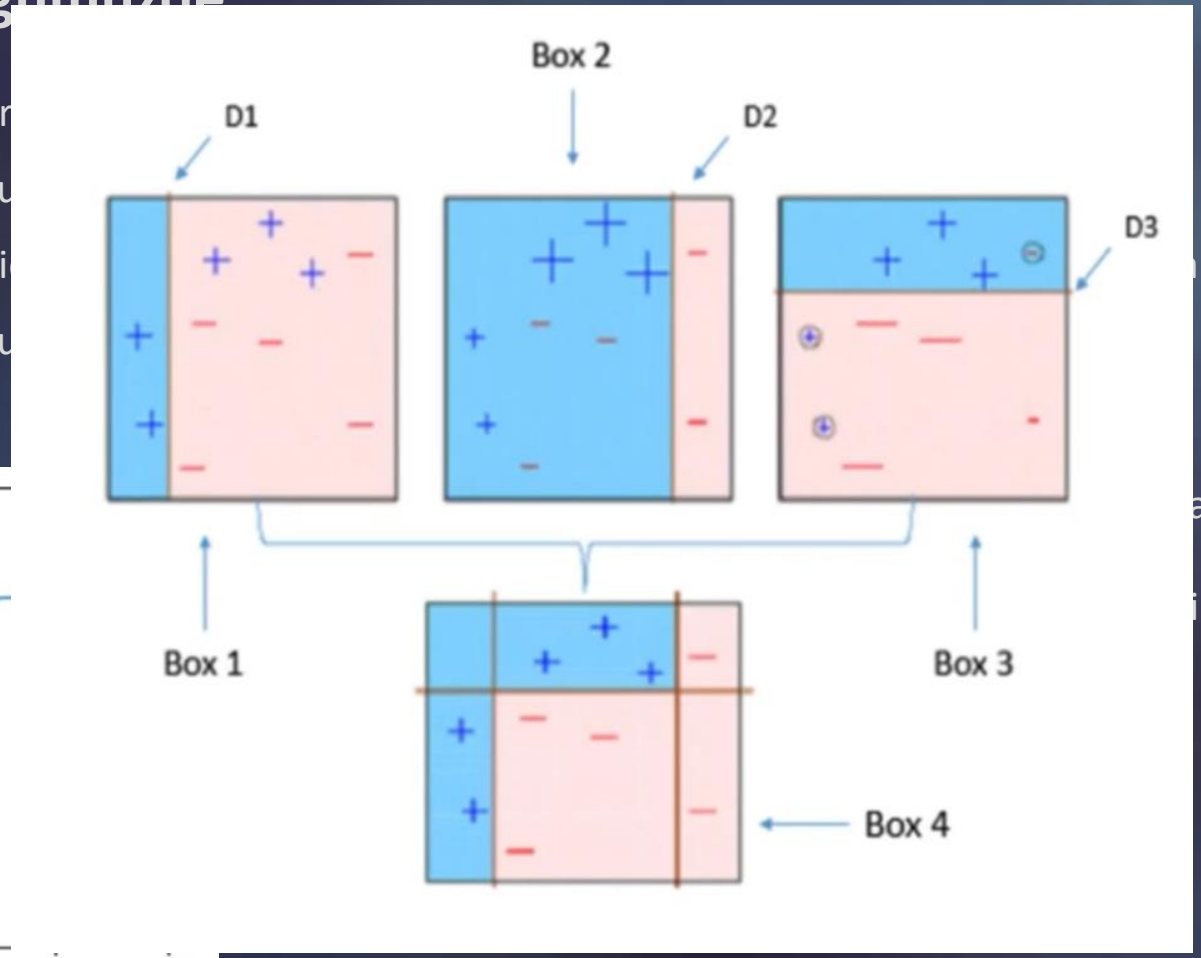
GBM ‘e tekrar döndüğümüzde:

- Hatalar/artıklar üzerine tek bir
- Tek bir tahminsel model formu
- Seri içerisindeki bir model seri
- Tek bir tahminsel model formu

rak (fit) oluşturulur.

Additive nedir?

