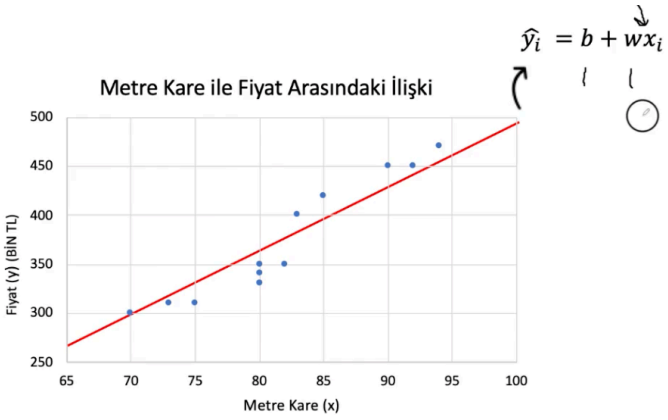
Doğrusal Regresyon

Linear regression is a widely used statistical technique for predicting a **continuous outcome variable based on one or more predictor variables**. While there are numerous libraries and tools available that offer ready-made implementations of linear regression, understanding how to implement it from scratch provides valuable insights into the underlying mathematics and concepts. In this article, we will walk through the process of implementing linear regression from scratch using Python.

metre_kare (x_i)	fiyat (y_i)
70	300
73	310
75	310
80	330
80	340
80	350
82	350
83	400
85	420
90	450
92	450
94	470



- Problem bu doğruyu nereye koyacağız?
  - **Cost function:**

    Makine öğrenmesinde **maliyet fonksiyonu**, bir modelin tahminlerinin (predicted values) gerçek değerlerden (actual/true values) ne kadar saptığını ölçen matematiksel bir fonksiyondur. Esasen, modelimizin ne kadar "iyi" veya "kötü" performans gösterdiğini nicel olarak ifade etmemizi sağlar. Amaç, bu maliyet fonksiyonunun değerini **minimize etmek** veya bazı durumlarda **maksimize etmek**tir. Regresyon modellerinde genellikle minimizasyon hedeflenir.

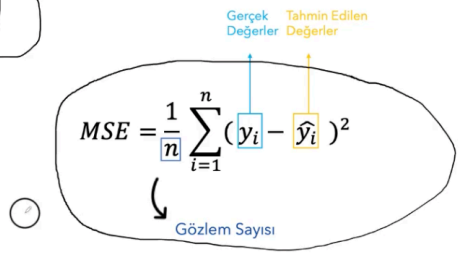
    Bu fonksiyon, modelin **parametrelerini (ağırlıklar ve sapmalar)** optimize etmek için bir pusula görevi görür. Model eğitilirken, parametreler maliyet fonksiyonunun değerini en aza indirecek şekilde ayarlanır. Bu

optimizasyon süreci genellikle **gradyan iniş (gradient descent)** gibi algoritmalarla yapılır.

$$Cost(b,w) = \frac{1}{2m}\sum_{i=1}^{m}((b + wx_i) - y_i)^2$$

Regresyon Modellerinde Başarı Değerlendirmesi: MSE / RMSE / MAE

fiyat (y_i)	fiyat tahmin ($\hat{y}_i$)	hata ($e_i = y_i - \hat{y}_i$)	hata_kare ($e_i = y_i - \hat{y}_i$)^2
300	320	-20	400
310	280	30	900
310	290	20	400
330	329	1	1
340	330	10	100
350	352	-2	4
350	400	-50	2500
400	410	-10	100
420	460	-40	1600
450	420	30	900
450	410	40	1600
470	480	-10	100



$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Parametrelerin Tahmin Edilmesi (Ağırlıkların Bulunması)

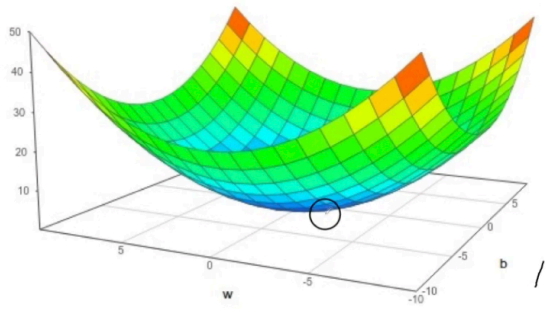
- **Parametre Tahmini: Modelin DNA'sını Keşfetmek**

  Makine öğrenmesinde **parametreler**, bir modelin öğrenme sürecinde verilerden edindiği bilgileri depolayan içsel değişkenlerdir. Regresyon bağlamında, bu parametreler genellikle **ağırlıklar (weights)** ve **sapmalar (biases)** olarak adlandırılır. Bir doğrusal regresyon modelinde, örneğin Y=wX+b denklemindeki w (eğim) ve b (kesişim) değerleri bu

parametrelerdir. Modelin amacı, girdi verileri (X) ile çıktı verileri (Y) arasındaki ilişkiyi en iyi temsil eden w ve b değerlerini bulmaktır.

