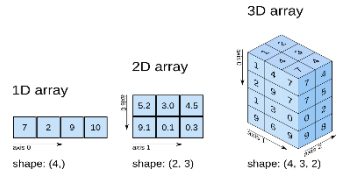
NumPy KÜTÜPHANESİ

NumPy kütüphanesi, veri manipülasyonun temel işlemlerinde kullanılan "Python" kütüphanesidir. Bu kütüphane bize çok boyutlu dizilerde yüksek performansta çalışmayı sağlar.

NumPy'ı projemizde kullanabilmek için **import** etmemiz gerekmektedir:

- `import numpy as np`

NUMPY ARRAY'İ NEDİR?



- `a = np.array([1,2,3])`
- `b = np.array([(1.5,2,3), (4,5,6)],dtype = float)`
- `c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]], dtype = float)`

NUMPY ARRAY OLUŞTURULMASI

- `np.zeros(size)=>` Tüm elemanları sıfır olan size boyutunda bir np array oluşturur.
- `np.ones(size)=>` Tüm elemanları bir size boyutunda bir np array oluşturur.
- `np.full(size,sayı)=>` Tüm elamanları "sayı" parametresi olan size boyutunda np arrayi oluşturur.
- `np.arange(aralık,artış)=>`Verilen aralığa ve artış miktarına göre numpy dizisi oluşturur.
- `np.eye(sayı)=>`Verilen sayı boyutunda birim matris oluşturur.
- `np.empty(size)=>`Verilen boyut kadar boş bir dizi oluşturur.
- `np.array([a,b,c])=>`Verilen arrayi np arrayine dönüştürür.
- `np.random.randint(size)=>`Rastgele değerlere sahip bir dizi oluştururuz.

NUMPY ARRAY ÖZELLİKLERİ

- `a.shape=>` Boyut bilgisi (a) yada (a, b) şeklinde gösterir.
- `len(a)=>` Arayin uzunluğunu verir.
- `b.ndim=>` Boyut sayısını 1-2-3 şeklinde gösterir.
- `e.size=>` Toplam eleman sayısını verir.
- `b.dtype=>` Dizinin veri tipini gösterir.
- `b.dtype.name=>` Veri türünün adını döndürür.
- `b.astype(int)=>` Bir diziyi farklı bir veri tipine dönüştürür.

NUMPY ARRAYLERİNDE ARİTMETİK İŞLEMLER

- `np.subtract(a,b) =>` Çıkarma işlemi
- `np.add(b,a) =>` Toplama işlemi
- `np.multiply(a,b) =>` Çarpma işlemi
- `np.exp(b)=>` Üs alma işlemi
- `np.sqrt(b)=>` Karekök alma işlemi
- `np.sin(a)=>` Bir dizinin sinüslerini yazdır.
- `np.cos(b)=>` Eleman eleman kosinüslerini yazdırır.
- `np.log(a)=>` eleman eleman logaritmalarını yazdırır.
- `e.dot(f)`

NUMPY ARRAYLERİNDE AGGREGATE İŞLEMLERİ

- `a.sum()=>` dizi bazında toplam
- `a.min()=>` Dizi bazında minimum değeri
- `b.max(axis=0) =>` Bir dizi satırının maksimum değeri
- `b.cumsum(axis=1) =>` Öğelerin kümülatif toplamı
- `a.mean() =>` Ortalama
- `b.median() =>` Ortanca Değer
- `a.corrcoef() =>` Korelasyon katsayısı
- `np.std(b) =>` Standart sapma

NUMPY ARRAYLERİNDE KOPYALAMA VE SIRALAMA İŞLEMLERİ

- `h = a.view() =>` Aynı verilerle dizinin bir görünümünü oluşturun
- `np.copy(a) =>` Dizinin bir kopyasını oluşturun
- `h = a.copy() =>` Dizinin derin bir kopyasını oluşturun
- `a.sort() =>` Dizileri küçükten büyüğe sıralar
- `c.sort(axis= 0)=>` Bir dizinin ekseninin öğelerini sıralama

NUMPY ARRAYLERİNDE SLICING VE FANCY İŞLEMLERİ

Bunları birkaç alt başlıkta inceleyeceğiz:

❖ Subsetting

- `a[2] =>` 2. dizindeki öğeyi seçin : 3
- `b[1,2] =>` 1. satır sütunundaki öğeyi seçin : 2 6.0 (eşiti `b[1][2]`)

