28.12.2023

YAPAY ÖĞRENMENİN TEMELLERİ

Proje Raporu

Doğrusal Regresyon (Linear Regression)

Ad: Umut Can | Ad: Emircan

Soyad: YILDIZ | Soyad: ŞAHAN

İÇİNDEKİLER

1. Proje özeti ve görevin açıklanması

Proje kapsamında, Python kullanarak Doğrusal Regresyon'u adım adım uygulama amacımız, Facebook gönderi metrikleri üzerinde tahminler yapabilecek bir model oluşturmaktır. Veri setimizdeki çeşitli öznitelikler arasında kategori, sayfa beğenileri, gönderi türleri, yayın zamanı (ay, saat, gün), ve reklam durumu gibi faktörler bulunmaktadır. Temel hedefimiz, bu verileri kullanarak gönderilerin toplam etkileşimlerini ("Total Interactions") tahmin edebilecek bir regresyon modeli oluşturmaktır.

Projenin ilk adımı, veri setinin indirilmesi ve keşifsel veri analiziyle başladı. Bu aşamada, veri seti yapısını anlamak ve içeriğini incelemek için çeşitli analizler yaptık. Daha sonra, veri ön işleme adımlarıyla seçilen özniteliklere odaklandık ve model oluşturmak için gerekli veri setini hazır hale getirdik.

Veri setini eğitim ve test setlerine ayırdık (%80 eğitim, %20 test) ve ardından model eğitimi için adımlar izledik. Ağırlık vektörünü tanımladık ve Gradyan İnişi yöntemini kullanarak modelimizi eğittik. Bu aşamada modelin performansını ölçmek için Toplam Kare Hatası, Ortalama Mutlak Hata (MAE) ve Ortalama Kare Hatası (MSE) gibi metrikleri kullandık.

Modelimizin iyileştirilmesi için öznitelikler üzerinde değişiklikler yaparak daha iyi sonuçlar elde etmeye çalıştık. Yaptığımız değişikliklerin modelin doğruluğunu ve tahmin yeteneğini nasıl etkilediğini görsel grafiklerle analiz ederek yorumladık.

Bu proje, Doğrusal Regresyon'un teorik temellerini anlamak, pratik uygulama adımlarını takip etmek ve gerçek veriler üzerinde modelleme yaparak model performansını iyileştirmek isteyenler için faydalı bir kaynak olmayı amaçlıyor.

2. Lineer Regresyon

a. Lineer Regresyon nedir?

Doğrusal regresyon bir makine öğrenimi modelidir ve bu model, bağımsız değişkenlerin (X) bağımlı değişken üzerindeki etkisini modellemek için kullanılır. Temel olarak, bağımlı değişkenin (Y) bağımsız değişkenler tarafından ne ölçüde açıklanabileceğini anlamaya çalışır.

Bir doğrusal regresyon modeli, şu formülle ifade edilir:

$$Y=\beta0+\beta1X1+\beta2X2+\ldots+\beta nXn+\varepsilon$$

Burada:

- Y : Bağımlı değişken
- X1,X2,…,Xn : Bağımsız değişkenler
- β0 : Bias terimi (kesme noktası)
- β1,β2,…,βn : Bağımsız değişkenlerin katsayıları
- ε : Hata terimi (gerçek değer ile tahmin arasındaki fark)

Model, veri setini kullanarak katsayıları (β 'ları) bulmaya çalışır. Bu katsayılar, veri setindeki bağımsız değişkenlerin bağımlı değişkendeki etkisini temsil eder.

Doğrusal regresyonun temel amacı, veri noktalarının dağılımı içinde en iyi uyumu sağlayacak doğruyu (veya düzlemi, eğer çoklu değişken varsa) bulmaktır. Bu genellikle "en küçük kareler yöntemi" olarak bilinen yöntemle yapılır; yani, modelin tahminlerinin gerçek değerler ile olan farklarının karelerinin toplamını en aza indirecek şekilde katsayılar belirlenir.

- ✓ **Lineer regresyonun kullanımı:**

Tahmin: Yeni veriler üzerinde tahmin yapmak için kullanılabilir. Örneğin, bir reklam harcaması ile satışlar arasındaki ilişkiyi modellemek ve gelecekteki satışları tahmin etmek.

İlişkiyi Anlama: İki veya daha fazla değişken arasındaki ilişkiyi anlamak için kullanılır. Örneğin, bir ilaç dozu ile hastalık semptomları arasındaki ilişkiyi anlamak.

Öznitelik Seçimi: Hangi değişkenlerin modelde yer alacağını belirlemek için kullanılır. Bu, gereksiz veya etkisiz değişkenlerin modelden çıkarılmasına yardımcı olur.

Anomalileri Belirleme: Anomali tespiti için kullanılabilir. Eğer model, beklenmedik bir davranış gösteriyorsa bu, dikkate alınması gereken potansiyel bir anormallik olabilir.

Doğrulama ve Test: Modelin performansını değerlendirmek için kullanılır. Gerçek değerlerle model tarafından tahmin edilen değerler arasındaki farkları analiz ederek modelin doğruluğunu değerlendirebiliriz.

Öngörülebilirlik: Gelecekteki olayları, trendleri veya ilişkileri anlamak ve tahmin etmek için kullanılır. Örneğin, ekonomik verilerle gelecekteki bir endeksin tahmin edilmesi.

