# Veriyi Tanıma ve Ön İşleme

### a) Veri setini kaynaklardan öğrenerek tanıtınız. Veri setindeki 13 özellik nelerdir (features), kısaca açıklayınız.

CRIM Şehir başına düşen suç oranı

ZN 25,000 üstü ikamet amaçlı arsa oranı

INDUS Şehir başına perakende olmayan iş alanlarının yüzdesi

CHAS Kukla değişken, yol nehri sınırlarsa 1 değilse 0 oluyor.

NOX nitrik oksit konsantrasyonu (10 milyonluk parça başına)

RM Konut başına ortalama oda sayısı

AGE 1940dan önce inşa edilen sahipli inşa oranı

DIS beş Boston istihdam merkezine ağırlıklı mesafeler

RAD radyal otoyollara erişilebilirlik endeksi

TAX 10.000 $ başına tam değerli emlak vergisi oranı

PTRATIO şehir bazında öğrenci-öğretmen oranı

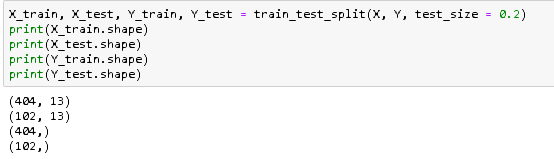
B 1000(Bk - 0.63)^2 formulu Bk şehirdeki siyahilerin şehre oranı

LSTAT Toplumdaki düşük statü oranı

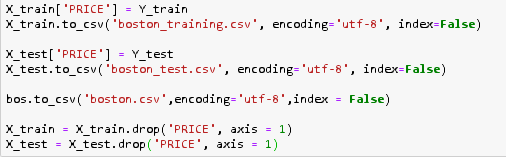
MEDV Sahibi olan evlerin fiyatlarının ortalama $1000lik değeri

### b) Kaç tane eğitim ve kaç tane test verisi programda kullanılıyor?

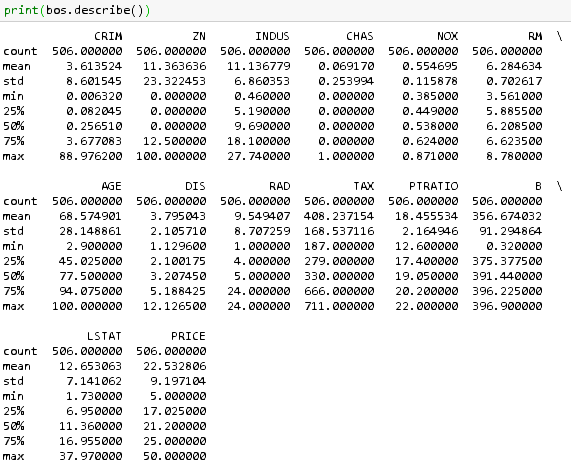
Ödevimizde toplamda 506 adet verimiz var, bunu 404 eğitim, 102 tane ise test olarak ayırıyoruz. Kodumuzdaki X yukarıda bahsettiğimiz özelliklere, Y ise Boston konut fiyatına denk gelmektedir.Modellerimizi X\_train, Y\_train ile eğitip X\_test, Y\_test verilerimizle test etmeliyiz.

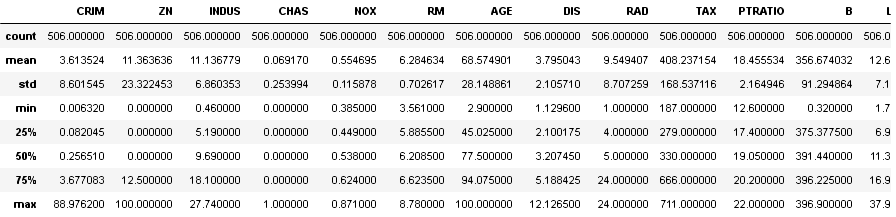


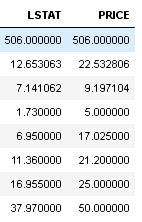
### c) Programdaki eğitim ve test verilerini CVS formatında iki ayrı dosyaya (boston\_training.cvs ve boston\_test.cvs) yazınız. Ayrıca bütün verilerin olduğu boston.cvs dosyasını da üretin.



### d) Verinizi bir makine öğrenme algoritmasına uygulamadan önce verilerinizi biraz tanımak iyi bir fikirdir. Yazacağınız basit bir Python programınya (boston\_pre.py) veri setindeki 13 sütunun istatistiğini alınız. Yani: örnek sayısı (count), ortalama (mean), standart sapma (std), min ve max ve diğerleri (aşağıdaki örneğe bakınız).

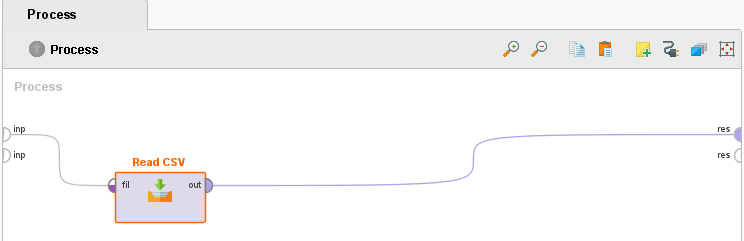




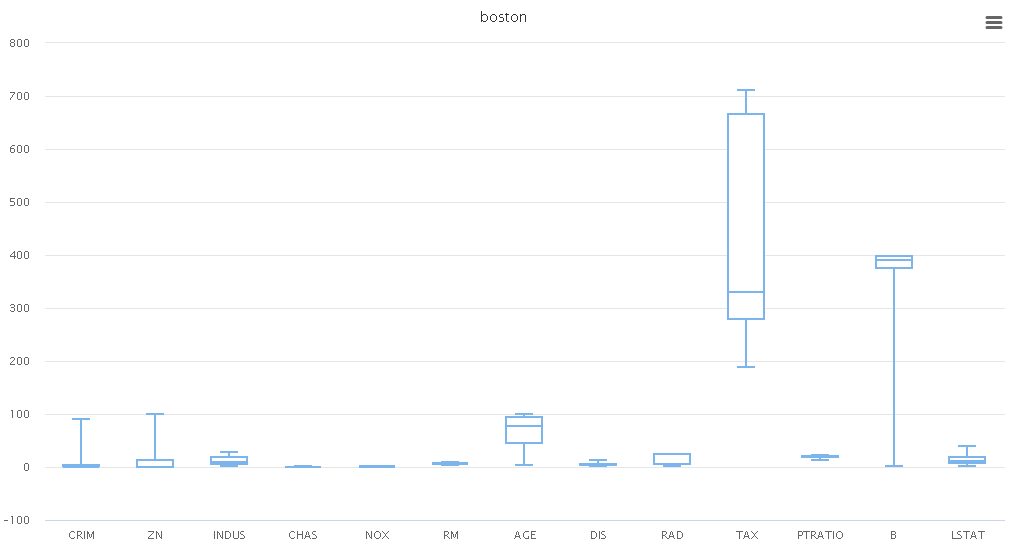


### e) Oluştuduğunuz boston.cvs dosyasını RMS’da okuyacak bir proses diyagramı oluşturun (RMS dosya adları: boston\_pre.rmp ve boston\_pre.properties). Bu proses diyagramı ve RMS çıktısını (Boston Ev Fiyatları tablosu) raporunuz koyunuz. Bu veri setindeki 13 özelliğin (attribute) Boxplot kullanarak grafiğini (chart) oluşturun ve lab raporunuza bu çıktıyı koyun. Grafikteki 13 özellik için 5- sayı özetini (5-number summary) yani min, Q1, median, Q3, max değerlerini raporunuza yazınız.

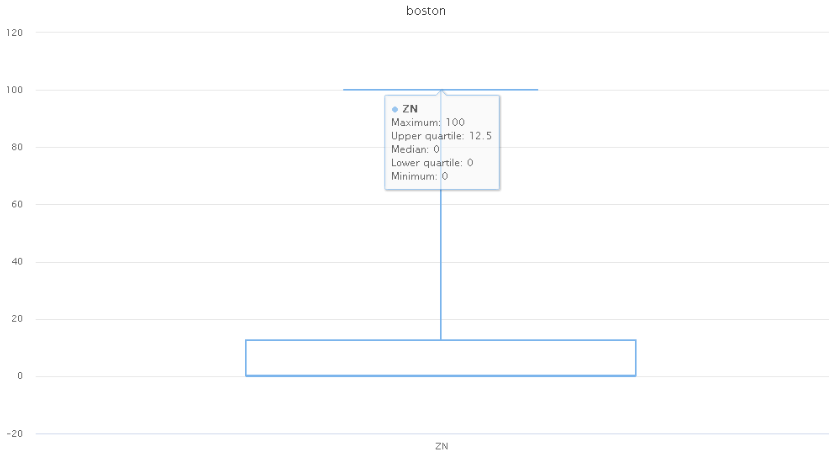
Processimiz :



