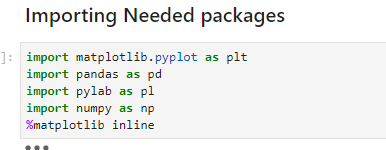
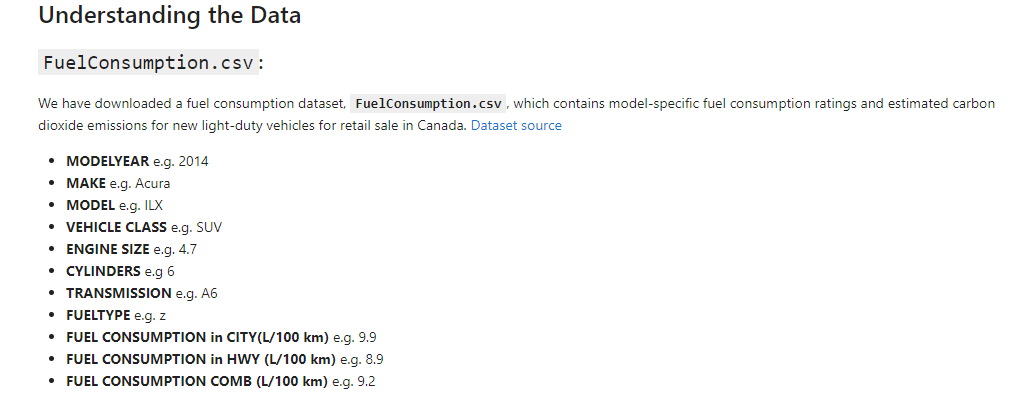
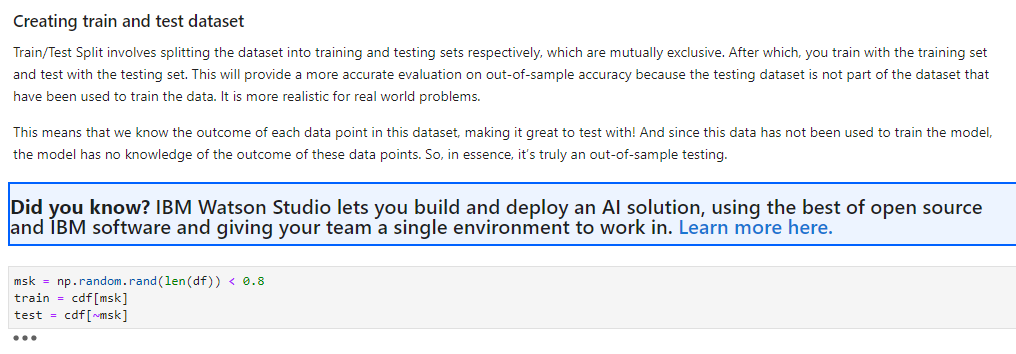
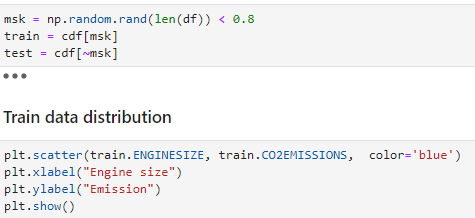
Multiple Linear Regression