Bu parametrelerin tahmin edilmesi süreci, temelde bir **optimizasyon problemidir**. Amacımız, modelimizin tahminleri ile gerçek değerler arasındaki farkı (yani **maliyet fonksiyonunun** çıktısını) minimize eden parametre setini bulmaktır.

$$Cost(b,w) = \frac{1}{2m}\sum_{i=1}^{m}((b + wx_i) - y_i)^2$$



Analitik Çözüm: Normal Denklemler Yöntemi (En Küçük Kareler Yöntemi)

Simple Linear Regression

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (b_0 + b_1 x_i))^2$$

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

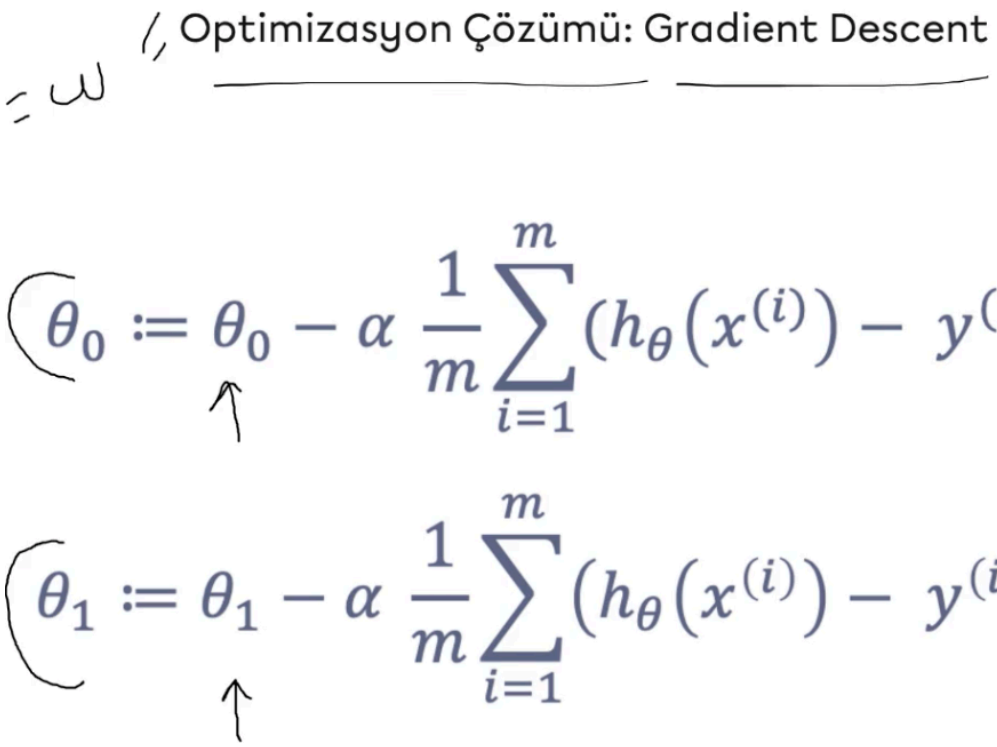
$$b_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

Multiple Linear Regression

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\hat{\beta} = (X^T.X)^{-1}X^T.Y$$

Niye Gradient Descend kullanılır?



Doğrusal Regresyon için Gradient Descent (Gradient Descent for Linear Regression)

Tanım:

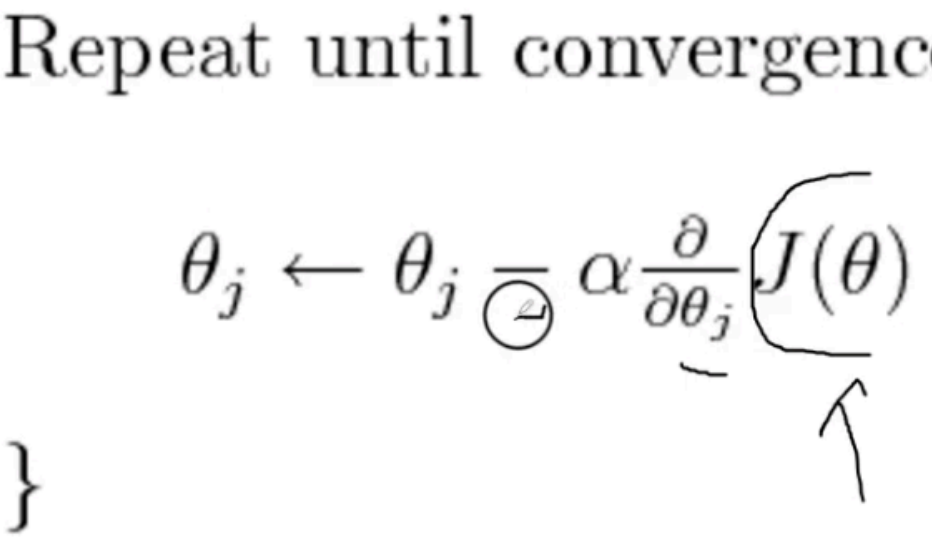
- Gradyanin negatifi olarak tanımlanan '**en dik iniş'** yönünde iteratif olarak parametre değerlerini güncelleyerek ilgili fonksiyonun minimum degerini verebilecek parametreleri bulur.
- Cost fonksiyonunu minimize edebilecek parametreleri bulmak için kullanılır.

Parametre tahmininde en yaygın ve temel algoritma **Gradyan İniş (Gradient Descent)**'tir. Bu algoritma, maliyet fonksiyonunun parametre uzayındaki "en dik iniş" yönünü bularak maliyeti adım adım azaltmayı hedefler.

1. **Başlangıç Noktası:** Parametreler (ağırlıklar ve sapmalar) genellikle rastgele değerlerle başlatılır. Bu, maliyet fonksiyonu yüzeyinde herhangi bir noktadan başlamak gibidir.
2. **Gradyan Hesaplama:** Her iterasyonda, maliyet fonksiyonunun her bir parametreye göre kısmi türevi (gradyanı) hesaplanır. **Gradyan, maliyet fonksiyonunun o anki parametre değerlerinde hangi yönde en hızlı**

arttığını gösteren bir vektördür. Bizim amacımız maliyeti azaltmak olduğu için, gradyanın tersi yönde hareket ederiz.