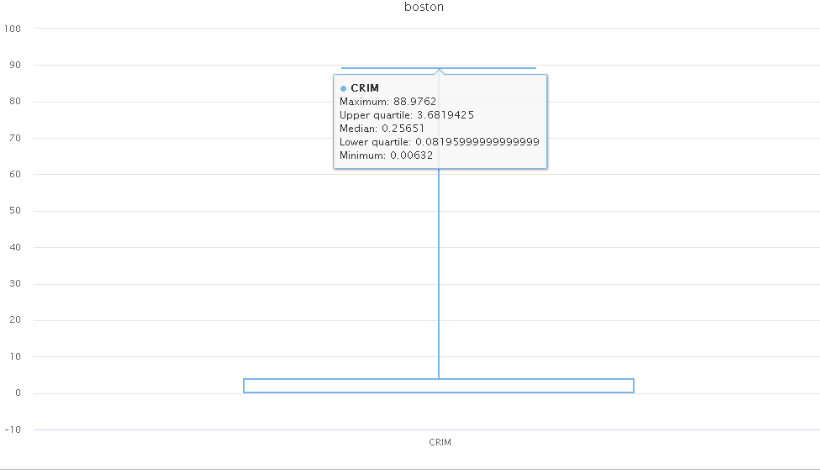
Tüm özelliklerimiz birlikte Boxchartımız…



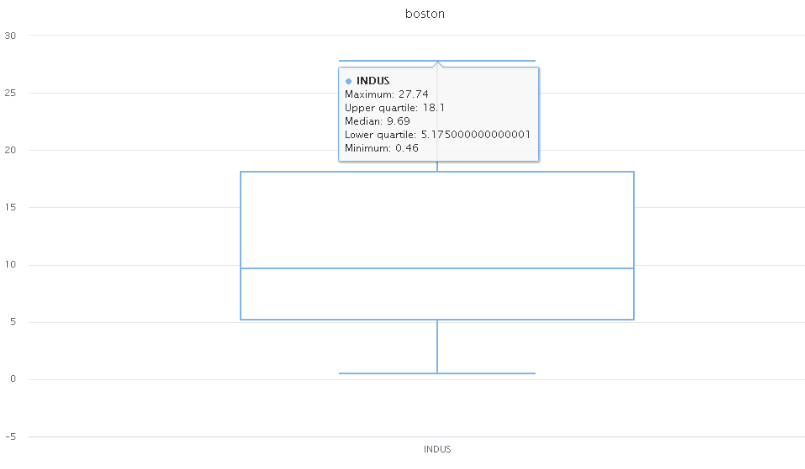
ZN Box Chartımız ve 5-sayı özeti :



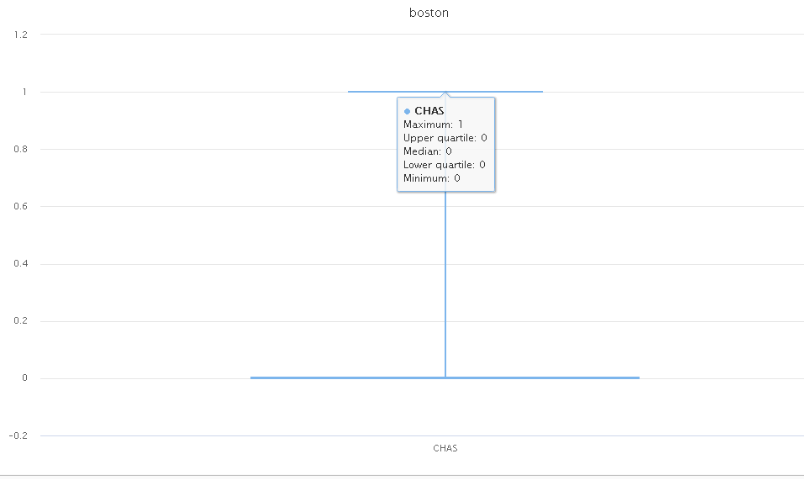
CRIM Box Chartımız ve 5-sayı özeti :



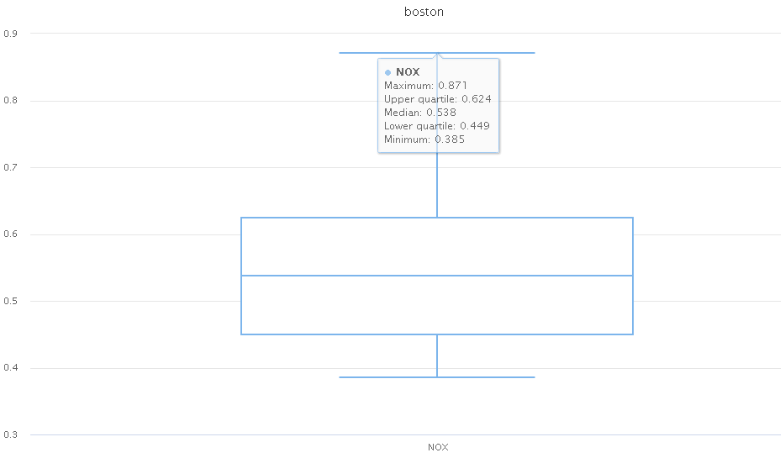
INDUS Box Chartımız ve 5-sayı özeti :



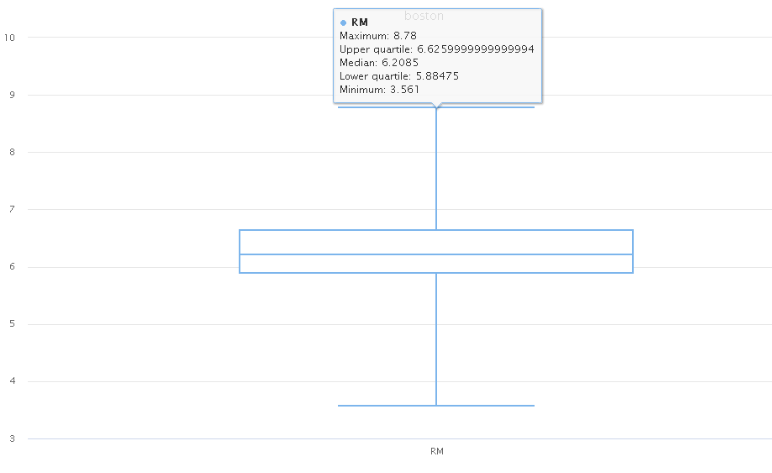
CHAS Box Chartımız ve 5-sayı özeti :



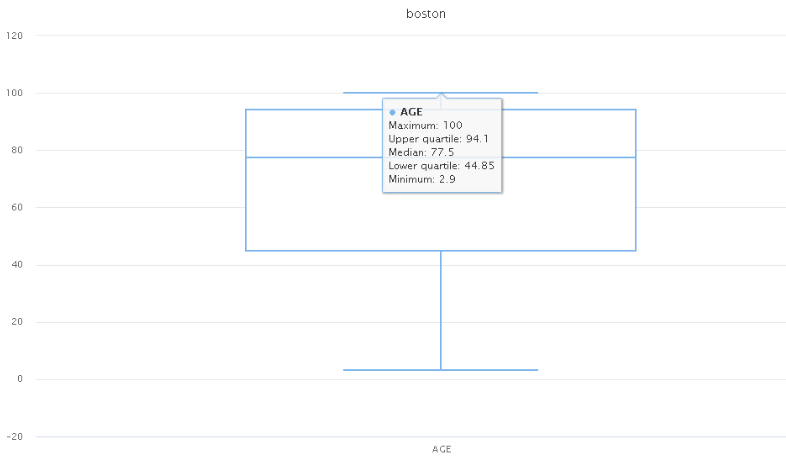
NOX Box Chartımız ve 5-sayı özeti :



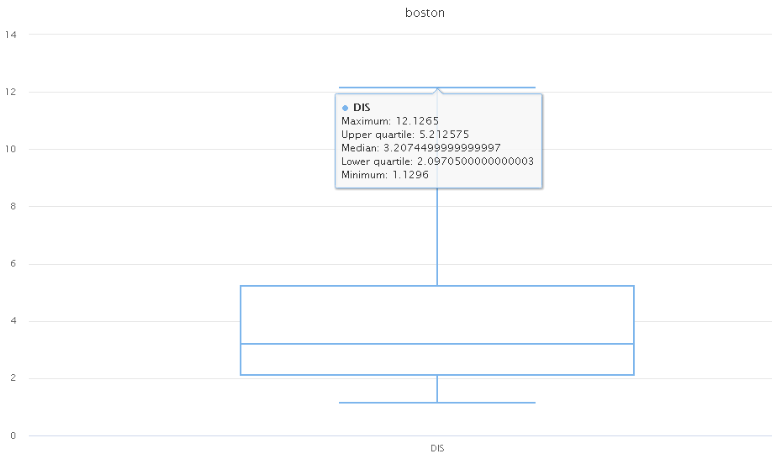
RM Box Chartımız ve 5-sayı özeti :



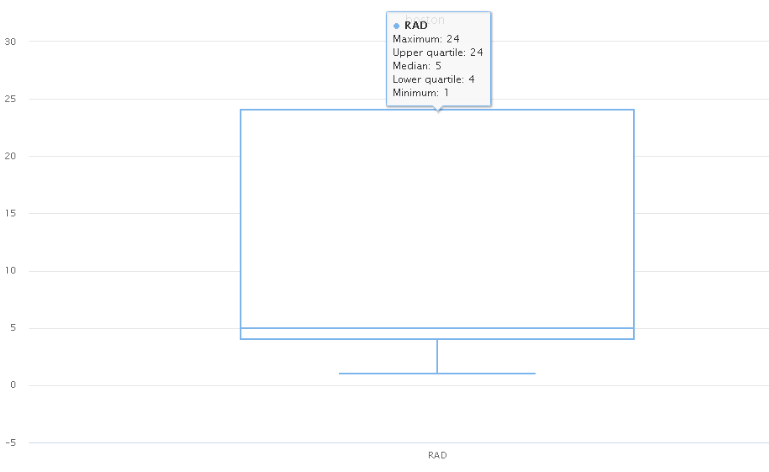
AGE Box Chartımız ve 5-sayı özeti :