❖ Slicing

- `a[0:2] =>` 0 ve 1 dizisindeki öğeleri seçin : array ([1, 2])
- `b[0:2,1] =>` 1. sütunda 0 ve 1. satırdaki öğeleri seçin :array([2., 5.])
- `b[:1] =>` 0 satırındaki tüm öğeleri seç array([[1.5, 2., 3.]]) (eşiti `b[0:1, :]`)
- c[1,...] aynıdır [1,:,:] array([[[3., 2., 1.], [4., 5., 6.]]])
- a[: :-1] bir diziyi ters çevirir : array([3, 2, 1])

❖ FANCY Indexing

- `b[[1, 0, 1, 0],[0, 1, 2, 0]]` Seçilen Elemanlar: (1,0),(0,1),(1,2) and (0,0) array([4. , 2. , 6. , 1.5])
- `b[[1, 0, 1, 0]][:,[0,1,2,0]]` => Matrisin satır dizisinin ve sütunlarının bir alt kümesini seçin

NUMPY ARRAYLERİNDE MANİPÜLASYON İŞLEMLERİ

Bunları birkaç alt başlıkta inceleyeceğiz:

❖ Transposing array

- `i = np.transpose(b) =>` Dizi boyutlarınca izin ver
- `i.T =>` Dizi boyutlarınca izin ver

❖ Changing Array Shape

- `b.ravel() =>` Diziyi düzleştir
- `g.reshape(3,-2) =>` Yeniden şekillendirin, ancak verileri değiştirmeyin

❖ Adding/Removing Elements

- `h.resize((2,6)) =>` Şekli olan yeni bir dizi döndür : (2,6)
- `np.append(h,g) =>` Diziye bir eleman ekler
- `np.insert(a, 1, 5) =>` Bir diziye öğe ekleme
- `np.delete(a,[1]) =>` Diziden eleman siler

❖ Combining Arrays

- `np.concatenate((a,d),axis=0) =>` Dizileri birleştir

array([1, 2, 3, 10, 15, 20])

- `np.vstack((a,b))` Dizileri dikey olarak yığar (satır satır)

array([[1. , 2. , 3.],

[1.5, 2. , 3.],

[4. , 5. , 6.]])

- `np.r_[e,f]` Dizileri dikey olarak yığar (satır satır)
- `np.hstack((e,f))` Dizileri yatay olarak yığar (sütun sütun)

array([[7., 7., 1., 0.],

[7., 7., 0., 1.]])

- `np.column_stack((a,d))=>` Yığılmış sütun bazında diziler oluşturun

array([[1, 10],

[2, 15],

[3, 20]])

- `np.c_[a,d]` Yığılmış sütun bazında diziler oluşturun

❖ Splitting Arrays

- `np.hsplit(a,3) =>` Diziyi 3'te yatay olarak bölün

[array([1]),array([2]),array([3])] index

- `np.vsplit(c,2)=>` Diziyi 2. dizinde dikey olarak böl

PANDAS KÜTÜPHANESİ

Pandas kütüphanesi NumPy kütüphanesinden sonra ve onun üzerine kurulmuştur. Python programalama dili için kullanımı kolay veri yapıları ve veri analizi araçları sağlamaktadır.

Pandas kütüphansini projemizde kullanabilmek için **import** etmemiz gerekmektedir.

- İmport pandas as pd

PANDAS VERİ YAPILARI

Pandas kütüphanesinde tanımamız gereken en önemli iki kavram pandas serileri ve dataframelerdir.

❖ Pandas Serileri

Herhangi bir veri türünü tutabilen “Tek Boyutlutlu” etiketli dizilerdir.

```
s = pd.Series([3, -5 , 7, 4], index=[
‘a’, ‘b’, ‘c’, ‘d’])
```

bu kodun çıktısı =>

a	3
b	-5
c	7
d	4

❖ Pandas Dataframe

Potansiyel olarak farklı türlerde sütunlara sahip iki boyutlu etiketli bir veri yapısıdır.

Columns →

Index →	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

```
Data = {‘Country’:[‘Belgium’, ‘Indıa’,
‘Brazil’], ‘Capital’ : [‘ Brussels’,
‘New Delhi’ , ‘Brasilia’], ‘Population’
:  [11190846, 1303171035, 207847528 ] }
Df = pd.DataFrame(data, columns =
[‘Coutry’, ‘Capital’, ‘Population’])
```

.DROP() VERİ SİLME FONKSİYONU

Drop(silinecek alan adı, axis) fonksiyonu kullanılmaktadır. axis parametressi alır ve “axis = 0” ise satırları, “axis = 1” ise seçilimiş sütunları siler.

```
s.drop([ ‘a’ ,  ‘c’])
df.drop( ‘Country’ , axis = 1)
```