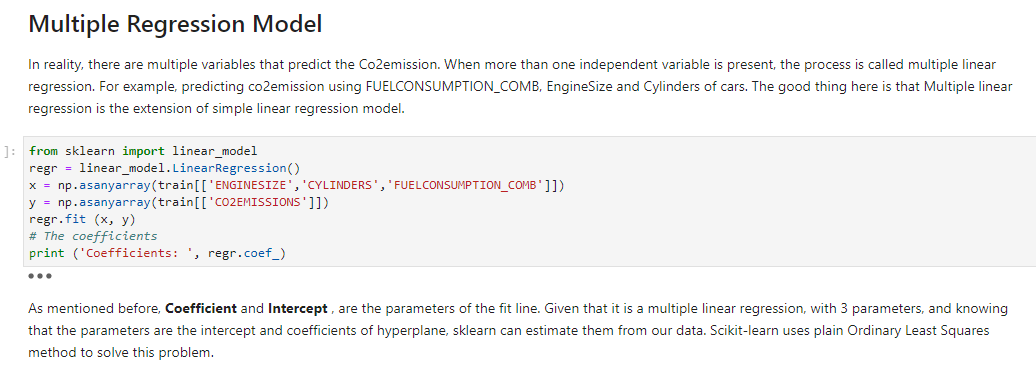








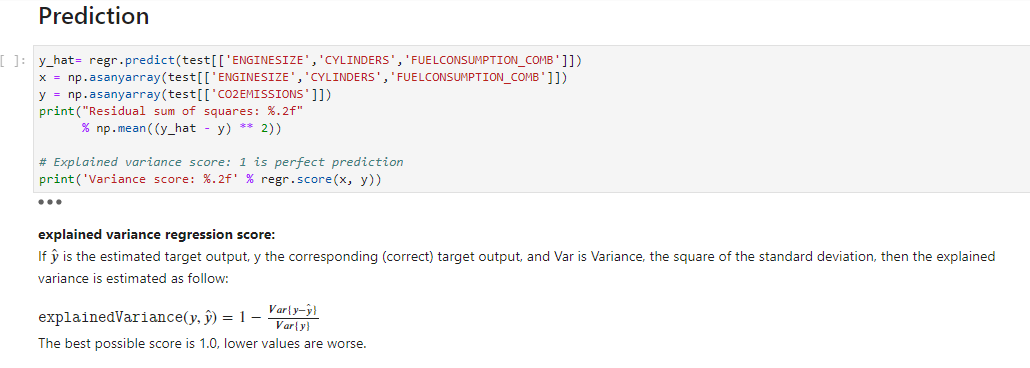


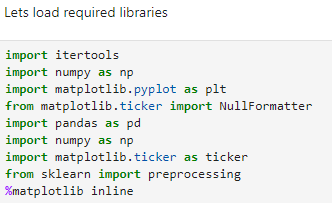


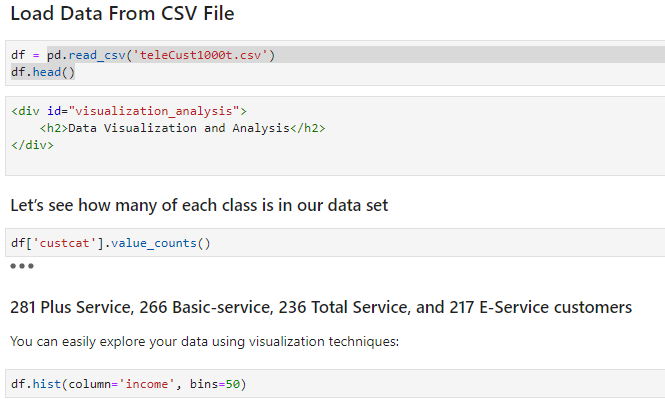
**Ordinary Least Squares (OLS)**

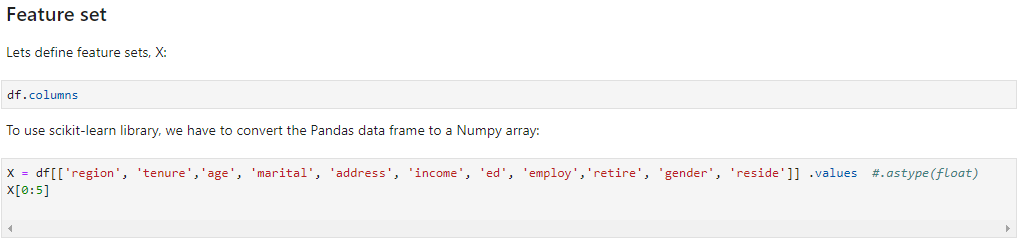
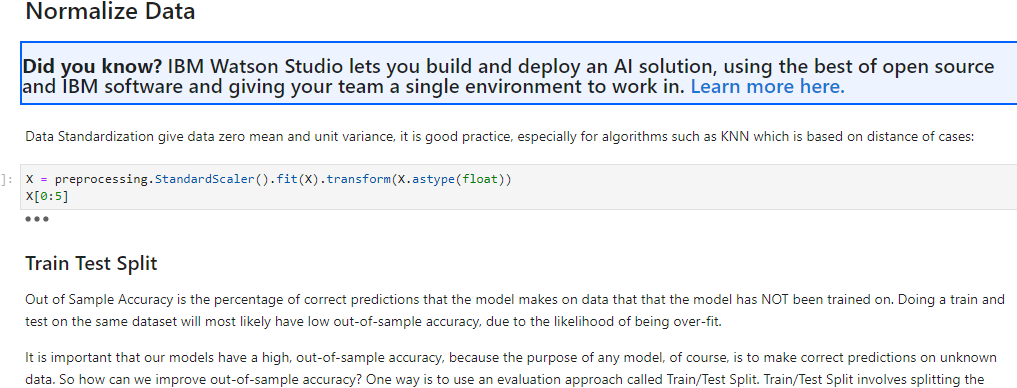
OLS is a method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by minimizing the sum of the squares of the differences between the target dependent variable and those predicted by the linear function. In other words, it tries to minimizes the sum of squared errors (SSE) or mean squared error (MSE) between the target variable (y) and our predicted output (𝑦̂ y^) over all samples in the dataset.

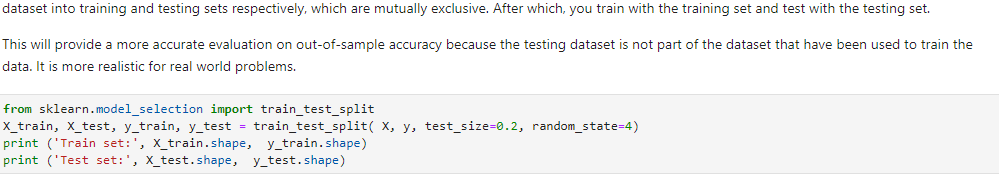
OLS can find the best parameters using of the following methods: - Solving the model parameters analytically using closed-form equations - Using an optimization algorithm (Gradient Descent, Stochastic Gradient Descent, Newton’s Method, etc.)



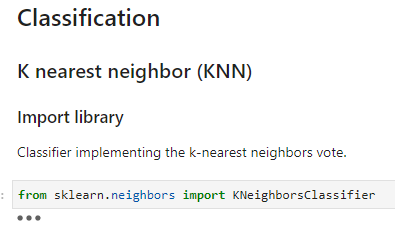


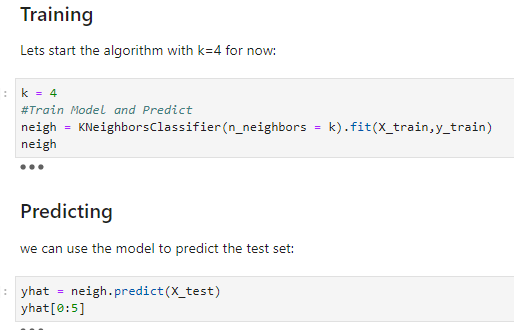


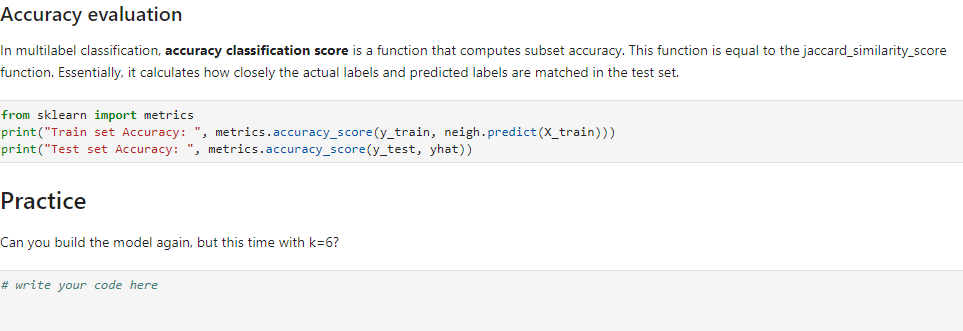
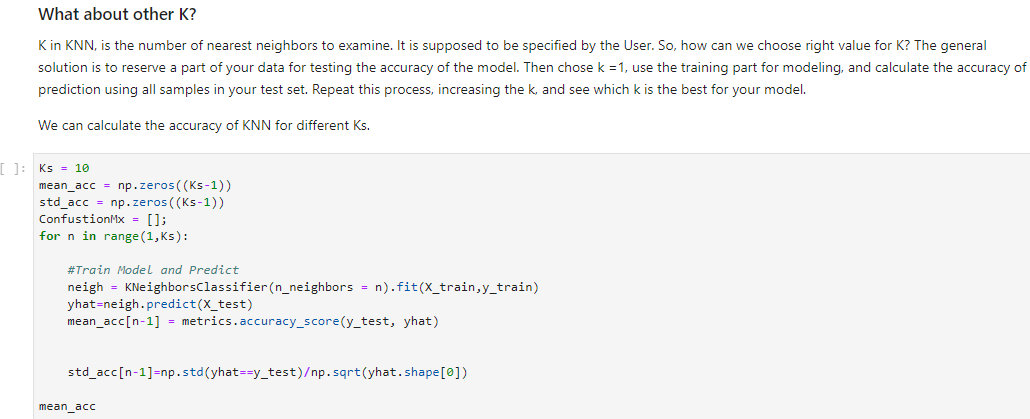