Lineer regresyon, temel yapısı ve kullanım kolaylığı nedeniyle genellikle ilk adım olarak tercih edilir. Ancak, veri setinin karmaşıklığı arttıkça veya doğrusal olmayan ilişkiler devreye girdiğinde, daha karmaşık modellerin kullanılması gerekebilir.

b. Yazılan Kodda Lineer Regresyonun Açıklanması

✓ LinearRegression Sınıfının Başlatılması (__init__ Fonksiyonu):

- X ve Y verileriyle çağrılır.
- X verileri normalize edilir ((X - mean) / std).
- Özellikleri birleştirerek, bağımsız terimi içeren X matrisi oluşturulur.
- İlgili veriler sınıfa atanır ve gerekli özellikler hesaplanır (örneğin, örnek sayısı m ve özellik sayısı n).

```
def __init__(self, X, Y):
    np.random.seed(42)  # Verilerin tekrar üretilebilirliği için rastgele bir tohum değeri belirler
    self.mean = np.mean(X, axis=0)  # X verisinin sütunları için ortalamaları alır
    self.std = np.std(X, axis=0)    # X verisinin sütunları için standart sapmaları hesaplar
    X = (X - self.mean) / self.std  # Özellikleri normalize eder
    ones = np.ones(X.shape[0])      # Bias birimini temsil eden 1'lerden oluşan bir dizi oluşturur
    X = np.column_stack((ones, X))  # Bias birimini X verisine ekler
    self.X = X                      # Normalize edilmiş X verisini saklar
    self.Y = Y                      # Etiketleri saklar
    self.m = X.shape[0]             # Eğitim örneklerinin sayısını saklar
    self.n = X.shape[1]             # Özellik sayısını saklar
    self.theta = np.random.randn(X.shape[1])  # Rastgele theta değerleri oluşturur
```

✓ Maliyet Fonksiyonunun Hesaplanması (computeCostFunction Fonksiyonu):

- Hesaplanan hipotez (h = X * theta) kullanılarak maliyet hesaplanır (J = (1 / (2 * m)) * sum((h - Y) ** 2)).

```
def computeCostFunction(self):
    h = np.matmul(self.X, self.theta)  # Tahminleri hesaplar
    self.J = (1 / (2 * self.m)) * np.sum((h - self.Y) ** 2)  # Hata fonksiyonunu hesaplar
    return self.J
```

✓ Gradient İnişin Yapılması (performGradientDescent Fonksiyonu):

- Belirli bir iterasyon sayısında (num_of_iter) ve öğrenme oranında (alpha) gradient inişi uygular.
- Her iterasyonda maliyeti ve theta değerlerini günceller.

```
def performGradientDescent(self, num_of_iter, alpha):
    self.Cost_history = []        # Her iterasyondaki maliyet değerlerini saklamak için bir liste oluşturur
    self.theta_history = []       # Her iterasyondaki theta değerlerini saklamak için bir liste oluşturur
    for x in range(num_of_iter):  # Belirlenen iterasyon sayısı kadar döngü yapar
        h = np.matmul(self.X, self.theta)  # Tahminleri hesaplar
        J = self.computeCostFunction()     # Hata fonksiyonunu hesaplar
        self.Cost_history.append(J)       # Maliyet değerini listeye ekler
        self.theta_history.append(self.theta)  # Theta değerlerini listeye ekler
        temp = h - self.Y                  # Geçici bir değişken oluşturur
        self.theta = self.theta - (alpha / self.m) * (self.X.T.dot(temp))  # Gradient iniş adımını uygular
    return self.theta, self.Cost_history, self.theta_history  # En iyi theta değeri ve maliyet geçmişini döndürür
```

✓ **Tahmin Yapılması (predictt Fonksiyonu):**

- Verilen test verileri üzerinde tahminleme yapar.
- Test verilerini normalize eder ve uygun formata getirir.
- theta değerlerini kullanarak tahmin yapar.

```
def predictt(self, X_test):
    # X_test verisini eğitim setinin ortalaması ve standart sapması ile normalize eder
    X_test = (X_test - self.mean) / self.std
    ones = np.ones(X_test.shape[0])    # Bias birimini temsil eden 1'lerden oluşan bir dizi oluşturur
    X_test = np.column_stack((ones, X_test))  # Bias birimini X_test verisine ekler

    if X_test.shape[1] != self.X.shape[1]:  # X_test'teki özellik sayısını kontrol eder
        raise ValueError(f"X_test'teki özellik sayısı ({X_test.shape[1]}) eğitimde kullanılan özellik sayısıyla

    Y_pred = np.matmul(X_test, self.theta)  # Tahminleri yapar
    return Y_pred  # Tahminleri döndürür
```

✓ **Normal Denklemin Kullanılmasıyla Tahmin Yapılması (predictUsingNormalEquation Fonksiyonu):**

- Normal denklemi kullanarak theta değerlerini hesaplar ve tahmin yapar.

```
def predictUsingNormalEquation(self, X_test, Y_test):
    # X_test verisini eğitim setinin ortalaması ve standart sapması ile normalize eder
    X_test = (X_test - self.mean) / self.std
    ones = np.ones(X_test.shape[0])    # Bias birimini temsil eden 1'lerden oluşan bir dizi oluşturur
    X_test = np.column_stack((ones, X_test))  # Bias birimini X_test verisine ekler

    inv = np.linalg.inv(np.matmul(self.X.T, self.X))  # X'in transpozesi ile X'in çarpımının tersini alır
    self.w = np.matmul(np.matmul(inv, self.X.T), self.Y)  # Normal denklemi kullanarak theta'yı hesaplar
    y_pred = np.matmul(X_test, self.w)  # Tahminleri yapar

    return y_pred, (abs(Y_test - y_pred) / Y_test) * 100  # Tahminleri ve yüzde hata değerlerini döndürür
```

✓ **R-kare Skorunun Hesaplanması (calculateR2 Fonksiyonu):**

- Gerçek ve tahmin edilen değerler arasındaki ilişkiyi görselleştirir ve R-kare skorunu hesaplar.

```
def calculateR2(self, Y_true, Y_pred):
    # R-kare skorunu hesaplar ve görselleştirir
    r2 = r2_score(Y_true, Y_pred)
    plt.scatter(Y_true, Y_pred, color='blue', label=f'R2 Skoru: {r2:.2f}')
    plt.plot([min(Y_true), max(Y_true)], [min(Y_true), max(Y_true)], linestyle='--', color='red', label='Mükemmel Uyum')
    plt.xlabel('Gerçek Değerler')
    plt.ylabel('Tahmin Edilen Değerler')
    plt.title('Gerçek vs Tahmin Edilen Değerler')
    plt.legend()
    plt.show()
    return r2
```

3. METODOLOJİ

✓ **Veri Seti Keşfi ve İndirme:**

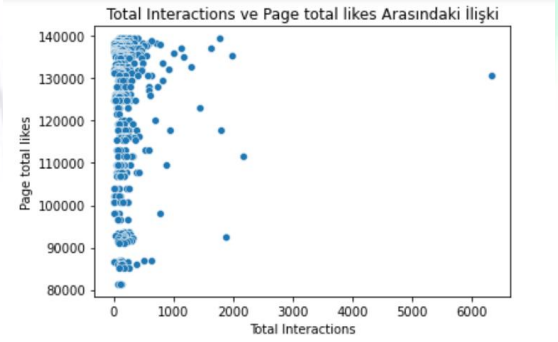
- Facebook gönderi metrikleri veri setinin indirilmesi.
- Keşifsel veri analizi ile veri setinin yapısının incelenmesi.