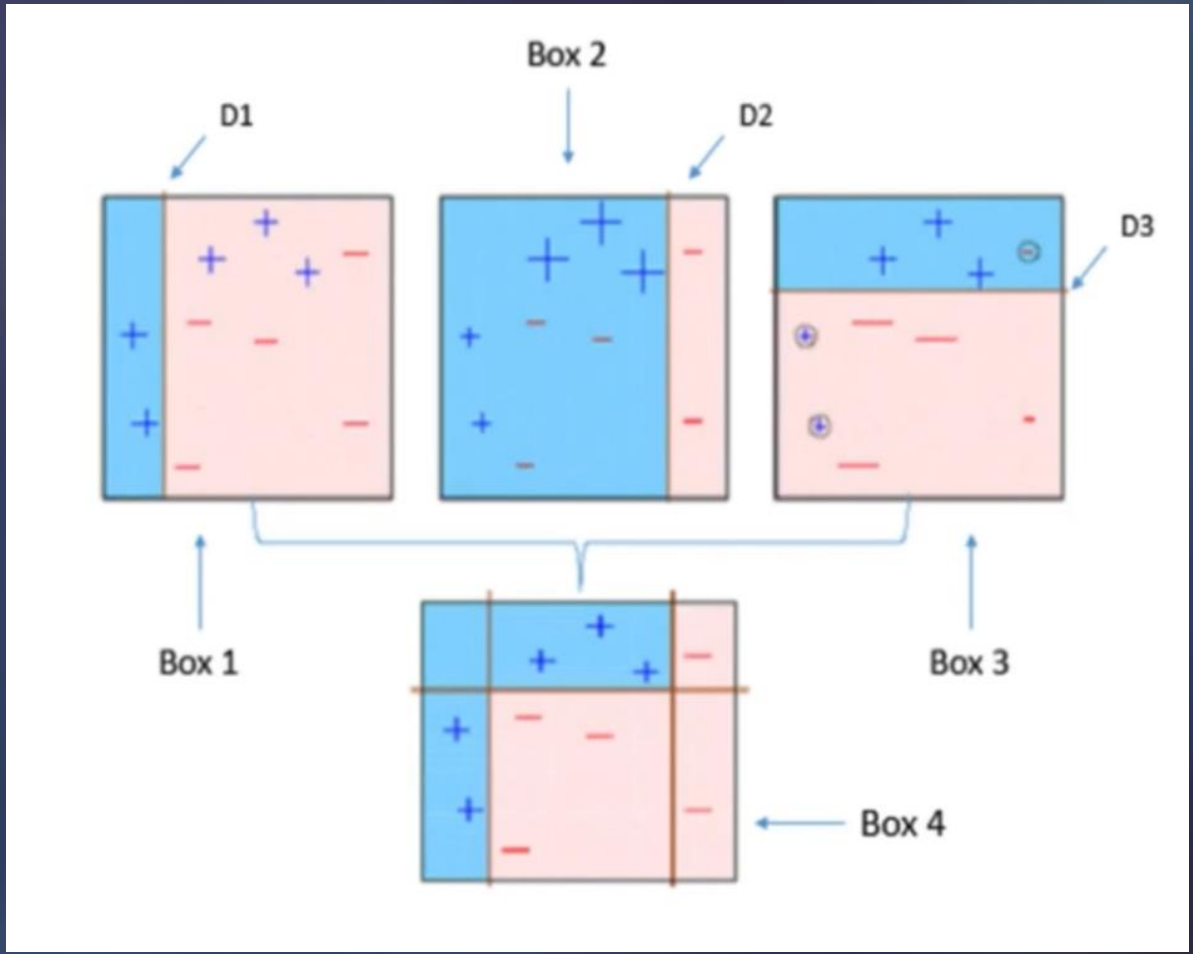




assaslaştı.

belirli bölgelere ayırıyordu;