OKUMA VE YAZMA CSV VEYA EXCELE

```
pd.read_csv(‘filename.csv’,  header  =
None, nrows= 5)
df.to_csv( ‘myDataFrame.csv)
pd.read_excel(‘file.xlsx’)
df.to_excel('dir/myDataFrame.xlsx'
'Sheet1')
```

SEÇİM İŞLEMLERİ

❖ Getting işlemleri

- Bir eleman seçmek işlemleri => `s[‘b’]`
- DataFrameden eleman seçme işlemleri => `df[1:]`

❖ Selecting, Boolean Indexing & Setting

Konumuna Göre Seçim

- `df.iloc[[0],[0]]` => Satır ve sütuna göre tek bir değer seçin
- `df.iat([0],[0])`

Etikete Göre Seçim

- `df.loc[[0], [ 'Country']]` => Satır ve sütun etiketlerine göre tek bir değer seçin
- df.at([0], ['Country'])

Hem Konuma Hem Etikete Göre Seçim

- `df.ix[2]` => tek satırlık satır alt kümesini seç
- `df.ix[:,’Capital’ ]`=> Sütunların alt kümesinin tek bir sütununu seçin.
- `df.ix[1, ’Capital ]` => Satırları ve sütunları seçiniz

İndekslerin Bool Kontrolü

- `s[~(s > 1)]` => dizide değeri -1 olmayan değerleri döndürür.
- `s[(s < -1) | (s > 2)]` =>değeri -1 den küçük veya 2 den büyük değerleri döndürür.

- `df[df[ ]>1200000000]` => DataFrame'i ayarlamak için filtreyi kullanın

Var Olan Değerleri Değiştirme

- `s['a'] = 6` => Dizi s'nin a indeksini 6'ya ayarla

SERİLER VE DATAFRAMELERİN BİLGİLERİNİ ÖĞRENME

- `df.shape` =>(satır_sayısı,sütun_sayısı)
- `df.index` => Dizini tanımlar
- `df.columns` => Dataframin sütunlarını tanımlar
- `df.info()` => Dataframe hakkında bilgi verir
- `df.count()` =>NA olmayan değerlerin sayısını verir
- `df.sum()` => Değerlerin toplamını verir
- `df.cumsum()` => Değerlerin kümülatif toplamını verir.
- `df.min()/df.max()` => Maksimum ve minimum değerleri verir.
- `df.idxmin()/df.idxmax()` => Maksimum ve minimum değerlerin indexlerini verir.
- `df.describe()` => istatistiklerin özetini gösterir.
- `df.mean()` => Ortalamasını verir.
- `df.median()` => Ortanca değerini verir.

FONKSİYON UYGULAMA

- `f = lambda x: x*2`
- `df.apply(f)` => Fonksiyonu uygular
- `df.applymap(f)` =>Fonksiyonu element element uygular.

VERİ HİZALAMA

NA değerleri, örtüşmeyen indekslere verilir. Örneğin s serimiz 4 etikete sahip bir seriydi 3 etiketli bir seri ile toplarsak sonucumuz şu şekildedir:

- `s3 = pd.Series([7, -2, 3], index=[ 'a' , 'c' ,'d'])`
- `s + s3`

a	10
b	Nan
c	5
d	3

Eğerki NaN sonucu elde etmek istemiyorsanız doldurma metotlarıyla kendinizde yapabilirsiniz. Örneğin:

- `s.add(s3, fill_values=0)`

a	10
b	-5
c	5
d	3

- `s.sub(s3, fill_value=2)`
- `s.div(s3, fill_value=4)`
- `s.mul(s3, fill_value=3)`

SCIKIT-LEARN KÜTÜPHANESİ

Scikit-learn, birleşik bir arayüz kullanarak bir dizi makine öğrenimi, ön işleme, çapraz doğrulama ve görselleştirme algoritmaları uygulayan açık kaynaklı bir Python kütüphanesidir.

Basit Bir Örneği:

```
from sklearn import neighbors, datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()
X, y = iris.data[:, :2], iris.target >>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
Scaler= preprocessing.StandardScaler().fit( X_train )
X_train = scaler.transform(X_train)
 X_test = scaler.transform(X_test)
knn = neighbors.KNeighborsClassifier( n_neighbors=5)
 knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_score(y_test, y_pred)
```

VERİLERİMİZİ YÜKLEME

Verilerimizin sayısal olması ve Numpy dizileri veya Scipy sparse matrisleri olarak saklanması gerekmektedir. Pandas dataframeleri gibide sayısal dizilere dönüştürülebilen diğer türlerde kabul edilmektedir.

```
import numpy as np
X = np.random.random((10,5))
y = np.array([ 'M', 'M' ,'F' ,'F' ,'M' ,'F' ,'M' ,'M', 'F', 'F', 'F'])
X[X < 0.7] = 0
```

