**Python’da Doğrusal Regresyon**

Yazarlar :Enes Duran ,Umur Abdullah Can

[B200109040@subu.edu.tr](mailto:B200109040@subu.edu.tr)

[B200109042@subu.edu.tr](mailto:B200109042@subu.edu.tr)

**Giriş**

Bu çalışmada Facebook Posts Metrics Verisetini kullanarak Pythonda Sıfırdan a Doğrusal Regresyon (Linear Regression) sıfırdan uygulayıp modelleme yapıldı.

Modelleme Yaparken adım adım Verileri İndirip sonra indirilen verileri okuyup verilerimizi

X(bağımsız değişkenler) ve Y(bağımlı değişkenler) olarak ikiye böldük.

Scikit-learn kütüphanesinde bulunan train\_test\_split fonksiyonunu import ederek

Bu ikiye bölünen verileri Eğitim ve Test setlerine ayırdık . Eğitim ve Test veri setlerini varsayılan olarak

%80 eğitim ve %20 test olacak şekilde ayarladık . Lineer regresyonumuza Bir ağırlık (θ) vektörünü

Tanımladık tüm bu bilgileri kullanarak projemizde Gradyan İnişini (Gradient Descent) uyguladık bu

sayede ağırlıklarımızı güncelleme işleminide Gradient Descent yardımı ile yapmış olduk.

Eğitim ve test verileri için Toplam Kare Hatası'nın (SSE) hesapladık ve bu hesaplanan değerleri kaydettik.

Eğitim ve test hatalarını içeren ağırlık matrisinin ve diğer değerlerin döndürülmesi yapılmıştır.

Son olarakta Eğitim ve test hatalarının grafiklerle görselleştirilmesi ve yorumlanması yapılmıştır.

LİNEER REGRESYON

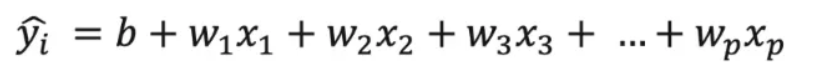
Lineer regresyon, iki veya daha fazla değişken arasındaki ilişkiyi modellemek ve anlamak için kullanılan bir istatistiksel yöntemdir. Temelde, bir bağımlı değişkenin (sonuç değişkeni) bağımsız değişkenlerle (özellikler veya tahminleyiciler) nasıl ilişkili olduğunu anlamak için kullanılır. Lineer regresyon, bu ilişkiyi bir doğru (lineer fonksiyon) ile temsil etmeye çalışır.

**Tek değişkenli doğrusal regresyon modeli**

yazı tipi, metin, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Çok değişkenli doğrusal regresyon modeli**

****

\* b sabiti literatürde intercept, bias olarak da ifade edilir.

\* wᵢ ifadesi kat sayı yani ağırlıktır. Bazı kaynaklarda coefficient olarak da ifade edilir. Kat sayısı olduğu değişkeni ifade etmek ve etkilerini belirlemek için kullanılır.

**Regresyon Modellerinde Başarı Değerlendirme ( MSE, RMSE, MAE )**

**MSE: (Mean Square Error)**: Hata kareler ortalamasıdır. Gerçek Değerler ile Tahmin Edilen Değerler arasındaki farkların (yani hataların) karelerinin toplamının gözlem sayısına bölünmesidir.

metin, yazı tipi, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**RMSE: (Root Mean Square Error)** MSE metriğinde kare alma işleminden ötürü oluşan şişmeyi önlemek ve daha doğru bir biçimde ölçümlendirme yapmak için kullanılır

yazı tipi, beyaz, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**MAE: (Mean Absolute Error)**

RMSE ile benzer amaçla kullanılan MAE, gerçek değerler ve tahmin edilen değerlerin farklarını (yani hata değerlerini) mutlak değer içerisinde değerlendirerek hem negatif/pozitif olma problemini hem de kare alma sonucu oluşan hata karelerin şişmesi problemine çözüm getirir.

yazı tipi, beyaz, diyagram, simge, sembol içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Parametrelerin Tahmin Edilmesi ( Ağırlıkların Bulunması )**

Parametrelerin tahmin edilmesi kapsamında yapılan işlem, en küçük hatayı verecek olan sabiti ve ağırlıkları bulmaktır.

Bu süreçte MSE ifadesinin benzeri olan Cost Fonksiyonu kullanılır.

