1. Regresyon/Nümerik Tahmin (Prediction)

Bir verisetinde sayısal tahmin/prediction yapıyorsak o zaman "Regression" modelleri/algoritmaları kullanırız. Maaş, hava sıcaklığı, bitcoin borsa değeri, döviz tahmini vs birer nümerik/sayısal tahmin problemidir. ÖNEMLİ NOT: Zamana bağlı (belli bir saat veya tarih) tahmin Time-Series problemidir. Sayısal tahmin olsa da regresyondan farklıdır.

Genel olarak regresyon modelleri doğrusal (linear) ve doğrusal olmayan (non-linear) olarak iki grupta incelenir. Algoritma isimleri üzerinden bakarsak

- Linear/Doğrusal Modeller: Linear Regression
- Non-linear/Doğrusal Olmayan Modeller: SVR, Random Forests
- NOT: Değer tahmini şu an bilinmeyen (gelecek zamanı dikkate almayan) bugüne ait bir tahmin olabildiği gibi, sayısal değerin ileriki bir zamanda tahmin edilmesi şeklinde de olabilir. Örneğin bir güneş panelinin geçmiş tarihlerde ürettiği enerji miktarını gelecek zamanda tahmin edip panelin verimliliği hesaplanabilir. (Aslında bir zaman serisi problemi biraz farklı!)

Bu bölümde aşağıdaki regresyon algoritmaları görülecektir.

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Support Vector for Regression (SVR)
- Decision Tree Regression (Classification içinde kullanılıyor!)
- Random Forest Regression (Classification içinde kullanılıyor!)

NOT: Bölüme ait tüm Kodlar ve Datasetler

https://www.superdatascience.com/machine-learning/ linkinde PART 2 Regression başlığının altında yer almaktadır.

1.1 Linear Regresyon

Problem: Aşağıdaki veri setinde bir personelin yıl cinsinden tecrübesine karşılık maaşını tahmin eden bir model üretilmek istenmektedir. Bu problem ile Linear Regresyon çalışma mantığına bakalım.

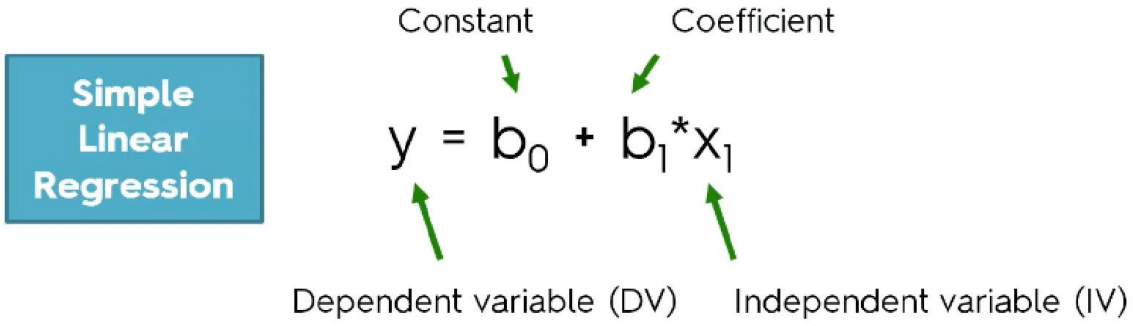
Dataset: Yıllık tecrübeden maaşı tahmin edeceğiz.

Tecrübe Yılı	Maaş
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957

SORU: Verisetine bakınca görülen ilişki nedir?

CEVAP: Tecrübe maaş ile orantısal (yüzde yüz doğrusal veya değil) artıyor gibi.

AMAÇ: Bağımlı değişken olan x değeri ile bağımsız değişken olan y değeri arasında bir doğrusal bir ilişki tanımlarsak bunu matematiksel olarak nasıl ifade ederdik?



Bir doğru denklemiyle! Bu denklemde b0 x'in y eksenini kestiği nokta, b1 ise eğim olur. Elimizde x=tecrübe yılı ve y= maaş şeklindedir. b0 ve b1 tanımlarıyla problemi yeniden düzenlersek:

SORU: Bu denklem ne yapar?

CEVAP: Bu model sayısal kestirim yapar. Yani 10 senelik tecrübeye sahip birinin (bu dataset te yer almayan bir değer) maaşını eldeki verilerden faydalanarak bulabilir. Bu denklem sayısal kestirim yapmakta kullanılan bir formülden ibarettir.

SORU: Noktaların ve çizginin (doğru) anlamı?

CEVAP: Noktalar dataset'in X-Y (tecrübe, maaş) şeklinde çizilmesi. Doğru ise bizim tahminde kullandığımız model.

SORU: Kestirim nasıl yapılır?

CEVAP: (Yeni) tecrübe noktamızdan (veri setinde olmayan!) doğrumuza bir dikme çıkar ve doğrunun Y düzleminde karşılığını buluruz.

SORU: Modelin doğruluğu ne ile orantılı?

CEVAP: "Çizdiğimiz tahmin doğrusu" oradaki noktalara ne kadar uyumluysa (regresyon felsefesi) o kadar yakın tahmin yapabilir. Ama doğrumuz veri setine nasıl uyacak? Eğim değişirse tahmin yeteneği değişir! Şekilde görebiliyor muyuz ?