EĞİTİM VE TEST DATALARI

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=0)
```

MODEL OLUŞTURMA

❖ Denetimli Öğrenme

```
lr.fit(X, y) =>
```
Modeli verilere sığdırır

```
knn.fit(X_train, y_train)
svc.fit(X_train, y_train)
```

❖ Denetimsiz Öğrenme

```
k_means.fit(X_train) =>
```
Modeli verilere sığdırır

```
pca_model = pca.fit_transform(X_train) =>
```
Modeli veriye sığdırır, sonra dönüştürür

TAHMİN

❖ Denetimli Tahminciler

```
y_pred = svc.predict(np.random.random((2,5))) =>
```
Etiketleri tahmin et

```
y_pred = lr.predict(X_test) =>
```
Etiketleri tahmin et

```
y_pred = knn.predict_proba(X_test)=>
```
Bir etiketin tahmini olasılığı

❖ Denetimsiz Tahminciler

```
y_pred = k_means.predict(X_test) =>
```
Kümeleme algoritmalarında etiketleri tahmin edin

VERİLERİ ÖN İŞLEME

❖ Standardizasyon

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
standardized_X = scaler.transform(X_train)
standardized_X_test = scaler.transform(X_test)
```

❖ Normalizasyon

```
from sklearn.preprocessing import Normalizer
scaler = Normalizer().fit(X_train)
normalized_X = scaler.transform(X_train)
normalized_X_test = scaler.transform(X_test)
```

❖ Binarizasyon

```
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=0.0).fit(X)
binary_X = binarizer.transform(X)
```

❖ Kategorik Özelliklerin Kodlanması

```
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=0.0).fit(X)
binary_X = binarizer.transform(X)
```

❖ Eksik Değerleri Atama

```
from sklearn.preprocessing import Imputer
imp = Imputer(missing_values=0, strategy= , a
xis=0)
imp.fit_transform(X_train
```

❖ Polinom Özellikleri Oluşturma

```
from sklearn.preprocessing import
PolynomialFeatures
poly = PolynomialFeatures(5)
poly.fit_transform(X)
```

MODEL OLUŞTURMA

❖ Lineer Regresyon

```
from sklearn.linear_model import Linear
Regression
lr = LinearRegression(normalize = True )
```

❖ Support Vektor Machines (SVM)

```
from sklearn.svm import SVC
svc = SVC(kernel= 'linear' )
```

❖ Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

❖ KNN

```
from sklearn import neighbors
knn =
neighbors.KNeighborsClassifier(n_neighbors=5)
```

❖ Principal Component Analysis (PCA)

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0. 95)
```

❖ K means

```
from sklearn.cluster import KMeans
k_means = KMeans(n_clusters=3, random_state=0)
```

MODELİN PERFORMANSINI DEĞERLENDİRME

❖ Sınıflandırma Metrikleri

Doğruluk Puanı

```
knn.score(X_test, y_test)=> Tahmin edici puan
metodu
from sklearn.metrics import accuracy_score =>
metrik puanlama fonksiyonu
accuracy_score(y_test, y_pred)
```

Sınıflandırma Raporu

```
from sklearn.metrics import classification_report
=> Kesinlik, geri çağırma, f1- puanı ve destek
print(classification_report(y_test, y_pred))
```

Karışıklık Matrisi

```
from sklearn.metrics import confusion _matrix
print (confusion _matrix ( y _test, y _pred))
```

❖ Regresyon Metrikleri

Ortalama Mutlak Hata

```
from sklearn.metrics import mean_absolute_error
y_true = [3, 0.5, 2]
mean_absolute_error(y_true, y_pred)
```

Ortalama Kare Hatası

```
from sklearn.metrics import mean_s quared_error
mean_s quared_error(y_test, y_pred)
```

R^2 Score

```
from sklearn.metrics import r2_score
r2_score(y_true, y_pred)
```

❖ Kümeleme Metrikleri

Düzeltilmiş Rand Endeksi

```
from sklearn.metrics import adjusted_rand_score
adjusted_rand_score(y_true, y_pred)
```

Homojenlik

```
from sklearn.metrics import homogeneity_score
homogeneity_score(y_true, y_pred)
```

V-Ölçümü

```
from sklearn.metrics import homogeneity_score
homogeneity_score(y_true, y_pred)
```

❖ Çapraz Geçerleme

```
from sklearn.cross_validation import
cross_val_score
print(cross_val_score(knn, X_train, y_train, cv=
4))
print(cross_val_score(lr, X, y, cv=2))
```