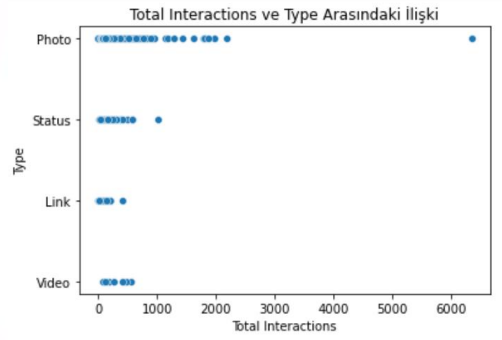
Bağımlı değişkenimiz yani **'Total Interactions'** ile diğer sütunlar arasındaki ilişkiyi görselleştirerek gözlemliyoruz. Amacımız bağımlı ve bağımsız değişkenler arasındaki farkı(yani lineer bir tablo arıyoruz) minimalize ederek bir model oluşturmak. Oluşturduğumuz tabloları inceliyoruz.

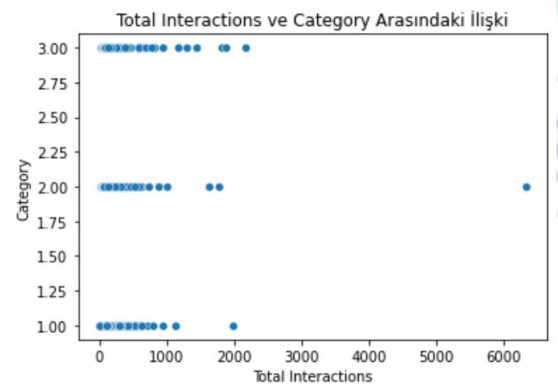
Yazılan kod :

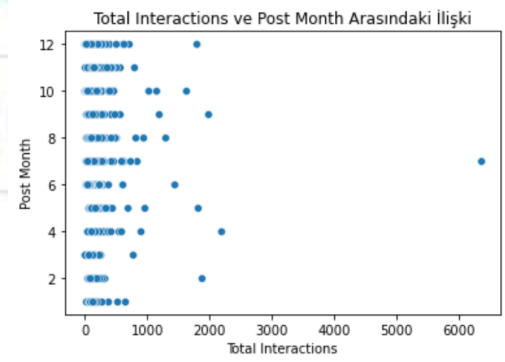
```
# 'Total Interactions' ile tüm özelliklerin scatter plot ile görselleştirilmesi
for column in data.columns:
    if column != 'Total Interactions':  # 'Total Interactions' özelliğini diğerleriyle karşılaştırma
        sns.scatterplot(x='Total Interactions', y=column, data=data)
        plt.title(f'Total Interactions ve {column} Arasındaki İlişki')
        plt.xlabel('Total Interactions')
        plt.ylabel(column)
        plt.show()
```

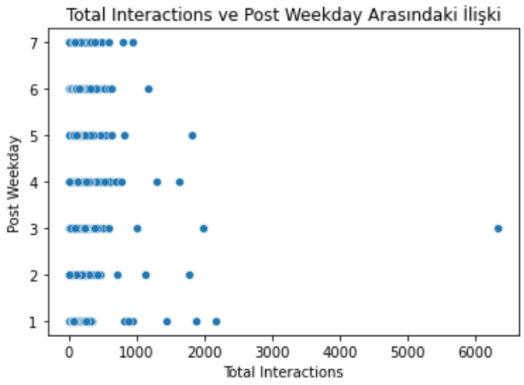
Çıktısı:

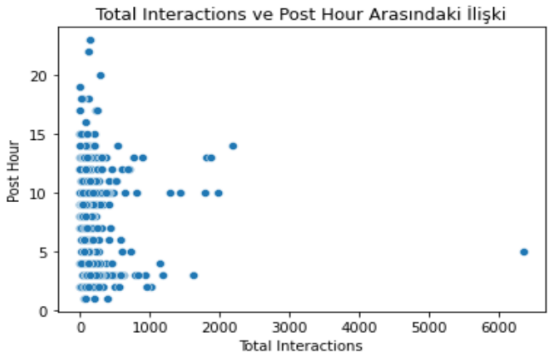


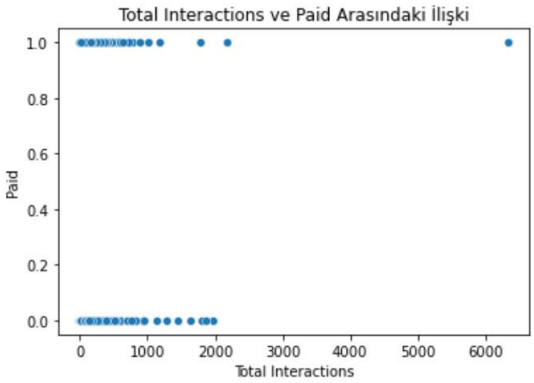


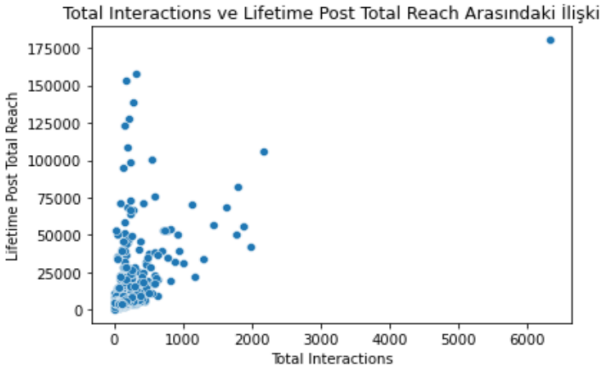


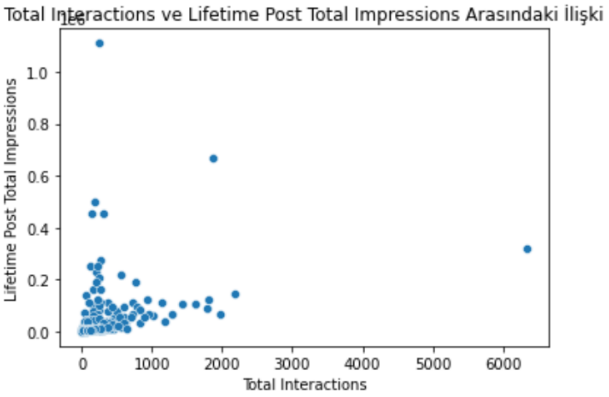


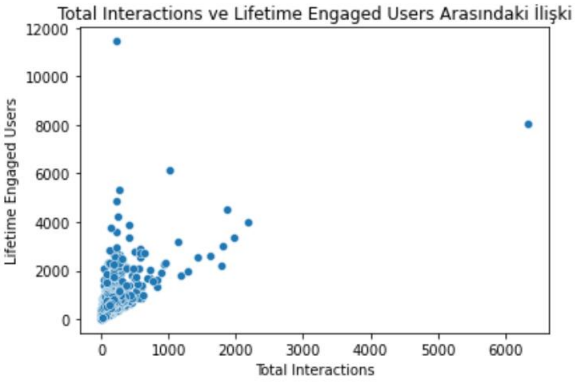

Total Interactions ve Post Weekday Arasındaki İlişki
Post Weekday
Total Interactions

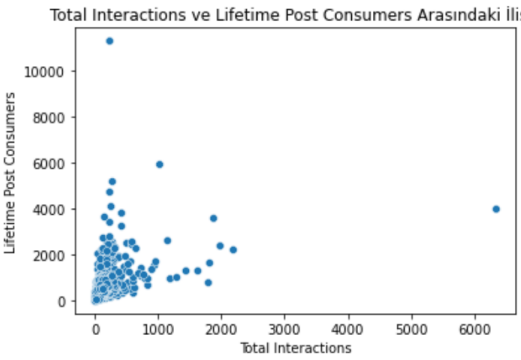

Total Interactions ve Post Hour Arasındaki İlişki
Post Hour
Total Interactions

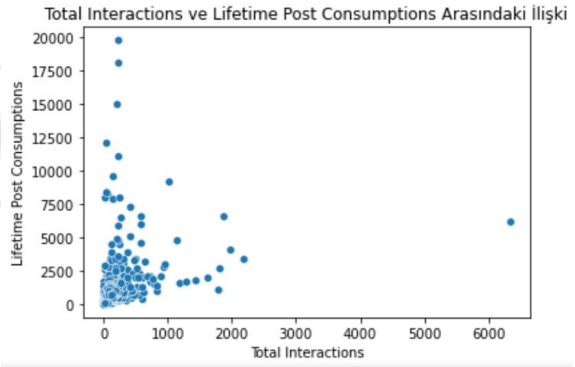

Total Interactions ve Paid Arasındaki İlişki
Paid
Total Interactions

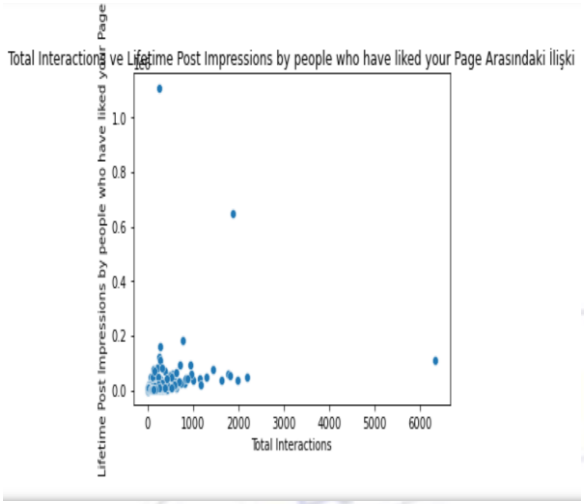

Total Interactions ve Lifetime Post Total Reach Arasındaki İlişki
Lifetime Post Total Reach
Total Interactions

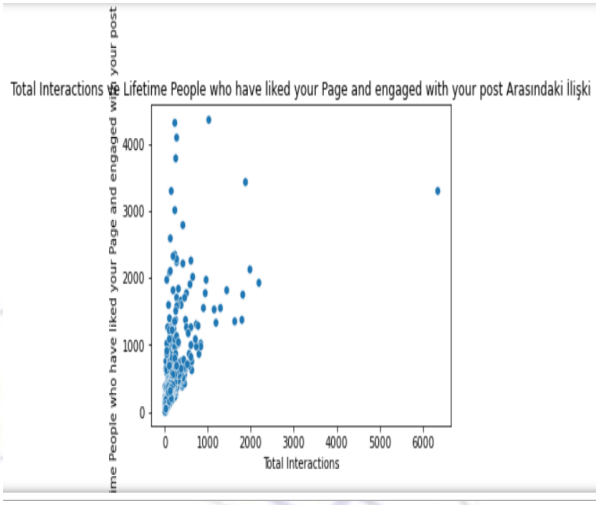

Total Interactions ve Lifetime Post Total Impressions Arasındaki İlişki
1e6
Lifetime Post Total Impressions
Total Interactions


Total Interactions ve Lifetime Engaged Users Arasındaki İlişki
Lifetime Engaged Users
Total Interactions