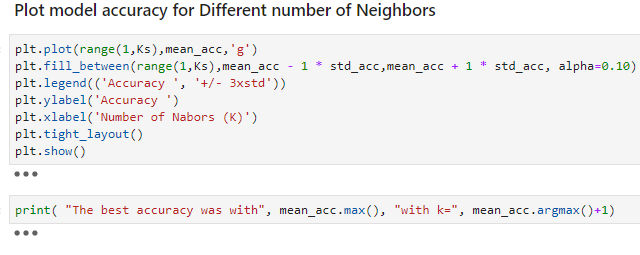


**Polinominal Regresyon:**

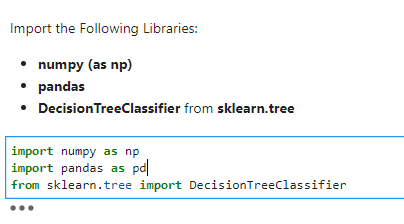








DECİSİON TREE:





Using **my\_data** as the Drug.csv data read by pandas, declare the following variables:

* **X**as the **Feature Matrix**(data of my\_data)
* **y**as the **response vector (target)**

Remove the column containing the target name since it doesn't contain numeric values.



As you may figure out, some features in this dataset are categorical such as **Sex** or **BP**. Unfortunately, Sklearn Decision Trees do not handle categorical variables. But still we can convert these features to numerical values. **pandas.get\_dummies()** Convert categorical variable into dummy/indicator variables.

, Sklearn Karar Ağaçları kategorik değişkenleri işlemez.



Burada kategorıksel olan kavramları sayısala cevırdik. Yoksa karar agacı onları yok sayar.

Şimdi ise hedef degıskenını dolduracagız.



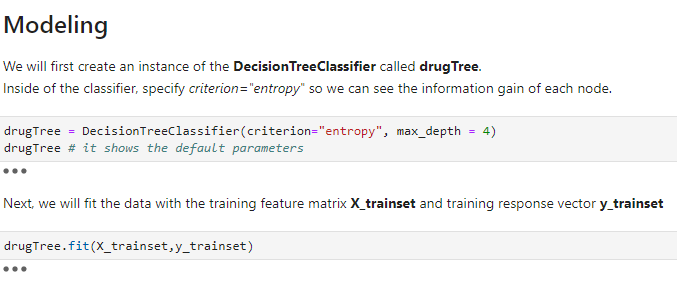
**Karar Agacı Kurulması:**



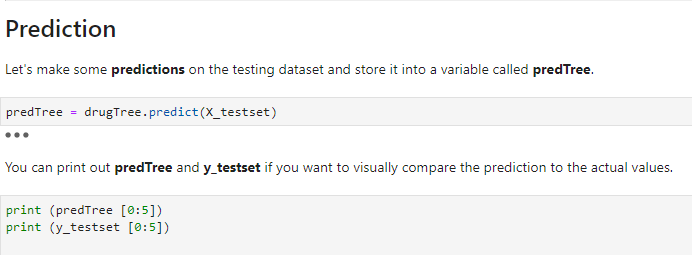
Now **train\_test\_split**will return 4 different parameters. We will name them:  
X\_trainset, X\_testset, y\_trainset, y\_testset  
  
The **train\_test\_split**will need the parameters:  
X, y, test\_size=0.3, and random\_state=3.  
  
The **X** and **y** are the arrays required before the split, the **test\_size** represents the ratio of the testing dataset, and the **random\_state** ensures that we obtain the same splits.

**Random state** sonunda karsılastırmayı yapabılmemız ıcın her defasında aynı bolunmeyı elde etmemizi saglar.



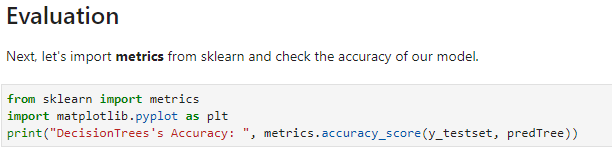


İlk olarak drugTree adlı bir DecisionTreeClassifier örneği oluşturacağız. Sınıflandırıcının içinde, her düğümün bilgi kazancını görebilmemiz için ölçüt = "entropi" belirtin.



Test veri kümesinde bazı tahminler yapalım ve şu değişkene depolayalım:

Tahmini gerçek değerlerle görsel olarak karşılaştırmak istiyorsanız **predTree** ve **y\_testset** yazdırabilirsiniz.



Ardından, sklearn'den metrikleri içe aktaralım ve modelimizin doğruluğunu kontrol edelim.

**Accuracy classification score(**Doğruluk sınıflandırma puanı) altküme doğruluğunu hesaplar: bir örnek için tahmin edilen etiket kümesi y\_true içindeki karşılık gelen etiket kümesiyle tam olarak eşleşmelidir.

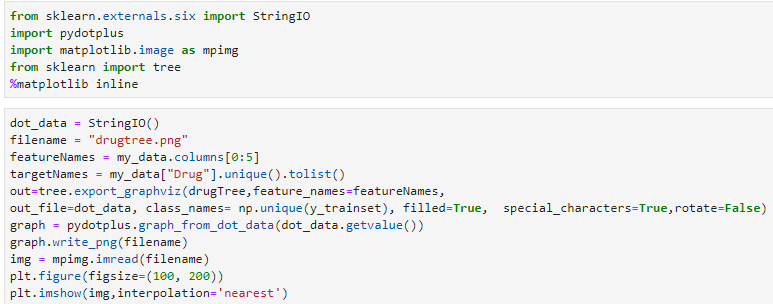
Çok etiketli sınıflandırmada, fonksiyon altküme doğruluğunu döndürür. Bir örnek için öngörülen etiketlerin tamamı gerçek etiket kümesiyle tam olarak eşleşiyorsa, alt kümenin doğruluğu 1.0 olur; aksi takdirde 0.0 olur.