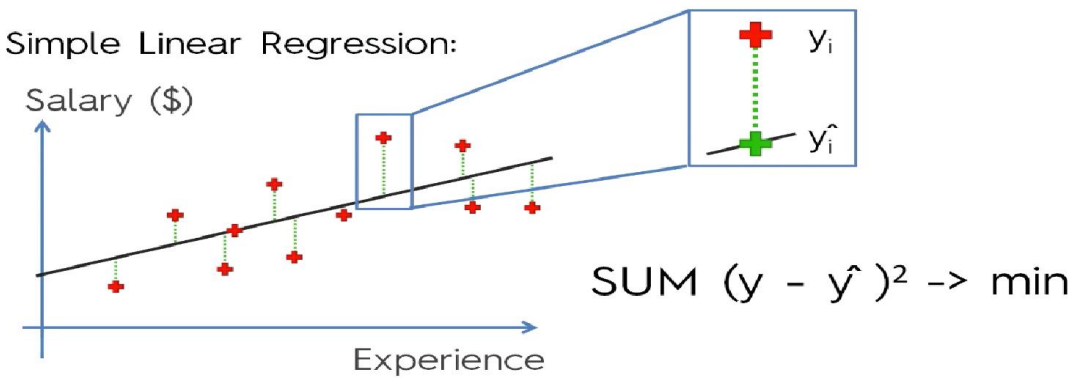
SORU: Bu doğruyu optimum (noktalara en yakın geçecek şekilde) nasıl ayarlayabiliriz? Eğim değiştikçe birkaç doğruya yakınlaştıkça başkalarından uzaklaşacaktır?

https://en.wikipedia.org/wiki/File:Least_Squares.gif

inceleyelim..

CEVAP: Tüm doğruları dikkate alarak "toplam hatayı" küçülten/minimize eden bir yol kullanacağız. Burada

Hata = | Tahmin Değeri – Gerçek Değer | olacaktır. Hata'yı (SUM) küçülten algoritmaya Least Mean Squares yöntemi denir.



Somut ifade edersek:

y (kırmızı) gerçek değer, y^ (yeşil) doğrunun tahmin değeri

min : Doğru nasıl şekillenmelidir ki bu hatalar toplamı sıfıra en yakın olsun

NOT: Aynı dataset üstünde çalışan regresyon algoritmalarının karşılaştırılması için en sık kullanılan metrik MAE, MSE ve RMSE'dir. Genel olarak gerçek değer (y) ile tahmin (y') farkı ortalaması olarak hesaplanan bu metriklerde 0 değerine yaklaştıkça modelin tahmin yeteneği daha iyidir diye ele alınır.

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

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

Mean Squared Error (MSE) is the mean of the squared errors:

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

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

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

YANİ: MAE farkların ortalaması, MSE farkların karelerinin ortalaması, RMSE farkların karalerinin ortalamasının kökü

Simple Linear Regression LAB Çalışmasını yapalım

***

1.2. Multiple Linear Regression (Çok değişken X1, X2, …Xn ile Y tahmini)

ÖNEMLİ SORU: Linear Regresyon her zaman iyi sonuçlar üretirmi?

ÖNEMLİ CEVAP: Bir regresyon problemi bir tahmin değişkeni (X) ve tahmin edilecek değerden (Y) ibaret ise linear regresyon yeterli seviyede hassas kabul edilebilir. Ya Y değerini tahmin etmek için birden fazla değişken varsa? Örneğin aşağıdaki problemi ele alalım:

Arge Harcamaları	Yönetim Harcamaları	Pazarlama Harcamaları	Şehir	Kar
44069.9	51283.1	197029	Ankara	89949.1
542.05	51743.2	0	İstanbul	35673.4
20229.6	65947.9	185265	İstanbul	81229.1
38558.5	82982.1	174999	Ankara	81005.8
27892.9	84710.8	164471	İzmir	77798.8
46014	85047.4	205518	İstanbul	96479.5
142107	91391.8	366168	İzmir	166188
100672	91790.6	249745	Ankara	144259
23640.9	96189.6	148001	Ankara	71498.5
77044	99281.3	140575	İstanbul	108552
131877	99814.7	362861	İstanbul	156991

Burada Arge harcamaları, Yönetim Harcamaları, Pazarlama Harcamaları, şehir ve bunlara bağlı olarak “Kar” tahmin edeceğiz. → Multiple Linear Regression

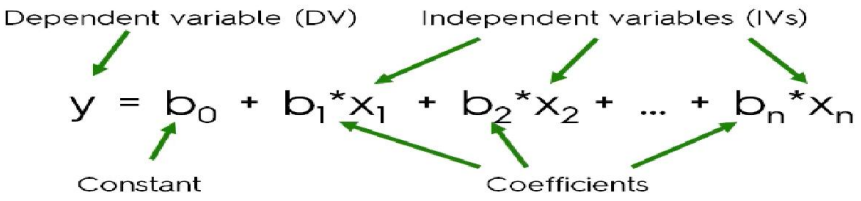
SORU: Matematiksel model olarak Multiple Linear Regression?

CEVAP: Tek değişkenle y = b0+ b1*x şeklinde bir denklem elde etmiştik. Buna benzer şekilde çok sayıda Linear Regression toplamı gibi yazarsak aşağıdaki formülasyonu elde ederiz:

Simple Linear Regression

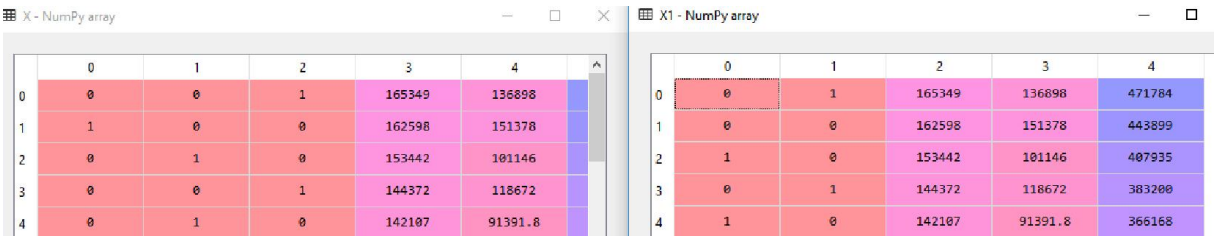
$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression



Buna göre y değerinin (yukarıda kar) tahmini için x1, x2, x3, x4 (Arge harcamaları, Yönetim Harcamaları, Pazarlama Harcamaları, Şehir) değerlerini kullanıyoruz.

