**Kanada’daki Evlerin Regresyon Modelleri Kullanılarak Tahmin Edilmesi**

**Giriş:**

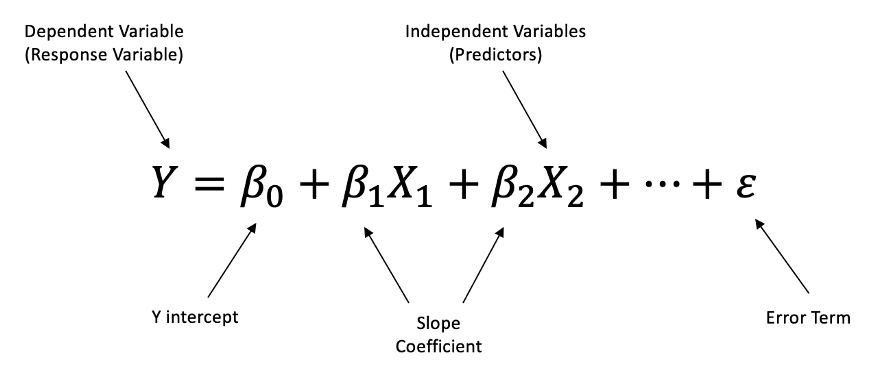
Regresyon nedir? Regresyon, istatistiksel bir analiz yöntemidir. Bir bağımlı değişken ile bir veya daha fazla bağımsız değişken arasındaki ilişkiyi modelleyerek bağımlı değişkeni tahmin etmek için kullanılır. Kesikli veri için uygun değildir, sürekli veri için tercih edilir.

**Modellerin Açıklanması:**

Bu ödevde 6 farklı regresyon modeli kullanılıyor. Sırasıyla bakalım:

**1- Linear Regression**

Ya da Türkçeye geçmiş haliyle Lineer Regresyon, adından da anlaşılabileceği üzere değişkenler arasındaki doğrusal bir ilişkiyi modellemeyi amaçlar. Ödevde kullandığımız veri için oldukça uygundur, zira bir evin fiyatı konumu, yatak odası sayısı gibi etkenlerle doğrusal ilişkilidir genelde. Formülü:



Dependent Variable: Tahmin ettiğimiz değişken.

Y intercept: Regresyon çizgisinin y eksenini kestiği değer (x = 0 değeri).

Independent Variables: Tahmin ederken kullandığımız değişken/ler.