Amaç belirli sabit ve ağırlık kombinasyonlarına karşılık elde edilen hata değerlerini gözlemleyip, bu sabit ve ağırlık değerlerini optimum olacak şekilde yani Cost fonksiyonunu en küçük verecek şekilde seçmeye çalışmaktır.

yazı tipi, metin, beyaz, hat sanatı, kaligrafi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu amaçla optimum ağırlıkları ve sabit değerini bulmak için **Normal Denklemler Yöntemi** ve **Gradient Descent** Yöntemi kullanılır.

**Normal Denklemler Yöntemi: (En Küçük Kareler Yöntemi)** Türeve dayalı, matris formunda, Analitik bir çözümdür. OLS yöntemi olarak da isimlendirilir.

metin, yazı tipi, el yazısı, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Gradient Descent Yöntemi: (Optimizasyon Çözümü**) Gradient Descent makine öğrenmesinden bağımsız olarak bir optimizasyon yapma yöntemidir. Bu yöntemin amacı bir fonksiyonu minimum yapacak parametre değerlerini bulmaktır.

**Gradient Descent yöntemi** ile herhangi bir türevlenebilir fonksiyon, ilgili fonksiyonu minimum yapabilecek parametreleri bulmak adına optimize edilebilir.

**\*\***bu süreçte optimize edilmek istenen fonksiyonun ilgili parametreye göre kısmi türevleri alınır. Elde edilen türev sonuçlarına (gradyan değeri) göre güncelleme işlemi gerçekleştirilir.

**\*\***Türev sonuçları ilgili fonksiyonun maksimum artış yönünü verir. Dolayısıyla maksimum artış yönünü veren gradyanın tersine doğru belirli bir şiddet ile gidilerek (yani parametrenin eski değerinde değişiklik yaparak) her iterasyonda hatanın azalması sağlanır.

yazı tipi, hat sanatı, kaligrafi, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**LİNEER REGRESYON YAPISI**

diyagram, çizgi, metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

**GRADIENT DESCENT YAPISI**

metin, diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Metodoloji**

*import* pandas *as* pd

-Bu Python kodu, Pandas kütüphanesini **pd** ismiyle içe aktarır (import eder).

*import* numpy *as* np

--Bu Python kodu, Numpy kütüphanesini **np** ismiyle içe aktarır (import eder).

*import* matplotlib.pyplot *as* plt

--Bu Python kodu, **matplotlib** kütüphanesini **plt** ismiyle içe aktarır (import eder). **matplotlib**, grafik ve görselleştirmeler oluşturmak için kullanılan bir kütüphanedir

*from* sklearn.preprocessing *import* StandardScaler

--Bu Python kodu, **scikit-learn** kütüphanesinden **StandardScaler** sınıfını içe aktarır (import eder). **StandardScaler**, veri setlerini standartlaştırmak veya normalize etmek için kullanılan bir ölçekleme yöntemini sağlayan bir sınıftır.

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

--Bu Python kodu, scikit-learn kütüphanesinden **train\_test\_split** fonksiyonunu içe aktarır (import eder). **train\_test\_split**, veri setini eğitim ve test setlerine bölme amacıyla kullanılan bir fonksiyondur.

*from* sklearn.preprocessing *import* OneHotEncoder

Bu Python kodu, scikit-learn kütüphanesinden **OneHotEncoder** sınıfını içe aktarır (import eder). **OneHotEncoder**, kategorik (nominal) verileri sayısal formata dönüştürmek için kullanılan bir kodlama tekniği olan "One-Hot Encoding" işlemini uygular.

*import* statsmodels.api *as* sm

**import statsmodels.api as sm** ifadesi, **statsmodels** kütüphanesini içe aktarmak ve bu kütüphanenin **sm** takma adıyla kullanılmasını sağlamak için kullanılmıştır. **statsmodels**, istatistiksel modeller oluşturmak, doğrusal regresyon analizi yapmak ve hipotez testleri gibi istatistiksel yöntemleri uygulamak için kullanılan bir Python kütüphanesidir.

