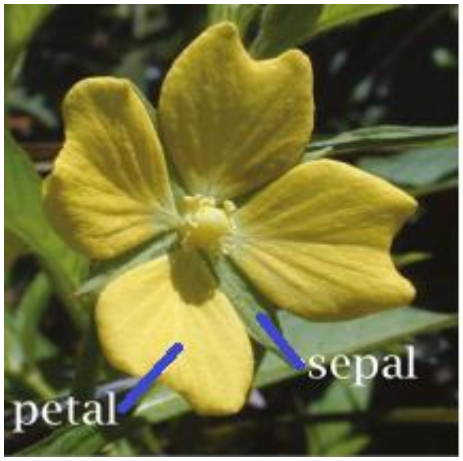
İST468. MAKİNE ÖĞRENMESİ

VERİSETİ:

İris çiçeği veri kümesi veya Fisher'ın İris veri kümesi, İngiliz istatistikçi ve biyolog Ronald Fisher tarafından tanıtılan çok değişkenli bir veri kümesidir. Veri seti, üç İris türünün (Iris setosa, Iris virginica ve Iris versicolor) her birinden 50 örnekten oluşur. Her numuneden dört özellik ölçüldü: taç ve çanak yaprakların uzunluğu ve genişliği santimetre cinsinden. Bu dört özelliğin kombinasyonuna dayanarak, Fisher türleri birbirinden ayırmak için doğrusal bir diskriminant model geliştirdi.

Bu, veri kümesinin dört sütun içerdiği anlamına gelir—sepal uzunluk, sepal genişlik, petal uzunluk ve petal genişlik. Bir taç yaprağı ve sepalın ne olduğunu merak ediyorsanız, sağdaki şekile bakablirsiniz. Daha detaylı bilgi için, https://archive.ics.uci.edu/ml/datasets/Iris adresini tıklayabilirsiniz.



Veri seti aşağıdaki gibidir.

	sepal uzunluğu	sepal genişiliği	petal uzunluğu	petal genişliği	sınıf
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

İST468. MAKİNE ÖĞRENMESİ

PROBLEM:

Yeni bir çiçeğin sepal ve petal yaprak uzunluk ve genişliklerini girerek, hangi sınıfa ait olduğunu tahmin etmektir.

KULLANILACAK MODELLER:

- Lojistik regresyon
- Karar ağaçları
- En yakın K komşuluğu
- Lineer diskriminant analizi
- Gauss Naive Bayes algoritması

Kullanılacak kütüphaneler:

```
# kütüphanelerinin sürümlerini kontrol etme
# Python sürümü
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

```
# kütüphanelerinin sürümlerini kontrol etmee

# Python sürümü
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

```
Python: 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
scipy: 1.4.1
numpy: 1.16.3
matplotlib: 3.1.3
pandas: 1.0.1
sklearn: 0.22.1
```

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Verileri doğrudan UCI makine Öğrenme deposundan yükleyebiliriz.

Verileri yüklemek için pandas kullanıyoruz. Verileri hem tanımlayıcı istatistikler hem de veri görselleştirme ile keşfetmek için de pandas kullanılacaktır. Verileri yüklerken her sütunun adlarını belirttiğimizi unutmayın. Bu, verileri incelediğimizde daha sonra yardımcı olacaktır.

```
# veri yükleme
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-uzunluğu', 'sepal-genişliği', 'petal-uzunluğu', 'petal-genişiliği', 'sınıf']
dataset = read_csv(url, names=names)
```

```
# veri boyutu
print(dataset.shape)
```

```
(150, 5)
```

```
# tanımlayıcı istatististikler
print(dataset.describe())
```

```
       sepal-uzunluğu  sepal-genişiliği  petal-uzunluğu  petal-genişiliği
count      150.000000        150.000000      150.000000        150.000000
mean         5.843333          3.054000        3.758667          1.198667
std          0.828066          0.433594        1.764420          0.763161
min          4.300000          2.000000        1.000000          0.100000
25%          5.100000          2.800000        1.600000          0.300000
50%          5.800000          3.000000        4.350000          1.300000
75%          6.400000          3.300000        5.100000          1.800000
max          7.900000          4.400000        6.900000          2.500000
```

```
# sınıf dağılımı
print(dataset.groupby('sınıf').size())
```