DİP NOT: Regresyon problemlerinde kategorik ifadeler (burada Şehir) kullanılamaz. Bunun için dönüşüm yapmalıyız. Bu dönüşüm sırasında şehir kolonu 3 kolonla ifade edilebileceği gibi, bir öznitelik sayısını azaltacak şekilde de dönüşüm yapılabilir. Yani soldaki kolonlara bakarsak ortadaki "0" kolonu olmadan da 01, 00,10 şeklinde (ankara, İstanbul izmir gibi) ifade etmek mümkündür.



	0	1	2	3	4
0	0	0	1	165349	136898
1	1	0	0	162598	151378
2	0	1	0	153442	101146
3	0	0	1	144372	118672
4	0	1	0	142107	91391.8

	0	1	2	3	4
0	0	1	165349	136898	471784
1	0	0	162598	151378	443899
2	1	0	153442	101146	407935
3	0	1	144372	118672	383200
4	1	0	142107	91391.8	366168

Yani ilk kolonu dahil etmeden, basit bir array/dizi işlemi ile

X = X[:, 1:] (1 kolonu dahil tüm kolonları ve satırları alarak X e atama yap) komutunu kullanacağız.

NOT: Python kütüphaneleri genellikle bu sorunu otomatik çözerler ancak gerektiğinde kullanmak adına buraya not edildi.

Multiple Linear Regression LAB Çalışmasını yapalım

***

1.3. Polinom Tabanlı Regresyon (Polynomial Regression, Doğru yerine noktalara daha yakın bir fonksiyon yani polinom ile modelleme)

SORU: X-Y düzlemindeki noktaları içine alacak şekilde (train data) BASİT bir fonksiyon olarak bir doğruyu kullandık. Ancak train datayı daha iyi temsil edecek ve dolayısıyla daha iyi tahmin yapacak bir yol yok mu?

CEVAP: Train data'ya daha yakın öğrenecek doğru yerine eğri (polynom derecesine göre değişecektir) kullanabiliriz.

PROBLEM: Bir firmaya iş için başvuran kişilerin tecrübe-pozisyon bilgilerine göre onlara verilecek maaşı tahmin etmek.

Pozisyon	Seviye	Maaş
İş Analisti	1	45000
Junior Yazılımcı	2	50000
Uzman Yazılımcı	3	60000
Yönetici	4	80000
İl Müdürü	5	110000
Bölge Müdürü	6	150000
Genel Müdür	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

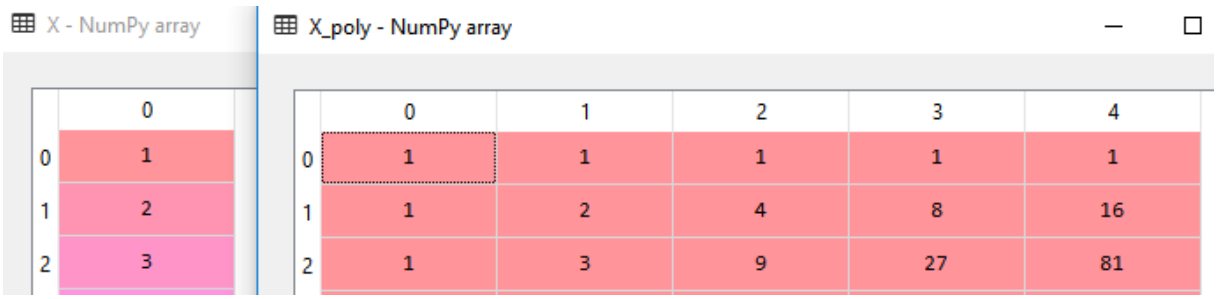
NOT: Excell'de datanın dağılımını görelim ve ne tür bir fonksiyon uydurabileceğimizi test edelim. Uydurulan eğri için derece seçimi de yapılabiliyor!