---- Lineer Regresyon fonksiyonumuzu tanımlayıp başlatıyoruz.

def \_\_init\_\_(*self*, *learning\_rate*, *no\_of\_itr*):

*self*.learning\_rate = *learning\_rate*

*self*.no\_of\_itr = *no\_of\_itr*

**Bu metodun parametreleri:**

**self: Bir sınıf örneğini temsil eden referans.**

**learning\_rate:** Modelin eğitilirken her adımda ne kadar öğrenme yapması gerektiğini belirten bir parametre.

**no\_of\_itr:** **Gradyan iniş (gradient descent)** veya **eğitim döngüsü** sırasında kaç iterasyon (adım) gerçekleştirileceğini belirten bir parametre.

Bu parametreler, sınıfın örneğinin oluşturulması sırasında kullanıcı tarafından belirlenen değerlerdir.

def fit(*self*, *X*, *Y*):

*self*.m, *self*.n = *X*.shape

*self*.w = np.zeros((*self*.n, 1))

*self*.b = 0

*self*.X = *X*

*self*.Y = *Y*

*for* i *in* range(*self*.no\_of\_itr):

*self*.update\_weigths()

# Lineer Sınıf Parametreler başlatılıyor.

**self.m, self.n = X.shape:** Bu satır giriş özelliği matrisinden satır sayısını ( self.m) ve sütun sayısını ( ) çıkarır . eğitim örneklerinin sayısını ve özelliklerin sayısını temsil eder.

**self.w = np.zeros((self.n, 1)):** Bu çizgi ağırlık vektörünü ( self.w) sıfırlardan oluşan bir sütun vektörüne başlatır.

**for i in range(self.no\_of\_itr):** Gradyan İnişinin uygulanması

**self.update\_weigths()**

def update\_weigths(*self*):

        Y\_prediction = *self*.predict(*self*.X)

        dw = -(*self*.X.T).dot(*self*.Y - Y\_prediction)/*self*.m

        db = -np.sum(*self*.Y - Y\_prediction)/*self*.m

*self*.w = *self*.w - *self*.learning\_rate \* dw

*self*.b = *self*.b - *self*.learning\_rate \* db

**def update\_weigths(self):**Ağırlığımızı güncellemek için bir fonskiyondur

**Y\_prediction = self.predict(self.X)=** Mevcut ağırlıklara ve sapmalara göre tahmin edilen değerleri hesaplar .

**Gradyanların hesaplanması:**

dw = -(self.X.T).dot(self.Y - Y\_prediction)/self.m

db = -np.sum(self.Y - Y\_prediction)/self.m

**Ağırlıkları ve biası güncelleme**

self.w = self.w - self.learning\_rate \* dw

self.b = self.b - self.learning\_rate \* db

def predict(*self*, *X*):

*return* *X*.dot(*self*.w) + *self*.b

**predict** Sınıftaki yöntem, eğitilmiş **Linear\_Regression** doğrusal regresyon modelini kullanarak tahminler yapmaktan sorumludur.

self: Sınıfın örneğini ifade eder.

X: Tahmin yapılması gereken giriş özelliklerini (veri matrisi) temsil eder.

X.dot(self.w): Giriş özelliklerinin nokta çarpımını Xağırlık vektörüyle gerçekleştirir

self.w. Bu adım esas olarak girdi özelliklerinin ve bunlara karşılık gelen ağırlıkların doğrusal kombinasyonunu hesaplar.

**b:**bias değeri

 def gradient\_descent(*self*):

        Y\_prediction = *self*.predict(*self*.X)

*# Gradyanları hesaplama*

        dw = -(*self*.X.T).dot(*self*.Y - Y\_prediction) / *self*.m

        db = -np.sum(*self*.Y - Y\_prediction) / *self*.m

**def gradient\_descent(self):** Gradyan inişini gerçekleştirir

**dw = -(self.X.T).dot(self.Y - Y\_prediction) / self.m:**ağırlıkların gradyanını hesaplar

**db = -np.sum(self.Y - Y\_prediction) / self.m:** biasın gradyanını hesaplar

def print\_weights(*self*):

        print('İlgili özellikler için ağırlıklar :')

        print(*self*.w)

        print()

        print('Regresyon için bias değeri ', *self*.b)

**def print\_weights(self):**Ağırlığı yazdırmak için bir fonksiyon.

**print(self.w):**Ağırlığı yazdırır

**print('Regresyon için bias değeri ', self.b):**bias değerini yazdırır

veriseti = pd.read\_csv('dataset/dataset\_Facebook.csv',*delimiter*=";")