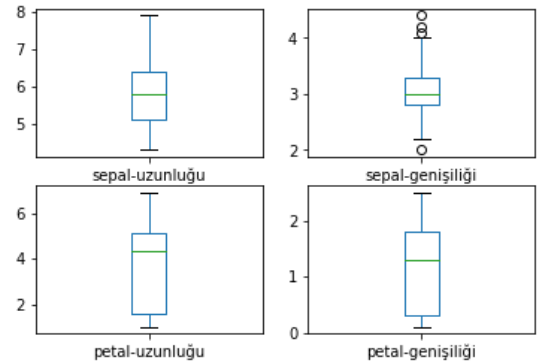
```
sınıf
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Şimdi veriler hakkında temel bir fikrimiz var. Bunu bazı görselleştirmelerle genişletmeliyiz.

İki tür grafik sözkonusudur.

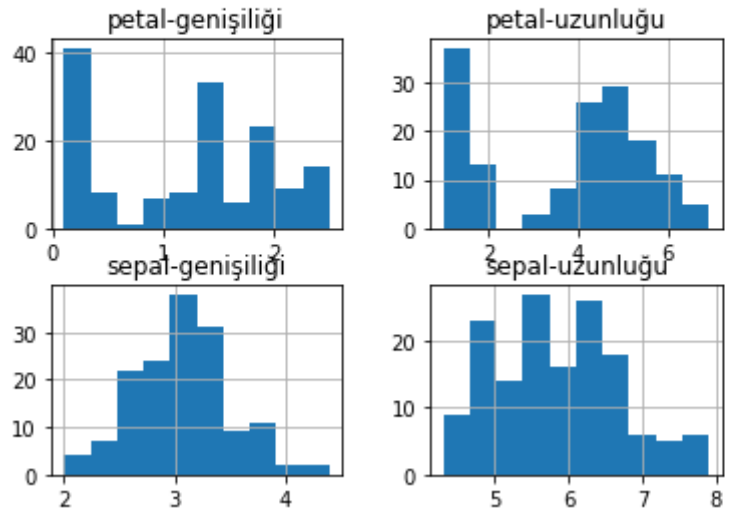
1. Her bir özelliği daha iyi anlamak için tek değişkenli grafikler.
2. Nitelikler arasındaki ilişkileri daha iyi anlamak için çok değişkenli grafikler.

```
# box and whisker grafikleri
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
pyplot.show()
```



Dağıtım hakkında bir fikir edinmek için her giriş değişkeninin histogramını da oluşturabiliriz.

```
# histogramlar
dataset.hist()
pyplot.show()
```

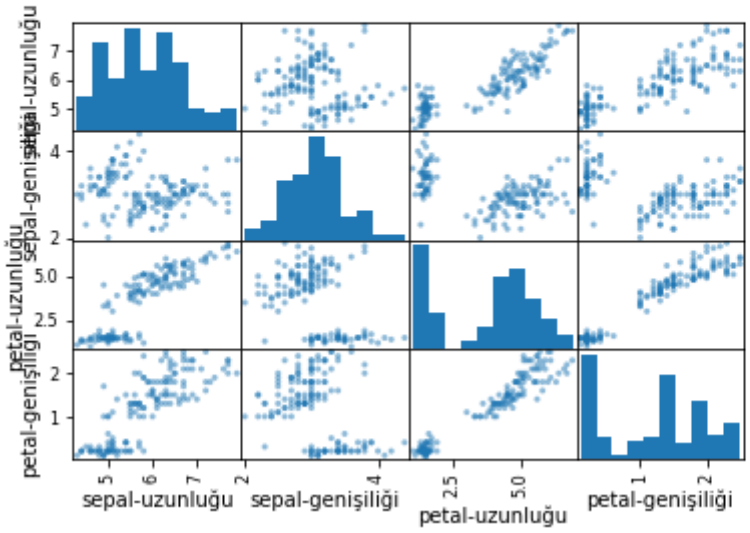


Belki de özniteliklerden ikisinin bir Normal dağılımı var gibi görünüyor. Bu varsayımdan yararlanabilecek algoritmaları kullanabileceğimiz için bunu not etmek yararlıdır.

Şimdi değişkenler arasındaki etkileşimlere bakabiliriz.

İlk olarak, Tüm özellik çiftlerinin scatterplot (serpme grafiği)'larına bakalım. Bu, öznitelikler arasındaki yapılandırılmış ilişkileri tespit etmek için yararlı olabilir.