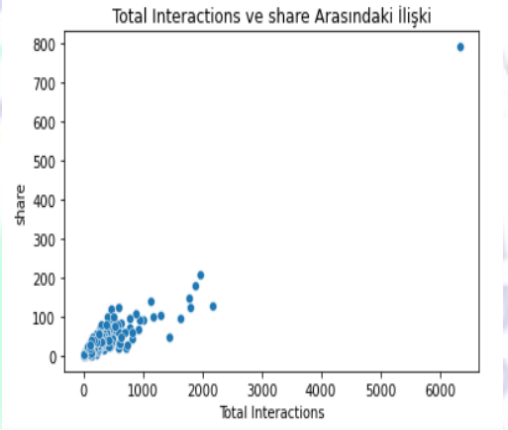

Total Interactions ve Lifetime Post Consumers Arasındaki İlişki
Lifetime Post Consumers
Total Interactions

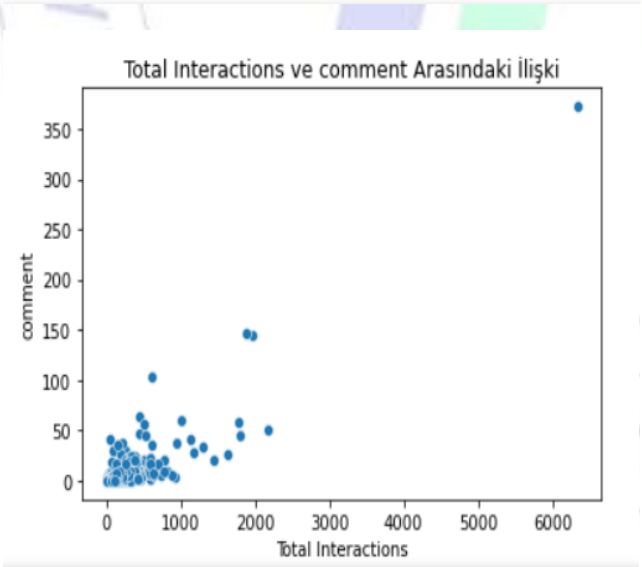

Total Interactions ve Lifetime Post Consumptions Arasındaki İlişki
Lifetime Post Consumptions
Total Interactions

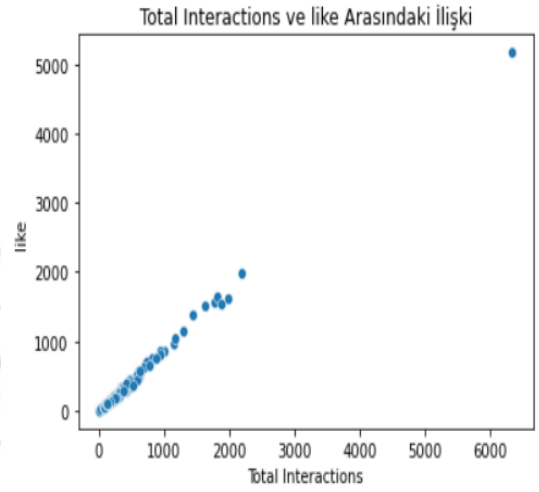

Total Interactions ve Lifetime Post Impressions by people who have liked your Page Arasındaki İlişki
Lifetime Post Impressions by people who have liked your Page
Total Interactions


Total Interactions ve Lifetime People who have liked your Page and engaged with your post Arasındaki İlişki
Total Interactions


Total Interactions ve Lifetime Post reach by people who like your Page Arasındaki İlişki
Lifetime Post reach by people who like your Page
Total Interactions


Total Interactions ve share Arasındaki İlişki
share
Total Interactions


Total Interactions ve comment Arasındaki İlişki
comment
Total Interactions


Total Interactions ve like Arasındaki İlişki
like
Total Interactions

Bağımlı değişkenimiz yani **'Total Interactions'** ile diğer sütunlar arasındaki ilişkiyi bulabilmek için her sütun özelinde bir Lineer Resgesyon kuruyoruz. Çıkan sonuçlar doğrultusunda özniteliklerimizi seçeceğiz.

Yazılan kod:

```
# Kategorik sütunları one-hot encoding ile dönüştürme
X_encoded = pd.get_dummies(data.drop('Total Interactions', axis=1))

# 'Total Interactions'ı hedef değişken olarak belirleme
y = data['Total Interactions']

# Her bir sütun için ayrı bir regresyon modeli oluşturma
for column in X_encoded.columns:
    X = X_encoded[[column]]  # Her sütunu bağımsız değişken olarak seçme

    # Lineer Regresyon modelini oluşturuyoruz ve eğitiyoruz
    model = LinearRegression()
    model.fit(X, y)

    # Model katsayılarını yazdırma
    print(f"Column: {column}")
    print(f"Coefficient: {model.coef_}")
    print("----------------------")
```

Çıktısı:

Column: Page total likes

Coefficient: [0.00105686]

Column: Category

Coefficient: [56.77012128]

Column: Post Month

Coefficient: [2.11064711]

Column: Post Weekday

Coefficient: [-15.17579881]

Column: Post Hour

Coefficient: [-2.38663474]

Column: Paid

Coefficient: [91.37743805]

Column: Lifetime Post Total Reach

Coefficient: [0.00900552]

Column: Lifetime Post Total Impressions

Coefficient: [0.00169988]

Column: Lifetime Engaged Users

Coefficient: [0.22086325]

Column: Lifetime Post Consumers

Coefficient: [0.15273933]

Column: Lifetime Post Consumptions

Coefficient: [0.04524993]

Column: Lifetime Post Impressions by people who have liked your Page

Coefficient: [0.00159037]

Column: Lifetime Post reach by people who like your Page

Coefficient: [0.03058535]

Column: Lifetime People who have liked your Page and engaged with your post

Coefficient: [0.30566945]

Column: comment

Coefficient: [15.52720408]

Column: like

Coefficient: [1.17385292]

Column: share

Coefficient: [8.30898855]

Column: Type_Link

Coefficient: [-128.73906428]

Column: Type_Photo

Coefficient: [30.13386626]

Column: Type_Status

Coefficient: [5.41147741]

Column: Type_Video

Coefficient: [84.92610837]

✓ **Veri Ön İşleme:**

- Öznitelik seçimi: Model için uygun özniteliklerin belirlenmesi.