**Visulation(Görüntüleme)**

**# Uyarı: Bunları daha önce yüklemediyseniz pydotplus ve graphviz kitaplıklarını açmanız ve yüklemeniz gerekebilir.**

**#! conda install -c conda-forge pydotplus -y**

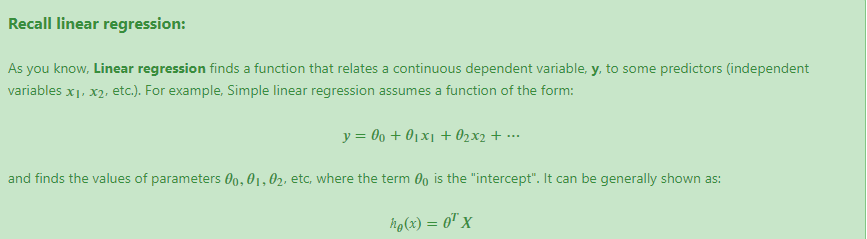
**#! conda install -c conda-forge python-graphviz -y**



**LOJISTIK REGRESYON :**

LINEAR(DOGRUSAL)-LOJISTIK FARKI:

Doğrusal Regresyon sürekli değerleri (örneğin konut fiyatını tahmin etmek) tahmin etmek için uygun olsa da, gözlemlenen bir veri noktasının sınıfını tahmin etmek için en iyi araç değildir. Bir veri noktasının sınıfını tahmin etmek için, o veri noktası için en olası sınıfın ne olacağına dair bir çeşit rehberliğe ihtiyacımız var. Bunun için Lojistik Regresyon kullanıyoruz.

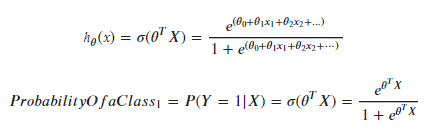


Doğrusal regresyonu hatırlayın:

Bildiğiniz gibi, Doğrusal regresyon, sürekli bağımlı bir değişkeni (y), bazı yordayıcılarla (bağımsız değişkenler 𝑥1, 𝑥2 vb.) İlişkilendiren bir işlev bulur. Örneğin, Basit doğrusal regresyon, formun bir işlevini varsayar: SEKILDEKI FORMULLERDEN YOLA CIIKABILIRIZ.

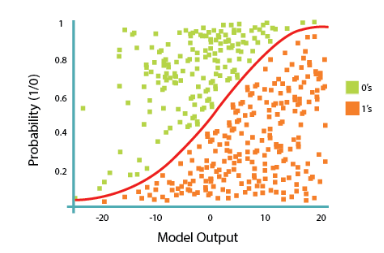
Lojistik Regresyon, gözlemlenen bağımlı değişken y, kategorik olduğunda faydalı olan Lineer Regresyonun bir varyasyonudur. Bağımsız değişkenlerin bir fonksiyonu olarak sınıf etiketinin olasılığını tahmin eden bir formül üretir.

Lojistik regresyon, lineer regresyonu alarak ve sayısal tahmini, sigmoid fonksiyon called olarak adlandırılan aşağıdaki fonksiyonla bir olasılığa dönüştürerek özel bir s-şekilli eğriye uyar:



Bu denklemde, 𝜃𝑇𝑋 regresyon sonucudur (katsayılar tarafından tartılan değişkenlerin toplamı), exp üstel fonksiyon ve 𝜎 (𝜃𝑇𝑋) lojistik eğri olarak da adlandırılan sigmoid veya lojistik fonksiyonudur. Yaygın bir "S" şeklidir (sigmoid eğrisi).

Kısaca, Lojistik Regresyon, girdiyi lojistik / sigmoidden geçirir, ancak sonucu bir olasılık olarak ele alır:



Lojistik Regresyon algoritmasının amacı, each (𝑥) = 𝜎 (𝜃𝑇𝑋) için modelin her bir vakanın sınıfını en iyi öngöreceği şekilde en iyi parameters parametrelerini bulmaktır.

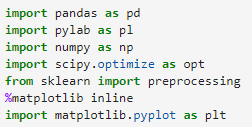
### **Customer churn with Logistic Regression:**

(müşteri Karmaşası)

Soru:

**Bir telekomünikasyon şirketi, kablo hattı rakipleri için sabit hat işlerinden ayrılan müşteri sayısından endişe duymaktadır. Kimin ayrıldığını anlamalılar. Bu şirkette analist olduğunuzu ve kimin neden ve neden ayrıldığını bulmanız gerektiğini düşünün.**

**Öncelikle gerekli kitaplıkları içe aktarmaya izin verir:**



Veri kümesi hakkında

Müşteri karmaşasını tahmin etmek için bir telekomünikasyon veri kümesi kullanacağız. Bu, her satırın bir müşteriyi temsil ettiği geçmiş bir müşteri veri kümesidir. Verileri anlamak nispeten kolaydır ve hemen kullanabileceğiniz bilgileri açığa çıkarabilirsiniz. Tipik olarak müşterileri tutmak yenilerini elde etmekten daha ucuzdur, bu nedenle bu analizin odak noktası şirkette kalacak müşterileri tahmin etmektir.

Bu veri kümesi, hangi davranışın müşterileri elde etmenize yardımcı olacağını tahmin etmenize yardımcı olacak bilgiler sağlar. İlgili tüm müşteri verilerini analiz edebilir ve odaklanmış müşteri tutma programları geliştirebilirsiniz.

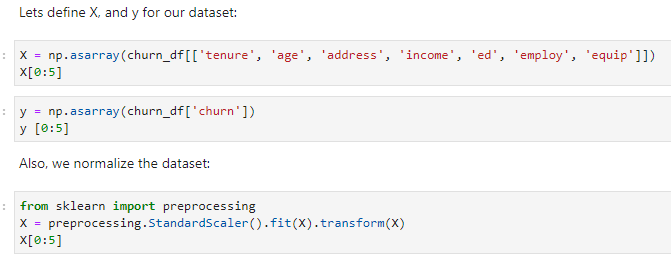
Veri kümesi aşağıdakiler hakkında bilgi içerir:

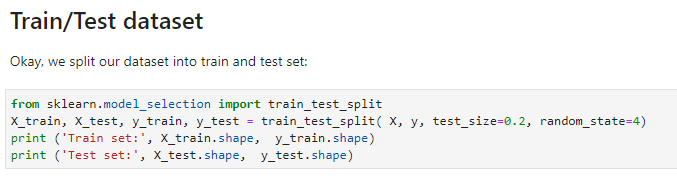
Geçen ay içinde ayrılan müşteriler - sütuna Churn denir