**#**verisetimizi yükleyip veriseti değişkenimize atama işlemi gerçekleştiriyoruz

**# delimiter=";** CSV dosyasındaki değerlerin 'dataset\_Facebook.csv'noktalı virgülle ( ;) ayrıldığını açıkça belirtir.

veriseti.dropna()

veriseti['Paid'].fillna(1, *inplace*=True)

veriseti['like'].fillna(1, *inplace*=True)

veriseti['share'].fillna(1, *inplace*=True)

**veriseti.dropna():**verisetindeki null değerlerini öne çıkarma

**veriseti['Paid'].fillna(1, inplace=True)**

**veriseti['like'].fillna(1, inplace=True)**

**veriseti['share'].fillna(1, inplace=True)**

**fillnaNaN** (boş) değerleri belirtilen değerle (bu durumda 1) değiştirmek için kullanılır. Parametre **inplace=True**, değişikliklerin **doğrudan 'veriseti' DataFrame'de** yapılmasını sağlar.

xSutun =['Category', 'Page total likes', 'Post Month', 'Post Hour', 'Post Weekday', 'Paid']

ySutun = ['Total Interactions']

Datasetini **x ve y** olarak sütunlara bölme işlemi

x = veriseti.iloc[:, [0, 2, 3, 4, 5, 6,13,15,16,17]]

y = veriseti.iloc[:, 18:]

Datasetteki sütün sıralarına göre hangisi x ve y ‘ nin hangi sütunların hangi satılarını tamamen alacağı işlemini yapıyoruz

mesela ‘y’ 18.endexli sütunun tüm satılarını kapsar.

**iloc**, tam sayı konumuna dayalı indeksleme için kullanılır**.**

ohe = OneHotEncoder()

c = veriseti.iloc[:, 1:2]

c = ohe.fit\_transform(c).toarray().astype(int)

tip = pd.DataFrame(*data*=c, *index*=range(500), *columns*=['Link', 'Photo', 'Status', 'Video'])

x = pd.concat([tip, x.iloc[:, :]], *axis*=1)

**ohe = OneHotEncoder():** Sınıfın bir örneği OneHotEncoderoluşturulur. Tek sıcak kodlama, etiket olarak temsil edilen kategorik verileri, tahminleri iyileştirmek için makine öğrenimi algoritmalarına sağlanabilecek bir formata dönüştürmek için kullanılan bir işlemdir.

**c = ohe.fit\_transform(c).toarray().astype(int)=** Type' sütununu numerik değerlere dönüştürüp doğruluğu arttırır.

**tip=pd.DataFrame(data=c):** Bu, tek sıcak kodlanmış diziyi ‘c’ veri olarak kullanarak 'tip' adında yeni bir Pandas DataFrame oluşturur.

**index=range(500):** DataFrame'e 0 ila 499 arasında değişen bir dizin verilir. Bu, kullanılarak belirtilir range(500)

**columns=['Link', 'Photo', 'Status', 'Video']:** DataFrame'in sütunları 'Bağlantı', 'Fotoğraf', 'Durum' ve 'Video' olarak adlandırılır.

**pd.concat([tip, x.iloc[:, :]], axis=1):** Bu satır iki DataFrame'i sütunlar boyunca birleştirir.

x\_train ,x\_test ,y\_train ,y\_test =train\_test\_split(x,y,*test\_size*=0.2,*random\_state*=10)

**train\_test\_split:** Bu, veri kümenizi karıştırıp iki alt kümeye bölen bir scikit-learn işlevidir: eğitim ve test kümeleri.

**x ve y :** Özellik ve hedef veri kümeleri.

**test\_size=0.2:** Bu parametre, test bölmesine dahil edilecek veri kümesinin oranını belirtir. Burada %20'ye ayarlanmıştır; bu, **verilerin %80'inin eğitim** için ve **%20'sinin test** için kullanılacağı anlamına gelir.

model = Linear\_Regression(*learning\_rate*=0.001,

*no\_of\_itr*=20000)

scaler=StandardScaler()

Linear\_Regression()=tanımladığımız Lineer regresion sınıfımız.

**learning\_rate=0.001:** Bu, gradyan iniş optimizasyon algoritmasında kullanılan öğrenme oranı hiper parametresidir

**no\_of\_itr=20000:** Bu parametre, gradyan iniş optimizasyon algoritmasının çalışacağı yineleme veya dönem sayısını temsil eder. Algoritma, hatayı en aza indirmek için model parametrelerini yinelemeli olarak günceller.