  Bu bölümü **Veri Seti Keşfi** başlığı altında inceledik ve kullanmaya karar verdiğimiz değişkenlerimiz:

- Eksik veri kontrolleri ve düzenlemeleri.

  Eksik verileri ortalama değerlerle doldurduk:

```
#verileri kontrol edilip boşlukları ortalama değer ile doldurma kısmı
print(data.isnull().sum())
numeric_columns = data.select_dtypes(include='number').columns.tolist()
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())
```

✓ **Veri Bölme:**

- Veri setinin eğitim ve test setlerine %80 eğitim, %20 test olacak şekilde böldük.

```
#veri setini %80 eğitim ve %20 test verisi olacak şekilde böldük/ayırdık
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=39)
```

✓ **Model Eğitimi:**

- Ağırlık vektörünün tanımladık.
- Gradyan İnişi kullanılarak modelin eğitimini tamamladık.

✓ **Performans Değerlendirmesi:**

- Toplam Kare Hatası (SSE), Ortalama Mutlak Hata (MAE), ve Ortalama Kare Hatası (MSE) gibi metriklerle modelin performansının değerlendirilmesini yaptık.

✓ **Model İyileştirme:**

- Lineer regresyon fonksiyonunda özniteliklerin değiştirilmesi ve modelin iyileştirilmesi.
- Yeni özelliklerin veya değişikliklerin model performansına etkisinin değerlendirilmesi.

✓ **Sonuçların Analizi:**

- Grafikler ve istatistiklerle elde edilen sonuçların detaylı bir şekilde analizlerini **Sonuçlar** başlığı altında inceledik.

4. Bulgular ve Tartışma

İki farklı lineer regresyon modeli incelendiğinde, elde edilen sonuçlar arasında belirgin farklar görülmektedir. İlk model, "Category", "Page total likes", "Post Month", "Post Hour", "Post Weekday", "Paid" ve "Combined_Column" gibi çeşitli öznitelikleri kullanıldı ama tahmin gücü oldukça düşüktü. Bu durumun birkaç nedeni olabilir. İlk olarak, bu modelde kullanılan özniteliklerin, toplam etkileşim sayısını doğrudan etkileyen özellikler olmaması mümkündür. Ayrıca, belki de modelin karmaşıklığı yetersizdi ve veri setindeki doğrusal olmayan ilişkileri yeterince açıklayamadı. Farklı modeller ile aynı öznitelikler üzerinde işlemler yaptığımızda doğruluğun ideal seviyelerde bulunduğunu gözlemledik.

(Random Forest Regressor - R-kare Değeri: 0.854601134021153)

(Gradient Boosting Regressor - R-kare Değeri: 0.5543556583227105)

Sonuç olarak birinci seçilen öznitelikler ile doğru sonuçlar verebilmesi için modelimizin karmaşıklığının arttırılması gerektiği noktasında hemfikir olduk.

Diğer yandan, ikinci modeldeki öznitelikler "Page total likes", "Lifetime Post reach by people who like your Page", "like" ve "Lifetime People who have liked your Page and engaged with your post" olarak belirlendi. Bu öznitelikler, modelin tahmin gücünü oldukça artırdı. Özellikle, bu özniteliklerin, toplam etkileşim sayısını daha iyi tahmin etmeye yönelik olduğu görülüyor. Örneğin, "like" veya "Page total likes" gibi özniteliklerin, toplam etkileşim sayısını doğrudan etkileme olasılığı yüksek.

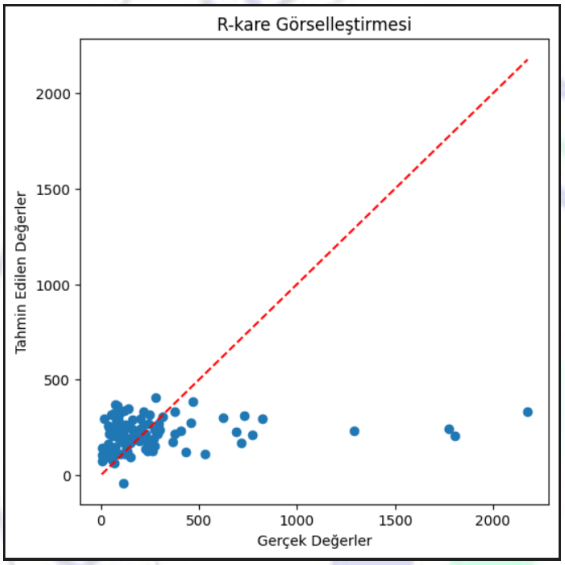
Ağırlık vektörlerine bakıldığında, ilk modeldeki özniteliklerin etkisi oldukça düşüktü. Bu durum, bu özniteliklerin modelin tahmin yeteneğine sınırlı bir katkıda bulunduğunu gösteriyor. Diğer yandan, ikinci modeldeki özniteliklerin ağırlık matrisi daha büyük değerlere sahipti, bu da bu özniteliklerin modelin tahmin gücüne daha fazla katkı sağladığını gösteriyor. İlk modelin ağırlık vektörü [35.806, 67.151, -47.379, -21.654, -24.476, 36.258] şeklindeyken, ikinci modelin ağırlık vektörü [4.610, 32.134, 318.887, 17.725] olarak hesaplandı.

Sonuç olarak, veri setindeki özniteliklerin seçimi ve modelin karmaşıklığı, modellerin performansını belirleyen kritik faktörler olarak ortaya çıkıyor. İlk modelde kullanılan öznitelikler, tahmin yeteneğini sınırlarken, ikinci modeldeki öznitelikler daha doğru tahminler yapmaya yardımcı oldu. Bu analiz, doğru öznitelik seçiminin ve modelin uygun karmaşıklığının, lineer regresyon modellerinin tahmin gücünü önemli ölçüde etkilediğini gösteriyor.