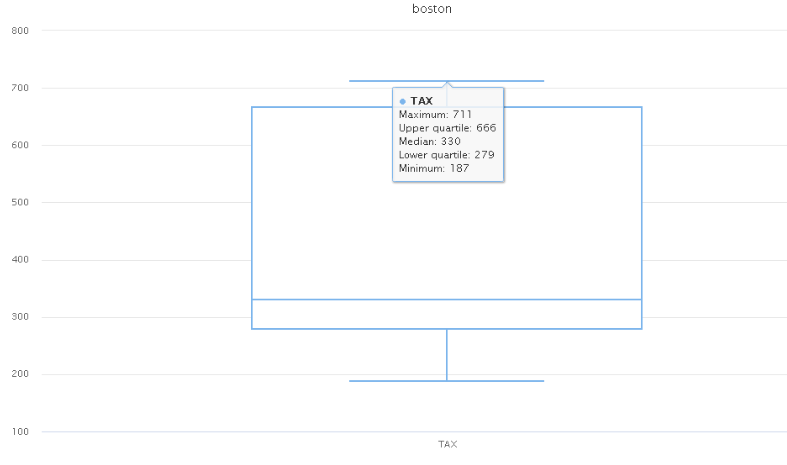
DIS Box Chartımız ve 5-sayı özeti :



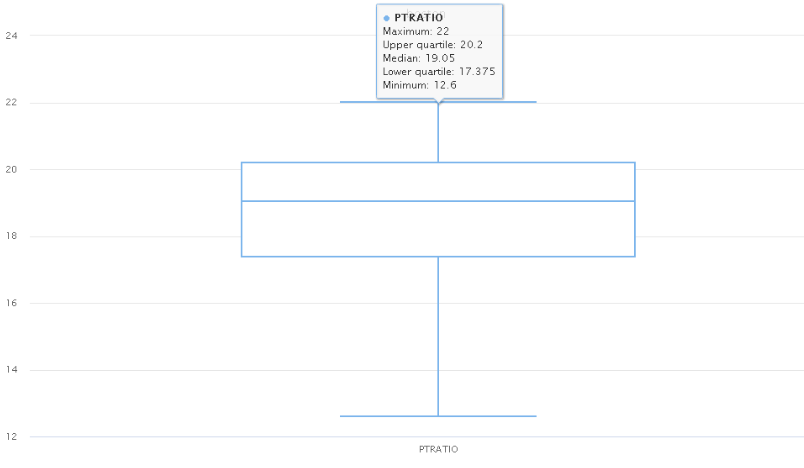
RAD Box Chartımız ve 5-sayı özeti :



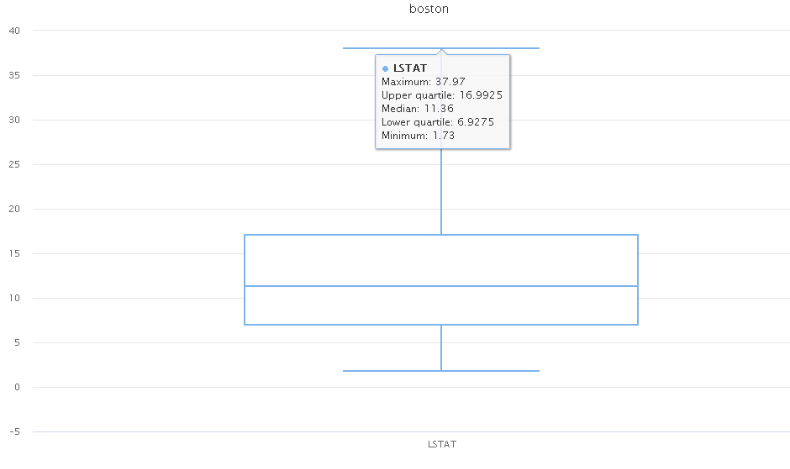
TAX Box Chartımız ve 5-sayı özeti :



PTRATIO Box Chartımız ve 5-sayı özeti :

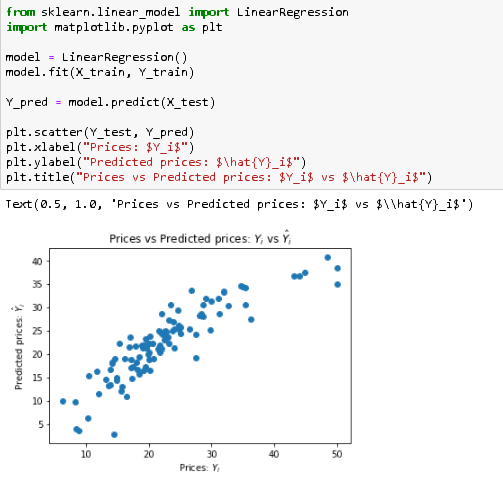


LSTAT Box Chartımız ve 5-sayı özeti :



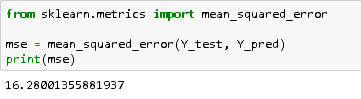
# 2. Linear Regression

### a) LinearRegression()’ı ve %80’lik eğitim verisini (X\_train ve Y\_train) kullanarak, Python ile bir doğrusal (linear) model oluşturun. Daha sonra %20’lik test verisi (X\_test) için ev fiyatlarını (Y\_pred) tahmin edin. Gerçek fiyatları (Y\_test) X ekseninde ve tahmini fiyatları (Y\_pred) Y ekseninde gösteren bir grafik üretiniz ve bu grafiği yorumlayınız. Nasıl bir grafik bulmanız gerekiyordu ve bu grafik ne anlama geliyor?



Burda elde ettiğimi grafikte modelimizin verilen özelliklere göre tahmin ettiği fiyat ile gerçek fiyatını karşılaştırıyoruz.Eğer modelimiz %100 başarı oranına sahip olsaydı grafiğimizin düz bir çizgi olmasını beklerdik fakat gördüğümüz gibi böyle bir durum mümkün değil.Fakat bazı örnekler için modelimiz aynı ya da çok yakın değerler tahmin etmiş, bazı değerlerimiz için de çok farklı tahmin edilmiş.Bu bizim modelimizin eğitilmesiyle , modelimizin başarı oranıyla orantılı. Daha karışık bir model kullansaydık(linear regressiondan farklı) , daha fazla , daha çeşitli eğitim verilerimiz olsaydı daha gerçeğe yakın değerler tahmin edebilirdik.

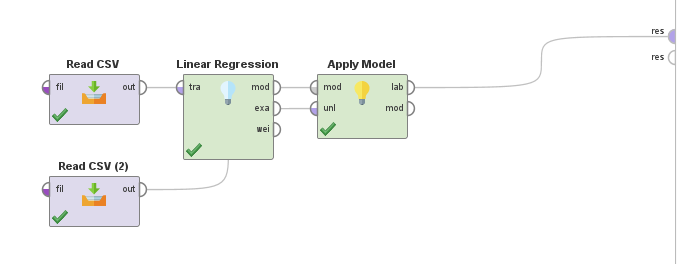
### b) Doğrusal modelinizin hatasını MSE (Mean Squeare Error) yöntemiyle bulan Python satırlarını koda ekleyin. Bulduğunuz MSE nedir? Sonucu yorumlayınız.



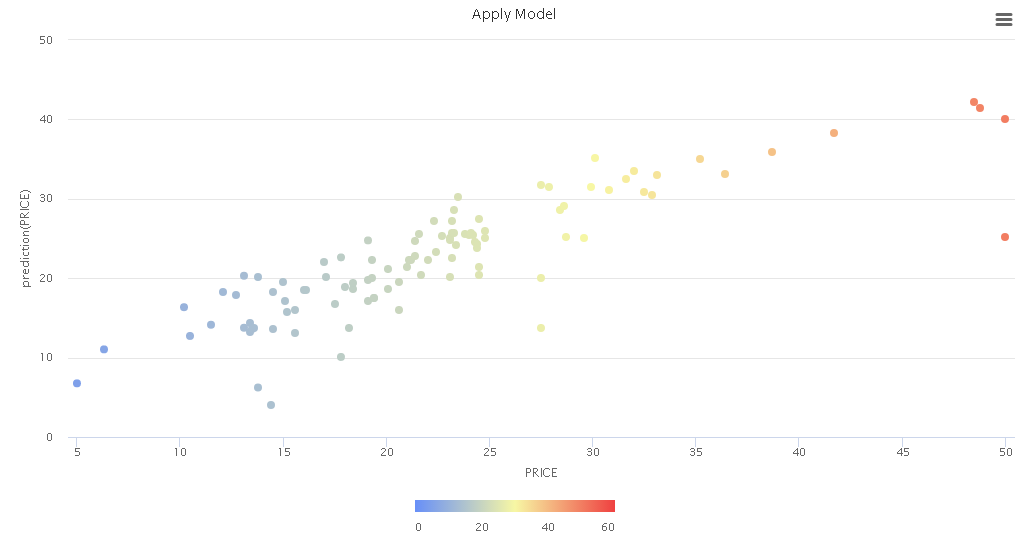
Burda modelimizin hatasını yani MSE hesaplanması beklenmiş.MSE bize gerçek değerlerimiz ile tahmin ettiğimiz değerlerimizin arasındaki farkların toplamının ortalamasını vermektedir.Bu değerin çok olması modelimizin olması gerekenden çok farklı değerler bulduğunu, az olması ise gerçeğe yakın değerler bulduğumuzu gösterir.Bizim bulduğumuz 16.28 değeri yüksek bir değer fakat başlangıç için kabul edilebilir.Bu oranı düşürmek için daha fazla veri ile eğitim yapılabilir, daha karmaşık bir model seçilebilir.