XGBoost Algoritması

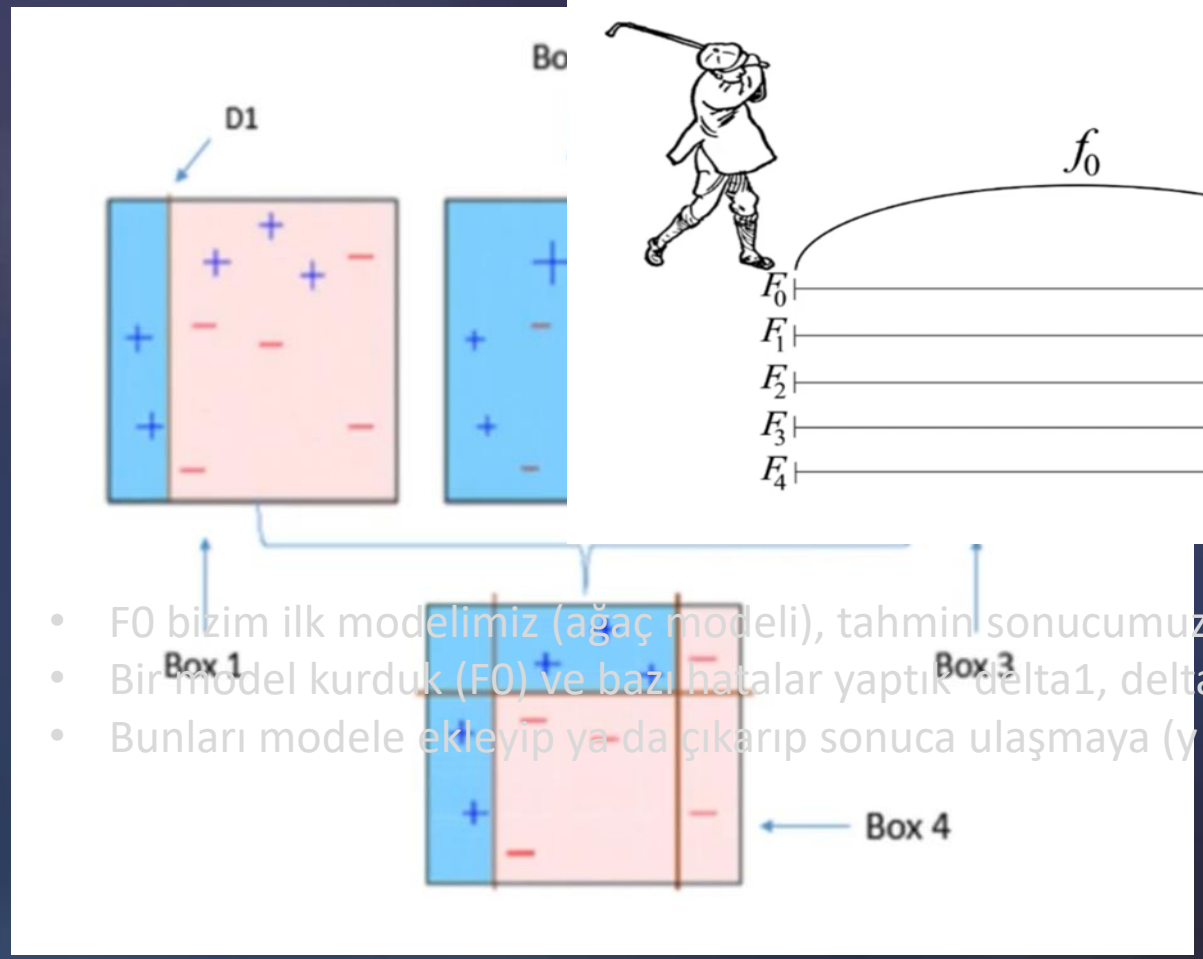


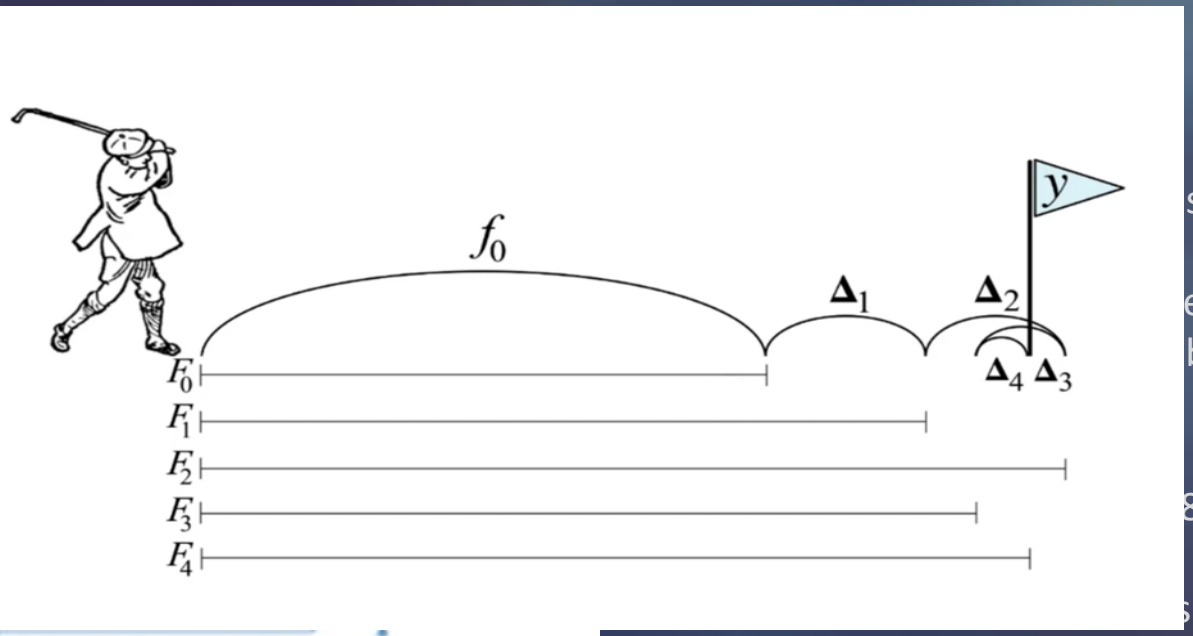
- Burada ki ayırımları daha hassas yapsak?