Her müşterinin kaydolduğu hizmetler - telefon, çoklu hatlar, internet, çevrimiçi güvenlik, çevrimiçi yedekleme, cihaz koruması, teknik destek ve TV ve film akışı

Müşteri hesabı bilgileri - ne kadar süredir müşteri, sözleşme, ödeme yöntemi, kağıtsız faturalandırma, aylık ücretler ve toplam ücretler

Müşterilerle ilgili demografik bilgiler - cinsiyet, yaş aralığı, ortakları ve bakmakla yükümlü oldukları kişiler





## **Modeling (Logistic Regression with Scikit-learn):**

Modelimizi Scikit-learn paketinden LogisticRegression kullanarak derleyelim. Bu işlev lojistik regresyonu uygular ve "newton-cg", "lbfgs", "liblinear", "sag", "saga" çözücüleri dahil olmak üzere parametreleri bulmak için farklı sayısal optimize ediciler kullanabilir. İnternette arama yaparsanız, bu optimize edicilerin artıları ve eksileri hakkında geniş bilgi bulabilirsiniz.

Scikit-learn'ta Lojistik Regresyon sürümü, düzenlileştirmeyi destekler. *Düzenleme, makine öğrenme modellerinde aşırı uydurma problemini çözmek için kullanılan bir tekniktir*. C parametresi, pozitif bir şamandıra olması gereken regülasyon kuvvetinin tersini gösterir. Daha küçük değerler daha güçlü bir düzenlenme belirtir. Şimdi modelimizi tren seti ile sığdıralım:



**Evaluation:**

jakar dizini;

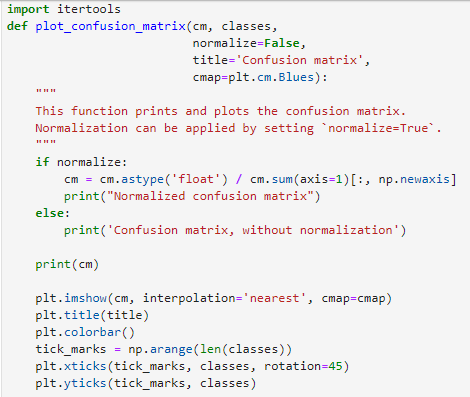
Doğruluk değerlendirmesi için jaccard indeksini deneyelim. jakarı, kavşak büyüklüğünün iki etiket kümesinin birleşim boyutuna böldüğü şekilde tanımlayabiliriz. Bir örnek için öngörülen etiketlerin tamamı gerçek etiket kümesiyle tam olarak eşleşiyorsa, alt kümenin doğruluğu 1.0 olur; aksi takdirde 0.0 olur.

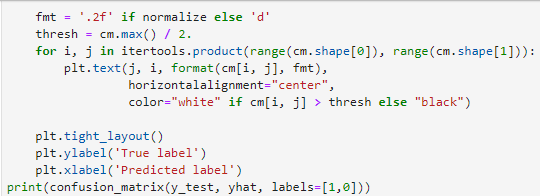


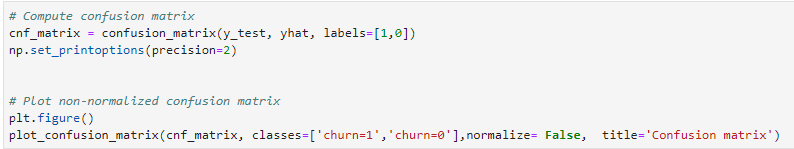
**Confusion Matrix :**

Another way of looking at accuracy of classifier is to look at **confusion matrix**.

(Sınıflandırıcının doğruluğuna bakmanın bir başka yolu da karışıklık matrisine bakmaktır.)





İlk satıra bakın. İlk satır, test setindeki gerçek çalkalama değeri 1 olan müşteriler içindir. Hesaplayabileceğiniz gibi, 40 müşteriden 15'inin çalkalama değeri 1'dir. Ve bu 15'ten sınıflandırıcı 6 tanesini doğru olarak tahmin etti. 1 ve 9 tanesi 0'dır.

Bu, 6 müşteri için gerçek çalkalama değerinin test setinde 1 olduğu ve sınıflandırıcının da bunları 1 olarak doğru şekilde tahmin ettiği anlamına gelir. Bununla birlikte, 9 müşterinin gerçek etiketi 1 iken, sınıflandırıcı 0 olarak tahmin etti, bu çok iyi değil . İlk satır için modelin hatası olarak düşünebiliriz.

Kayma değeri 0 olan müşteriler ne olacak? İkinci satıra bakalım. Kayıp değerlerinin 0 olduğu 25 müşteri olduğu anlaşılıyor.

Sınıflandırıcı 24 tanesini doğru, 0 tanesini yanlış olarak 1 olarak tahmin etti. Bu nedenle, müşterileri 0 karmaşası ile tahmin etmede iyi bir iş çıkardı. veya sınıfları ayırın. Bu örnek gibi belirli bir ikili sınıflandırıcı durumunda, bu sayıları gerçek pozitif, yanlış pozitif, gerçek negatif ve yanlış negatif sayısı olarak yorumlayabiliriz.  


Her bölümün sayısına bağlı olarak, her bir etiketin hassasiyetini ve hatırlanmasını hesaplayabiliriz:

**Hassasiyet(Precision),** bir sınıf etiketinin öngörülmesi şartıyla, doğruluğun bir ölçüsüdür. Şununla tanımlanır: duyarlık = TP / (TP + FP)

**Hatırlama(Recall)** gerçek pozitif oran. Şu şekilde tanımlanır: Hatırlama = TP / (TP + FN)

Böylece, her sınıfın hassasiyetini ve hatırlamasını hesaplayabiliriz.

**F1 puanı**: Şimdi her bir etiket için F1 puanlarını o etiketin kesinliğine ve hatırlanmasına göre hesaplayacağız.

F1 puanı, bir F1 puanının 1'de en iyi değerine (mükemmel hassasiyet ve hatırlama) 0'da en kötü değere ulaştığı hassasiyet ve geri çağırma harmonik ortalamasıdır. hatırlama ve hassasiyet.

Ve son olarak, bu sınıflandırıcı için ortalama doğruluğun her iki etiket için de F1 skorunun ortalaması olduğunu söyleyebiliriz, bu da bizim durumumuzda 0,72'dir.