### c) Rapid Miner’da training.cvs ve test.cvs veri setlerini kullanan ev fiyat tahmini yapan Linear Regresyon operatörünün kullanıldığı bir proses diyagramı kurunuz ve raporunuza koyarak kısaca açıklayınız.

Öncelikle RapidMinerda training ve test verilerimizi okuyoruz, bunlardan training verimizi Linear Regression modeline bağlıyoruz, elde ettiğimiz modeli ve eğitimde kullanmadığımız test verilerimizi de Apply Model kısmına bağlıyoruz.RapidMinerdaki Linear Regresyon operatörünün kullanıldığı proses diyagramımız :



Prosesimizden elde ettiğimiz, gerçek fiyatı ve tahmin edilen fiyatı gösteren grafiğimiz :



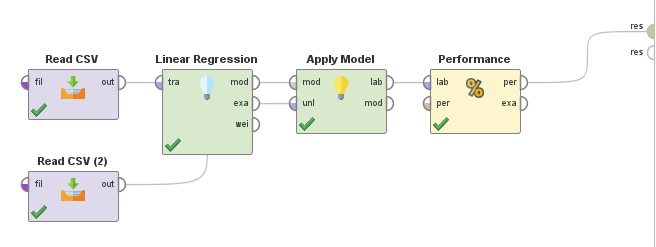
Buna ek olarak performansımızı ölçen Performance kısmını da prosesimize ekleyelim ve ordan aldığımız sonucu da görelim.

### d) Rapid Miner’da elde ettiğimiz model nedir? Kısaca modeli yorumlayınız.

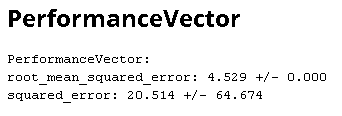
Rapid Miner’da elde ettiğimiz model, elimizde olan training verileriyle eğitilen ve daha önce karşılaşmadığı , verilen 13 özelliğe göre tahmini bir fiyat hesaplaması yapan bir makine öğrenmesi modelidir.İdeal olan bir model düz bir çizgi çıkarması beklenmektedir fakat böyle bir şey mümkün olmadığından modelimiz düz bir çizgi çıkaramamaktadır.

### e) Rapid Miner’da elde ettiğimiz başarı sonucunu MSE (Performace Vector) nedir? Yukarıda (b) şıkkında Python kodu ile elde ettiğiniz MSE ile karşılaştırarak kısaca karşılaştırınız.

B şıkkında Pythonda 16.28 değerini bulmuştuk.Bakalım RapidMinerda nasıl bir sonuçla karşılaşacağız bunun için prosesimize Performance kısmını ekliyoruz.



Performance sonucumuz :



Burdan görüyoruz ki bizim Pythonda yazdığımız Linear Regression ile aralarında çok az fark var fakat Pythonda yazdığımız modelin MSE’si daha az.Bunun sebeplerinden birisi Python da kullanılan modellerin daha bilimsel hesaplamalar kullanılması, hazır kütüphanelerinin daha güçlü olması olabilir.Fakat programlama bilgisi gerektirmektedir, RapidMinerda ise programlama bilgisi olmadan bu modeller kullanılıp test edilebilir.İkisinin de kendilerine özel artıları ve eksileri vardır.

# 3. Sinir Ağı

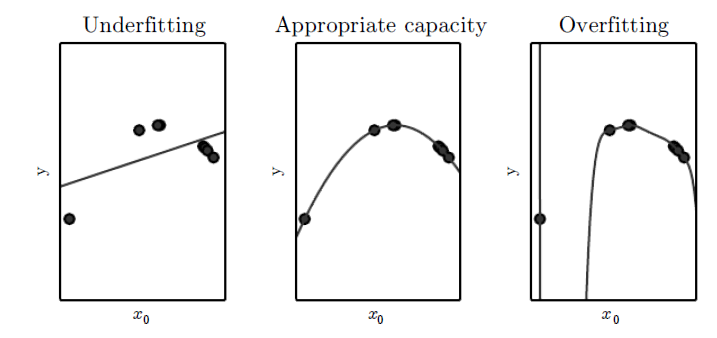
### a) Verilen örnek program, açıklamaları ve ders notlarını kullanarak, bu sinir ağının şu ayrıntılarını anlatınız:

#### a. Gizli katman sayısının işlemlere ve tahminlere olan etkisi, gizli katman sayısı nasıl seçilir?

Gizli katmanlar içlerindeki nöronların ağırlıklarının güncellenmesiyle veri setini bir anlamda öğrenirler.Daha çok sayıda katman daha karışık problemlerin öğrenmesine yardımcı olabilir.Fakat karışık olmayan problemlerde bu sayının fazla olması karşımıza overfitting olarak döner, az olması underfitting olarak döner.Bu yüzden gizli katman sayısının doğru seçilmesi önemlidir.Bunun için kesin bir kural yoktur.Tam olarak doğru gizli katman sayısını seçmek mümkün değildir genel olarak deneme yapılarak, testte elde ettiğimiz başarıya göre tahmini bir sonuça ulaşılabilir(Cross validation gibi)

#### b. Gizli katman sayısı arttırıldıkça ne oluyor? Bu daha iyi tahmin yapmak için iyi midir, yoksa kötü müdür? Tam uygunluk (Overfitting) ile ilişkisi nedir? Raporunuza çıktılar koyarak bu soruları açıklayınız.

Gizli katman sayısının artması, modelimizin başarısını arttırabilir de azaltabilir de, bu bizim problemimizin karmaşasına bağlı bir durum.Çünkü çok fazla hidden layer kullandığımız zaman modelimiz verileri öğrenmektense ezberleyebilir, görmediği veriler hakkında çok doğru tahmin yapamayabilir.Az seçtiğimizde ise modelimizin az eğitilme durumu var bu durumda da modelimizin başarısı düşük olur.Sırasıyla az hidden layer, gerektiği hidden layer ve olması gerekenden fazla hidden layera sahip bir modelin sonuçlarını görebiliyoruz.



#### c. Aktivasyon fonksiyonu nedir ve nasıl seçilir, programda hangisi nasıl kullanılmış?

Aktivasyon fonksiyonu hücreye giren tek ya da birden fazla inputtan bir output elde edilmesine yarar.Bu programda RELU aktivasyon fonksiyonu kullanılmıştır.Genel olarak sigmoid ve tanh fonksiyonları hidden layerlar için uygun değildir, gizli katmanlar için relu tercih etmek mantıklıdır.Çıktı layerında ise 2 den fazla classımız olan bir problemse sigmoid, değilse linear fonksiyonlar kullanılabilir.