- Sabit olan ortalama değerine eğer (y = 30 gibi) additive model ile dokunabilirsek, ki burada x değerlerimiz artıklar/hatalar olacaktır, bu durumda daha başarılı tahminler elde edebiliriz.
- y = 30 mu olmalı? 29 mu? 28 mi? Bunu nasıl modelleyebilirim?
- Ağaç yönteminde dallanmasına budaklanmasına en fazla dokunabiliyorduk. Artık hataları/artıkları modelleyebiliyoruz. 30 olan tahmin sonucumuzu, artıkları modelleyerek çıkan yeni (artık) tahmin modelimizi eklediğimizde 30+(-5) 'den 25 yeni tahmin sonucumuz olacaktır.

! Amacımız tahmin modeline ekleme veya çıkarma yaparak optimum sonuca gitmek.

XGBoost Algoritması





ssas yapsak?

e eğer (y = 30 gibi) additive
burada x değerlerimiz
durumda daha başarılı

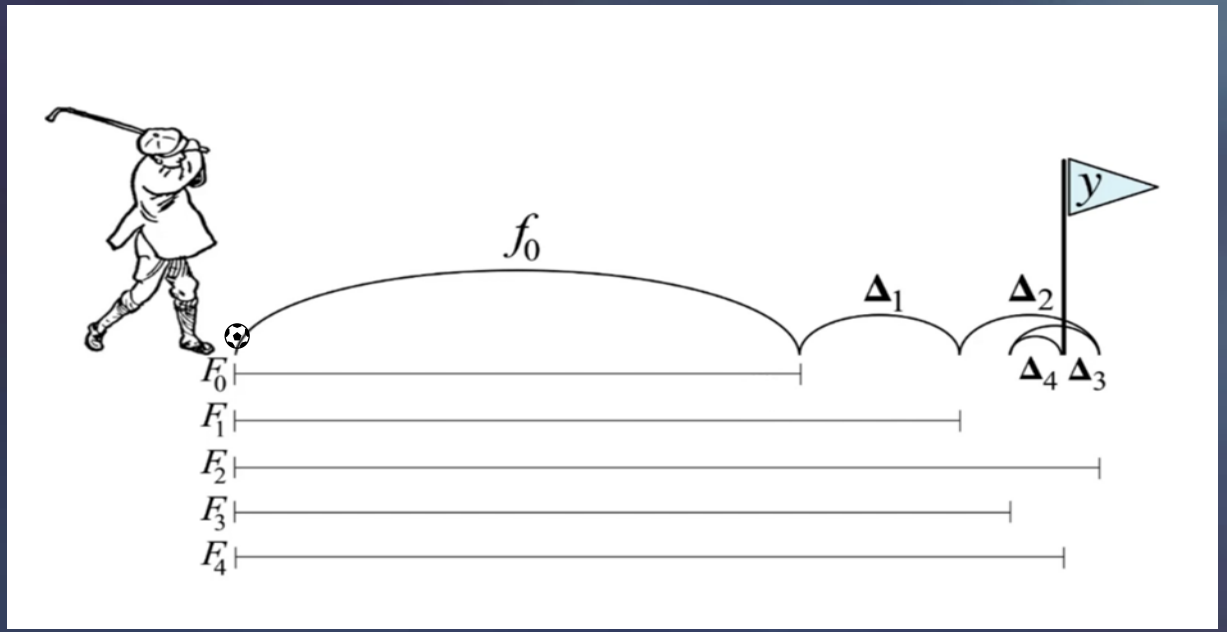
8 mi? Bunu nasıl

sına budaklanmasına en fazla
dokunabiliyorduk. Artık hataları/artıkları modelleyebiliyoruz.
30 (ilk tahmin sonucumuz, ağaçları modelleyerek çıkan
yeni tahmin sonucumuz olacaktır.