NOT: Python kodunda linear regression kütüphanesinden türetilen polynomial regression temel olarak polinom derecesine bağlı olarak giriş değişkenini polinomlar toplamı (dereceler) şeklinde ifade etmektir. Yani her iki regresyon modelini kıyaslarsak

```
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

Linear regresyon X matrisini kullanırken, polinom tabanlı regresyon X_poly tabanlı matrisi kullanır. Bu iki değişkenin elemanlarını yan yana LAB da inceleyelim.



	0
0	1
1	2
2	3

	0	1	2	3	4
0	1	1	1	1	1
1	1	2	4	8	16
2	1	3	9	27	81

SORU: Polynom derecesi ile modelin tahmin hassasiyeti arasında bir ilişki var mıdır? Bu derece arttıkça tahmin yeteneği nasıl değişir? Bu değişimin özel bir anlamı var mı? Excell de test edelim.

Polynomial Regression LAB Çalışmasını yapalım ve Linear Regresion İle KIYASLAYALIM

***

1.3.1 Tahminde En Etkili Değişken(leri) Bulmak- Backward Elimination (SONRA İNCELEYELİM)

Backward Elimination aslında temelde bir feature selection (öznitelik seçimi) yöntemidir. Bu konuya daha sonra detaylı dönülecektir. Ancak kısaca "feature selection nedir? ne işe yarar?" sorularını cevapladıktan sonra ona bağlı olarak backward elimination yöntemi nasıl çalışır incelenecektir.

Feature Selection: Daha önce kullandığımız dataset üzerinde konuşalım

Arge Harcamaları	Yönetim Harcamaları	Pazarlama Harcamaları	Şehir	Kar
44069.9	51283.1	197029	Ankara	89949.1
542.05	51743.2	0	İstanbul	35673.4
20229.6	65947.9	185265	İstanbul	81229.1
38558.5	82982.1	174999	Ankara	81005.8
27892.9	84710.8	164471	İzmir	77798.8
46014	85047.4	205518	İstanbul	96479.5
142107	91391.8	366168	İzmir	166188
100672	91790.6	249745	Ankara	144259
23640.9	96189.6	148001	Ankara	71498.5
77044	99281.3	140575	İstanbul	108552
131877	99814.7	362861	İstanbul	156991

Kar alanı/kolonu dört tane bağımlı alan üzerinden tahmin edilmektedir. Burada amaç en az bir en fazla dört değişkeni (kolonu) kullanarak kar kolonunu en hassas şekilde tahmin etmektir. Burada kritik fayda nedir? Eğer kar kolonunu bir kolona bağlı olarak tahmin edebilirsek o zaman hem dataseti daha iyi anlamış hem de ML eğitimi sırasında değişken sayısı azaldığı için hesapsal yükü azaltmış oluruz. Burada kritik nokta şudur:

i) Tüm kolonları kullanırken elde edilen başarım referans olarak tutulur

ii) Bir feat selection yöntemiyle yine tahmin yapılır ve elde edilen başarım ilk adımla kıyaslanır.

iii) Kıyaslama sonucunda başarım yeterli kabul edilirse (biz karar vereceğiz) daha az sayıda kolon içeren datasetle çalışmaya devam edilir.

ML literatüründe çok sayıda Feat Selection yöntemi olmakla beraber burada giriş seviyesinde Backward Elimination'u ve bunun Python da uygulamasını inceleyeceğiz

SORU: Backward Elimination?

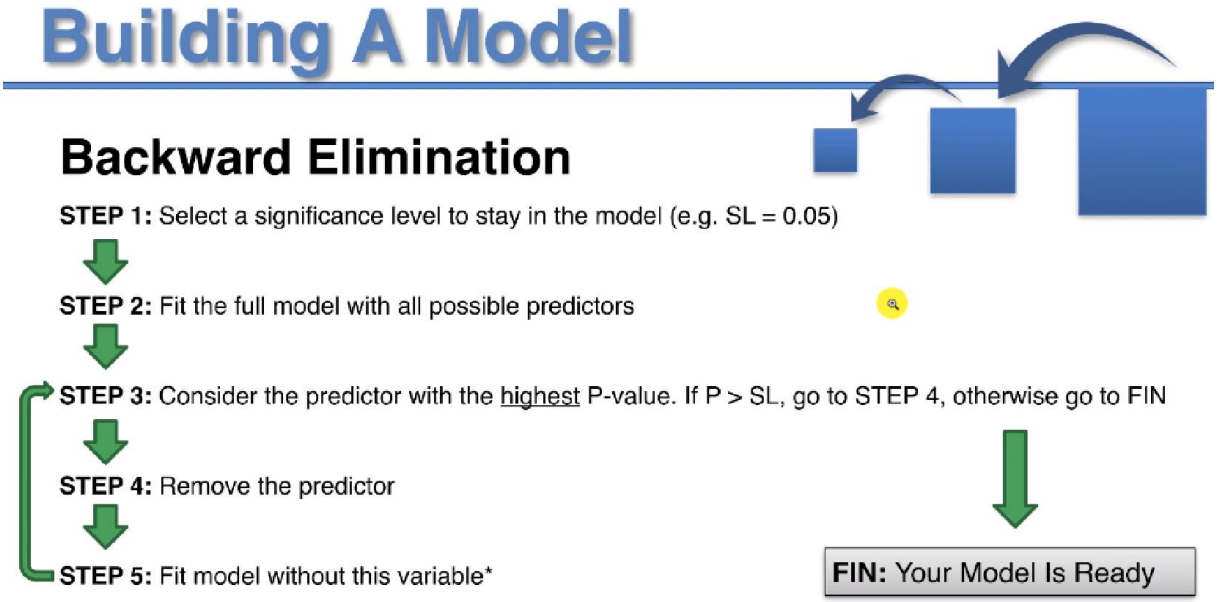
CEVAP: Adım adım eldeki değişkenlerin sayısını azaltarak ve performansı ölçerek yürüyen bir süreç

Yani:

i) Bağımlı değişkenler olan kolonların (X'ler) bir istatistiksel yöntemle ağırlıkları hesaplanır.

ii) En az puanlı veya en yüksek puanlı'dan başlayarak (hesaplama yöntemine bağlı değişir) birer birer kolonlar elimine edilerek (tüm kolonların kümesinden, tek kolonlu kümeye doğru) performans gözlenir. Elimine etmek çoktan aza olursa backward, azdan çoğa olursa (ekleye ekleye gidilirse) forward eliminasyon olur.

iii) Performans ölçümüne göre durulması gereken hedef nokta belirlenir ve eldeki dataset ile devam edilir. Aşağıda modelin çalışma biçimini görelim.



Multiple-Linear Regression Deneyindeki Kod ve Datasetle devam edeceğiz. Burada feature seçme yöntemi olarak istatistiksel bir yöntem olan "p-value" yi kullanacağız (şekilde SL=0.05 bu p değerine bağlı belirlenen eşik). Detaylara girmeden bir kolonun/değişkenin "p-value" değeri ne kadar yüksekse o oranda elimine edilmesi uygundur. Yani adım adım giderken kolonları bir bir elerken, p-value değerine bakacağız. Literatürde p-value için 0.1 veya 0.05 eşik değeri seçilmesi tavsiye edilmiştir.

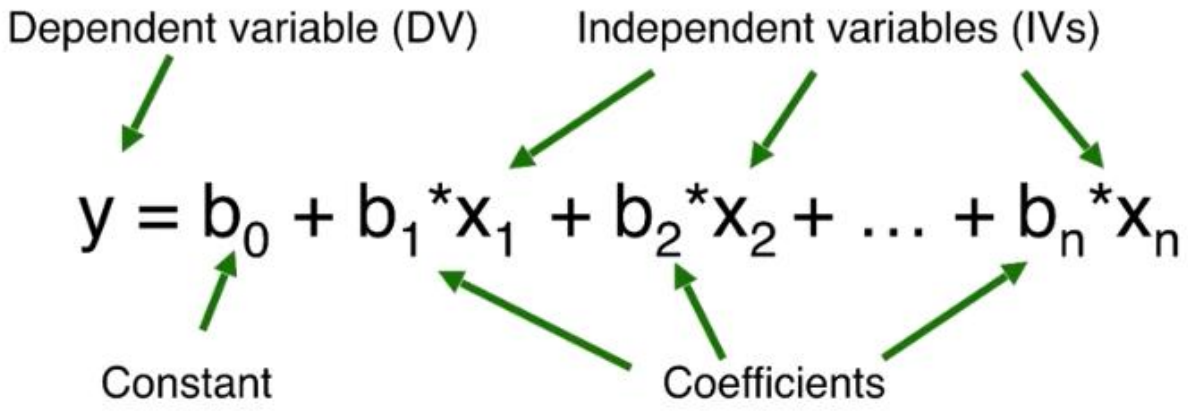
BKZ: http://www.jerrydallal.com/lhsp/simplify.htm

Şimdi deneyle ilerleyelim.

P-value Backward Elimination LAB Çalışmasını yapalım

***

Deneye geçmeden önce verisetinde ufak bir manipülasyon yapmalıyız. Öncelikle Multiple Linear Regression denklemini hatırlayalım:



x1, x2, ..xn için b1, b2.. bn katsayıları var olduğu için Backward Elimination onları dikkate alacaktır. Ancak burada b0'ın da gerçekte katsayısı x0 gibi bir vektördür. b0'ı dikkate almak için 1'lerden oluşan bir x0 vektörüyle çarpılmış gibi (b0 aynen kalmış oluyor) düşüneceğiz. Burada dikkat etmemiz gereken nokta linear reg veya multiple reg algoritmalarının b0'ı dikkate aldığı. Bu ekleme Backward Elimination algoritmasından kaynaklanıyor.

Bu ekleme için

X = np.append(arr = np.ones((50,1)).astype(int), values = X, axis = 1) kodunu kullanacağız.

Bu kod ise X'in soluna 50 satırlık 1'lerle dolu kolonu (dikey eksende axis = 1) ekler. Artık b0, algoritma tarafından bir kolon olarak ele alınıyor.

NOT: bu deneyde istatistiksel olarak üretilen çok sayıda metrik var. Yani "fit" edilmiş modelin feature'larla ilişkisi aşağıdaki şekilde görünüyor. P-value kısmına dikkat edelim.

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

Omnibus:	14.782	Durbin-Watson:	1.283
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.266
Skew:	-0.948	Prob(JB):	2.41e-05
Kurtosis:	5.572	Cond. No.	1.45e+06

Seçimi (eliminasyonu) pvalue > 0.05 ve en büyük değer olarak yapıyoruz. Buna göre elimine edilecek kolon 0.990 ile x2 olur.

Tüm seçimler buna göre