Matematiksel olarak, bir θ parametresi için güncelleme kuralı şu şekildedir:



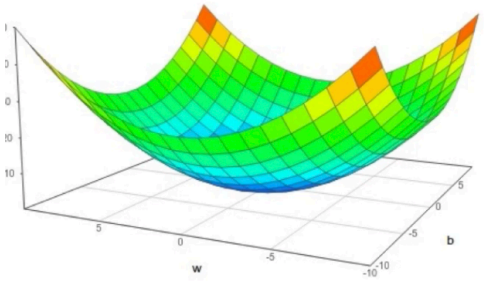
$$\theta_{yeni} = \theta_{eski} - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$

Burada:

- θ: Modelin parametresi (ağırlık veya sapma)
- $J(\theta)$: Maliyet fonksiyonu
- $\frac{\partial J(\theta)}{\partial \theta}$: Maliyet fonksiyonunun θ'ya göre gradyanı (kısmi türevi)
- α (alfa): **Öğrenme Oranı (Learning Rate)**. Bu, her adımda ne kadar büyük bir sıçrama yapacağımızı belirleyen kritik bir hiperparametredir.

3. **Parametre Güncelleme:** Hesaplanan gradyan ve öğrenme oranı kullanılarak parametreler güncellenir. Bu, maliyet fonksiyonu yüzeyinde bir sonraki adıma geçmek anlamına gelir.
4. **İterasyon ve Yakınsama:** Bu adımlar, maliyet fonksiyonunun değeri yeterince küçük olana veya belirli bir iterasyon sayısına ulaşılana kadar

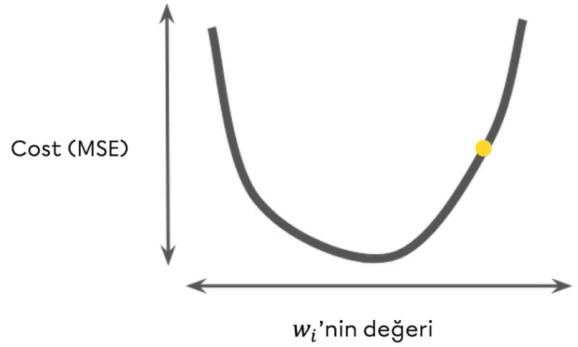
tekrarlanır. Amaç, maliyet fonksiyonunun **global minimumuna** (veya bazen yerel minimumuna) yakınsamaktır.



$$Cost(b,w) = \frac{1}{2m}\sum_{i=1}^{m}((b + wx_i) - y_i)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\underset{\theta_0,\theta_1}{minimize} J(\theta_0, \theta_1)$$



$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\underset{\theta_0,\theta_1}{minimize} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x^{(i)}$$

🔄 Gradient Descent Çeşitleri

Tür	Açıklama
Batch Gradient Descent	Tüm veri seti kullanılır. Yavaş ama kararlı.
Stochastic Gradient Descent (SGD)	Her bir örnek için güncelleme yapılır. Hızlı ama gürültülü.
Mini-Batch Gradient Descent	Veri küçük gruplara (batch) bölünür. Dengeli performans sunar.

📊 Learning Rate Kullanımı

Algoritma	Learning Rate?	Notlar
Linear Regression (SGD)	✅ Evet	`alpha` parametresi ile ayarlanır.
Deep Learning (NN, CNN)	✅ Evet	Optimizer'larda (Adam, RMSprop) kullanılır.
XGBoost/LightGBM	✅ Evet	`learning_rate` veya `eta` parametresi.
Decision Tree/Random Forest	❌ Hayır	Doğrudan kullanılmaz.

Gradyan İnişin Çeşitleri: Ölçek ve Veri Hacmi

Veri setinin büyüklüğüne ve hesaplama kaynaklarına bağlı olarak Gradyan İniş'in farklı varyantları kullanılır:

- **Batch Gradient Descent (Toplu Gradyan İniş):** Her iterasyonda **tüm eğitim veri setini** kullanarak gradyanı hesaplar. Bu, maliyet fonksiyonunun gerçek gradyanını verir ve daha kararlı bir yakınsama sağlar. Ancak, çok büyük veri setleri için hesaplama açısından maliyetli ve yavaş olabilir.
- **Stochastic Gradient Descent (SGD - Stokastik Gradyan İniş):** Her iterasyonda **sadece bir eğitim örneğini** rastgele seçerek gradyanı hesaplar. Bu, çok hızlıdır ve büyük veri setleri için uygundur. Ancak, gradyan hesaplaması gürültülü olduğundan, yakınsama yolu daha dalgalı olabilir ve global minimum etrafında salınım yapabilir.
- **Mini-Batch Gradient Descent (Mini-Toplu Gradyan İniş):** Batch GD ve SGD arasında bir denge kurar. Her iterasyonda **küçük bir alt küme (mini-batch)** kullanarak gradyanı hesaplar. Bu, hem hesaplama verimliliği sağlar hem de SGD'ye göre daha kararlı bir yakınsama yolu sunar. Modern derin öğrenme uygulamalarında en yaygın kullanılan yöntemdir.

Öğrenme Oranı (Learning Rate) ve Optimizörler

- **Öğrenme oranı (α)**, gradyan inişin en hassas hiperparametrelerinden biridir.

