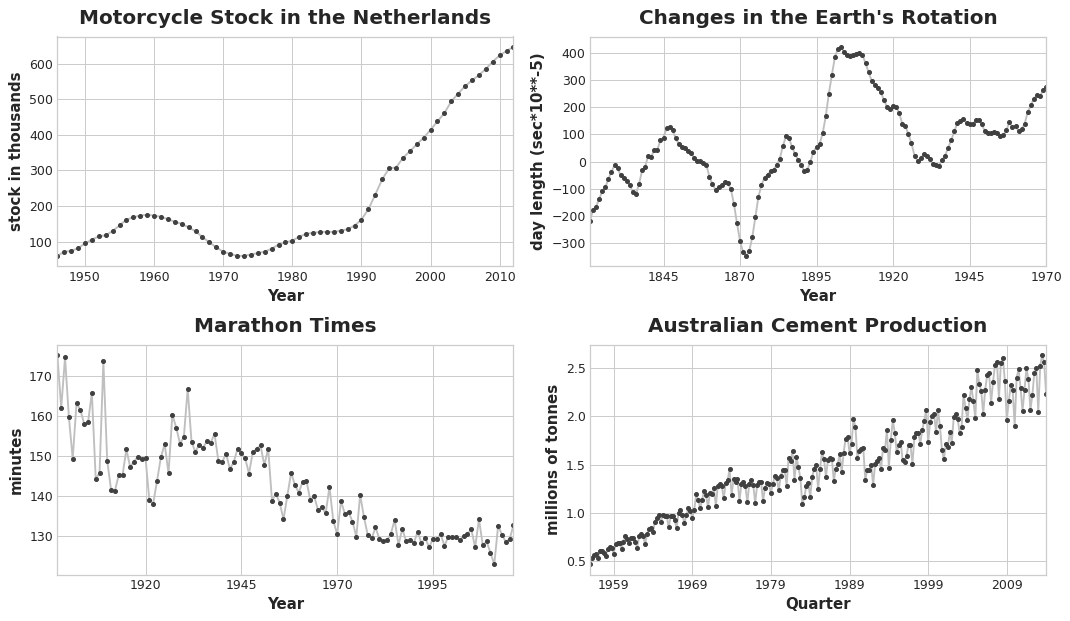
What is Trend?[¶](https://www.kaggle.com/code/ryanholbrook/trend#What-is-Trend?)

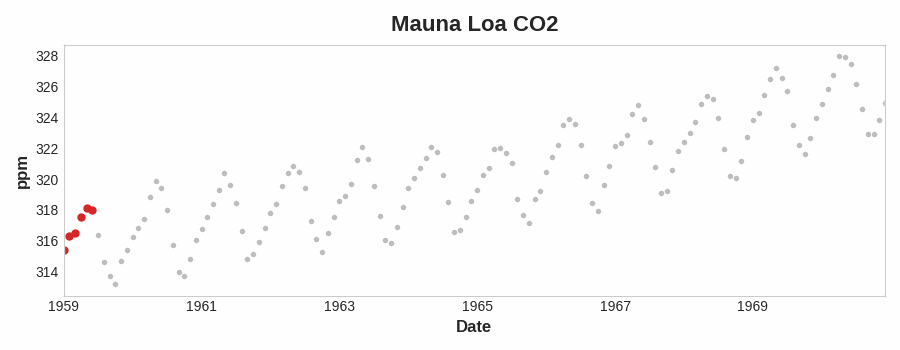
Bir zaman serisinin **trend** bileşeni, serinin ortalamasındaki kalıcı, uzun vadeli bir değişikliği temsil eder. Trend, bir serinin en yavaş hareket eden, en büyük öneme sahip zaman ölçeğini temsil eden kısmıdır. Bir ürün satışları zaman serisinde, artan bir trend, giderek daha fazla insanın üründen haberdar olmasıyla pazarın genişlemesinin bir etkisi olabilir.