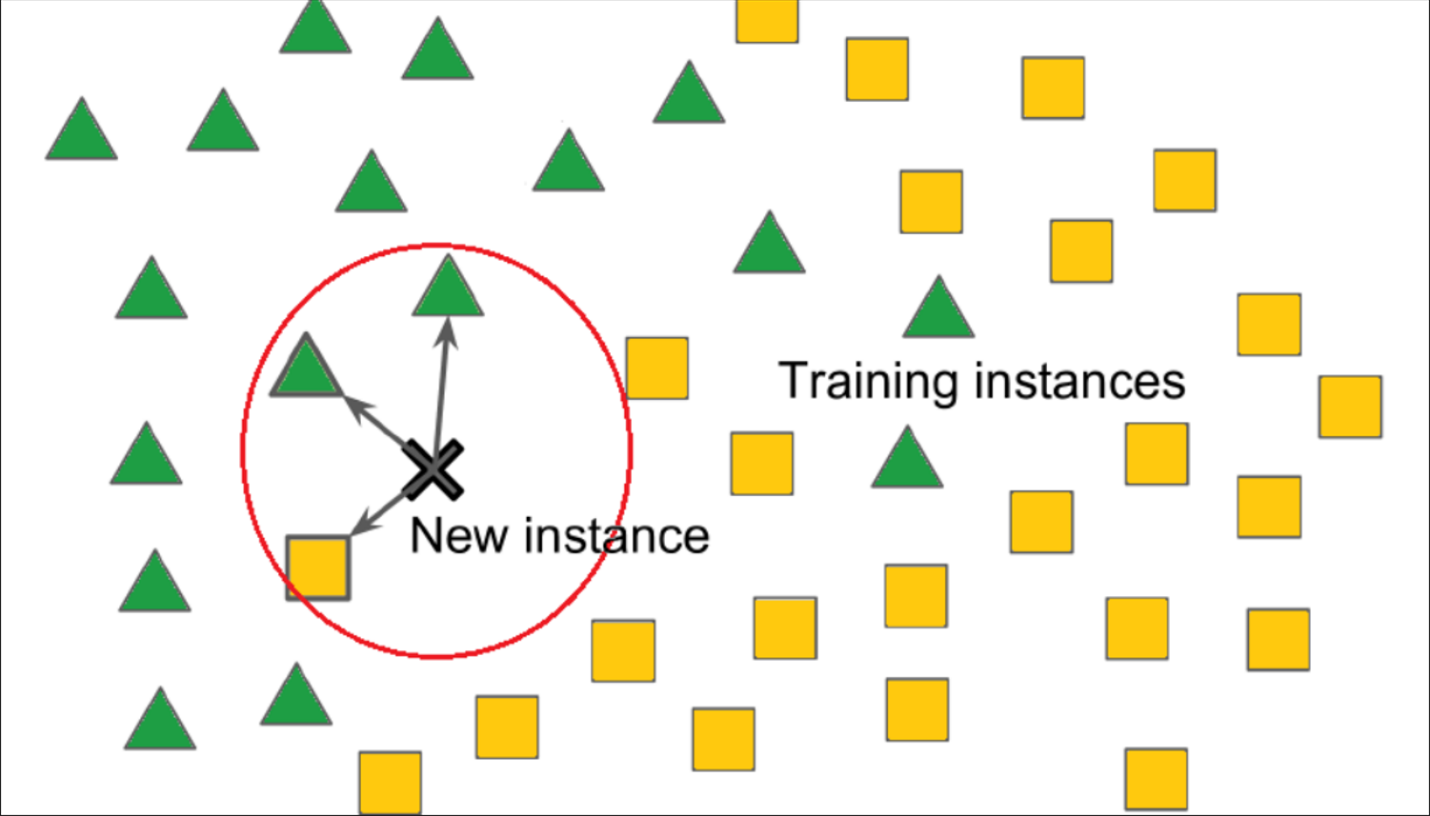
Slope Coefficient: Eğim katsayısı.

Error Term: Hata payı.

**2- kNN Regression**

kNN Regression, tıpkı kNN Classifier gibi k-Nearest Neighbors (k-En Yakın Komşu) mantığında çalışır. kNN Classifier’ın aksine kNN Regression k değerinin tek sayı olması gibi bir gereksinime sahip değildir. Veri setinin düzensiz veya gürültülü (noisy) olduğu durumlarda kullanışlıdır. Bu modelin dezavantajları olarak ise büyük ve çok boyutlu setlerde performans sorunu yaşatması ile uzaklık ölçüsü ve uygun k değeri gibi parametrelerin ayarlanması verilebilir.

kNN mantığını genel olarak tahmin edilecek veri noktasına belirli uzaklık fonksiyonları kullanarak en yakın k sayıdaki komşuyu bulup (regresyon için) bu komşuların bağımlı değişkenlerinin ortalamasını almak oluşturur. Bu sebeple yüksek boyutlu verilerde yavaş kalabilmektedir. kNN mantığı temel olarak şu şekilde gösterilebilir:



**3- Random Forest Regression**

Bu modelden başlayarak kullandığımız regresyon modelleri daha da karmaşık hal alıyor. Bu model, Random Forest algoritmasının regresyon versiyonudur. Bu model, birden çok karar ağacını bir araya getirip bu ağaçların tahminlerini birleştirerek bir regresyon modeli oluşturur.

Öncelikle veri seti rastgele alt kümelere bölünür. Her alt küme üzerinde ayrı bir karar ağacı oluşturulur. Karar ağaçları, veri setini özeliklerine göre böler ve her bölgeye bir regresyon tahmini yapar. Sonrasında bu tahminler birleştirilir, birleştirmek için genelde tahminlerin ortalaması alınır.

Doğası gereği karmaşık ve gürültülü setler için elverişli olabilir. Yüksek boyutlu veri setleri için de iyi bir seçenektir.