    - Çok küçük bir öğrenme oranı, yakınsamanın çok yavaş olmasına neden olabilir.
    - Çok büyük bir öğrenme oranı, maliyet fonksiyonunun minimumu kaçırmasına ve hatta ıraksamasına (diverge) neden olabilir.
- Bu zorluğu aşmak için, öğrenme oranını eğitim süreci boyunca dinamik olarak ayarlayan veya **farklı parametreler için farklı öğrenme oranları kullanan daha gelişmiş optimizasyon algoritmaları (optimizers)** geliştirilmiştir:
    - **Momentum:** Gradyan inişin geçmiş adımlarındaki "ivmesini" kullanarak daha hızlı ve kararlı yakınsama sağlar. Yerel minimumlardan kaçınmaya yardımcı olabilir.
    - **AdaGrad (Adaptive Gradient Algorithm):** Her parametre için geçmiş gradyanların karesine göre öğrenme oranını uyarlar. Sık güncellenen parametreler için öğrenme oranını düşürür, seyrek güncellenenler için artırır.
    - **RMSprop (Root Mean Square Propagation):** AdaGrad'ın bir varyantıdır, ancak geçmiş gradyanların karelerinin üssel hareketli ortalamasını kullanır. Bu, öğrenme oranının sürekli azalmasını engeller.
    - **Adam (Adaptive Moment Estimation):** Momentum ve RMSprop'un avantajlarını birleştiren, en popüler ve genellikle en iyi performans gösteren optimizörlerden biridir. Her parametre için hem ilk moment (ortalama) hem de ikinci moment (varyans) tahminlerini kullanarak öğrenme oranlarını uyarlar.
- Bir doğrusal regresyon modelinin Gradient Descent ile eğitiminde öğrenme hızını (learning rate) çok büyük seçmek ne tür bir sonuca yol açabilir?
    - **Yakınsamama (Divergence):** Öğrenme hızı çok büyük olduğunda, modelin ağırlıkları (parametresi), optimum noktadan (minimum maliyet fonksiyonu) her adımda daha da uzaklaşabilir. Her güncelleme, maliyet fonksiyonunun değerini artırır ve modelin asla doğru sonuca ulaşamamasına neden olur. Bu durum, eğitim sırasında maliyet fonksiyonunun değerinin sürekli yükseldiğini gözlemleyerek anlaşılabilir.
    - **Salınım (Oscillation):** Optimum noktayı tamamen ıskalamasa bile, model bu noktanın etrafında sürekli olarak ileri geri sıçrayabilir ve asla istikrarlı bir şekilde o noktaya yerleşemez. Maliyet fonksiyonunun değeri düşse

bile, istikrarlı bir minimuma ulaşmak yerine sürekli dalgalanmalar gösterir.

- Ağırlık güncellemelerinin sıkça atlandığı yerel minimumlarda sıkışma riskini artırabilir

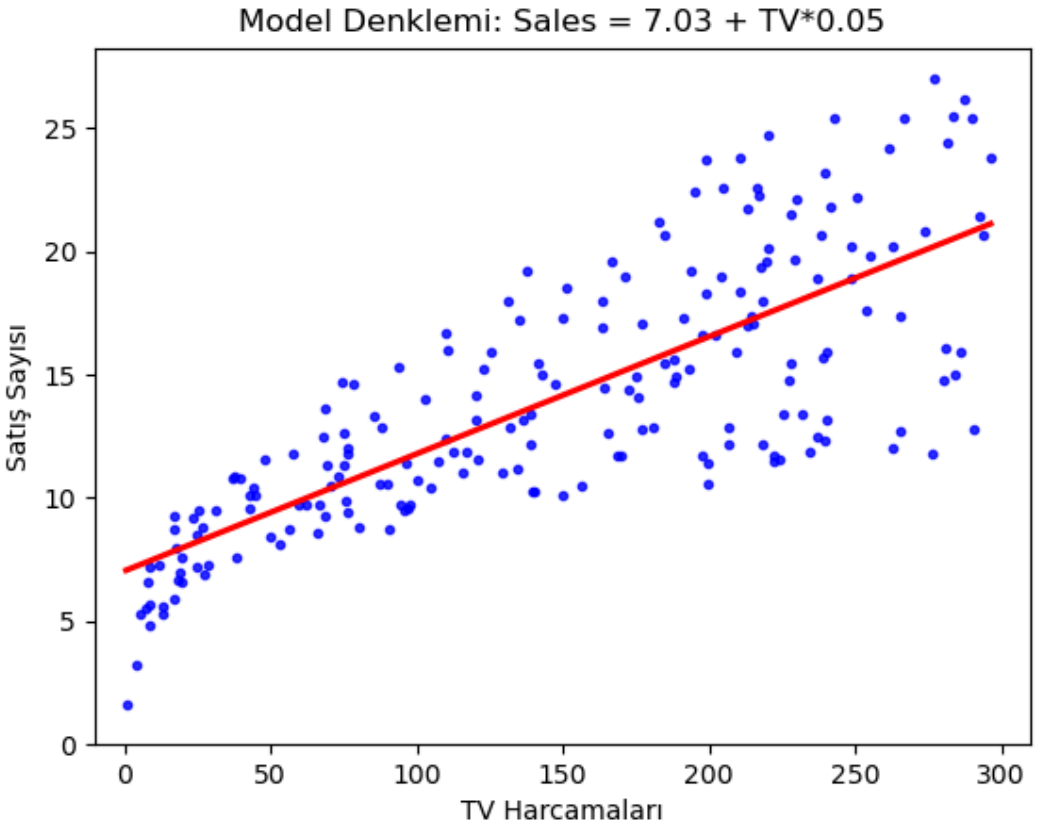
- Bu durumlar, modelin doğru ağırlıkları öğrenmesini engeller ve sonuç olarak **eğitim sürecinin başarısız olmasına** ve **modelin asla iyi performans gösterememesine** yol açar.