5. Sonuçlar

✓ Model Performansı Üzerine Genel Bir Bakış:

Sonuç olarak yukardaki başlıklarda da bahsedildiği gibi, iki farklı lineer regresyon modeli eğitimi gerçekleştirildi. Her iki modelde de farklı öznitelik setleri kullanılarak ayrı eğitimler yapıldı. İlk modelde "Category", "Page total likes", "Post Month", "Post Hour", "Post Weekday", "Paid" ve "Combined_Column" sütunları kullanılırken, ikinci modelde ise öznitelikler "Page total likes", "Lifetime Post reach by people who like your Page", "like" ve "Lifetime People who have liked your Page and engaged with your post" sütunları tercih edildi. Bu farklı öznitelik seçimleri, her bir modelin genel performansını etkilemektedir.



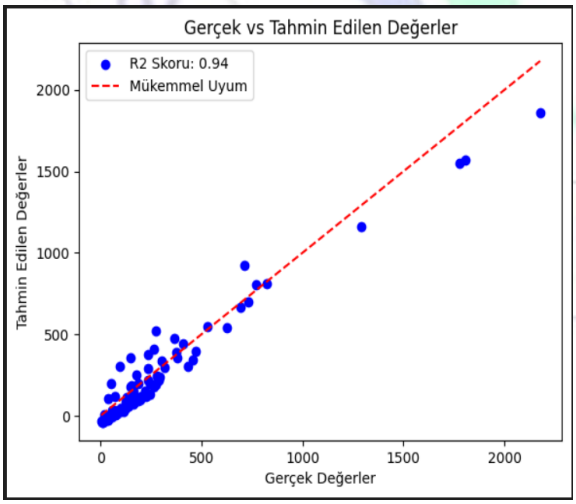
MODEL 1 R2,MSE ve MEA değerleri

Modelin performansını değerlendirmek adına elde ettiğimiz R-kare, MSE ve MAE değerlerine odaklanıldığında, modelin yukarda belirtilen özniteliklerle ile iyi bir uyum sağladığı söylenemez.

```
R-squared (R2) Score: 0.2669055358621818
```

```
Eğitim Seti MSE Değeri: 141562.2663911306
Test Seti MSE Değeri: 126116.26274533392
```

```
Eğitim Seti MAE Değeri:152.55195951631106
Test Seti MAE Değeri:189.8391811126613
```



MODEL 2 R2,MSE ve MEA değerleri

İkinci modelde kullanılan öznitelikler ile elde edilen R-kare, MSE ve MAE değerleri aşağıda belirtilmiştir.

```
R-squared (R2) Score: 0.9351769732331993
```

```
Eğitim Seti MAE Değeri: 200.57172944850092
Test Seti MAE Değeri: 255.4216409678172
```

```
Eğitim Seti MSE Değeri: 649.2459265398985
Test Seti MSE Değeri: 1339.588320234465
```

Bu durum,ikinci öznitelik seti ile eğitilen modelin, veri setinin özelliklerine daha farklı bir şekilde tepki verdiğini ve modelin performansının ilk öznitelik seti ile eğitilen modele kıyasla **daha iyi bir sonuç verdiğini göstermektedir.**

- ✓ **Önemli Bulguların Vurgulanması**:
  Bu çalışmada, iki farklı öznitelik seti üzerinde eğitilen lineer regresyon modellerinin performansını değerlendirdik. Elde ettiğimiz önemli bulgular arasında, belirli özelliklerin tahmin üzerindeki belirleyici etkisi ve ağırlık değerlerinin bu özelliklere olan duyarlılığı bulunmaktadır. Özellikle 'like' ve 'Lifetime Post reach by people who like your Page' gibi öznitelikler, modellerin başarısında önemli bir rol oynamaktadır.Ayrıca, modelin belirli bir yönüne odaklanarak elde ettiği başarılar, uygulama alanlarını belirleme konusunda bize değerli ipuçları sunmaktadır.Bu bulgular, modelin güçlü ve zayıf yönlerini anlamamıza yardımcı olarak, modelin uygulama bağlamında daha etkili olmasını sağlamak adına değerli bir yol haritası sunmaktadır.

- ✓ **Çalışmanızın Katkısı**:
  Farklı öznitelik setleriyle eğitilen lineer regresyon modelleri üzerinde yürütülen analizlerle elde edilen sonuçlar sayesinde birçok yandan katkı sunmaktadır.Aynı zamanda, farklı öznitelik setlerinin performans karşılaştırmaları, öznitelik seçiminin model performansı üzerindeki etkisini anlamak adına önemli bir rehberlik sunmaktadır Lineer regresyon modellerinin belirli uygulama bağlamlarında nasıl kullanılabileceği ve geliştirilebileceği konusunda değerli bir temel oluşturmaktadır, bu da gelecekteki çalışmalar ve uygulamalar için önemli bir referans noktası olarak kullanılabilir.

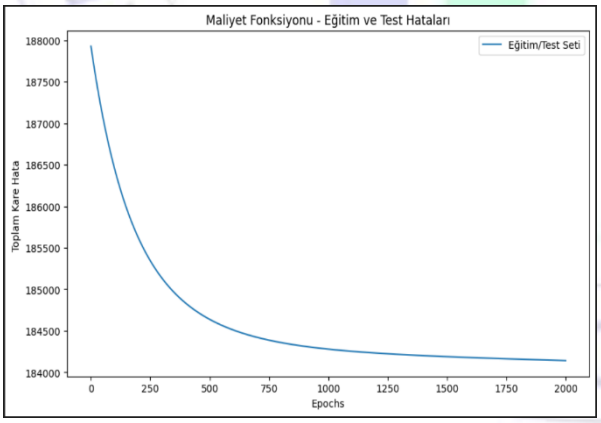
- ✓ **Çalışmanızın Sınırlılıkları:**
  Bu projenin bir dizi sınırlılığı bulunmaktadır ve bu sınırlılıkların sonuçlar üzerindeki etkilerini değerlendirmek önemlidir.Öncelikle, kullanılan veri setinin boyutu sınırlıdır ve bu durum, modelin genelleme yeteneği üzerinde etkili olmaktadır.Daha büyük veri setleri kullanılarak elde edilen sonuçlar, modelin daha genel bir bağlamda nasıl performans göstereceği konusunda daha güvenilir bir değerlendirme sağlayabilir. Ayrıca, belirli özelliklerin seçimi konusundaki sınırlılıklar da göz önüne alınmalıdır. Bu çalışmada seçilen öznitelikler, belirli bir bağlamda etkili olabilir ancak farklı bir bağlamda veya uygulama alanında farklı sonuçlar alınabilir.Son olarak, kullanılan modelin karmaşıklığı da bir sınırlılık olabilir. Daha karmaşık modeller kullanılarak elde edilen sonuçlar, aynı zamanda daha fazla hesaplama gücüne ihtiyaç duyabilir. Bu sınırlamaların bilincinde olmak, çalışmanın genel değerlendirmesinde dikkate alınmalıdır.