**scaler = StandardScaler():**

Bu satır StandardScalerscikit-learn'den sınıfın bir örneğini oluşturur.

**StandardScaler** veri önişleme aşamasında kullanılır.

x\_train\_scaled=scaler.fit\_transform(x\_train)

x\_test\_scaled=scaler.transform(x\_test)

y\_train\_scaled=scaler.fit\_transform(y\_train)

y\_test\_scaled=scaler.transform(y\_test)

**scaler.fit\_transform(x\_train):** Bu yöntem, eğitim setinden ( x\_train) standardizasyon için gerekli ortalama ve standart sapmayı hesaplar ve ardından hesaplanan bu değerleri kullanarak eğitim setini ölçeklendirir.

**fit\_transform**: yöntem burada kullanıldı çünkü eğitim setinden ortalama ve standart sapmayı hesaplamak ve ardından dönüşümü uygulamak istiyoruz.

**scaler.transform(x\_test)**: Bu yöntem , eğitim setinden hesaplanan ortalama ve standart sapmayı kullanarak aynı dönüşümü (ölçeklemeyi) test setine ( ) uygular . Burada bu transformyöntem kullanılır çünkü test setini ölçeklendirmek için eğitim setinden hesaplanan ortalama ve standart sapmayı kullanıyoruz.

Aynı şekilde tüm hesaplamalar y\_train ve y\_test içinde uygulanır.

model.fit(x\_train\_scaled, y\_train\_scaled)

model.print\_weights()

tahmin=model.predict(x\_test\_scaled)

hata\_test=y\_test\_scaled-tahmin

tahmin\_egitim=model.predict(x\_train\_scaled)

hata\_egitim=y\_train\_scaled-tahmin\_egitim

**model.fit(x\_train\_scaled, y\_train\_scaled):** bir makine öğrenimi modelini eğitmek için kullanılır.

**model.print\_weights**():eğitilen lineer regresyon modelinin öğrenilen ağırlıklarını ve bias değerini ekrana yazdırmaktır.

**tahmin=model.predict(x\_test\_scaled):**ölçeklendirilmiş test verileri (x\_test\_scaled) üzerinde tahminler yapmaktadır.

**hata\_test=y\_test\_scaled-tahmin**: modelin tahminleri ile gerçek test etiketleri arasındaki hataları hesaplar.

**tahmin\_egitim=model.predict(x\_train\_scaled)**: eğitilmiş bir modeli kullanarak, ölçeklendirilmiş eğitim verisi (x\_train\_scaled) üzerinde tahminler yapmaktadır.

**hata\_egitim=y\_train\_scaled-tahmin\_egitim**: modelin eğitim verisi üzerindeki tahminler ile gerçek eğitim etiketleri arasındaki hataları hesaplar.

plt.scatter(np.arange(len(hata\_egitim)),hata\_egitim,*label*="Eğitim Hataları")

plt.scatter(np.arange(len(hata\_test)),hata\_test,*label*="Test Hataları")

plt.scatter(np.arange(len(hata\_egitim)),hata\_egitim,*label*="Eğitim Hataları")

eğitim verisi üzerindeki hataların dağılımını görselleştirmek amacıyla bir saçılım (scatter) grafiği oluşturur.

plt.scatter(np.arange(len(hata\_test)),hata\_test,*label*="Test Hataları")

test verisi üzerindeki hataların dağılımını görselleştirmek amacıyla bir saçılım (scatter) grafiği oluşturur

grafiğini daha anlaşılır hale getirmek ve grafiği görsel olarak düzenlemek için kullanılır.

plt.xlabel('Veri noktası')

x ekseni etiketi

plt.ylabel('Hata')

y ekseni etiketi

plt.legend()

etiketleri göster

plt.show()

grafiği gösterir

residuals = y\_test\_scaled - tahmin

sm.qqplot(residuals, *line*='s')

plt.title('QQ Çizgisi')

plt.show()

**residuals = y\_test\_scaled - tahmin:** Modelin tahminleri ile gerçek test etiketleri arasındaki hatalar (residuals) hesaplanır. Bu hatalar, modelin ne kadar başarılı olduğunu gösteren önemli bir ölçüdür.