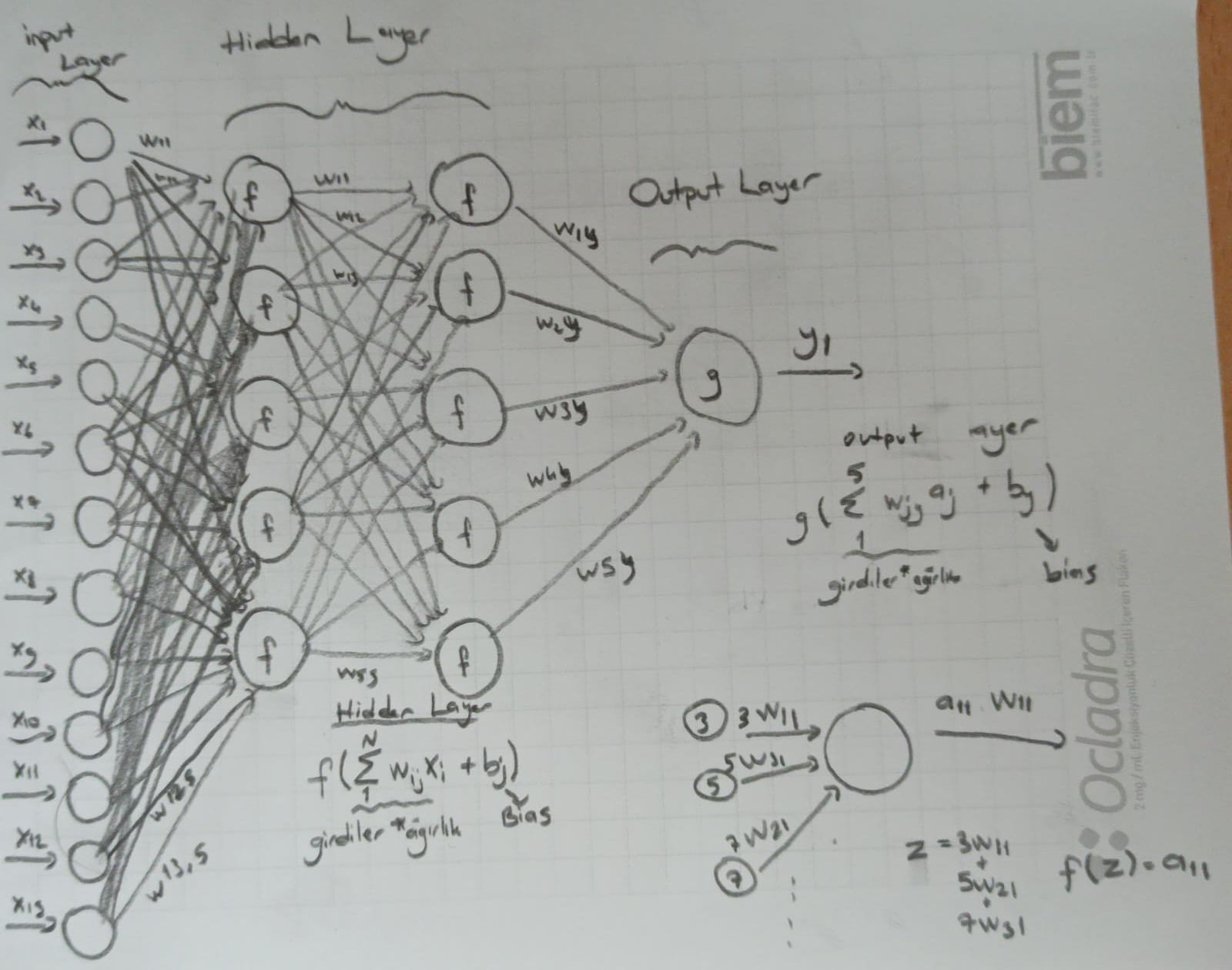
#### d. Programdaki sinir ağının öğrenme parametreleri nelerdir? Bu parametreler nasıl seçilir, ağ nasıl öğrenir, ağ nasıl tahminde bulunur?

Programdaki öğrenme parametreleri hidden layer sayısı, hidden layerlardaki nöron sayısı, aktivasyon fonksiyonu, nöron ağırlıkları, öğrenme oranı, tekrar sayısıdır. Bu parametreleri seçmek için genel bir kural olmasa da bazı tavsiyeler vardır. Problemin karmaşıklığına göre hidden layer sayısı belirlemek lazımdır, layerlardaki nöron sayısı girdi ile çıktı sayısının ortasında olması mantıklıdır, layerlar için doğru aktivasyon seçmek önemlidir yukarıda bahsettiğimiz gibi hidden layerlarda sigmoid ve tanh yerine Relu daha mantıklı bir seçimdir.Çıktı için sigmoid ya da linear fonksiyonlar mantıklıdır.Nöron ağırlıkları genelde random atanır, öğrenme oranının çok büyük ya da çok küçük olmamasına dikkat edilir, küçükse çok zaman alır, büyükse verileri öğrenmeden geçebilir, tekrar sayısı çok fazla olursa ezberlemeye gidebilir modelimiz, az olursa da tam anlamıyla öğrenemeyebiliriz.

Genel olarak ağlar , eğitim aşamasında her tekrar sayısında ileri yayılım dediğimiz olayı gerçekleştirir.Bu kısımda nöronlara gelen inputlar ağırlıklarla çarpılır , bias eklenir ve aktivasyon fonksiyonuna göre bir çıktı verir, bu tüm layerlar ve nöronlar için tekrarlanır, output kısmında bulduğumuz ile gerçekte olması gerekeni karşılaştırırız ve bir kayıp fonksiyonuyla bir değer buluruz ve bu bulduğumuz değeri kullanarak bu sefer de geri yayılım olayını gerçekleştiririz.Bu kısımda türev kullanarak nöronları ağırlıklarını güncelliyoruz.

Tahmin etmek içinse ağımız sadece bir kez ileri yayılım yapar, output layerda oluşan bizim çıktımız olur.

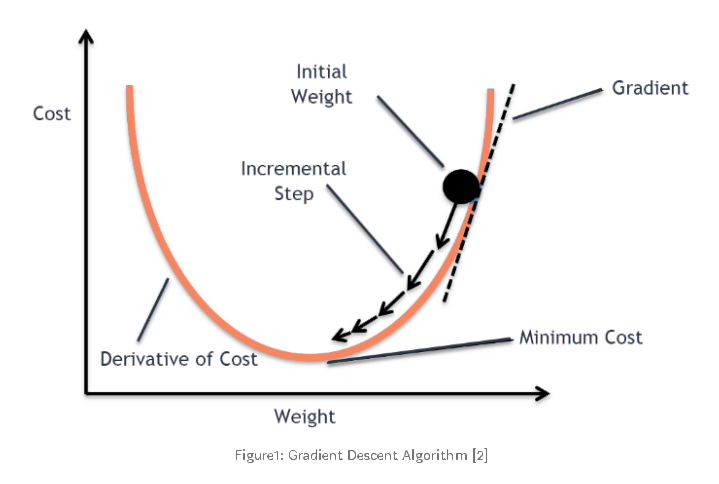
### b) Programın layer\_sizes = [13, 5, 5, 1] ifadesiyle verilen parametrelere göre göre oluşturulan sinir ağının diyagramını çiziniz. Bu diyagramda giriş ve çıkışların isimleri, nöron yapıları ve bağlantıları tam olarak gösterilecek. Ayrıca bir nöronun ayrıntılı giriş ve çıkış ilişkisini gösteren basit diyagramı çizerek matematiksel ilişkilerini yazınız.



Çok karışık gözüküceğinden tüm isimleri gösteremedim tüm ilişkilerin üzerinde nerden geldiği bilgisi yazmalıdır.

### c) Verilen programda Gradient Decent algoritması nasıl çalışıyor? Batch Gradient Descent algoritmasının farkları ve avantajları nelerdir, karşılaştırınız?

Gradient Decent kayıp fonksiyonlarını iyileştirmek için kullandığımız birkaç yöntemden birisidir.Minimuma ulaşmamızı sağlayan bir algoritmadır.Genel olarak her tekrarda nöron ağırlıklarını değiştirerek optimal duruma ulaşmamıza yarar.

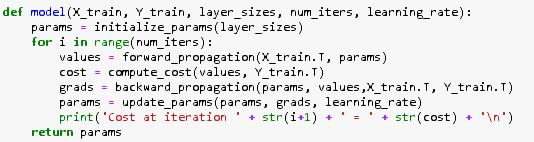


Batch Gradient Descent ise eğitimde tüm verileri kullanarak parametreleri güncellemek için tüm veriler üzerinde bir kayıp fonksiyonu hesaplar.Bu değere göre parametreyi günceller.Diğerlerinden farkı eğitim setindeki her veri için için kayıp hesaplayıp güncellemektense , parametreler sadece tüm eğitim verileri geçtikten sonra güncellenir.Avantajlar :

* Genel minimum için daha az adım atılır
* Tek bir veri için parametre güncellemektense tüm veriler için ortalama alınıp güncellenir
* Tüm verileri beraber işleme hızını arttırır
* Hesaplama için daha efektiftir, tek bir örnek için hesaplamaktansa hepsi için hesaplanmaktadır.

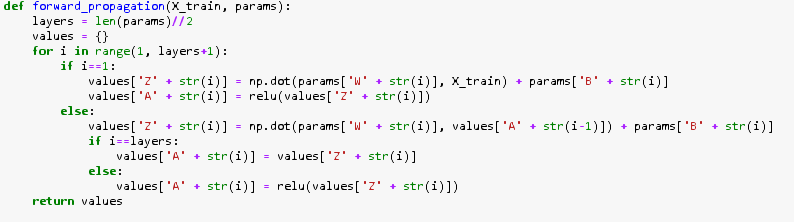
#### d) Verilen programın model, forward\_propagation, backward\_propagation ve predict gibi metotlarını ve önemli değişkenlerini kısaca açıklayınız.

Model:



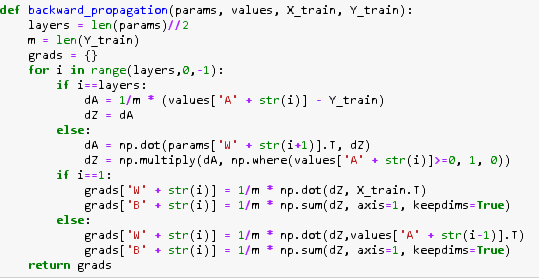
Burda en başta layerların sizelarını vererek ağırlıkları belirliyoruz daha sonra seçtiğimiz tekrar sayısına göre programımızı eğitiyoruz.Model ilk başta ileri yayılım yaparak değerler buluyor(output değerleri) daha sonra bu değerlerle gerçek değerleri karşılaştırarak bir cost buluyoruz.Daha sonra geri yayılım yaparak gradientleri hesaplıyoruz ve bu aşamada parametrelerimizi güncelliyoruz.Bu süreci seçtiğimiz sayı karar tekrarlıyoruz ce modelimizi böylece eğitmiş oluyoruz.