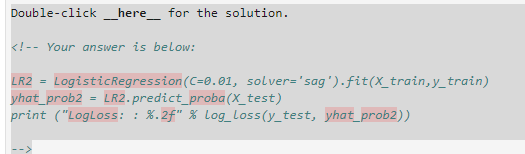
**Log loss :**

Şimdi, değerlendirme için günlük kaybını deneyelim. Lojistik regresyonda çıktı, müşteri karmaşasının evet (veya 1'e eşit olması) olasılığı olabilir. Bu olasılık 0 ile 1 arasında bir değerdir. Günlük kaybı (Logaritmik kayıp), tahmin edilen çıktının 0 ile 1 arasında bir olasılık değeri olduğu bir sınıflandırıcının performansını ölçer.



## Practice

Try to build Logistic Regression model again for the same dataset, but this time, use different \_\_solver\_\_ and \_\_regularization\_\_ values? What is new \_\_logLoss\_\_ value?

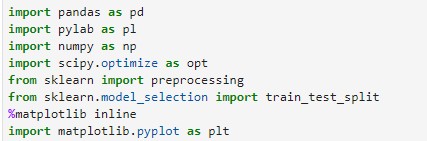


## **SVM (Support Vector Machines)**

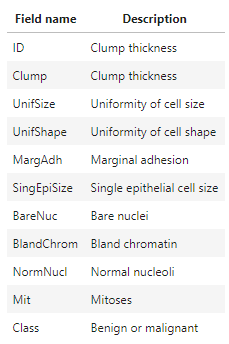
**(DESTEK VEKTOR MAKINELERI**)

Bu not defterinde, insan hücresi kayıtlarını kullanarak bir model oluşturmak ve eğitmek ve hücreleri numunelerin iyi huylu veya kötü huylu olup olmadığına göre sınıflandırmak için SVM (Destek Vektör Makineleri) kullanacaksınız.

SVM, verileri doğrusal olarak ayrılamaz olsa bile, veri noktalarının sınıflandırılabilmesi için verileri yüksek boyutlu bir özellik alanına eşleyerek çalışır. Kategoriler arasında bir ayırıcı bulunur, daha sonra veriler ayırıcı bir hiper düzlem olarak çizilebilecek şekilde dönüştürülür. Bunu takiben, yeni verilerin özellikleri, yeni bir kaydın ait olması gereken grubu tahmin etmek için kullanılabilir.



Örnek, UCI Machine Learning Depository'den (Asuncion ve Newman, 2007) [http://mlearn.ics.uci.edu/MLRepository.html] genel kullanıma açık bir veri kümesine dayanmaktadır. Veri kümesi, her biri bir dizi hücre özelliğinin değerlerini içeren birkaç yüz insan hücre örneği kaydından oluşur. Her kayıttaki alanlar:



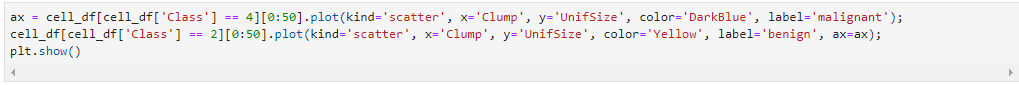
Load Data From CSV File



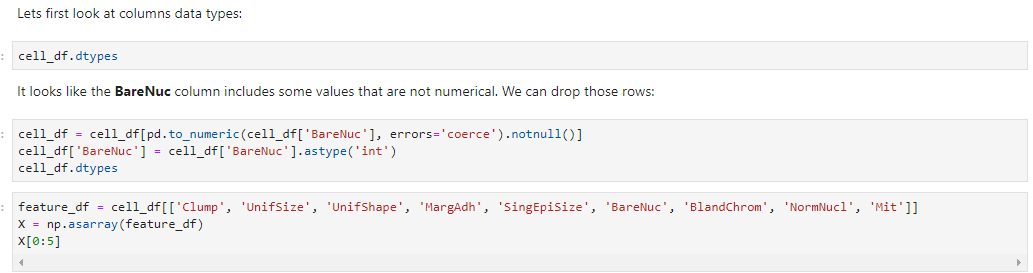
Kimlik alanı hasta tanımlayıcılarını içerir. Her hastadan alınan hücre numunelerinin özellikleri, Clump to Mit alanlarında bulunur. Değerler 1'den 10'a kadar derecelendirilir, 1'i benign'e en yakındır.

Sınıf alanı, numunelerin iyi huylu (değer = 2) veya kötü huylu (değer = 4) olup olmadığı konusunda ayrı tıbbi prosedürlerle onaylandığı gibi tanı içerir.

Sınıfların Dağılım kalınlığı ve hücre boyutunun Tekdüzeliği temelinde dağılımına bakalım:

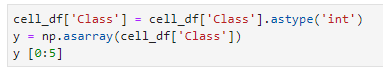


**Veri ön işleme ve seçim(Data pre-processing and selection)**

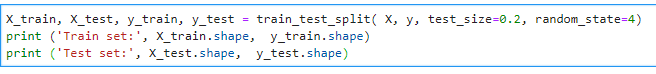


We want the model to predict the value of Class (that is, benign (=2) or malignant (=4)). As this field can have one of only two possible values, we need to change its measurement level to reflect this.

(Modelin Sınıf değerini (yani, iyi huylu (= 2) veya kötü huylu (= 4)) tahmin etmesini istiyoruz. Bu alan sadece iki olası değerden birine sahip olabileceğinden, bunu yansıtmak için ölçüm seviyesini değiştirmemiz gerekir.)



## **Train/Test dataset**



## **Modeling (SVM with Scikit-learn)**

he SVM algorithm offers a choice of kernel functions for performing its processing. Basically, mapping data into a higher dimensional space is called kernelling. The mathematical function used for the transformation is known as the kernel function, and can be of different types, such as:

1.Linear

2.Polynomial

3.Radial basis function (RBF)

4.Sigmoid

Each of these functions has its characteristics, its pros and cons, and its equation, but as there's no easy way of knowing which function performs best with any given dataset, we usually choose different functions in turn and compare the results. Let's just use the default, RBF (Radial Basis Function) for this lab.