**4- Support Vector Regression**

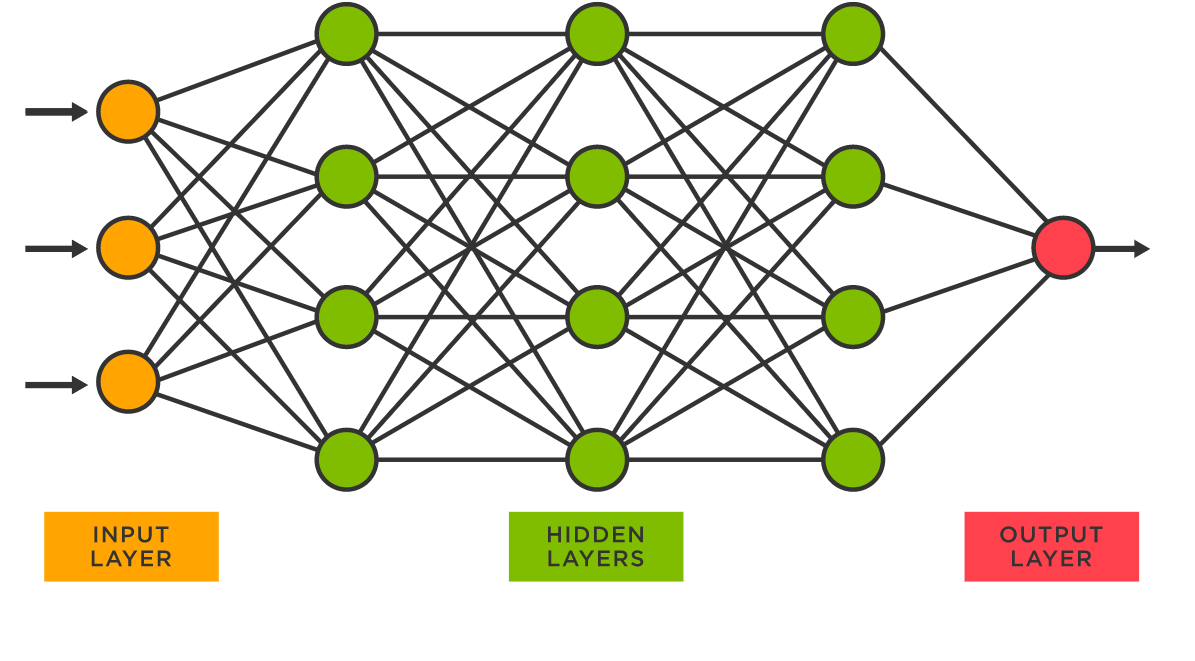
Support Vector Machines (destek vektör makineleri) algoritmalarının regresyon versiyonudur. Özellikle doğrusal olmayan veriler için kullanışlıdır.

Önce eğitim verileri yüksek boyutlu bir uzaya haritalanır. Sonra hedef değişkenin etrafında bir tolerans aralığı belirlenir. Bu aralık içindeki vektörler destek vektör olarak kabul edilir. Bu destek vektörler sonrasında bu uzayda bir “hiper düzlem” oluşturur. Bu düzlem regresyon tahminlerinde kullanılır.

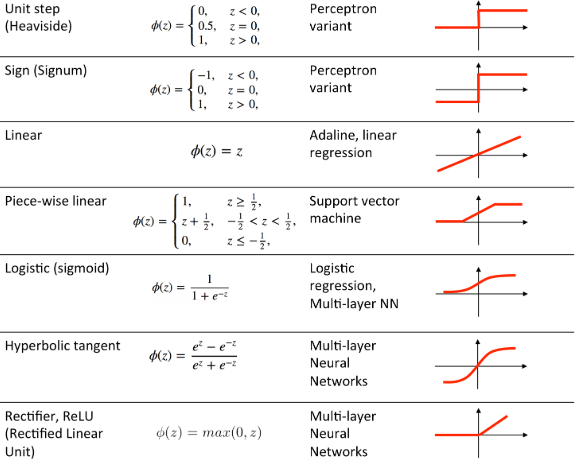
Daha önce açıklananın dışında, küçük ve overfittinge eğimli veriler üzerinde de kullanışlıdır.

**5- Neural Network Regression (MLP Regression)**

MLP (Multi-Layer Perceptron) Regression, yapay sinir ağlarını kullanan regresyon yöntemidir. Yapay sinir ağları, beynimizdeki biyolojik sinir ağlarını (nöronlar) taklit etmeyi amaçlar. En basit MLP yapısında bir adet giriş katmanı (input layer), bir adet gizli katman (hidden layer), bir adet de çıkış katmanı (output layer) bulunur. Karmaşıklaştıkça gizli katman sayısı artabilir. Katmanların boyutu kullanılan veriye göre değişkenlik gösterip her bir katmandaki her bir nöron, kendinden sonraki katmandaki her bir nörona belirli ağırlıklarla bağlıdır.