Forward\_propagation:



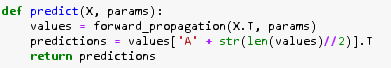
Her bir nöron için giriş değerlerini, ağırlıklarını ve bias değerini kullanarak Z değerini hesaplıyoruz.Bu Z değerini Relu fonksiyonuna soktuğumuzda ise karşımıza A değeri çıkıyor.Bunu tüm nöronlar için tekrarlıyoruz.

Backward\_propagation:



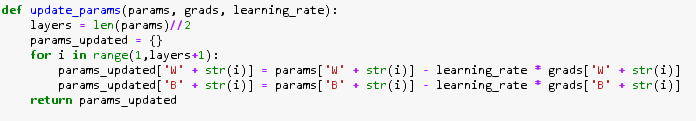
Forwardda hesaplanan değerleri türev yardımıyla kullanarak Gradient değerlerini hesaplamak için geri yayılım yapıyoruz, bu değerleri kaydediyoruz.Bu değerleri kullanıp parametreleri güncellyo olucağız diğer kısımda.

Predict:



İleri yayılımda output layerdan çıkar değerler bizim tahminlerimiz oluyor.Burada onu yapıyor ve predictions vectörünü dönüyor.Bunu hem cost hesaplamak için hem de tahmin etmek için kullanabiliriz.

Update\_params:



Geri yayılımda hesapladığımız gradient değerlerini kullanarak nöronlarımızın ağırlıklarını learning rate yardımıyla güncelliyoruz.

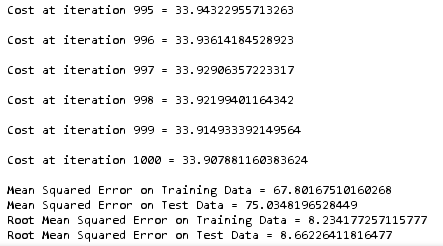
Relu:



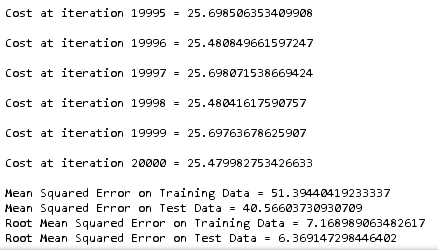
Hidden layerlar için relu fonksiyonumuz, 0 ile z arasından maximum olanı alıyor.Bu sayede nöron çıktılarının negatif olmasını engelliyor.

#### e) Program çalıştırıldığında ekran çıktısını raporunuz koyarak, iterasyonlar, maliyet (cost), RMSE gibi çıktıları kısaca açıklayınız. Bunlar ne gösteriyor?

Çalıştırdığımızda elde ettiğimiz ekran çıktısı:



Burdan görebildiğimiz üzere Cost azalmaya devam ediyor. Bu da demek oluyor ki aslında daha büyük bir tekrarla modelimizi eğitmeye devam edebiliriz.Çünkü cost azalmaya devam ediyor.Deneyelim.



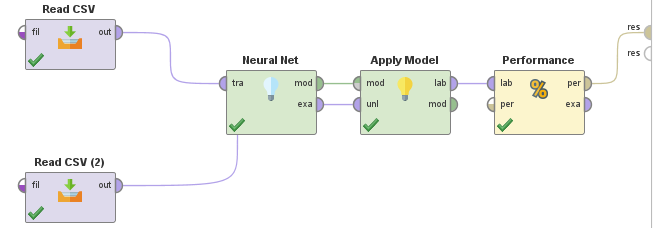
Gördüğümüz gibi her şey aynı fakat tekrar sayısı 20 000 olduğunda daha iyi bir sonuç elde ettik.Burda iterasyon kaç kere eğiticeğimizi gösteriyor.Mean Squared Error on Training, eğitimde karşılaşılan errorların(farkların) karelerinin toplamının ortalamasını veriyor.MSE Test ise görmediği datada olan Mean Squared Errorunu veriyor.Bizim dikkat etmemiz gereken değer bu çünkü eğitimde bu sayının az çıkması modelin eğitildiğinin tam kanıtı değildir.Model ezberlemiş olabilir.

#### f) Verilen programa scikit-learn kütüphanesinden Confusion Matrix oluşturmak için gereken ek satırları yazarak bu matrisi oluşturun ve bu yeni dosyayı neural\_networkCM.py adıyla saklayın. Çıktıyı raporunuza koyarak kısaca açıklayınız.

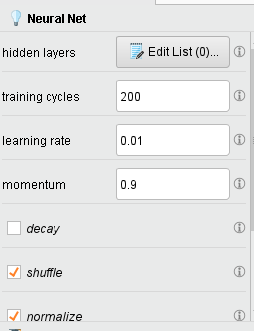
Hocam problemimiz bir regression problemi olduğu için bu probleme Confusion Matrix yazmamız mümkün değil.Çünkü confusion matrix sadece classification problemleri için geçerli bir değerlendirme ölçeğidir.Biz problemimizde MSE(Mean Squared Error) kullandığımız için modelimizi değerlendirebildik.

#### g) Rapid Miner’da training.cvs ve test.cvs veri setlerini kullanan ev fiyat tahmini yapan Neural Network operatörünün kullanıldığı bir proses diyagramı (neural\_network.rmp) kurunuz ve bu diyagramı ve yaptığınız ayarları raporunuza koyarak kısaca açıklayınız.

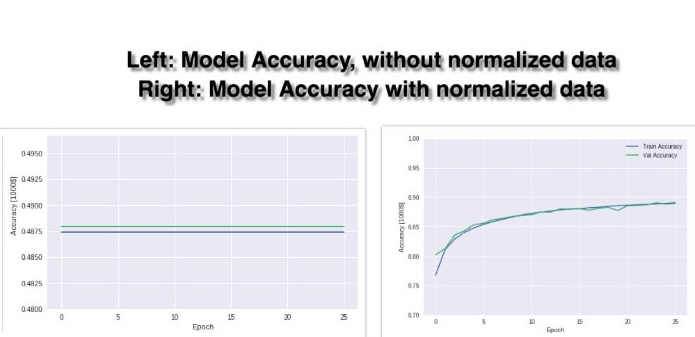
RapidMiner proses diagramımız.Üstteki CSV filemiz training, alttaki ise test inputlarımız.Eğitim verilerini Neural Net’e bağlıyoruz, ağımızı eğittikten sonra Neural Net modelini test verileri ile Apply Model’e bağlıyoruz.Daha sonra ise Performance’e bağlıyoruz.



Neural Net ayarlarımız :

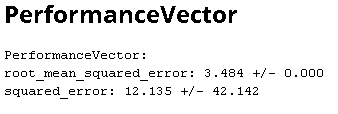


Burada bizim Python kodumuzdan ekstra Momentum, Shuffle, Normalize kullanıldığınız görüyoruz.Momentum minimumdan uzaksak daha hızlı adım atlamamızı sağlıyor(yani süreci hızlandırıyor) Shuffle veriyi karıştırıp tekrar vererek verilerin daha iyi kullanılmasını sağlıyor.Normalization ise verileri ortak bir ölçeğe aralarındaki farklılıkları bozmadan değerlerini değiştirmektir.Bu da bazı durumlarda bizim modelimizin başarı oranını arttırır.İnternetten bulduğumuz güzel bir örnek:



#### h) Rapid Miner’da elde ettiğimiz model ve başarı sonuçları nedir? Modeli ve başarı sonuçlarını raporunuza koyarak kısaca yorumlayınız. Yukarıda (g) şıkkında Python kodu ile elde ettiğiniz Confusion Matrix ile karşılaştırarak kısaca açıklayınız.

Modelimizin başarı oranını ölçmek için Performance Vector kullanıyoruz.Burda elde ettiğimiz veri ise Pythonda elde ettiğimizden çok daha iyi bir şekilde olduğunu görüyoruz.Bunun sebebi yukarıda bahsettiğimiz gibi RapidMinerdeki kullandığımız ekstra parametreler, özellikler.



# 4. Clustering / Outlier Analysis

#### a) Kümeleme (Clustering) ve Aykırı Analizi (Outlier Analysis) veri madenciliğinde ne zaman hangi teknikler kullanılarak yapılır?

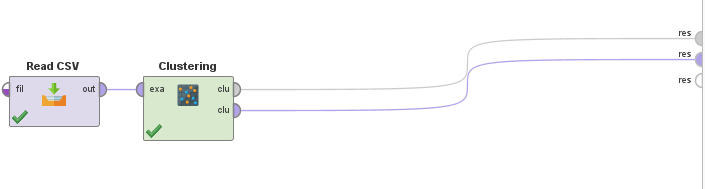
Kümeleme ve aykırı analizi elimizdeki verilerin genel olarak bilgisini öğrenebilmemiz için gereklidir.Veri hakkında bilgi sahibi olmamızı sağlar.Kümeler bize benzer verileri, aykırılar ise bize veri setimizdeki uçuk verileri gösterir.Aykırı verileri bazen eğitimin dışında tutmak bize daha başarılı modeller eğitebilmemize yardımcı olur.Kümeler ise bize benzer verileri gösterir. Bazı çeşitleri ise :

* Bölümleme Yöntemi
* Hiyerarşik Yöntem
* Yoğunluk Tabanlı Yöntem
* Sistem Tabanlı Yöntem(Grip-based)
* Model Tabanlı Yöntem
* Kısıtlamaya Dayalı Yöntem

Bunlar popüler olarak kullanlan kümeleme yöntemlerinden bir kaçı.