(SVM algoritması, işlenmesini gerçekleştirmek için bir dizi çekirdek işlevi sunar. Temel olarak, verileri daha yüksek boyutlu bir alana eşleme, çekirdekleme olarak adlandırılır. Dönüşüm için kullanılan matematiksel işlev çekirdek işlevi olarak bilinir ve aşağıdaki gibi farklı türlerde olabilir:

1.Linear 2.Polynomial 3.Radyal temel işlevi (RBF) 4.Sigmoid

Bu işlevlerin her birinin özellikleri, artıları ve eksileri ve denklemleri vardır, ancak hangi işlevin herhangi bir veri kümesiyle en iyi performansı gösterdiğini bilmenin kolay bir yolu olmadığından, genellikle sırayla farklı işlevler seçer ve sonuçları karşılaştırırız. Bu laboratuvar için varsayılan RBF'yi (Radyal Temel Fonksiyon) kullanalım.)

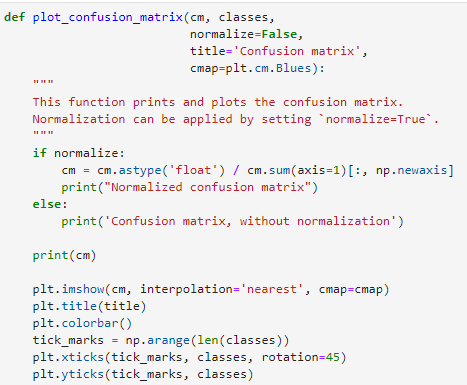


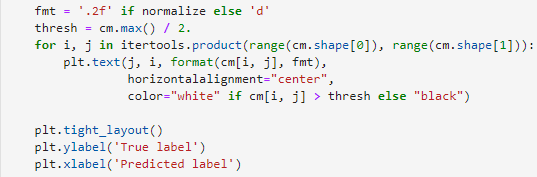
fter being fitted, the model can then be used to predict new values

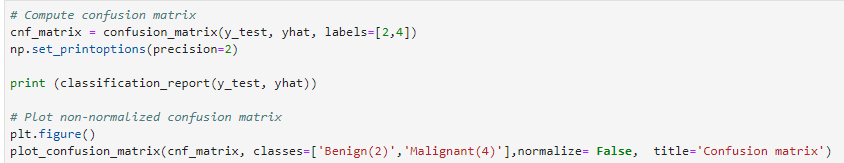


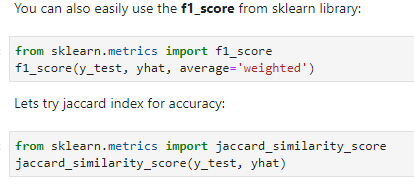
## **Evaluation**



devamı







**K-MEANS CLUSTERING**

Using k-means for customer segmentation.

K-araçlarının gerçek dünyadaki bazı uygulamaları:

Müşteri segmentasyonu

Bir web sitesini ziyaret edenlerin neyi başarmaya çalıştığını anlayın

Desen tanıma

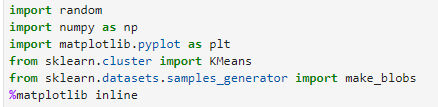
Makine öğrenme

Veri sıkıştırma

Bu not defterinde 2 örnekle k-ortalama kümelemesi uyguluyoruz:

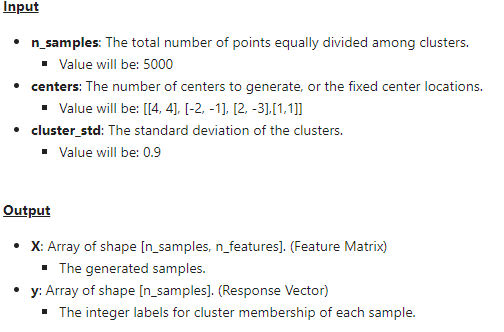
\*k-rastgele oluşturulmuş bir veri kümesinde

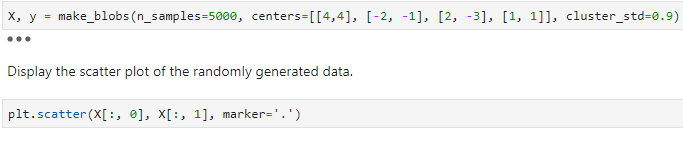
\*Müşteri segmentasyonu için k-araçlarını kullanma



# k-Means on a randomly generated dataset







\*\*\*Şimdi rastgele verilerimize sahip olduğumuza göre, K-Ortalamalar Kümelememizi ayarlayalım.

KMeans sınıfının kullanılabilecek birçok parametresi vardır, ancak bu üçünü kullanacağız:

init: Centroidlerin başlatma yöntemi.

Değer şöyle olacaktır: "k-++ anlamına gelir"

k-means ++: Yakınsamanı hızlandırmak için k-ortalama kümeleme için akıllı kümeleme merkezlerini seçer.

n\_clusters: Oluşturulacak küme sayısı ve üretilecek sentroid sayısı.

Değer: 4 olacaktır (4 merkezimiz olduğu için)

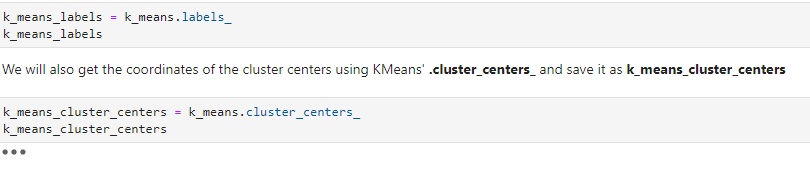
n\_init: k-ortalama algoritmasının farklı sentroid tohumlarla çalıştırılma sayısı. Nihai sonuçlar, eylemsizlik açısından n\_init ardışık çalışmaların en iyi sonucu olacaktır.

Değer olacak: 12

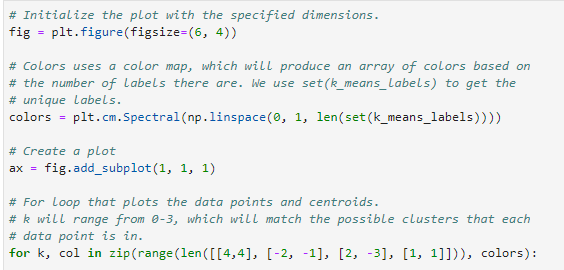


Şimdi KMeans modelini yukarıda oluşturduğumuz özellik matrisi ile sığdıralım, X;



Şimdi KMeans'in .labels\_ niteliğini kullanarak modeldeki her nokta için etiketleri alalım ve k\_means\_labels olarak kaydedelim. 