- F0 bizim ilk modelimiz (ağaç modeli), tahmin sonucumuz olacak. (Model; belirli filtrelere göre ortalama değer)
- Bir model kurduk (F0) ve bazı hatalar yaptık delta1, delta2 gibi.
- Bunları modele ekleyip ya da çıkarıp sonuca ulaşmaya (y 'nin gerçek değerine ulaşmaya) çalışacağız.

! Amacımız tahmin modeline ekleme veya çıkarma yaparak optimum sonuca gitmek.

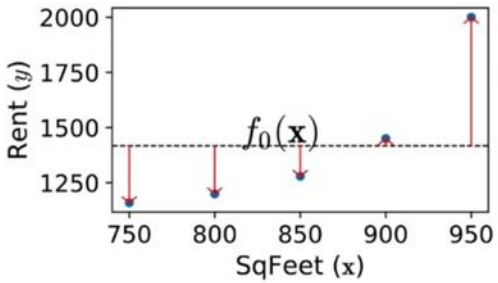
XGBoost Algoritması



- F0 bizim ilk modelimiz (ağaç modeli), tahmin sonucumuz olacak. (Model; belirli filtrelere göre ortalama değer)
- Bir model kurduk (F0) ve bazı hatalar yaptık delta1, delta2 gibi.
- Bunları modele ekleyip ya da çıkarıp sonuca ulaşmaya (y ‘nin gerçek değerine ulaşmaya) çalışacağız.

XGBoost Algoritması

sqfeet	rent	F_0	$y - F_0$
750	1160	1418	-258
800	1200	1418	-218
850	1280	1418	-138
900	1450	1418	32
950	2000	1418	582

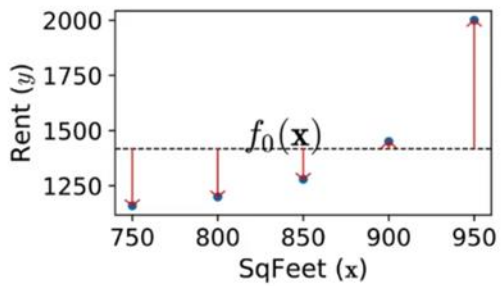


F0 : İlk modelimiz. İlk tahminlerimiz. | **Sqfeet** : Metrekare bilgisi. | **Rent** : Kira bilgisi.

- Normalde evin kirasını tahmin etmeye çalışırsak, rentlerin ortalamasını alırız, sqfeetlerin ortalamasını alırız ve ortalama bir metrekare de ortalama fiyatı buluruz. AdaBoost 'ta gördüğümüz o tablolar gelsin aklınıza.
- F0 : Bu ilk modelimiz (ağaç yok yani), tahmin sonucumuz olacak değer. Gerçek değerlere göre ortalama değer)
- Bir modelimiz var (F0) ve bazı hatalar yaptık delta1, delta2 gibi.
- Bunları modele ekleyip ya da çıkarıp sonuca ulaşmaya (y 'nin gerçek değerine ulaşmaya) çalışacağız.

XGBoost Algoritması

sqfeet	rent	F_0	$\mathbf{y} - F_0$
750	1160	1418	-258
800	1200	1418	-218
850	1280	1418	-138
900	1450	1418	32
950	2000	1418	582



F0 : İlk modelimiz. İlk tahminlerimiz. | **Sqfeet** : Metrekare bilgisi. | **Rent** : Kira bilgisi.

- Normalde evin kirasını tahmin etmeye çalışırsak, rentlerin ortalamasını alırız, sqfeetlerin ortalamasını alırız ve ortalama bir metrekare de ortalama fiyatı buluruz. AdaBoost 'ta gördüğümüz o tablolar gelsin aklınıza.
- y-F0 : Buradaki y, gerçek değer yani rent. F0 ise tahmin edilen değer. Gerçek değerden tahmin edilen değeri çıkarttığımızda hatamızı buluyoruz.

