Introduction

Doğrusal regresyon, trendleri tahmin etmede (extrapolating) mükemmeldir ancak etkileşimleri öğrenemez. XGBoost, etkileşimleri öğrenmede mükemmeldir ancak trendleri tahmin edemez. Bu derste, birbirini tamamlayan öğrenme algoritmalarını birleştiren ve birinin güçlü yönlerinin diğerinin zayıflığını telafi etmesini sağlayan "hibrit" tahminciler oluşturmayı öğreneceğiz.

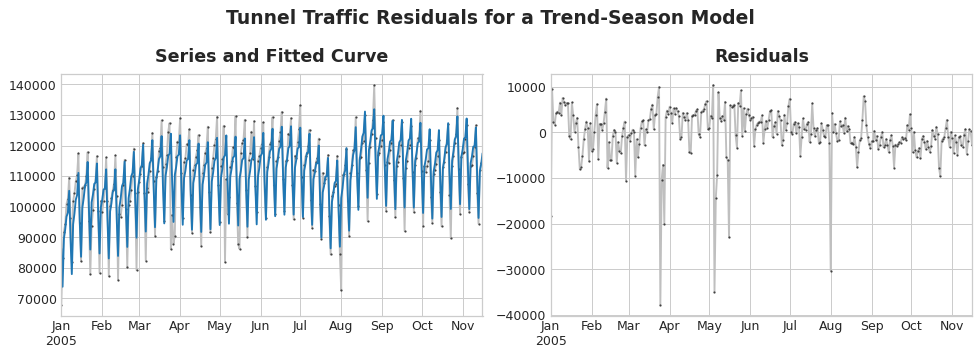
# Components and Residuals[¶](https://www.kaggle.com/code/ryanholbrook/hybrid-models" \l "Components-and-Residuals" \t "_self)

Etkili hibritler tasarlayabilmek için, zaman serilerinin nasıl inşa edildiğini daha iyi anlamamız gerekiyor. Şimdiye kadar üç bağımlılık kalıbı inceledik: **trend**, **mevsimler** ve **döngüler**. Birçok zaman serisi, sadece bu üç bileşenin ve esasen tahmin edilemez, tamamen rastgele bir **hata**nın eklenmesinden oluşan toplamsal bir modelle yakın bir şekilde tanımlanabilir:

series = trend + seasons + cycles + error

Bu modeldeki terimlerin her birine, zaman serisinin bir **bileşeni** deriz.

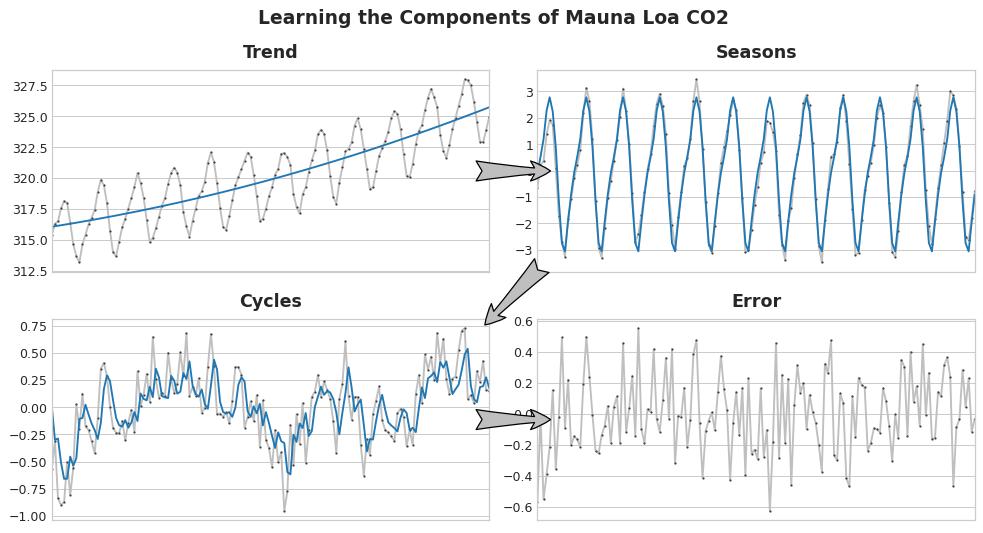
Bir modelin **rezidüelleri** (kalıntıları), modelin üzerinde eğitildiği hedef ile modelin yaptığı tahminler arasındaki farktır; başka bir deyişle, gerçek eğri ile uydurulmuş eğri arasındaki farktır. Rezidüelleri bir özelliğe karşı grafiğe döktüğünüzde, hedefin "artık" kısmını veya modelin o özellikten hedef hakkında öğrenemediği kısmı elde edersiniz.