Bu kursta, ortalamadaki trendlere odaklanacağız. Ancak daha genel olarak, bir serideki herhangi bir kalıcı ve yavaş hareket eden değişiklik bir trend oluşturabilir; örneğin, zaman serilerinin genellikle varyasyonlarında trendler bulunur.

# Moving Average Plots[¶](https://www.kaggle.com/code/ryanholbrook/trend" \l "Moving-Average-Plots" \t "_self)

Bir zaman serisinin ne tür bir trende sahip olabileceğini görmek için, bir **hareketli ortalama grafiği** kullanabiliriz. Bir zaman serisinin hareketli ortalamasını hesaplamak için, belirli bir genişliğe sahip kayan bir pencere içindeki değerlerin ortalamasını hesaplarız. Grafikteki her bir nokta, seride pencerenin her iki tarafına düşen tüm değerlerin ortalamasını temsil eder. Buradaki amaç, serideki kısa vadeli dalgalanmaları yumuşatarak geriye yalnızca uzun vadeli değişikliklerin kalmasını sağlamaktır.



Doğrusal bir eğilimi gösteren hareketli ortalama grafiği. Eğrideki her nokta (mavi), 12 boyutlu bir pencere içindeki noktaların (kırmızı) ortalamasıdır.

Yukarıdaki Mauna Loa serisinin yıldan yıla tekrarlanan bir yukarı ve aşağı harekete sahip olduğuna dikkat edin; bu, kısa vadeli, **mevsimsel** bir değişimdir. Bir değişimin trendin bir parçası olması için, herhangi bir mevsimsel değişimden daha uzun bir periyotta meydana gelmesi gerekir. Bu nedenle, bir trendi görselleştirmek için, serideki herhangi bir mevsimsel periyottan daha uzun bir periyot üzerinde bir ortalama alırız. Mauna Loa serisi için, her bir yıl içindeki mevsimi düzleştirmek amacıyla 12 büyüklüğünde bir pencere seçtik.

# Engineering Trend[¶](https://www.kaggle.com/code/ryanholbrook/trend#Engineering-Trend)

Bir trendin şeklini belirledikten sonra, onu bir zaman adımı özelliği kullanarak modellemeye çalışabiliriz. Zaman kuklasını kullanmanın doğrusal bir trendi nasıl modelleyeceğini zaten görmüştük:

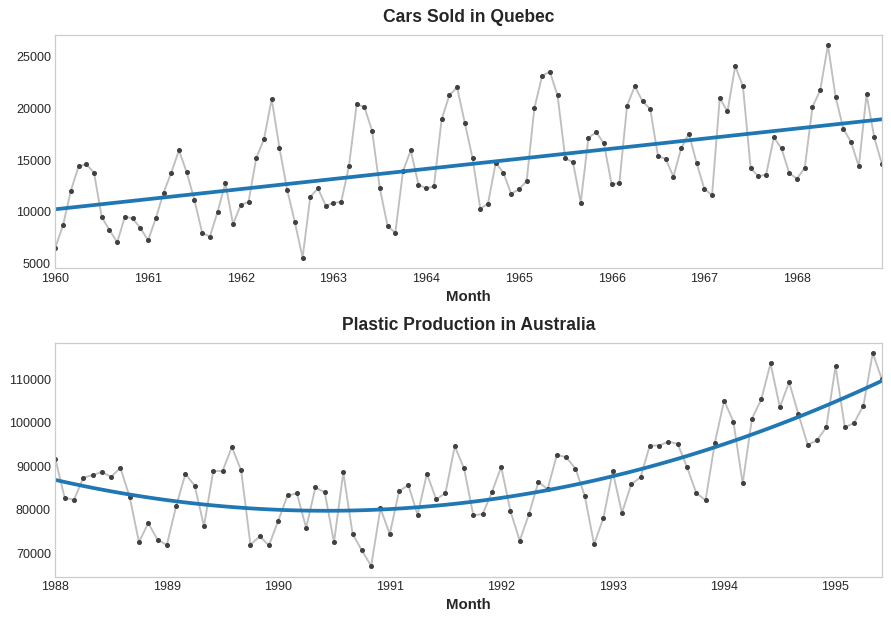
target = a \* time + b

Zaman kuklası üzerinde dönüşümler yaparak başka birçok trend türünü de uydurabiliriz. Eğer trend karesel (bir parabol) gibi görünüyorsa, özellik setine zaman kuklasının karesini eklememiz yeterlidir. Bu bize şunu verir:

target = a \* time \*\* 2 + b \* time + c

Doğrusal regresyon, a, b ve c katsayılarını öğrenecektir.

Aşağıdaki şekildeki trend eğrileri, bu tür özellikler ve scikit-learn'ün Doğrusal Regresyonu kullanılarak oluşturulmuştur:



***Top:****Series with a linear trend.****Below:****Series with a quadratic trend.*

Daha önce bu yöntemi görmediyseniz, doğrusal regresyonun çizgilerden başka eğrileri de uydurabileceğini fark etmemiş olabilirsiniz. Buradaki fikir, uygun şekildeki eğrileri özellik olarak sağlayabilirseniz, doğrusal regresyonun bunları hedefi en iyi şekilde yansıtan biçimde nasıl birleştireceğini öğrenebilmesidir.

# Example - Tunnel Traffic[¶](https://www.kaggle.com/code/ryanholbrook/trend" \l "Example---Tunnel-Traffic" \t "_self)

Bu örnekte Tünel Trafiği veri kümesi için bir trend modeli oluşturacağız.

from pathlib import Path

from warnings import simplefilter

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

simplefilter("ignore") *# ignore warnings to clean up output cells*

*# Set Matplotlib defaults*

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

plt.rc("figure", autolayout=True, figsize=(11, 5))

plt.rc(

"axes",

labelweight="bold",

labelsize="large",

titleweight="bold",

titlesize=14,

titlepad=10,

)

plot\_params = dict(

color="0.75",

style=".-",

markeredgecolor="0.25",

markerfacecolor="0.25",

legend=False,

)

%config InlineBackend.figure\_format = 'retina'

*# Load Tunnel Traffic dataset*

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

tunnel = pd.read\_csv(data\_dir / "tunnel.csv", parse\_dates=["Day"])

tunnel = tunnel.set\_index("Day").to\_period()

Bu, bir veri analizi veya makine öğrenimi uygulamasından alınan bir metin parçasıdır.

Bu serinin ne tür bir trende sahip olduğunu görmek için bir hareketli ortalama grafiği yapalım. Bu seri günlük gözlemlere sahip olduğu için, yıl içindeki kısa vadeli değişiklikleri yumuşatmak amacıyla 365 günlük bir pencere (window) seçelim.

Bir hareketli ortalama oluşturmak için, öncelikle pencere bazlı bir hesaplama başlatmak üzere **rolling** yöntemini kullanın. Ardından, pencere üzerindeki ortalamayı hesaplamak için **mean** yöntemini uygulayın. Gördüğümüz gibi, "Tunnel Traffic" trendi yaklaşık olarak doğrusaldır.

moving\_average = tunnel.rolling(

window=365, *# 365-day window*

center=True, *# puts the average at the center of the window*

min\_periods=183, *# choose about half the window size*

).mean() *# compute the mean (could also do median, std, min, max, ...)*

ax = tunnel.plot(style=".", color="0.5")

moving\_average.plot(

ax=ax, linewidth=3, title="Tunnel Traffic - 365-Day Moving Average", legend=False,

);