XGBoost Algoritması

sqfeet	rent	F_0	$y - F_0$	Δ_1	F_1	$y\text{-}F_1$	Δ_2	F_2	$y - F_2$	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000

F0 : İlk modelimiz. İlk tahminlerimiz. | **Sqfeet** : Metrekare bilgisi. | **Rent** : Kira bilgisi.

F0 : İlk modelimiz. İlk tahminlerimiz. | **Sqfeet** : Metrekare bilgisi. | **Rent** : Kira bilgisi.

- Metrekare (bağımsız değişkenimiz) ile y, F0 (yeni bağımlı değişkenimiz) 'ı modelliyoruz. Buradan bazı hata [illegible] ve ortalama bir metrekare de ortalama fiyatı buluruz. AdaBoost 'ta gördüğümüz o tablolar gelsin aklınıza.
- [illegible] -145.5 çıkmış. Burası bir grup, [illegible] değerdir.
- [illegible] ilk model ile topluyoruz ve yeni değerler elde ediyoruz. Yeni model kurmuş oluyoruz. Daha sonra tekrar hatalarımı hesaplıyorum (delta2).
- Delta2 'de ilk iki satırın bir grup, son 3 satırın da ayrı bir grup olduğunu gözlemliyorum.
- Bir kez daha işlemlere sokuyorum. En son artık F3 'te 1. Ve 2. Satırın bir grup, 3. Ve 4. Satırın başka bir grup ve 5. Satırın da diğer bir grup olduğunu buluyorum.

XGBoost Algoritması

sqfeet	rent	F_0	$y - F_0$	Δ_1	F_1	y-F_1	Δ_2	F_2	y - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000

F0 : İlk modelimiz. İlk tahminlerimiz. | **Sqfeet** : Metrekare bilgisi. | **Rent** : Kira bilgisi.

- Metrekare (bağımsız değişkenimiz) ile y-F0 (yeni bağımlı değişkenimiz) ‘i modelliyoruz. Buradan bazı hata katsayıları yakalıyoruz.
- İlk 4 değişken delta1 ‘de bir arada yani -145.5 çıkmış. Burası bir grup, y-F0 ‘ın ilk dört değerinin ortalamasıdır.
- F1 ‘e geldiğimizde hata sonuçlarımızı ilk model ile topluyoruz ve yeni değerler elde ediyoruz. Yeni model kurmuş oluyoruz. Daha sonra tekrar hatalarımı hesaplıyorum (delta2).
- Delta2 ‘de ilk iki satırın bir grup, son 3 satırın da ayrı bir grup olduğunu gözlemliyorum.
- Bir kez daha işlemlere sokuyorum. En son artık F3 ‘te 1. Ve 2. Satırın bir grup, 3. Ve 4. Satırın başka bir grup ve 5. Satırın da diğer bir grup olduğunu buluyorum.

XGBoost Algoritması

```python
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import warnings
from sklearn.model_selection import cross_validate, GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier


pd.set_option('display.max_columns', None)
warnings.simplefilter(action='ignore', category=Warning)


df = pd.read_csv("datasets/diabetes.csv")

y = df["Outcome"]
X = df.drop(["Outcome"], axis=1)


########################################## GRADIENT BOOSTING MACHINES (GBM) ##############################################

## Zayıf öğrenicilerden güçlü bir öğrenici oluşturma. Diğer hatalardan yararlanıp daha güçlü bir ağaç oluşturma.

gbm_model = GradientBoostingClassifier()

gbm_model.get_params() #parametrelerimize baktığımız kısım.
```

XGBoost Algoritması

```
In [4]: gbm_model.get_params() #parametrelerimize baktığımız kısım.
Out[4]:
{'ccp_alpha': 0.0,
 'criterion': 'friedman_mse',
 'init': None,
 'learning_rate': 0.1,
 'loss': 'deviance',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_iter_no_change': None,
 'random_state': None,
 'subsample': 1.0,
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': 0,
 'warm_start': False}
```

Bunları kullanarak hiper parametre optimizasyonu yapmamız gerekiyor. Fakat önce bir hatamıza bakalım.