Hedef seri ile tahminler arasındaki fark (mavi) artıklar serisini verir.

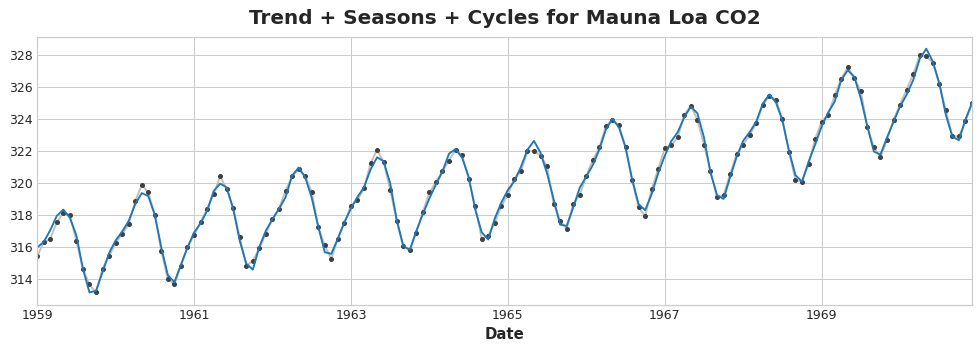
Yukarıdaki şeklin sol tarafında, **Tunnel Traffic** serisinin bir bölümü ve 3. Dersten elde edilen trend-mevsimsel eğri bulunmaktadır. Uygun eğriyi çıkardığımızda sağdaki rezidüeller (kalıntılar) kalır. Rezidüeller, trend-mevsimsel modelin **Tunnel Traffic**'ten öğrenemediği her şeyi içerir.

Bir zaman serisinin bileşenlerini öğrenmeyi, tekrarlayan bir süreç olarak düşünebiliriz: önce trendi öğrenip seriden çıkarırız, ardından trendden arındırılmış rezidüellerden mevsimselliği öğrenip mevsimleri çıkarırız, sonra döngüleri öğrenip döngüleri çıkarırız ve sonunda geriye yalnızca tahmin edilemez hata kalır.



Mauna Loa CO2'nin bileşenlerini adım adım öğrenin. Bir sonraki adımda seriyi elde etmek için, uydurulmuş eğriyi (mavi) serisinden çıkarın.

Öğrendiğimiz tüm bileşenleri bir araya eklediğimizde, tam modeli elde ederiz. Bu, doğrusal regresyonun, trendi, mevsimleri ve döngüleri modelleyen eksiksiz bir özellik seti üzerinde eğitilmesi durumunda yapacağı şeyin temelidir.



Öğrenilen bileşenleri ekleyerek tam bir model elde edin.

# Hybrid Forecasting with Residuals[¶](https://www.kaggle.com/code/ryanholbrook/hybrid-models" \l "Hybrid-Forecasting-with-Residuals" \t "_self)

Önceki derslerde, tüm bileşenleri aynı anda öğrenmek için tek bir algoritma (doğrusal regresyon) kullanmıştık. Ancak, bazı bileşenler için bir algoritma, geri kalanı için ise başka bir algoritma kullanmak da mümkündür. Bu sayede her bir bileşen için her zaman en iyi algoritmayı seçebiliriz. Bunu yapmak için, orijinal seriyi uydurmak amacıyla bir algoritma, ardından kalan seriyi (rezidüel seriyi) uydurmak için ikinci bir algoritma kullanırız.

Ayrıntılı olarak süreç şu şekildedir:

# 1. Train and predict with first model

model\_1.fit(X\_train\_1, y\_train)

y\_pred\_1 = model\_1.predict(X\_train)

# 2. Train and predict with second model on residuals

model\_2.fit(X\_train\_2, y\_train - y\_pred\_1)

y\_pred\_2 = model\_2.predict(X\_train\_2)

# 3. Add to get overall predictions

y\_pred = y\_pred\_1 + y\_pred\_2

Genellikle, her bir modelin ne öğrenmesini istediğimize bağlı olarak farklı **özellik setleri** (yukarıdaki **X\_train\_1** ve **X\_train\_2**) kullanmak isteriz. Örneğin, ilk modeli eğilimi öğrenmek için kullanırsak, ikinci model için genellikle bir eğilim özelliğine ihtiyacımız olmaz.