```
# çok değişkenli serpme matrisi grafiği
scatter_matrix(dataset)
pyplot.show()
```



Bazı öznitelik çiftlerinin köşegen gruplandırmasına dikkat edin. Bu, yüksek bir korelasyon ve öngörülebilir bir ilişki olduğunu göstermektedir.

Şimdi bazı veri modelleri oluşturmanın ve görünmeyen verilerdeki doğruluğunu tahmin etmenin zamanı geldi.

İşte bu adımda ele alacağımız şey:

1. Bir doğrulama veri kümesini ayırın.
2. Test için, 10 kat çapraz doğrulama kullanacak şekilde ayarlayın.
3. Çiçek ölçümlerinden türleri tahmin etmek için birden fazla farklı model oluşturun
4. En iyi modeli seçin.

1. Oluşturulan modelin iyi olduğunu bilmemiz gerekiyor. Daha sonra, görünmeyen veriler üzerinde oluşturduğumuz modellerin doğruluğunu tahmin etmek için istatistiksel yöntemler kullanacağız. Ayrıca, görünmeyen verilerdeki en iyi modelin doğruluğunu, gerçek görünmeyen veriler üzerinde değerlendirerek daha somut bir tahmin yapmak istiyoruz.

Yani, algoritmaların göremeyeceği bazı verileri tutacağız ve bu verileri en iyi modelin gerçekte ne kadar doğru olabileceğine dair ikinci ve bağımsız bir fikir edinmek için kullanacağız.

Yüklenen veri kümesini ikiye böleceğiz, bunların %80'i modellerimiz arasında eğitmek, değerlendirmek ve seçmek için kullanacağız ve %20'si bir doğrulama veri kümesi olarak tutacağız.

```
# veri setini ayırma
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_eğitim, X_validasyon, Y_eğitim, Y_validasyon = train_test_split(X, y, test_size=0.20, random_state=1)
```

2. Model doğruluğunu tahmin etmek için tabakalı 10 kat çapraz doğrulama kullanacağız. Bu, veri kümemizi 10 parçaya bölecek, 9 parça ile veri eğitecek ve kalan 1 parça ile test edecek ve eğitim- test bölünmelerinin tüm kombinasyonları için tekrar edecektir.

Tabakalı, veri kümesinin her bir katının veya bölünmesinin, tüm eğitim veri kümesinde olduğu gibi sınıfa göre aynı örnek dağılımına sahip olmayı hedefleyeceği anlamına gelir.

3. Bu problemde hangi algoritmaların iyi olacağını veya hangi yapılandırmaların kullanılacağını bilmiyoruz. Grafiklerden, bazı sınıfların bazı boyutlarda kısmen doğrusal olarak ayrılabileceğine dair bir fikir ediniyoruz, bu yüzden Genel olarak iyi sonuçlar bekliyoruz.

Altı farklı algoritmayı test edelim:

- Lojistik regresyon (LR)
- Doğrusal Diskriminant analizi (LDA)
- K-En Yakın komşular (KNN).
- Sınıflandırma ve Regresyon Ağaçları (CART)
- Gauss naive Bayes (NB).
- Destek Vektör Makineleri (SVM)