Basit Doğrusal Regresyon Modeli (Linear Regression)

```
###########################################################
# Simple Linear Regression with OLS Using Scikit-Learn
###########################################################

df = pd.read_csv("datasets/advertising.csv")
df.shape

X = df[["TV"]]
y = df[["sales"]]


############################
# Model
############################

reg_model = LinearRegression().fit(X, y)

# y_hat = b + w*TV

# sabit (b - bias)→ Intercept
reg_model.intercept_[0]

# tv'nin katsayısı (w1)
reg_model.coef_[0][0]
```

Doğrusal Regresyonda Tahmin İşlemleri

```
############################
# Tahmin
############################

# 150 birimlik TV harcaması olsa ne kadar satış olması beklenir?

reg_model.intercept_[0] + reg_model.coef_[0][0]*150

# 500 birimlik tv harcaması olsa ne kadar satış olur?

reg_model.intercept_[0] + reg_model.coef_[0][0]*500

df.describe().T


# Modelin Görselleştirilmesi
g = sns.regplot(x=X, y=y, scatter_kws={'color': 'b', 's': 9},
                ci=False, color="r")

g.set_title(f"Model Denklemi: Sales = {round(reg_model.intercept_[0], 2)} +
TV*{round(reg_model.coef_[0][0], 2)}")
g.set_ylabel("Satış Sayısı")
g.set_xlabel("TV Harcamaları")
plt.xlim(-10, 310)
plt.ylim(bottom=0)
plt.show()
```

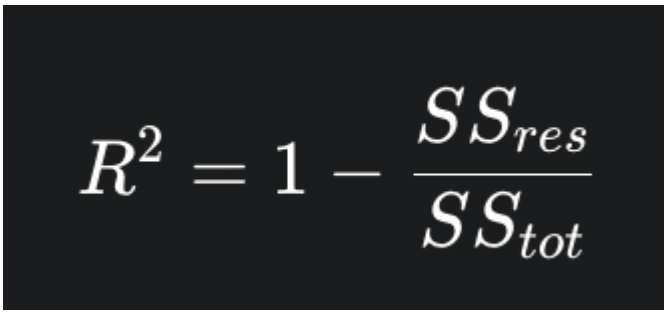


Doğrusal Regresyonda Tahmin İşlemleri

```
############################
# Tahmin Başarısı
############################

# MSE
y_pred = reg_model.predict(X)
mean_squared_error(y, y_pred)
# 10.51
y.mean()
y.std()

# RMSE
```

```
np.sqrt(mean_squared_error(y, y_pred))
# 3.24

# MAE
mean_absolute_error(y, y_pred)
# 2.54

# R-KARE →
reg_model.score(X, y)
```

- **R-Kare**, modelin bağımlı değişkendeki varyansın ne kadarını açıkladığını gösteren bir istatistiktir. Değeri 0 ile 1 arasında değişir (bazı durumlarda negatif olabilir). Formülü:



$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

- Burada SS_res (residual sum of squares), modelin açıklayamadığı varyansı; SS_tot (total sum of squares) ise bağımlı değişkendeki toplam varyansı ifade eder.
    - **Açıklama:** 0'a yakın bir R-Kare değeri, modelin bağımlı değişkendeki varyansın çok azını açıkladığını gösterirken, 1'e yakın bir değer, modelin varyansın büyük bir kısmını açıkladığını gösterir.
    - **Değerlendirme:** Genel olarak, daha yüksek R-Kare değeri daha iyi bir model anlamına gelir. Ancak, tek başına R-Kare'ye bakmak yeterli değildir, çünkü modele daha fazla bağımsız değişken eklemek R-Kare'yi her zaman artıracaktır (modelin gerçekten daha iyi olup olmadığına bakılmaksızın). Bu yüzden, ayarlanmış R-Kare (Adjusted R-squared) gibi metrikler de kullanılır.

Çoklu Doğrusal Regresyon Modeli (Multiple Linear Regression)

```
##########################################################
# Multiple Linear Regression
##########################################################

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

X = df.drop('sales', axis=1)

y = df[["sales"]]


##############################
# Model
##############################

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random
_state=1)

y_test.shape
y_train.shape

reg_model = LinearRegression().fit(X_train, y_train)

# sabit (b - bias)
reg_model.intercept_

# coefficients (w - weights)
reg_model.coef_
```

Çoklu Doğrusal Regresyonda Tahmin İşlemleri

```
############################
# Tahmin
############################

# Aşağıdaki gözlem değerlerine göre satışın beklenen değeri nedir?

# TV: 30
# radio: 10
# newspaper: 40

# 2.90
# 0.0468431 , 0.17854434, 0.00258619

# Sales = 2.90  + TV * 0.04 + radio * 0.17 + newspaper * 0.002

2.90794702 + 30 * 0.0468431 + 10 * 0.17854434 + 40 * 0.00258619

yeni_veri = [[30], [10], [40]]
yeni_veri = pd.DataFrame(yeni_veri).T

reg_model.predict(yeni_veri)
```

Çoklu Doğrusal Regresyonda Tahmin Başarısı

```
############################
# Tahmin Başarısını Değerlendirme
############################

# Train RMSE
y_pred = reg_model.predict(X_train)
np.sqrt(mean_squared_error(y_train, y_pred))
# 1.73

# TRAIN RKARE
reg_model.score(X_train, y_train)
```

```
# Test RMSE
y_pred = reg_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
# 1.41

# Test RKARE
reg_model.score(X_test, y_test)


# 10 Katlı CV RMSE
np.mean(np.sqrt(-cross_val_score(reg_model,
                        X,
                        y,
                        cv=10,
                        scoring="neg_mean_squared_error")))

# 1.69


# 5 Katlı CV RMSE
np.mean(np.sqrt(-cross_val_score(reg_model,
                        X,
                        y,
                        cv=5,
                        scoring="neg_mean_squared_error")))
# 1.71
```

Eğitim Seti Performansı

- **Train RMSE:** `1.73`
- **Train R-Kare:** (Değer verilmemiş, ancak regresyon modelinin `score` metodu genellikle R-Kare verir.)