İkiden fazla model kullanmak mümkün olsa da pratikte bu, özellikle faydalı görünmüyor. Aslında, **hibritleri oluşturmak için en yaygın strateji**, az önce tanımladığımızdır: Basit (genellikle doğrusal) bir öğrenme algoritması, ardından GBDT'ler (Gradyan Artırımlı Karar Ağaçları) veya derin bir sinir ağı gibi karmaşık, doğrusal olmayan bir öğrenici takip eder. Basit model, tipik olarak, onu izleyen güçlü algoritma için bir **"yardımcı"** olarak tasarlanır.

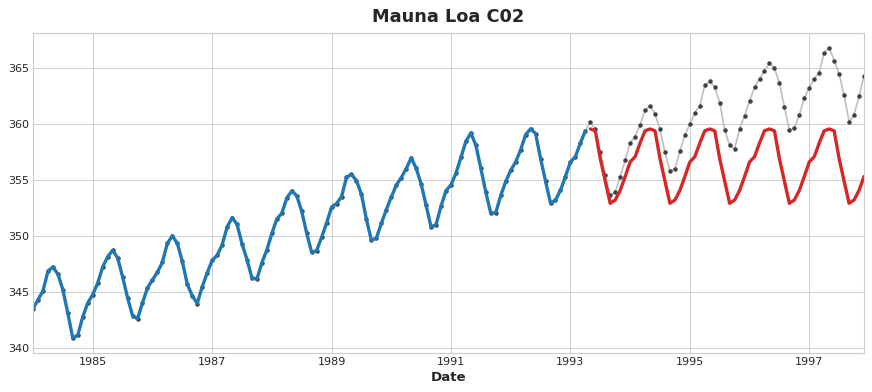
### **Designing Hybrids[¶](https://www.kaggle.com/code/ryanholbrook/hybrid-models" \l "Designing-Hybrids" \t "_self)**

Dersimizde ana hatlarını çizdiğimiz yöntemin dışında, makine öğrenimi modellerini birleştirmenin birçok yolu vardır. Ancak, modelleri başarılı bir şekilde birleştirmek, bu algoritmaların nasıl çalıştığına biraz daha derinlemesine inmemizi gerektirir.

Bir **regresyon algoritmasının tahmin yapmasının** genel olarak iki yolu vardır: ya **özellikleri dönüştürerek** ya da **hedefi dönüştürerek**.

* **Özellikleri dönüştüren algoritmalar**, girdileri özellik olarak alan ve ardından bunları birleştirip dönüştürerek eğitim setindeki hedef değerlerle eşleşen bir çıktı üreten matematiksel bir fonksiyon öğrenirler. **Doğrusal regresyon** ve **sinir ağları** bu türdendir.
* **Hedefi dönüştüren algoritmalar**, eğitim setindeki hedef değerleri gruplamak için özellikleri kullanır ve bir gruptaki değerlerin ortalamasını alarak tahmin yaparlar. Bir dizi özellik, sadece hangi grubun ortalamasının alınacağını belirtir. **Karar ağaçları** ve **en yakın komşular** bu türdendir.

Önemli olan şudur: **Özellikleri dönüştüren algoritmalar**, girdi olarak uygun özellikler verildiğinde hedef değerleri eğitim setinin ötesine **ekstrapole edebilirken**, **hedefi dönüştüren algoritmaların** tahminleri her zaman eğitim setinin aralığıyla sınırlı kalacaktır. Eğer zaman kuklası (dummy) zaman adımlarını saymaya devam ederse, doğrusal regresyon eğilim çizgisini çizmeye devam eder. Aynı zaman kuklası verildiğinde, bir karar ağacı ise eğitim verisinin son adımının gösterdiği eğilimi geleceğe doğru sonsuza kadar tahmin edecektir. **Karar ağaçları eğilimleri ekstrapole edemez.** Rastgele ormanlar ve gradyan artırımlı karar ağaçları (XGBoost gibi), karar ağaçlarının bir araya getirilmesiyle oluşan topluluklar (ensemble) oldukları için onlar da eğilimleri ekstrapole edemezler.



Bir karar ağacı, eğitim kümesinin ötesinde bir eğilimi genelleştirmede başarısız olacaktır.

Bu fark, bu dersteki hibrit tasarımı motive eden şeydir: **eğilimi ekstrapole etmek için doğrusal regresyonu kullanmak, eğilimi kaldırmak için hedefi dönüştürmek ve bu eğilimden arındırılmış kalıntılara XGBoost uygulamak.**

### Hibrit Model Oluşturma Yöntemleri

Bir sinir ağını (bir **özellik dönüştürücü**) hibrit hale getirmek için, başka bir modelin tahminlerini bir özellik olarak dahil edebilirsiniz. Sinir ağı bu özelliği kendi tahminlerinin bir parçası olarak kullanır.