```
import statsmodels.formula.api as sm
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

şeklinde olacaktır.

SONUÇ: İdeal modelde elimizde tek bir kolon olan Ar-Ge kolonu kaldı.

NOT: Bu model optimum başarımı sağlıyor mu? MLR normal ve MLR optimal kolonlar ile train/test yapılmalı ve sonuçlar “yanyana” karşılaştırılmalı.

1.4 SVR (Support Vector Regression) Modeli

Regresyon (sayısal kestirim) için çok sayıda algoritma kullanıldığından bahsetmiştik. Bu bölümde hem sınıflandırıcı olarak (özellikle metin madenciliğinde) hem de sayısal tahmin algoritması olarak çok kullanılan SVR modelini inceleyeceğiz.

1.4.1 SVR Kısa Teori

SVR temel olarak SVM sınıfından bir yöntem olup Support Vector Machine hem sınıflandırma hem de regresyon için kullanılmaktadır. Matematiksel olarak yoğun bir arka planı olan yöntemin detaylarına girmeden doğrusal regresyon mantığı ile kıyaslarsak:

Doğrusal regresyonda eğitim sırasında tahmin ile gerçek veri arasındaki farkı yani hata miktarını Least Mean Squares gibi bir algoritmayla minimize etmeye çalışıyoruz. SVR ise hata için belli bir eşik değeri belirleyerek onun altında kalacak şekilde çözüm arar/model üretir. Bir problem ile mantığı anlamaya çalışırsak:

Diyelim ki, yarınki döviz kurunu tahmin etmek için bir grup geçmiş ekonomik veri ile SVR'yi eğitiyoruz. Burada üst sınır yanlış tahmin durumunda kaybetmeyi göze aldığımız parayı (maksimum kabul edeceğimiz hata payı) olur.

SVR doğrusal olmayan bir regresyon algoritmasıdır. Bir çok ML algoritmasında olduğu gibi SVR'nin eğitiminde kullanılan parametreler vardır. Bu parametreleri "sklearn" paketindeki haliyle inceleyelim:

from sklearn.svm import SVR (kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1)

Burada tüm parametreleri anlatmak yerine en çok kullanılan kernel parametresini inceleyeceğiz:

Kernel : SVR eğitiminde kullanılan fonksiyondur. Bir tür seçimi yapılmaz ise varsayılan fonksiyon radial basis function 'rbf' dir. Diğer fonskiyonlar 'linear', 'poly' ve 'sigmoid' şeklindedir. Burada degree parametresi yazılırsa sadece poly (polynomial/dereceli fonksiyon) için dikkate alınır.