  Yorum: Eğitim seti üzerindeki RMSE değeri, modelinizin eğitim verileri üzerinde ortalama olarak ne kadar hata yaptığını gösterir. Bu değerin kendi başına yorumlanması zordur; **test seti ve çapraz doğrulama sonuçlarıyla karşılaştırılması** gerekir. Eğer R-Kare değeri çok yüksek (örneğin 0.95 ve

üzeri) ve RMSE çok düşükse, bu durum **aşırı öğrenme (overfitting)** potansiyeline işaret edebilir. Yani model, eğitim verilerini ezberlemiş olabilir.

Gradient Descent ile Doğrusal Regresyon

1. Maliyet Fonksiyonu (Cost Function): `cost_function(Y, b, w, X)`

   Bu fonksiyon, modelimizin belirli b ve w parametreleri için ne kadar **yanlış** tahmin yaptığını ölçer. Doğrusal regresyonda en yaygın kullanılan maliyet fonksiyonlarından biri **Ortalama Kare Hata (Mean Squared Error - MSE)**'dir.

   Teorisi:
   MSE'nin amacı, her bir veri noktası için **tahmin edilen değer (yhat)** ile **gerçek değer (Y)** arasındaki farkın karesini alıp, bu kareli farkların ortalamasını bulmaktır. Kare alma işlemi, hem pozitif hem de negatif hataları pozitif yapar ve büyük hataları daha fazla cezalandırır. Amacımız bu MSE değerini minimize etmektir.

Matematiksel Formülü:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_{hat,i} - y_i)^2$$

veya

$$MSE = \frac{1}{m}\sum_{i=1}^{m}((b + w \times X_i) - Y_i)^2$$

```
############################################################
# Simple Linear Regression with Gradient Descent from Scratch
############################################################

# Cost function MSE
def cost_function(Y, b, w, X):
    m = len(Y)
    sse = 0
```

```
    for i in range(0, m):
        y_hat = b + w * X[i]
        y = Y[i]
        sse += (y_hat - y) ** 2

    mse = sse / m
    return mse
```

2. Ağırlıkları Güncelleme Fonksiyonu (Update Weights): `update_weights(Y, b, w, X, learning_rate)`

   Bu fonksiyon, maliyet fonksiyonunu minimize etmek için b (bias) ve w (weight) parametrelerini küçük adımlarla ayarlar. İşte burası **Gradient Descent**'in kalbidir.

   Teorisi:
   Gradient Descent, maliyet fonksiyonunun grafiği üzerinde en dik iniş yönünü (yani gradyanını) bularak maliyeti azaltmayı amaçlar. Bu yön, maliyet fonksiyonunun b ve w'ye göre kısmi türevleriyle belirlenir. Parametreler, gradyanın tersi yönde güncellenir.

```
# update_weights
def update_weights(Y, b, w, X, learning_rate):
    m = len(Y)
    b_deriv_sum = 0
    w_deriv_sum = 0
    for i in range(0, m):
        y_hat = b + w * X[i]
        y = Y[i]
        b_deriv_sum += (y_hat - y)
        w_deriv_sum += (y_hat - y) * X[i]
    new_b = b - (learning_rate * 1 / m * b_deriv_sum)
    new_w = w - (learning_rate * 1 / m * w_deriv_sum)
    return new_b, new_w
```

3. Eğitim Fonksiyonu (Train Function): `train(Y, initial_b, initial_w, X, learning_rate, num_iters)`

   Bu ana fonksiyon, Gradient Descent algoritmasını belirli bir sayıda iterasyon (döngü) boyunca çalıştırır ve b ile w parametrelerini optimum değerlerine yaklaştırır.

   Teorisi:
   Gradient Descent, maliyet fonksiyonu minimuma ulaşana veya belirli bir iterasyon sayısına ulaşana kadar `update_weights` fonksiyonunu tekrar tekrar çağırır. Her iterasyonda parametreler, maliyeti azaltacak yönde ayarlanır.

```
# train fonksiyonu
def train(Y, initial_b, initial_w, X, learning_rate, num_iters):

    print("Starting gradient descent at b = {0}, w = {1}, mse = {2}".format(init
ial_b, initial_w,
                                                cost_function(Y, initial_b, initial_w,
X)))

    b = initial_b
    w = initial_w
    cost_history = []

    for i in range(num_iters):
        b, w = update_weights(Y, b, w, X, learning_rate)
        mse = cost_function(Y, b, w, X)
        cost_history.append(mse)


        if i % 100 == 0:
            print("iter={:d}    b={:.2f}    w={:.4f}    mse={:.4}".format(i, b, w, ms
e))


    print("After {0} iterations b = {1}, w = {2}, mse = {3}".format(num_iters, b,
w, cost_function(Y, b, w, X)))
    return cost_history, b, w
```

4. Hiperparametreler ve Çalışma

   Kodunuzda tanımlanan hiperparametreler:

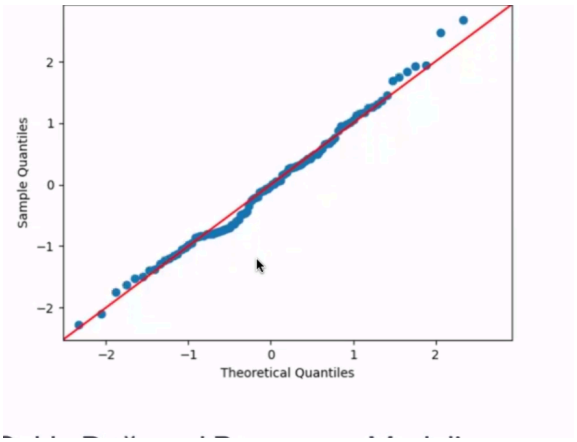
   - `learning_rate = 0.001`: Çok küçük bir adım boyutu, bu da algoritmanın minimuma yavaşça ve stabil bir şekilde yaklaşmasını sağlar.
   - `initial_b = 0.001`, `initial_w = 0.001`: b ve w için başlangıç değerleri. Bu değerler genellikle rastgele seçilir veya sıfıra yakın başlatılır.
   - `num_iters = 100000`: Algoritmanın 100.000 kez parametreleri güncelleyeceği anlamına gelir. Bu, modelin yakınsaması için yeterince büyük bir sayı olabilir.

```
df = pd.read_csv("datasets/advertising.csv")

X = df["radio"]
Y = df["sales"]

# hyperparameters
learning_rate = 0.001
initial_b = 0.001
initial_w = 0.001
num_iters = 100000

cost_history, b, w = train(Y, initial_b, initial_w, X, learning_rate, num_iters)
```