**Creating the Visual Plot**



Çizgiyi belirtilen boyutlarda başlatın.

fig = plt.figure (figsize = (6, 4))

# Renkler, aşağıdakilere dayalı bir renk dizisi oluşturacak bir renk haritası kullanır

# etiket sayısı var. Almak için set (k\_means\_labels) kullanıyoruz

# benzersiz etiket.

renkler = plt.cm Spektral (np. boşluk (0, 1, len (set (k\_means\_labels)))))

# Çizim oluştur

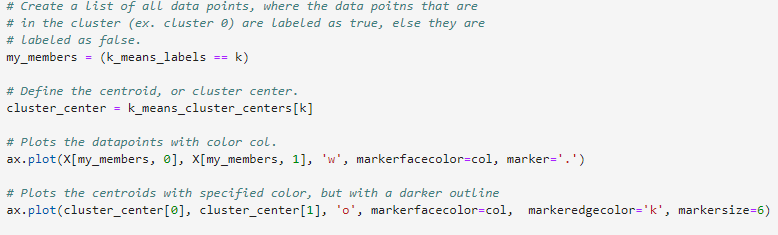
ax = fig.add\_subplot (1, 1, 1)

# Veri noktalarını ve sentroidleri çizen döngü için.

# k, 0-3 arasında değişecektir, bu da her birinin olası kümelerle eşleşecektir

# veri noktası var.

k için, zip olarak col (aralık (len ([[4,4], [-2, -1], [2, -3], [1, 1]]), renkler)):



# Tüm veri noktalarının bir listesini oluşturun.

    kümedeki # (örn. küme 0) true olarak etiketlenir, aksi takdirde küme

    # false olarak etiketlendi.

    üyelerim = (k\_means\_labels == k)

    # Merkez veya küme merkezini tanımlayın.

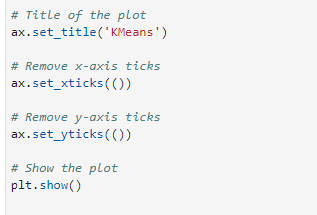
    cluster\_center = k\_means\_cluster\_centers [k]

    # Veri noktalarını renk sütunuyla çizer.

    ax.plot (X [üyelerim, 0], X [üyelerim, 1], 'w', markerfacecolor = col, marker = '.')

    # Centroidleri belirtilen renkle, ancak daha koyu bir anahatla çizer

    ax.plot (cluster\_center [0], cluster\_center [1], 'o', markerfacecolor = col, markeredgecolor = 'k', işaretçi boyutu = 6)



# Arsa başlığı

ax.set\_title ( 'kmeans')

# X ekseni kenelerini kaldırın

ax.set\_xticks (())

# Y ekseni kenelerini kaldır

ax.set\_yticks (())

# Grafiği göster

) (Plt.show

**Customer Segmentation with K-Means**

Bir müşteri veri kümeniz olduğunu ve bu geçmiş verilere müşteri segmentasyonu uygulamanız gerektiğini düşünün. Müşteri segmentasyonu, bir müşteri tabanını benzer özelliklere sahip bireyler gruplarına ayırma uygulamasıdır. Bir işletme bu belirli müşteri gruplarını hedefleyebildiği ve pazarlama kaynaklarını etkin bir şekilde tahsis edebileceği için önemli bir stratejidir. Örneğin, bir grup, yüksek kârlı ve düşük riskli, yani ürün satın alma veya bir hizmete abone olma olasılığı yüksek müşteriler içerebilir. Bir iş görevi, bu müşterileri elde tutmaktır. Başka bir grup, kar amacı gütmeyen kuruluşlardan müşteriler içerebilir. Ve bunun gibi.

Veri kümesini indirelim. Verileri indirmek için IBM Object Storage'dan indirmek için! Wget kullanacağız.



## **Pre-processing**

Gördüğünüz gibi, bu veri kümesindeki Adress kategorik bir değişkendir. k-means algoritması kategorik değişkenlere doğrudan uygulanamaz çünkü Öklid uzaklık fonksiyonu ayrık değişkenler için gerçekten anlamlı değildir. Şimdi bu özelliği bırakıp kümelemeyi çalıştıralım.,



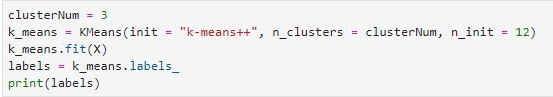
#### Normalizing over the standard deviation(Standart sapma üzerinde normalleştirme)

#### Şimdi veri kümesini normalleştirelim. Ama neden normalde normalleşmeye ihtiyacımız var? Normalleştirme, matematiksel tabanlı algoritmaların farklı büyüklük ve dağılımlara sahip özellikleri eşit olarak yorumlamasına yardımcı olan istatistiksel bir yöntemdir. Veri setimizi normalleştirmek için StandardScaler () kullanıyoruz.

#### 

## **Modeling**

Örneğimizde (k-ortalama algoritmasına erişimimiz olmasaydı), her müşteri grubunun birden fazla test ve deneyle belirli bir yaş, gelir, eğitim vb. Olacağını tahmin etmekle aynı olurdu. Ancak, K-araçları kümelemesini kullanarak tüm bu işlemleri çok daha kolay yapabiliriz.

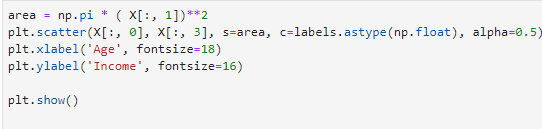


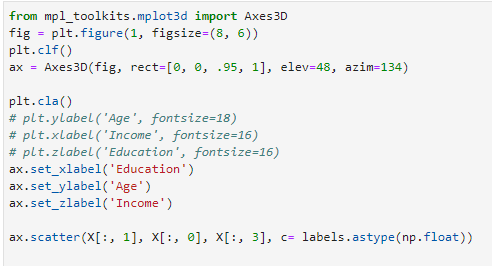
## **Insights**

Etiketleri veri çerçevesinde her satıra atarız.



Şimdi müşterilerin yaşlarına ve gelirlerine göre dağılımına bakalım





k-means müşterilerinizi birbirini dışlayan gruplara, örneğin 3 kümeye ayıracaktır. Her kümedeki müşteriler demografik olarak birbirine benzer. Şimdi her kümenin ortak özelliklerini göz önünde bulundurarak her grup için bir profil oluşturabiliriz. Örneğin, 3 küme şunlar olabilir:

ETKİLİ, EĞİTİMLİ VE ESKİ YAŞLI

ORTAÇAĞ VE ORTA GELİR

GENÇ VE DÜŞÜK GELİR