SORU: Farklı kernel farklı başarım üretir mi?

CEVAP: Evet

http://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html

linkindeki kod ile deneme yaparsak

```
from sklearn import datasets, svm
digits = datasets.load_digits()
X_digits = digits.data
y_digits = digits.target
svc = svm.SVC(C=1, kernel='sigmoid')
svc.fit(X_digits[:-100], y_digits[:-100]).score(X_digits[-100:], y_digits[-100:])
```

Kodu için sadece kernel'ı sırasıyla

```
In [6]: svc = svm.SVC(C=1, kernel='linear')
   ...: svc.fit(X_digits[:-100], y_digits[:-100]).score(X_digits[-100:], y_digits[-100:])
Out[6]: 0.98

In [7]: svc = svm.SVC(C=1, kernel='rbf')

In [8]: svc.fit(X_digits[:-100], y_digits[:-100]).score(X_digits[-100:], y_digits[-100:])
Out[8]: 0.42

In [9]: svc = svm.SVC(C=1, kernel='poly')

In [10]: svc.fit(X_digits[:-100], y_digits[:-100]).score(X_digits[-100:], y_digits[-100:])
Out[10]: 0.98

In [11]: svc = svm.SVC(C=1, kernel='sigmoid')

In [12]: svc.fit(X_digits[:-100], y_digits[:-100]).score(X_digits[-100:], y_digits[-100:])
Out[12]: 0.1
```

Test edersek (büyük sonuç daha iyi bu denemelerde!) sonucun değiştiğini görebilmekteyiz.

SVR için çoklukla kernel ve C (ve bazen gamma) parametreleri için değerler veriliyor. Burada önemli sorun/soru?

SORU: Bu parametreleri nasıl seçeceğiz?

CEVAP: gridsearch, parameter tuning, hyperparameter tuning aramalarıyla örnekler görebiliriz. Şimdilik bu seçimi varsayılan olarak bırakacağız.

NOT: Kodlama sırasında "normalizasyon" kavramına benzer şekilde "scaling/ölçekleme" yapılıyor. Bunun nedeni SVR'nin buna duyarlı olması. Yani SVR, anlamlı sonuçlar üretmek için feature'larının birbirine ölçekli/scale edilmesini istiyor. scale edilmiş ve edilmemiş modellerin performansını LAB uygulamasında görebiliriz.

SVR LAB Çalışmasını yapalım

1.5 Decision Tree (Karar Ağacı) ve Random Forest (Toplu Karar Ağacı) Regression

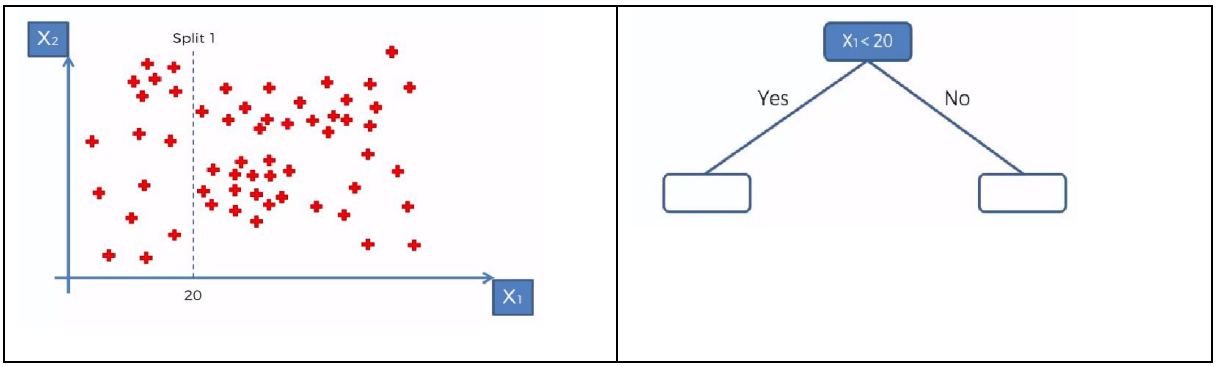
Karar Ağaçları entropy adı verilen kavram üzerinden verisetini tahmin yapılacak kolonu (sınıflandırma veya regresyon) gözetecek şekilde homojen/benzer gruplara bölmesi/split ile ortaya çıkan yapılardır. Ağaçların ne kadar böleceği (ağacın yüksekliği) ve hangi değerden başlayarak kök düğümü bulacağı entropy/split kavramları ile ilişkilidir. Örneğin X1, X2 kolonları ile bir Y kolonunu regresyonla tahmin etmek için (amacı unutmayalım) Karar Ağacı şöyle çalışacaktır.