Gizli katmandaki her bir nöron, kendinden önceki katmandaki nöronlardan gelen ağırlıklı girişleri alır ve bir aktivasyon fonksiyonuna sokar. Böylece sonraki katman için bir çıkış sağlamış olur. Bazı aktivasyon fonksiyonları şu şekilde gösterilebilir:



Çıkış katmanı da regresyonda tahmin edilen değeri sağlar. Bir çıkış nöronu, regresyon probleminde bir tahmin değerini sergiler.

Doğrusal olmayan, çok özellikli ve büyük veri setleri için kullanışlı bir yöntemdir.

**6- Histogram-based Gradient Boosting Regression**

Bu algoritma NaN değerlere karşı kendiliğinden bir mekanizmaya sahiptir. Eğitim sırasında ağacın oluşmasıyla ilgili olan mekanizma NaN değerlerin sağa mı yoksa sola mı gitmesi gerektiğini öğrenir. Düz Gradient Boosting Regression modelinin aksine büyük veriler için (+10.000) daha optimize çalışır.

Öncelikle set üzerinde basit bir karar ağacı oluşturulur. Bu ağaç hedef değişkeni tahmin etmeye çalışır. Sonra ilk ağaç tarafından yapılan tahminlerdeki hatalar hesaplanır ve bu hatalar sonraki ağaçta düzeltilmeye çalışılır. Bu hata tespit ve düzeltme aşamaları belli bir hedefe ulaşıncaya kadar devam eder. Son olarak oluşturulan karar ağaçları birleştirilir.

Büyük, çok boyutlu ve doğrusal olmayan ilişkilere dayalı veri setleri için uygundur. Ayrıca bir yere kadar da aykırı değerlere karşı dayanıklıdır.

**Ön İşleme:**

Ön işleme aşamasında neler yapıldığından sırasıyla bahsedecek olursak:

1- Veride herhangi bir NaN girdinin olup olmadığının kontrol edilmesi. NaN verilere ne yapılacağına karar verilmesi ileriki aşamalarda yapılsa da verinin genel incelenmesinde ilk aşamalarda kontrol edilmesi faydalı olur.

2- Veriyi anlamak için dağılımların incelenmesi. Gözle bakıldığında bile neyin ne olduğu biraz belli olduğu için Address özelliğini ilk olarak herhangi bir grafiğe sokmadan, basit bir yöntemle incelemeye başlıyoruz. Bu özellikteki eşsiz girdilere bakıyoruz çünkü adres, gerçek hayatta eşsiz bir veridir ve bu sette de benzer bir özellik göstermesi beklenmektedir. İncelendiğinde, gerçekten de 31.379 gibi inanılmaz fazla sayıda eşsiz girdi barındırıyor olup en fazla tekrar eden girdi bile aslında boş bir girdidir. Bu yüzden bu özelliğin dağılımını grafikle incelemek imkansızdır.

Geriye 9 adet özellik kaldığı için bu özelliklerin dağılımını 3x3’lük bir gride sokmak görünürlük ve inceleme kolaylığı açısından yararımıza olacaktır. Nitekim böyle yaptığımızda iki özelliğin daha standart bir histogramla pek incelenemeyeceğini fark ediyoruz. Bu iki özellik için eksenleri değiştirilmiş birer countplot hazırlayıp onların da dağılımlarını incelenebilir hale getiriyoruz.

4- Veriyi modellere sokmak için uygun hale getirme aşamasına giriyoruz. Öncelikle Address özelliğini kullanmamız pek mümkün olmadığı için veriden çıkarıyoruz. Sonra City özelliğine target encoding uyguluyoruz. Bu yöntem, özelliklerle hedef değişken arasında kesin bir ilişki olup olmadığı bilinmediğinde faydalı olabiliyor. Temel olarak encode edilecek özelliğin değerlerine göre bir gruplama yapıyor ve bu gruplamayı yaparken hedef değişkene, yapılan gruplandırmaya göre hedef değişkenin ortalamasını giriyor. Böylece bir ilişki varsa da kaçırmamış oluyoruz.