**sm.qqplot(residuals, line='s'):** Statsmodels kütüphanesinin qqplot fonksiyonu kullanılarak, hataların QQ çizgesi oluşturulur. veri setinin normal dağılıma ne kadar yakın olduğunu gösterir.

**plt.title('QQ Çizgisi'):** Çizgenin başlığı belirlenir.

**plt.show():** Çizge ekranda gösterilir.

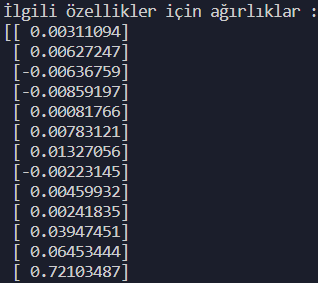
mse = np.mean((hata\_test)\*\*2)

Ortalama kare hatası

sse = np.sum((hata\_test)\*\*2)

toplam kare hatası

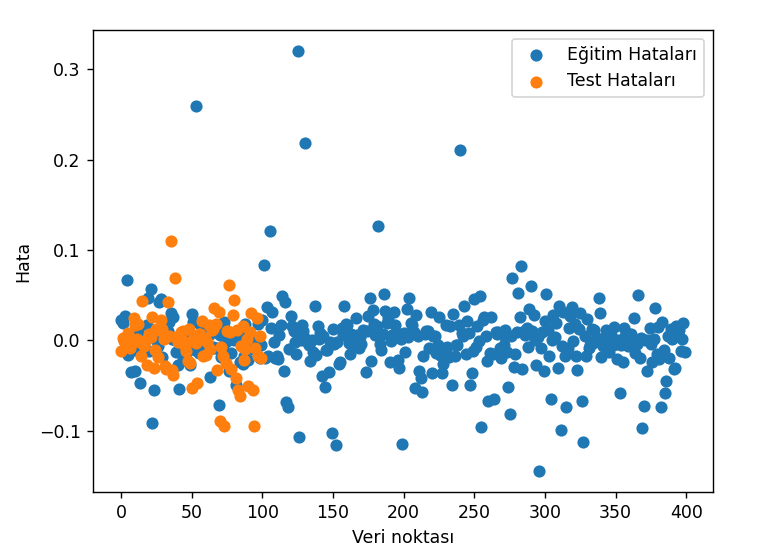
**Ekran Çıktıları**

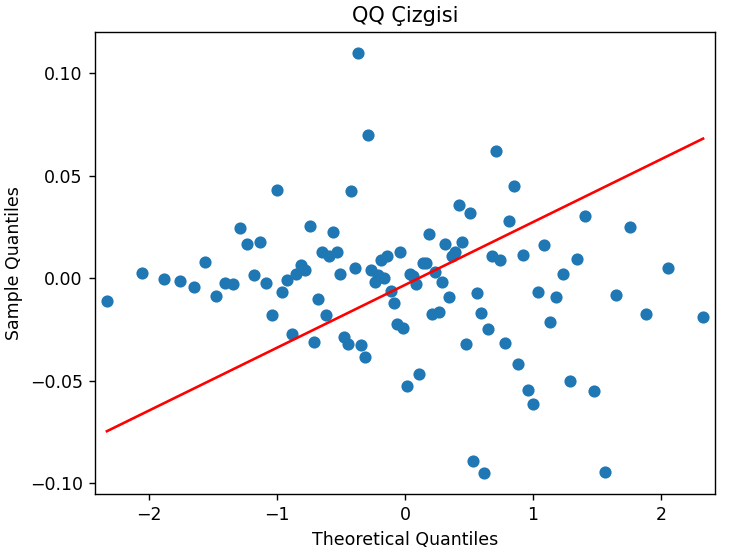
****

****









**Sonuçlar**

veri setindeki bir hedef değişkenin tahminlenmesi için basit bir lineer regresyon modelini eğitmekte ve modelin performansını çeşitli metriklerle değerlendirmektedir. Modelin başarısı, hata analizi ve görselleştirmeler kullanılarak değerlendirilmektedir.

**Kaynakça**

<https://medium.com/analytics-vidhya/machine-learning-implementing-various-regression-algorithms-to-predict-boston-house-prices-c87f961de981>

<https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155>

<https://www.geeksforgeeks.org/>

[www.youtube.com](http://www.youtube.com)

<https://en.wikipedia.org/wiki/Linear_regression>

<http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm>

<https://www.techtarget.com/searchenterpriseai/definition/linear-regression>