XGBoost Algoritması

```
#Hatalarımıza bakalım:
cv_results = cross_validate(gbm_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
#İşlemler hızlı olsun diye, 5 katlı çapraz doğrulama yaptım.
cv_results['test_accuracy'].mean() # 0.75
cv_results['test_f1'].mean() # 0.63
cv_results['test_roc_auc'].mean() # 0.82
```

Hatalarımızı not aldık, hiper parametre optimizasyonundan sonra nasıl değiştiklerine bakacağız.

```
gbm_params = {"learning_rate": [0.01, 0.1], #Arama işlemi uzayabilir, ne kadar küçük olursa train süresi o kadar uzar
              #Fakat ne kadar küçük olursa, o kadar başarılı tahminler yapar.
              "max_depth": [3, 8, 10],
              "n_estimators": [100, 500, 1000],
              "subsample": [1, 0.5, 0.7]} #Kaç tane gözlemin oransal olarak göz önünde bulundurulacağını belirler.
#Bütün değerlerimi göz önünde bulundurmalıyız? Yoksa bir kısmını mı?
gbm_best_grid = GridSearchCV(gbm_model, gbm_params, cv=5, n_jobs=-1, verbose=True).fit(X, y)

gbm_best_grid.best_params_
# learning_rate: 0.1
# max_depth: 8
# n_estimators: 100
# subsample: 0.5
gbm_final = gbm_model.set_params(**gbm_best_grid.best_params_).fit(X, y)
# Hatalarımıza bakalım:
cv_results = cross_validate(gbm_final, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
cv_results['test_accuracy'].mean() # 0.78
cv_results['test_f1'].mean() # 0.66
cv_results['test_roc_auc'].mean() # 0.82
```

XGBoost Algoritması

Şimdi bir de XGBoost ile modelimizi kuralım;

```python
import pandas as pd
import matplotlib
matplotlib.use('Qt5Agg')
import warnings

from sklearn.model_selection import cross_validate, GridSearchCV
from xgboost import XGBClassifier

pd.set_option('display.max_columns', None)
warnings.simplefilter(action='ignore', category=Warning)


df = pd.read_csv("datasets/diabetes.csv")

y = df["Outcome"]
X = df.drop(["Outcome"], axis=1)

################################################ XGBOOST ################################################

xgboost_model = XGBClassifier(random_state=17)

cv_results = cross_validate(xgboost_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
cv_results['test_accuracy'].mean() # 0.75
cv_results['test_f1'].mean() # 0.63
cv_results['test_roc_auc'].mean() #0.79

xgboost_model.get_params()
```

XGBoost Algoritması

```
Out[12]:
{'objective': 'binary:logistic',
 'use_label_encoder': False,
 'base_score': None,
 'booster': None,
 'callbacks': None,
 'colsample_bylevel': None,
 'colsample_bynode': None,
 'colsample_bytree': None,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'gamma': None,
 'gpu_id': None,
 'grow_policy': None,
 'importance_type': None,
 'interaction_constraints': None,
 'learning_rate': None,
 'max_bin': None,
 'max_cat_to_onehot': None,
 'max_delta_step': None,
 'max_depth': None,
 'max_leaves': None,
 'min_child_weight': None,
 'missing': nan,
 'monotone_constraints': None,
 'n_estimators': 100,
 'n_jobs': None,
 'num_parallel_tree': None,
 'predictor': None,
 'random_state': 17,
 'reg_alpha': None,
 'reg_lambda': None,
 'sampling_method': None,
 'scale_pos_weight': None,
 'subsample': None,
 'tree_method': None,
```

XGBoost Algoritması

```
xgboost_params = {"learning_rate": [0.1, 0.01],
                  "max_depth": [5, 8, None],
                  "n_estimators": [100, 500, 1000],
                  "colsample_bytree": [None, 0.7, 1]} #subsample ile aynı şey diyebiliriz.

xgboost_best_grid = GridSearchCV(xgboost_model, xgboost_params, cv=5, n_jobs=-1, verbose=True).fit(X, y)

xgboost_final = xgboost_model.set_params(**xgboost_best_grid.best_params_, random_state=17).fit(X, y)

cv_results = cross_validate(xgboost_final, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
cv_results['test_accuracy'].mean() # 0.75
cv_results['test_f1'].mean() # 0.63
cv_results['test_roc_auc'].mean() #0.81
```

Hata skorlarımızın düştüğünü/azaldığını gözlemleyebiliriz.

Teşekkürler.