* **Boosted hibritler**: Kalıntılara uydurma yöntemi, gradyan artırma algoritmasının kullandığı yöntemle aynıdır. Bu tür hibrit modellere **"boosted hibritler"** denir.
* **Stacked hibritler**: Tahminleri özellik olarak kullanma yöntemi ise **"istifleme"** (stacking) olarak bilinir. Bu tür hibrit modellere de **"stacked hibritler"** denir.

**Winning Hybrids from Kaggle Competitions**

For inspiration, here are a few top scoring solutions from past competitions:

* [STL boosted with exponential smoothing](https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/discussion/8125) - Walmart Recruiting - Store Sales Forecasting
* [ARIMA and exponential smoothing boosted with GBDT](https://www.kaggle.com/c/rossmann-store-sales/discussion/17896) - Rossmann Store Sales
* [An ensemble of stacked and boosted hybrids](https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/39395) - Web Traffic Time Series Forecasting
* [Exponential smoothing stacked with LSTM neural net](https://github.com/Mcompetitions/M4-methods/blob/slaweks_ES-RNN/118%20-%20slaweks17/ES_RNN_SlawekSmyl.pdf) - M4 (non-Kaggle)

# Example - US Retail Sales[¶](https://www.kaggle.com/code/ryanholbrook/hybrid-models" \l "Example---US-Retail-Sales" \t "_self)

### ABD Perakende Satış Veri Seti

"**ABD Perakende Satışları** veri seti, ABD Sayım Bürosu tarafından 1992'den 2019'a kadar toplanan çeşitli perakende sektörlerine ait aylık satış verilerini içerir. Amacımız, 2016-2019 yılları arasındaki satışları, önceki yılların satış verilerini kullanarak tahmin etmektir. Doğrusal regresyon ve XGBoost hibrit modelini oluşturmanın yanı sıra, bir zaman serisi veri setini XGBoost ile kullanmak için nasıl hazırlayacağımızı da göreceğiz."

from pathlib import Path

from warnings import simplefilter

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from statsmodels.tsa.deterministic import CalendarFourier, DeterministicProcess

from xgboost import XGBRegressor

simplefilter("ignore")

*# Set Matplotlib defaults*

plt.style.use("seaborn-whitegrid")

plt.rc(

"figure",

autolayout=True,

figsize=(11, 4),

titlesize=18,

titleweight='bold',

)

plt.rc(

"axes",

labelweight="bold",

labelsize="large",

titleweight="bold",

titlesize=16,

titlepad=10,

)

plot\_params = dict(

color="0.75",

style=".-",

markeredgecolor="0.25",

markerfacecolor="0.25",

)

data\_dir = Path("../input/ts-course-data/")

industries = ["BuildingMaterials", "FoodAndBeverage"]

retail = pd.read\_csv(

data\_dir / "us-retail-sales.csv",

usecols=['Month'] + industries,

parse\_dates=['Month'],

index\_col='Month',

).to\_period('D').reindex(columns=industries)

retail = pd.concat({'Sales': retail}, names=[None, 'Industries'], axis=1)

retail.head()

| Sales | |
| --- | --- |
| Industries | BuildingMaterials | FoodAndBeverage |
| Month |  |  |
| 1992-01-01 | 8964 | 29589 |
| 1992-02-01 | 9023 | 28570 |
| 1992-03-01 | 10608 | 29682 |
| 1992-04-01 | 11630 | 30228 |
| 1992-05-01 | 12327 | 31677 |

Öncelikle, her serideki eğilimi öğrenmek için doğrusal bir regresyon modeli kullanalım. Örnek olarak, ikinci dereceden bir eğilim kullanacağız. (Buradaki kod, önceki derslerdekiyle temelde aynı.) Uyum mükemmel olmasa da, ihtiyaçlarımızı karşılayacak kadar yeterli olacaktır.

y = retail.copy()

*# Create trend features*

dp = DeterministicProcess(

index=y.index, *# dates from the training data*

constant=True, *# the intercept*

order=2, *# quadratic trend*

drop=True, *# drop terms to avoid collinearity*

)

X = dp.in\_sample() *# features for the training data*

*# Test on the years 2016-2019. It will be easier for us later if we*

*# split the date index instead of the dataframe directly.*

idx\_train, idx\_test = train\_test\_split(

y.index, test\_size=12 \* 4, shuffle=False,

)

X\_train, X\_test = X.loc[idx\_train, :], X.loc[idx\_test, :]

y\_train, y\_test = y.loc[idx\_train], y.loc[idx\_test]