Sonra diğer kategorik değişken City için one-hot encoding uyguluyoruz. Eşsiz girdi sayısı daha az olduğu için verinin boyutu çok artmıyor, veri fazla karmaşıklaşmıyor. Sonrasında bu encoding sonucu oluşan dummy özelliklerin boolean değerlerini integer değerlere çeviriyoruz ki modele sokabilelim.

Sonrasında sıra outlier kontrolüne geliyor. Burada outlierları bulmak ve özelliklere göre mild/extreme olarak sınıflandırılmış bir şekilde çıktı vermek için iki fonksiyon tanımlıyoruz. Sonra kullanım kolaylığı için key değeri özellikler, value değeri ise outlierlar olan bir sözlük oluşturuyoruz. Outlier sayısı fazla olsa da veri de büyük olduğu için başta outlierları veriden atıyoruz, sonraysa outlier oranının biraz yüksek gelmesinden dolayı atmayıp ileride impute edebilmek için NaN değerlere dönüştürdüğümüz ikinci bir veri oluşturuyoruz. Tamamen deneysel amaçlı, ileride bir hata yapmamak için.

Sırada normalizasyon işlemi var. Z-score normalization istendiği için StandardScaler kullanarak iki veriyi de normalize ediyoruz.

Bundan sonra outlierları NaN ile değiştirdiğimiz verideki NaN verileri impute etmek geliyor. Bunun için henüz deneysel bir özellik olan IterativeImputer kullanıyoruz. Bu imputer NaN değerli her bir özelliği diğer özelliklere bağlı bir fonksiyon olarak yorumladığı için işlevsel bir imputing yöntemi. Sonrasında modelleri denemeye geçiyoruz.

**Modellerin Kıyaslanması:**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Mean Absolute  Error (MAE) | R-Squared | Mean Squared  Error (MSE) |
| Multiple Linear Regression | 8.975512421513416e-16 | 1.0 | 1.359978824211623e-30 |
| kNN Regression | 0.11089647283841042 | 0.9714547318108084 | 0.03309635206214318 |
| Random Forest Regression | 0.030071348182072393 | 0.995652593038812 | 0.0050405310747568405 |
| Support Vector Regression | 0.006920656362144246 | 0.9999064478165469 | 0.00010846757435331196 |
| Neural Network Regression | 0.023609831432632875 | 0.9972690925865836 | 0.0031663066748754197 |
| Gradient Regression | 0.023609831432632875 | 0.9972690925865836 | 0.0031663066748754197 |

Outlierları attığımız ve outlierlar yerine başka değerler impute ettiğimiz iki farklı veri olduğu için iki farklı tablo kullanacağız.

*Tablo 1 – Outlierlar atılmış veri için değerler*

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Mean Absolute  Error (MAE) | R-Squared | Mean Squared  Error (MSE) |
| Multiple Linear Regression | 0.0025320423572923623 | 0.9999542955439957 | 5.596807971617335e-05 |
| kNN Regression | 0.10953977129191876 | 0.9745440052722745 | 0.03117252160362328 |
| Random Forest Regression | 0.02928088424781753 | 0.9968747644869095 | 0.003827054200404868 |
| Support Vector Regression | 0.006236226642549993 | 0.9999295758991742 | 8.623889295582004e-05 |
| Neural Network Regression | 0.003979199914160274 | 0.9999035676846014 | 0.00011808764368476651 |
| Gradient Regression | 0.003979199914160274 | 0.9999035676846014 | 0.00011808764368476651 |

*Tablo 2 – Outlierlar impute edilmiş veri için değerler*

**Sonuç:**

İki tablo incelendiğinde de en iyi sonuçları Linear Regression modeli vermiş görünüyor. Bu beklenen bir sonuç çünkü bu verideki özellikler arasındaki ilişki doğrusal veya doğrusala yakın olacaktır. Bu durumda da doğrusal modellerin daha iyi sonuç vermesi beklenir. En kötü sonuçları da iki veride de kNN Regression veriyor. Buna sebep olarak da kNN mantığının doğrusal ilişkilerde pek başarılı olmaması verilebilir. Doğrusal bir ilişki için kNN doğruya en yakın k noktayı inceleyeceği için aslında doğrudan uzak değerleri tahmin edecektir. Bu da doğrusal bir ilişkiyi anlamada dezavantajlı olacaktır.