#### b) Boston Ev Fiyatlarını RMS’da Clustering (k-Means) operatörü kullanarak ve küme sayısı parametresiyle oynayarak farklı kümemelere ayıran, değişik renkli noktalar olarak scatter çizim (plot) ile gösteren prosesi (boston\_clustering.rmp) oluşturun ve çıktıyı üretiniz. Proses diyagramını raporunuza koyarak kullandığınız parametrelerle beraber yaptığınız kümeleme işlemini kısaca açıklayınız.

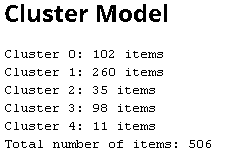
Prosesimiz :



Parametreler :

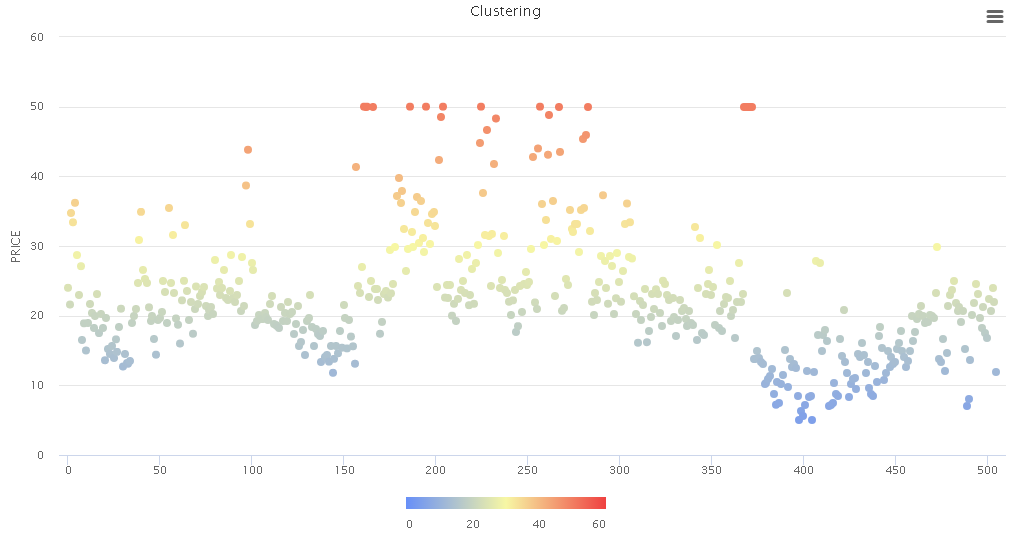
* K = 5(Bu sayı kaç adet parçaya böleceğimizi belitriyor)
* Measure types = BregmanDivergences(default)
* Max optimization steps = 100(default)
* Divengerce = Squared Euclidean Distance(uzaklığa göre gruplandırma)

Prosesimizi çalıştırdıktan sonra elde ettiğimiz Cluster Modelimiz :

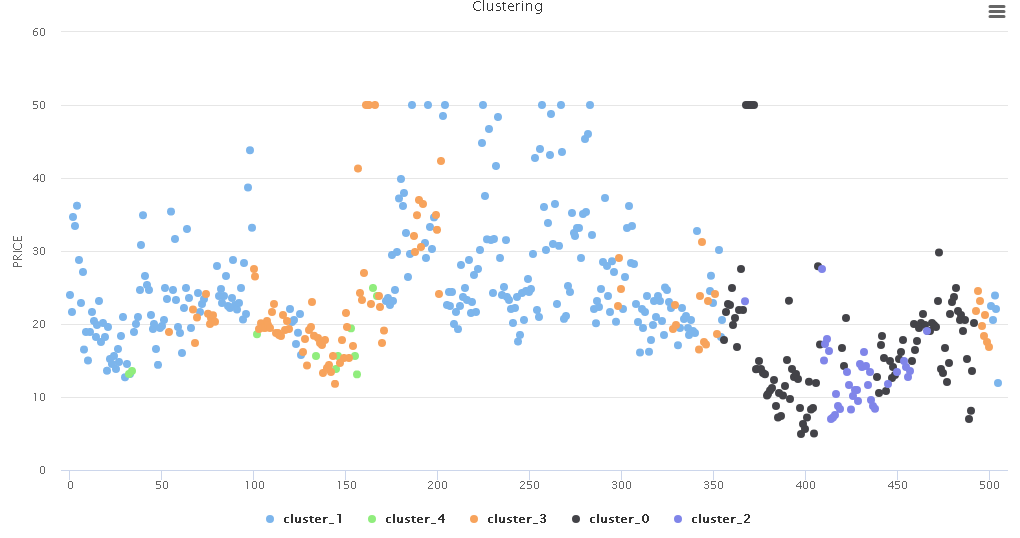


Burdan görüyoruz ki modelimiz 506 adet Boston verisini 5 adet kümeye bölmüş.Bu kümelere bakalım.

Cluster öncesi :



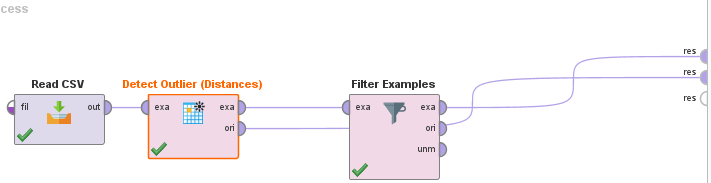
Cluster Sonrası:



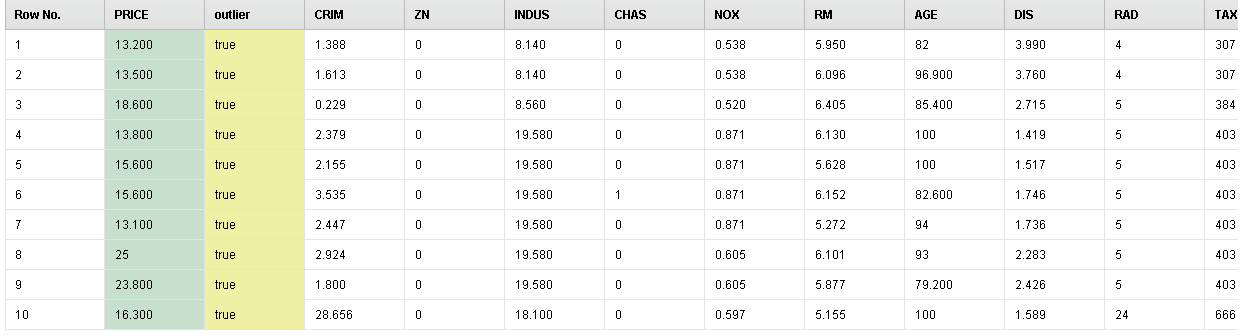
Bu iki resimde dikkat etmemiz gereken normalde fiyat aralığına göre bir parçalandırma yapan yukarıdaki resimin clustering sonrası özellikleri birbirine yakın olan verileri 5 K-Means kullanarak farklı bir şekilde ayırması.Bunun sebebi ise gruplandırmayı uzaklığa bağlı bir şekilde yapması.

#### c) Boston Ev Fiyatları veri setinindeki genel dağılımdan farklı aykırı ev fiyatları verilerini (outliers) RMS’da ilgili operatörleri kullanarak bir proses (boston\_outliers.rmp) oluşturarak tespit edin ve temizleyiniz. Proses diyagramını raporunuza koyarak kullandığınız parametrelerle beraber yaptığınız temizleme işlemini kısaca açıklayınız

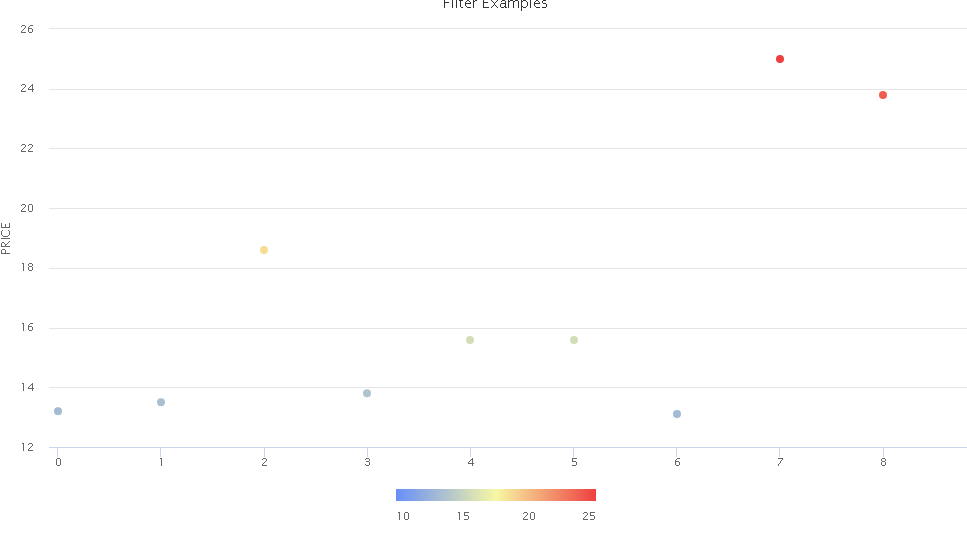
Prosesimiz :