OLS - Ordinary Least Squares - En Küçük Kareler Yöntemi

- OLS, doğrusal regresyon modellerindeki **regresyon katsayılarını (parametreleri)** tahmin etmek için kullanılan en yaygın ve temel yöntemdir. Amacı, **gerçek gözlem değerleri ile model tarafından tahmin edilen değerler arasındaki farkların (hata terimlerinin veya rezidüellerin) karelerinin toplamını minimize eden** bir doğru veya hiperdüzlem bulmaktır.

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.639
Model:                            OLS   Adj. R-squared:                  0.635
Method:                 Least Squares   F-statistic:                     173.4
Date:                Tue, 08 Jul 2025   Prob (F-statistic):           2.10e-23
Time:                        17:35:03   Log-Likelihood:                -591.78
No. Observations:                 100   AIC:                             1188.
Df Residuals:                      98   BIC:                             1193.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -25.0568     18.302     -1.369      0.174     -61.377      11.263
x              4.1435      0.315     13.169      0.000       3.519       4.768
==============================================================================
Omnibus:                        1.302   Durbin-Watson:                   2.421
Prob(Omnibus):                  0.521   Jarque-Bera (JB):                1.361
Skew:                           0.257   Prob(JB):                        0.506
Kurtosis:                       2.750   Cond. No.                         117.
==============================================================================
```



Matris Gösterimi

Bu modeli matris şeklinde şu şekilde ifade edebiliriz:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

Bu durumda model denklemimiz:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \epsilon_i, \quad i = 1, \ldots, n$$

şeklinde olur ve matris formunda:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

olarak yazılır.

Varsayımları:

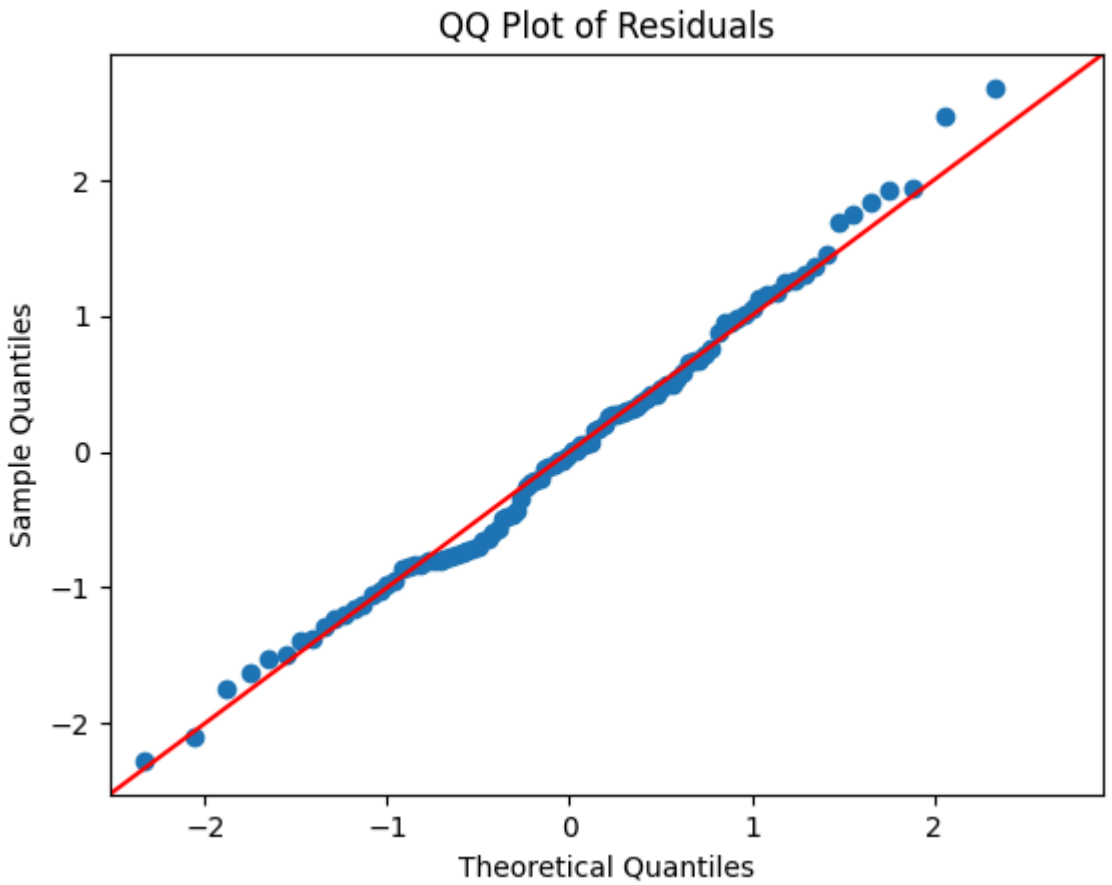
OLS'nin optimal özelliklere sahip olabilmesi ve sonuçlarının güvenilir olması için belirli varsayımların karşılanması gerekir:

1. **Doğrusallık (Linearity):** Bağımlı değişken ile bağımsız değişkenler arasında doğrusal bir ilişki olmalıdır.
2. **Bağımsızlık (Independence of Errors):** Hata terimleri birbirlerinden bağımsız olmalıdır. (Özellikle zaman serisi verilerinde otokorelasyon olmamalıdır.)
3. **Homoskedastisite (Homoscedasticity):** Hata terimlerinin varyansı, bağımsız değişkenlerin tüm değerleri için sabit olmalıdır. Yani, hata terimleri sabit bir varyansa sahip olmalıdır.

4. **Hata Terimlerinin Normal Dağılımı (Normality of Errors):** Hata terimleri normal dağılıma sahip olmalıdır. Bu varsayım, özellikle küçük örneklemler için katsayı tahminlerinin güven aralıkları ve hipotez testleri için önemlidir. Büyük örneklemlerde Merkezi Limit Teoremi nedeniyle daha az kritik olabilir.
5. **Multikolineerlik Yokluğu (No Multicollinearity):** Bağımsız değişkenler arasında yüksek korelasyon olmamalıdır. Yüksek multikolineerlik, katsayı tahminlerinin standart hatalarını artırabilir ve yorumlanmasını zorlaştırabilir.

Q-Q Plot:

- If the residuals are **normally distributed** (which is an assumption of linear regression), the points will fall approximately along the 45-degree reference line.