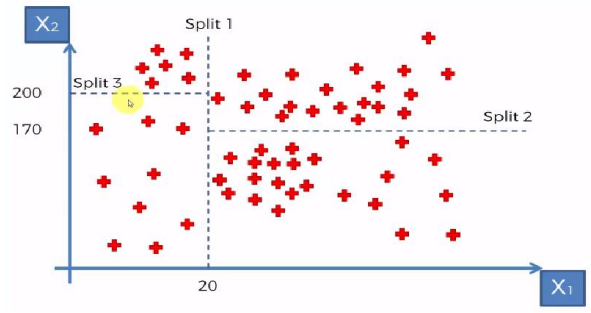
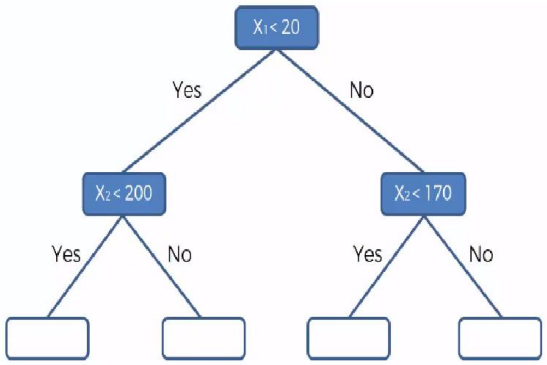
i) X1, X2 ve Y noktalarını (3 boyutlu çizmemiz gerekmektedir ancak Y boyutunu pas geçerek) düzleme aktaralım.

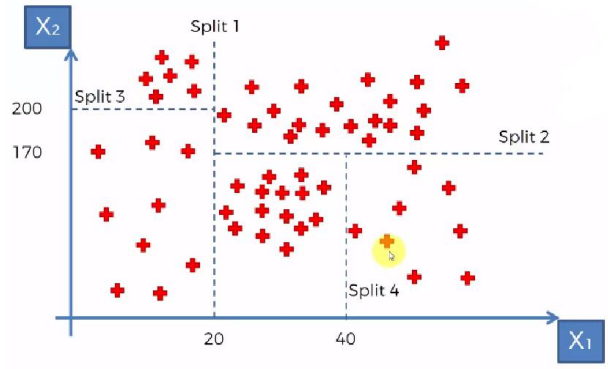
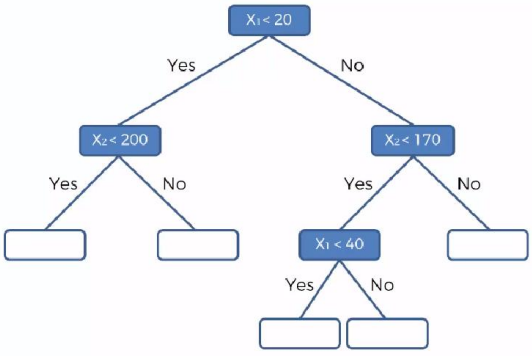
ii) Bu noktaları entropy kavramına göre en homojen bölümlere ayıralım. Başlangıç düğümündeki ve onu izleyen her alt düğümdeki bölümleme ağacımızı oluşturacaktır.

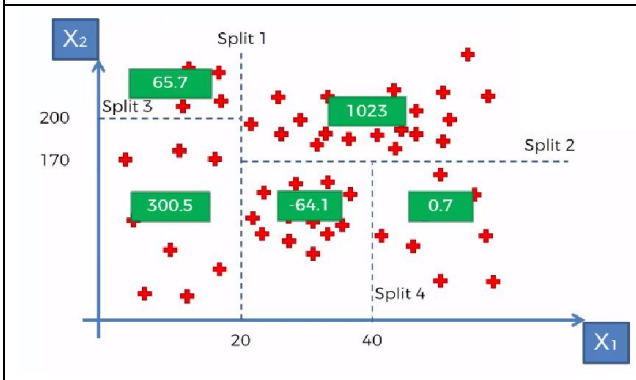
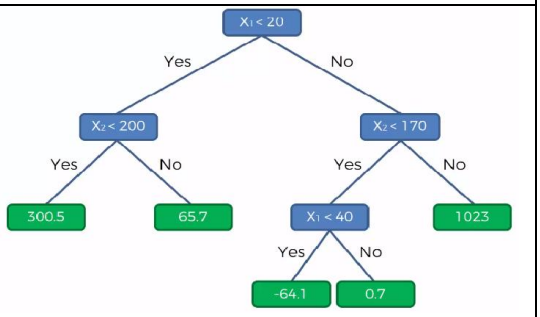
ÇOK ÖNEMLİ NOT: Neden bölme yapıyoruz?

CEVAP: Çünkü her bir bölmenin Y ortalaması bize tahmin edilecek olan Y değerini verecektir. Karar Ağacı yeni değer hangi bölmedeyse o bölme için tek bir değer tahmini yapacağından bölme sayısının artması (bir noktaya kadar!) hassas tahmini arttırır. En sonda bu tartışmayı hatırlayalım.

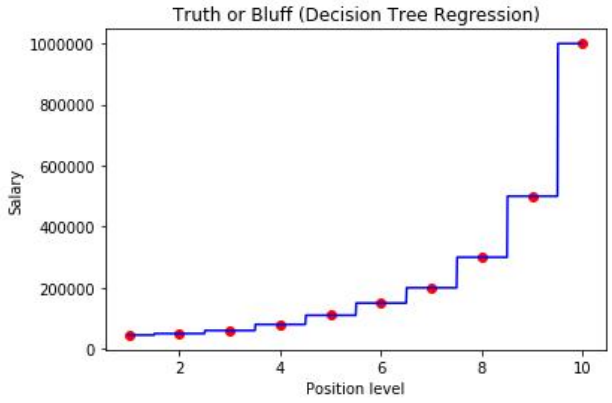


Noktalar iki bölüme ayrıldı kural ve düğümü yan tarafta görelim.	
 20 den büyük küçük olma durumunu referans alıp iki bölme daha yapalım ve bunu ağacımıza aktaralım.	 Dört bölüm için kural yazacak olsaydık: i) 20'den büyük küçük VE ii) 200 den büyük küçük VE iii) 170'ten büyük küçük

 Bölümleme işlemini maksimum performansı elde edecek şekilde bitirdiğimizi varsayarak şimdi ağacımızı çizip bu bölümler ne işe yarayacak görelim.	 SORU: Amaç neydi? CEVAP: Elimde X1=25 ve X2=100 gibi iki değer var. Buna göre Y nedir? X1 ve X2'ye göre ağacı izleyebiliriz. Bu ağacın sonundaki boş kutulara gelecek değerler tahmin edeceğimiz Y olacaktır. Bu değerler başta söylediğimiz grup ORTALAMALARI! Şimdi o formatı görelim

 Bu ortalamalarla birlikte ağacımızı çizelim.	 Buna göre yukarıdaki X1=25 ve ve X2=100 için Y değerini -64.1 elde ederiz.