```
# ALGORİTMALAR
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# MODEL METRİKLERİNİ HESAPLAMA
sonuçlar = []
names = []
for name, model in models:
        kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
        cv_sonuçlar = cross_val_score(model, X_eğitim, Y_eğitim, cv=kfold,
scoring='accuracy')
        sonuçlar.append(cv_sonuçlar)
        names.append(name)
        print('%s: %f (%f)' % (name, cv_sonuçlar.mean(), cv_sonuçlar.std()))
```

4. Şimdi her biri için 6 model ve doğruluk tahminimiz var. Modelleri birbirleriyle karşılaştırmalı ve en doğru olanı seçmeliyiz.
Yukarıdaki örneği çalıştırarak, aşağıdaki sonuçları elde ederiz

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

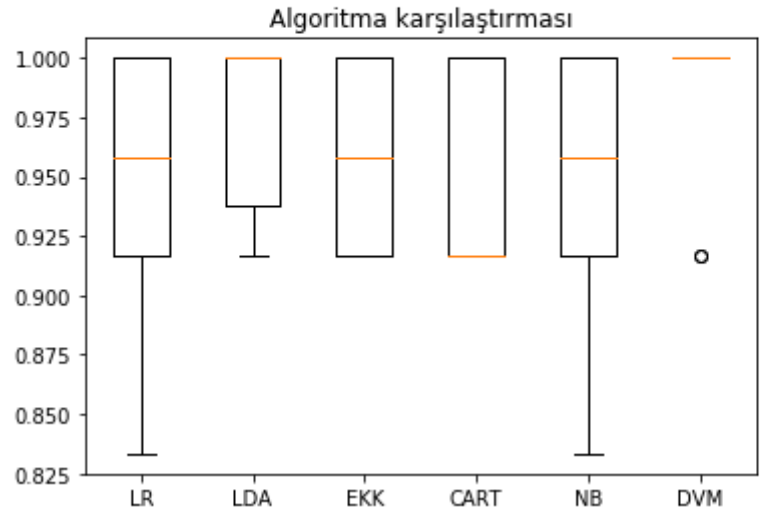
Sonuçlarınız, algoritmanın veya değerlendirme prosedürünün stokastik doğası veya sayısal doğruluktaki farklılıklar göz önüne alındığında değişebilir. Örneği birkaç kez çalıştırmayı düşünün ve ortalama sonuçları karşılaştırın.

Bu durumda, destek vektör makinelerinin (SVM) yaklaşık %0.98 veya 98'de en büyük tahmini doğruluk puanına sahip olduğunu görebiliriz.

Ayrıca, model değerlendirme sonuçlarının bir grafiğini oluşturabilir ve her modelin yayılımını ve ortalama doğruluğunu karşılaştırabiliriz. Her algoritma için bir doğruluk ölçüsü popülasyonu vardır, çünkü her algoritma 10 kez değerlendirilmiştir (10 kat çapraz doğrulama ile).

Her algoritma için sonuç örneklerini karşılaştırmanın yararlı bir yolu, her dağıtım için aşağıdaki gibi bir kutu grafiği oluşturmak ve dağılımları karşılaştırmaktır.

```
# ALGORİTMALARI KARŞILAŞTIRMA
pyplot.boxplot(sonuçlar, labels=names)
pyplot.title('Algoritma karşılaştırması')
pyplot.show()
```



Tahmin yapmak için kullanılacak bir algoritma seçmeliyiz.

Önceki bölümdeki sonuçlar, SVM'nin belki de en doğru model olduğunu göstermektedir. Bu modeli son modelimiz olarak kullanacağız.

Şimdi doğrulama setimizdeki modelin doğruluğu hakkında bir fikir edinmek istiyoruz. Bu bize en iyi modelin doğruluğu konusunda bağımsız bir son kontrol sağlayacaktır. Eğitim sırasında bir kayma yapmanız durumunda, eğitim setine aşırı öğrenme veya veri sızıntısı gibi bir doğrulama seti tutmak değerlidir. Bu sorunların her ikisi de aşırı iyimser bir sonuca yol açacaktır.

Tahminleri, doğrulama kümesindeki beklenen sonuçlarla karşılaştırarak değerlendirebilir, daha sonra sınıflandırma doğruluğunu, ayrıca bir karışıklık matrisini ve bir sınıflandırma raporunu hesaplayabiliriz.

```
# TAHMİNLER YAPMA
model = SVC(gamma='auto')
model.fit(X_eğitim, Y_eğitim)
tahminler = model.predict(X_validasyon)
```

```
# MODEL DOĞRULUĞU
print(accuracy_score(Y_validasyon, tahminler))
print(confusion_matrix(Y_validasyon, tahminler))
print(classification_report(Y_validasyon, tahminler))
```

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

Doğruluğun 0.966 veya Yaklaşık %96 olduğunu görebiliriz. Karışıklık matrisi, yapılan hataların bir göstergesini sağlar.

Son olarak, sınıflandırma raporu, hassasiyet, gerçek pozitif oranı (recall), f1 puanı ve destek değerleri için mükemmele yakın değerler sağlar.