- **Deviations** from the line (especially at the ends) indicate **non-normality** (e.g., skewness, heavy tails, outliers).
- Systematic curves or S-shapes suggest the residuals are not normal, which may affect the validity of statistical tests and confidence intervals in your regression.



R2:

R-kare, regresyon modelinizdeki bağımsız değişken(ler)in, bağımlı değişkendeki **varyansın ne kadarını açıkladığını gösteren istatistiksel bir ölçüttür.** Başka bir deyişle, modelinizin veriye ne kadar iyi uyduğunu (goodness-of-fit) gösterir.

📌 Doğru R² Formülü:

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\text{SSE}}{\text{SST}}$$

Terimler:

- y_i: Gerçek değer
- $\hat{y}_i$: Tahmin edilen değer
- $\bar{y}$: Gerçek değerlerin ortalaması
- **SST (Total Sum of Squares):** $\sum (y_i - \bar{y})^2$
- **SSE (Error Sum of Squares):** $\sum (y_i - \hat{y}_i)^2$
- **SSR (Regression Sum of Squares):** $\sum (\hat{y}_i - \bar{y})^2$

Düzeltilmiş R Kare

Çoklu regresyonda, değişken sayısı arttıkça R kare değeri otomatik olarak artar. Bu nedenle **düzeltilmiş R kare** kullanılır:

$$\bar{R}^2 = 1 - \left(\frac{SSE/(n-k-1)}{SST/(n-1)} \right)$$

Düzeltilmiş R kare, bağımsız değişken sayısını dikkate alarak modelin uyumunu daha doğru bir şekilde ölçer.

Ortalama Mutlak Yüzde Hata (MAPE - Mean Absolute Percentage Error)

Hata oranlarının yüzde olarak ortalamasıdır:

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

MAPE, tahmin hatasının göreceli büyüklüğünü anlamak için kullanılır ancak $y_i = 0$ durumlarında dikkatli olunmalıdır.

Hata Metrikleri Ozeti

Metrik	Açıklama	Düşük Değer İyi Mi?	Ölçüm Birimi
MSE	Ortalama Kare Hatası	Evet	y'nin karesi cinsinden
RMSE	Kök Ortalama Kare Hatası	Evet	y'nin birimiyle aynı
MAE	Ortalama Mutlak Hata	Evet	y'nin birimiyle aynı
MAPE	Ortalama Mutlak Yüzde Hata	Evet	Yüzde (%)

Regresyonda Bağımlı Değişken Transformasyonu

1. Temel Nokta: Ne Yapıyoruz?

   Regresyonda amaç, bağımsız değişkenlerle (X1, X2, ...) bağımlı değişkeni (Y) en iyi şekilde açıklamak ve
   tahmin etmektir. Ancak bu işlemin bazı varsayımları vardır. Eğer bu varsayımlar sağlanmazsa model
   güvenilmez hale gelir.

2. Multiple Regression Varsayımları

   Temel varsayımlar:
   - Doğrusallık: X'lerle Y arasında doğrusal ilişki
   - Hataların normal dağılımı
   - Sabit varyans (homoscedasticity)
   - Aykırı değer hassasiyeti
   - Multicollinearity olmaması
   Bu varsayımlar bozulursa modelin güvenilirliği düşer.

3. Transformasyon Neden Uygulanır?

   A. Doğrusallığı sağlamak için
   Y ile X'ler arasındaki ilişki doğrusal değilse (eğrisel ise), log veya sqrt dönüşümleriyle lineer hale
   getirilebilir.

   B. Heteroscedasticity düzeltmek için
   Y'nin varyansı artıyorsa, log(Y) gibi dönüşümlerle bu durum sabitlenir.

C. Aykırı değerleri yumuşatmak için
Y'nin içindeki büyük değerler modeli etkiliyorsa transformasyon uygulanır.

D. Yorumu kolaylaştırmak için
log(Y) modeli, katsayıları % değişim olarak yorumlamayı sağlar.

4. Örnek Senaryo

Y = Gelir, X1 = Eğitim, X2 = Yaş, X3 = Deneyim gibi bir modelde, Gelir sağa çarpık olabilir.
Bu durumda:
log(Gelir) = a + b1 * Eğitim + b2 * Yaş + b3 * Deneyim
şeklinde model kurmak, daha kararlı ve yorumlanabilir sonuçlar üretir.

5. Özet Tablo

Neden Y'ye Transformasyon Uygulanır?
- Doğrusallığı sağlamak
- Hata varyansını sabitlemek
- Aykırı değerlerin etkisini azaltmak
- Yorumu kolaylaştırmak