NOT: Y değerini tahmin ederken, yeni bir Y değeri hangi gruba düşerse o grubun ortalama tahmin değerini alacağı için Karar Ağacı grafikleri sabit basamaklar şeklindedir. Basamak ne kadar çoğalırsa (ne kadar çok bölümleme yaparsak) hassasiyet (bir noktaya kadar/aşırı uyum riski var) o kadar artar. Aşağıda LAB'da üreteceğimiz grafik örnek olarak verilmiştir.



NOT: Karar Ağacı Regresyonun parametreleri için Ctrl+i kombinasyonunu kullanarak detayları görebiliriz.

Decision Tree Regression LAB Çalışmasını yapalım

1.5.1 Random Forest (Ensemble Decision Tree) Regression

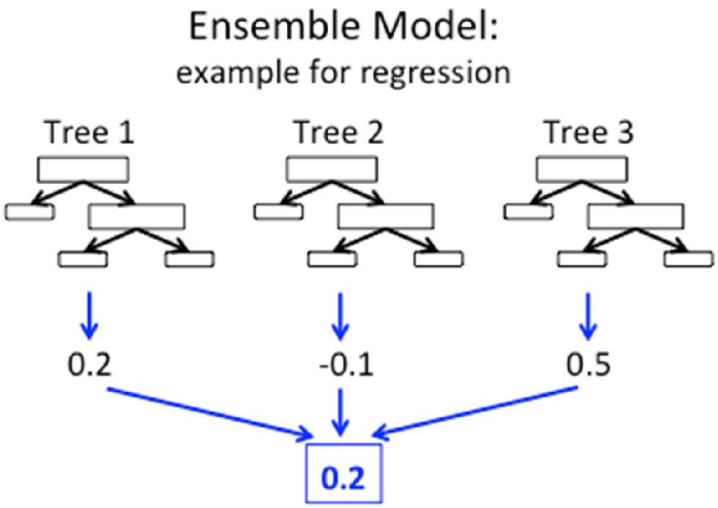
Regresyon için birden fazla karar ağacının oluşturduğu bir ensemble/topluluk öğrenmesi algoritmasıdır.

Ensemble Learning: Birden fazla algoritmanın birlikte öğrenerek tek başına tahmin yapmalarına göre daha hassas sonuçlar ürettikleri öğrenme biçimidir. Farklı parametrelere sahip aynı tip algoritma veya farklı algoritmalar bir araya gelerek veya verinin farklı bölümleri ile eğitilen algoritmaların topluluğu ile "diversity/ayrıklık" sağlanan eğitim biçimi. Daha sonra detaylı inceleyeceğiz. Bu arada önemli bir dip not olarak "karar ağacı tabanlı" Random Forests ve Gradient Boosting Trees ve çok popüler XGBoost algoritmaları en başarılı ve en çok kullanılan ML modelleri arasında yer alırlar.

Random Forests prensip olarak çok sayıda karar ağacından oluşmuştur demiştik. Peki topluluktaki -diyelim on ağacın her biri ve dolayısıyla topluluk- nasıl train (fit) ve test (predict) yapılıyor:

i) Bir "Karar Ağacı" train edilirken train için ayrılan tüm verileri kullanarak tek bir model oluşturuyor. Oysa Random Forest seçilen ağaç sayısına bağlı olarak her bir ağacı train verisinde seçtiği random verilerle (her ağaç için train verisinin tamamını kullanmıyor) eğitiyor. Dolayısıyla topluluktaki her ağaç birbirinden farklı modeller olarak eğitilmiş oluyor.

ii) Test aşamasında ise tahmin edilecek değerler ise tüm ağaçlara gönderilip her değer için üretilen değerlerin ortalaması alınıyor.



RF'nin, tek karar ağacına göre en önemli artıları:

i) Karar Ağacının veriyi aşırı öğrenmesi (overfitting ne demektir?) sorununu ortadan kaldırması (her ağaç verinin başka tarafıyla eğitiliyor aşırı öğrenme durumu olmuyor)

ii) Tahmin aşamasında daha esnek/hassas olması (birden fazla kararın ortalaması)

NOT: Eğer sınıflandırma problemi çalışılıyorsa o zaman sınıf tahmini çoğunluğun oylarıyla belirleniyor.

SORU: Parametreler?

CEVAP: Ctrl+i ile ulaşalım

RandomForestRegressor

Definition : `RandomForestRegressor(n_estimators=10, criterion="mse", max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0., max_features="auto", max_leaf_nodes=None, min_impurity_decrease=0., min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False)`

Type : Present in sklearn.ensemble.forest module

class RandomForestRegressor(ForestRegressor):

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if *bootstrap=True* (default).

Bizim için en önemli parametre n_estimators (ağaç sayısı) olsa da bu tarz algoritmalarda yine hyperparameter tuning (en iyi parametre kombinasyonunu bulmak, deep learning'de çok duyarsınız!) önemlidir.

Random Forests Regression LAB Çalışmasını yapalım

1.6 Regresyon Algoritmaları Değerlendirme Kriterleri

SORU: Aynı problemi çözmek için kullandığımız algoritmalardan birini ötekine tercih etmek için bir metrik var mı?

CEVAP: Araştıralım