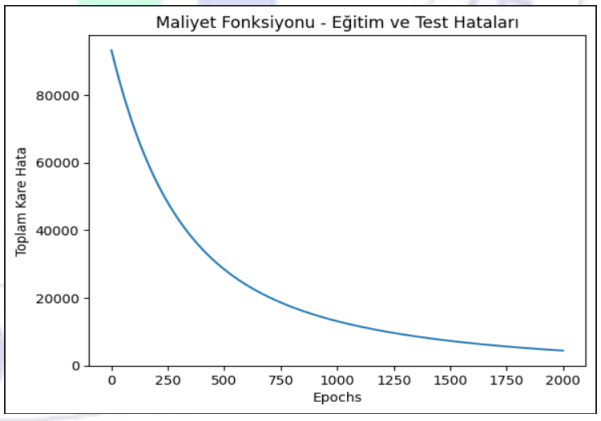
- ✓ **Gelecek İşaretleri ve Öneriler**:
  Çalışmanın devamında, modelin performansını artırmak ve daha geniş bir uygulama alanına yaymak adına bazı öneriler; ilk olarak daha büyük bir veri setinin kullanılması, modelin genelleme yeteneğini iyileştirebilir. Ayrıca, farklı öznitelik setleri ve daha karmaşık model yapılarıyla denemeler yaparak, modelin davranışları üzerindeki etkileri daha iyi anlamak mümkündür. Ayrıca, çeşitli hiperparametre ayarlamalarının ve optimizasyon tekniklerinin kullanılması, modelin daha iyi bir performans sergilemesine olanak tanıyabilir.

- ✓ **Sonuç ve Özet:**

Çalışma sonuçlarına göre, **ikinci model** genel başarıda daha etkili olmuştur, özellikle **'like'** üzerindeki tahmin performansı göze çarpmaktadır. Ancak, **birinci model** için gözlemlenen *overfitting* durumu, modelin belirli koşullarda aşırı uyum sağlamakta zorlandığını göstermiştir. Bu durum için, birinci modelin'in performansını iyileştirmek ve genel uygulanabilirliğini artırmak adına daha fazla inceleme ve düzenleme gerekmektedir.

Ayrıca, birinci modelin test veri setinde beklenen performansı sağlamakta zorlanması, modelin belirli bir epoch'tan sonra(**1000 civarı**) aşırı öğrenmeye eğilimli olduğunu ortaya koymaktadır.Modelin eğitim sürecini daha ayrıntılı bir şekilde değerlendirmek için maliyet fonksiyonlarına odaklanmak bu konu için daha mantıklı olacaktır. Modelin eğitim ve test veri setlerindeki performansını görselleştiren maliyet fonksiyonlarının grafikleri, modelin öğrenme sürecinin hangi aşamalarında iyileştiğini ve zorlandığını özellikle *overfitting* ve *underfitting* durumlarını belirlemede önemli bir araç olacak ve modelin genel performansının geliştirilmesine yönelik stratejilerin belirlenmesine katkı sağlayacaktır.





MODEL 1 **MODEL 2**

Sınırlamalar göz önüne alındığında, modelin genel performansını optimize etmek için özellikle **birinci model** üzerinde daha dikkatli bir ayarlama yapılması ortaya çıkmaktadır. Daha fazla veri kullanımı ve farklı öznitelik setleri ile yapılan denemeler ile modelin genel uygulanabilirliğini artırabilir ve overfitting durumlarını önleyebiliriz.

6. Referanslar

[1] Zhang, S., & Sun, Y. (2013). Automatically synthesizing SQL queries from input-output examples. In 2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings (pp. 224–234). https://doi.org/10.1109/ASE.2013.6693082

[2] Dette, Holger, and Axel Munk. "Validation of Linear Regression Models." *The Annals of Statistics*, vol. 26, no. 2, 1998, pp. 778–800. *JSTOR*, http://www.jstor.org/stable/120052. Accessed 25 Dec. 2023.

[3] Chen, P. P. S. (1976). The Entity-Relationship Model—toward a Unified View of Data. ACM Transactions on Database Systems (TODS), 1(1), 9–36. https://doi.org/10.1145/320434.320440

[4] Poole, M. A., & O'Farrell, P. N. (1971). The Assumptions of the Linear Regression Model. *Transactions of the Institute of British Geographers*, *52*, 145–158. https://doi.org/10.2307/621706

[5] Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., & Radev, D. R. (2018). SyntaxSqlnet: Syntax tree networks for complex and cross-domain text-to-SQL task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018 (pp. 1653–1663). Association for Computational Linguistics. https://doi.org/10.18653/v1/d18-1193

[6] Yuan, Z., & Yang, Y. (2005). Combining Linear Regression Models: When and How? *Journal of the American Statistical Association*, *100*(472), 1202–1214. http://www.jstor.org/stable/27590665

[7] Ritov, Y. "Estimation in a Linear Regression Model with Censored Data." *The Annals of Statistics*, vol. 18, no. 1, 1990, pp. 303–28. *JSTOR*, http://www.jstor.org/stable/2241545. Accessed 25 Dec. 2023.