*# Fit trend model*

model = LinearRegression(fit\_intercept=False)

model.fit(X\_train, y\_train)

*# Make predictions*

y\_fit = pd.DataFrame(

model.predict(X\_train),

index=y\_train.index,

columns=y\_train.columns,

)

y\_pred = pd.DataFrame(

model.predict(X\_test),

index=y\_test.index,

columns=y\_test.columns,

)

*# Plot*

axs = y\_train.plot(color='0.25', subplots=True, sharex=True)

axs = y\_test.plot(color='0.25', subplots=True, sharex=True, ax=axs)

axs = y\_fit.plot(color='C0', subplots=True, sharex=True, ax=axs)

axs = y\_pred.plot(color='C3', subplots=True, sharex=True, ax=axs)

for ax **in** axs: ax.legend([])

\_ = plt.suptitle("Trends")

Doğrusal regresyon algoritması çoklu çıktı regresyonu yapabilse de, **XGBoost algoritması bu özelliğe sahip değildir**. XGBoost ile birden fazla seriyi aynı anda tahmin etmek için, her sütunda bir zaman serisinin bulunduğu **geniş (wide)** formatı, her bir serinin satırlardaki kategorilerle indekslendiği **uzun (long)** formata dönüştürürüz.

*# The `stack` method converts column labels to row labels, pivoting from wide format to long*

X = retail.stack() *# pivot dataset wide to long*

display(X.head())

y = X.pop('Sales') *# grab target series*

| Sales |
| --- |
| Month | Industries |  |
| 1992-01-01 | BuildingMaterials | 8964 |
| FoodAndBeverage | 29589 |
| 1992-02-01 | BuildingMaterials | 9023 |
| FoodAndBeverage | 28570 |
| 1992-03-01 | BuildingMaterials | 10608 |

XGBoost'un iki zaman serimizi ayırt edebilmesi için, **'Industries'** (Endüstriler) için satır etiketlerini **etiket kodlaması (label encoding)** ile kategorik bir özelliğe dönüştüreceğiz. Ayrıca, zaman indeksinden ay numaralarını çekerek yıllık mevsimsellik için bir özellik oluşturacağız.

*# Turn row labels into categorical feature columns with a label encoding*

X = X.reset\_index('Industries')

*# Label encoding for 'Industries' feature*

for colname **in** X.select\_dtypes(["object", "category"]):

X[colname], \_ = X[colname].factorize()

*# Label encoding for annual seasonality*

X["Month"] = X.index.month *# values are 1, 2, ..., 12*

*# Create splits*

X\_train, X\_test = X.loc[idx\_train, :], X.loc[idx\_test, :]

y\_train, y\_test = y.loc[idx\_train], y.loc[idx\_test]

Şimdi daha önce yaptığımız trend tahminlerini uzun formata dönüştürüp orijinal seriden çıkaracağız. Bu, XGBoost'un öğrenebileceği trendden arındırılmış (artık) seriyi elde etmemizi sağlayacak.

*# Pivot wide to long (stack) and convert DataFrame to Series (squeeze)*

y\_fit = y\_fit.stack().squeeze() *# trend from training set*

y\_pred = y\_pred.stack().squeeze() *# trend from test set*

*# Create residuals (the collection of detrended series) from the training set*

y\_resid = y\_train - y\_fit

*# Train XGBoost on the residuals*

xgb = XGBRegressor()

xgb.fit(X\_train, y\_resid)

*# Add the predicted residuals onto the predicted trends*

y\_fit\_boosted = xgb.predict(X\_train) + y\_fit

y\_pred\_boosted = xgb.predict(X\_test) + y\_pred

Uyum oldukça iyi görünüyor, ancak XGBoost'un öğrendiği trendin, doğrusal regresyonla öğrenilen trend kadar iyi olduğunu görebiliyoruz; özellikle, XGBoost, 'BuildingMaterials' serisindeki zayıf uyum trendini telafi edemedi.

axs = y\_train.unstack(['Industries']).plot(

color='0.25', figsize=(11, 5), subplots=True, sharex=True,

title=['BuildingMaterials', 'FoodAndBeverage'],

)

axs = y\_test.unstack(['Industries']).plot(

color='0.25', subplots=True, sharex=True, ax=axs,

)

axs = y\_fit\_boosted.unstack(['Industries']).plot(

color='C0', subplots=True, sharex=True, ax=axs,

)

axs = y\_pred\_boosted.unstack(['Industries']).plot(

color='C3', subplots=True, sharex=True, ax=axs,

)

for ax **in** axs: ax.legend([])