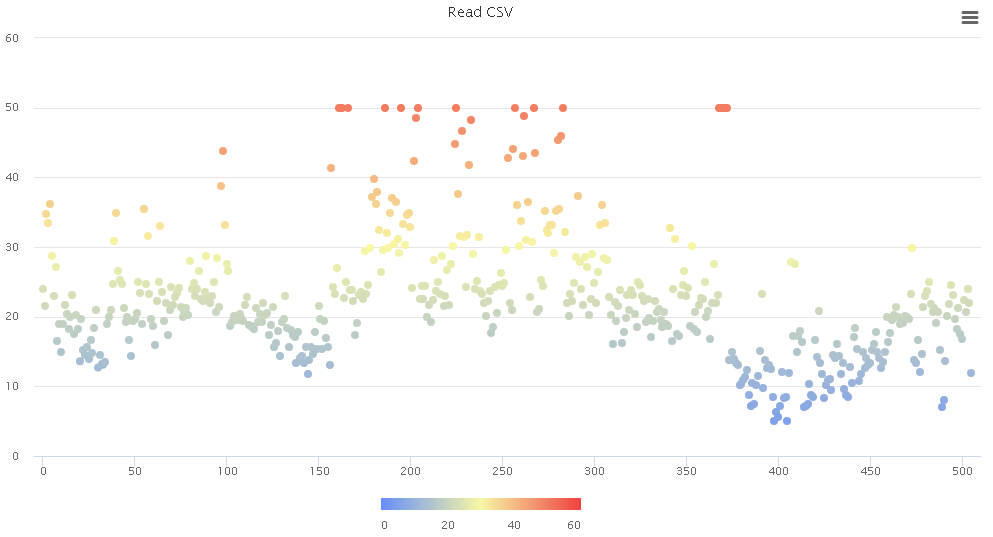
Verimizde bulunan outlierlar



Bulduğumuz outlierlar :

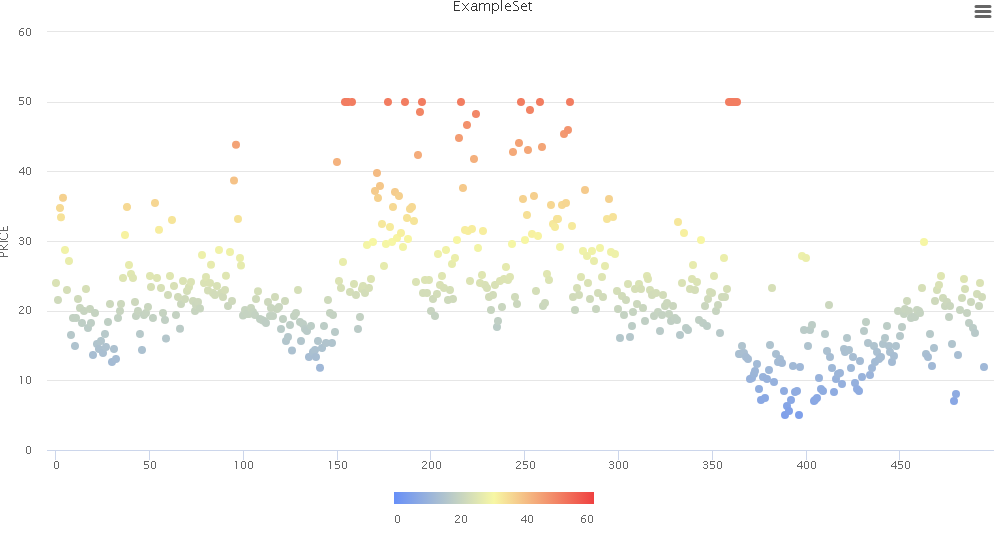


Outlierlar kaldırılmadan önce :



Outlierlar kaldırıldıktan sonra :

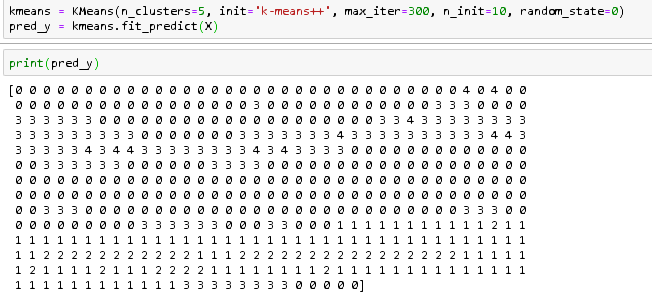




Outlier sayısı daha da arttırılabilir fakat fazla arttırılması veri kaybına yol açar.Genel olarak outlierların amacı çok uçuk olan değerlerin veri setimizden çıkarılması demektir.Bu uçuk değerler modeller eğitilirken eğitim sürecini bozabilir.

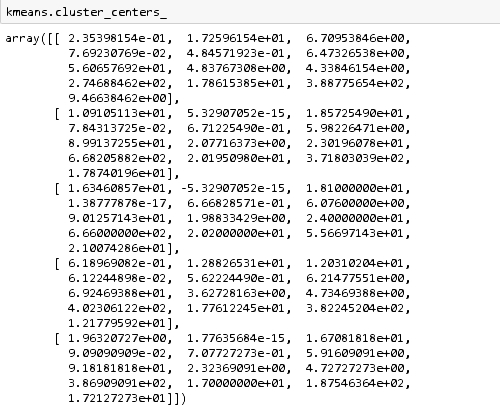
#### d) Boston Ev Fiyatları verisi setine RMS’de yaptığımız gibi benzer kümeleme ve aykırı durum analizini Pyhton kodu ile yapabilecek basit bir program geliştiriniz (clustering\_outliers.py) ve çıktılarını alıp raporunuza koyarak yaptığınız kümeleme ve temizleme işlemlerini kısaca açıklayınız.

Öncelikle Pythonda clustering işlemi yapabilmemiz için sklearn.cluster içinde olan Kmeans fonksiyonunu kullanıyoruz.Bu fonksiyon bizden cluster sayısını istiyor ve verdiğimiz sayıya göre verilerimizi clusterliyor ve sonucunda verdiğimiz sayı kadar çevresinde veri toplanmış noktalar veriyor.

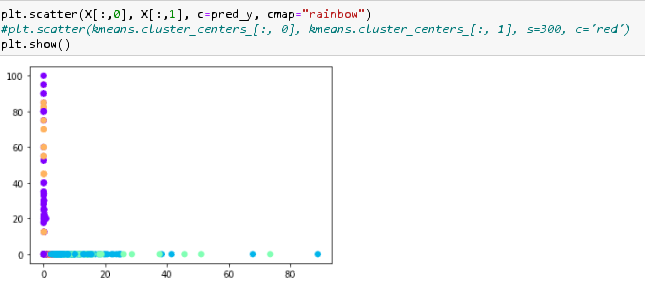


RapidMinerda 5 adet clusterımız olduğu için burda da 5 adet cluster kullanıyoruz.Gördüğümüz gibi verilerimiz kendilerine en yakın cluster’a atanmış.Bu noktalara cluster merkezleri diyoruz.Genel olarak clustering mantığı ise bu noktaları K Means bizde 5e bölebilecek en iyi noktalara getiriyor.

Cluster merkezlerimiz:



Elde ettiğimiz clusterları görselleştirelim:

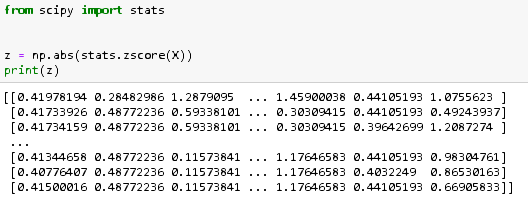


Outlier bulmak için birden fazla yolumuz var.Bunlar Box Plot, Scatter Plot, Z-Score, IQR Score.Biz Z score kullanıyo olucağız.

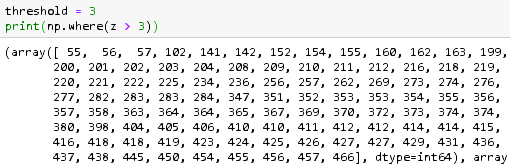
Wikipedia tanımı : Z-skoru, bir gözlem veya veri noktasının değerinin, gözlemlenen veya ölçülen şeyin ortalama değerinin üzerinde olduğu işaretli standart sapma sayısıdır.

Bu Z skorunu hesaplayıp, belli bir sınır(threshold) belirleyip bu sınırın üzerindeki değerlerimize outlier diyo olucağız.

Verimizin Z skorunu hesaplıyoruz:



Şimdi bahsettiğimiz sınırı belirleyip bu sınırın üstündekilere Outlier diyo olucağız.



Gördüğümüz gibi 3 değeri için 100 adet outlier bulduk.506 adet veri için bu sayı biraz fazla.Burdaki 3 değerini değiştirip istediğimiz bir sayıyı bulana kadar deneyebiliriz. İstediğimiz sayıya düşürdükten sonra tek yapmamız gereken indexleri gözüken verilerimizi , orijinal veri setimizden çıkarmak ve modelimizi o temizlediğimiz veri seti ile yapmak. Outlierlar genelde uçuk değerleri olduğu için( Z skorlarında bazı veriler 0.xx, 1.xx çıkarken bunlar +3 çıkmıştı) bu veriler eğitime zarar verir, temizlendiği zaman daha ideal bir model elde etmemiz gayet mümkündür.