Lesson 1'de (Ders 1'de) zaman kuklasını doğrudan Pandas'ta tasarlamıştık. Ancak, bundan sonra **DeterministicProcess** adında, **statsmodels** kütüphanesinden bir fonksiyon kullanacağız. Bu fonksiyonu kullanmak, zaman serileri ve doğrusal regresyonla ortaya çıkabilecek bazı zorlu hata durumlarından kaçınmamıza yardımcı olacak. **order** argümanı polinom derecesini ifade eder: doğrusal için **1**, karesel için **2**, kübik için **3** ve bu şekilde devam eder.

from statsmodels.tsa.deterministic import DeterministicProcess

dp = DeterministicProcess(

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

constant=True, *# dummy feature for the bias (y\_intercept)*

order=1, *# the time dummy (trend)*

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

)

*# `in\_sample` creates features for the dates given in the `index` argument*

X = dp.in\_sample()

X.head()

| const | trend |
| --- | --- |
| Day |  |  |
| 2003-11-01 | 1.0 | 1.0 |
| 2003-11-02 | 1.0 | 2.0 |
| 2003-11-03 | 1.0 | 3.0 |
| 2003-11-04 | 1.0 | 4.0 |
| 2003-11-05 | 1.0 | 5.0 |

(Bu arada, **deterministik süreç**; **sabit** ve **trend** serileri gibi, rastgele olmayan veya tamamen **belirlenmiş** bir zaman serisi için kullanılan teknik bir terimdir. Zaman indeksinden türetilen özellikler genellikle deterministik olacaktır.)

Trend modelimizi temelde daha önce olduğu gibi oluşturuyoruz, ancak **fit\_intercept=False** argümanının eklendiğine dikkat edin.

from sklearn.linear\_model import LinearRegression

y = tunnel["NumVehicles"] *# the target*

*# The intercept is the same as the `const` feature from*

*# DeterministicProcess. LinearRegression behaves badly with duplicated*

*# features, so we need to be sure to exclude it here.*

model = LinearRegression(fit\_intercept=False)

model.fit(X, y)

y\_pred = pd.Series(model.predict(X), index=X.index)

LinearRegression modelimiz tarafından keşfedilen trend, hareketli ortalama grafiğiyle neredeyse aynıdır; bu da bu durumda doğrusal bir trendin doğru karar olduğunu göstermektedir.

ax = tunnel.plot(style=".", color="0.5", title="Tunnel Traffic - Linear Trend")

\_ = y\_pred.plot(ax=ax, linewidth=3, label="Trend")

Tahmin yapmak için, modelimizi **"örneklem dışı"** özelliklere uygularız. "Örneklem dışı", eğitim verisinin gözlem periyodu dışındaki zamanları ifade eder. 30 günlük bir tahmini nasıl yapabileceğimiz aşağıda gösterilmiştir:

X = dp.out\_of\_sample(steps=30)

y\_fore = pd.Series(model.predict(X), index=X.index)

y\_fore.head()

2005-11-17 114981.801146

2005-11-18 115004.298595

2005-11-19 115026.796045

2005-11-20 115049.293494

2005-11-21 115071.790944

Freq: D, dtype: float64

Önümüzdeki 30 güne ait trend tahminini görmek için serinin bir bölümünü çizelim:

ax = tunnel["2005-05":].plot(title="Tunnel Traffic - Linear Trend Forecast", \*\*plot\_params)

ax = y\_pred["2005-05":].plot(ax=ax, linewidth=3, label="Trend")

ax = y\_fore.plot(ax=ax, linewidth=3, label="Trend Forecast", color="C3")

\_ = ax.legend()

Bu derste öğrendiğimiz trend modelleri birkaç nedenden dolayı faydalıdır. Daha gelişmiş modeller için bir başlangıç noktası olmanın yanı sıra, trendleri öğrenemeyen algoritmalarla (**XGBoost** ve **rastgele ormanlar** gibi) bir "**hibrit model**" içinde bir bileşen olarak da kullanılabilirler. Bu tekniği 5. Derste daha ayrıntılı olarak öğreneceğiz.