20220510_Wirtschaftsinformatik_FAT2

May 10, 2022

[1]:
```
# Support Vecvot MAchines
```

[2]:
```
# Intuition: Lineare SVMs ziehen eine Linie zwischen zwei Klassen und bilden␣
↪eine Trennung des Raums.
```

[3]:
```
# Alle Datenpunkte auf einer Seite der Linie gehören einer Kategorie.
```

[4]:
```
# Es gibt unendliche Anzahl an Lösungen. Aber nur eine Linie hat den maximalen␣
↪Abstand zu den Punkten.
```

[5]:
```
# Linear SVM
```

[6]:
```
import matplotlib.pyplot as plt
import numpy as np
```

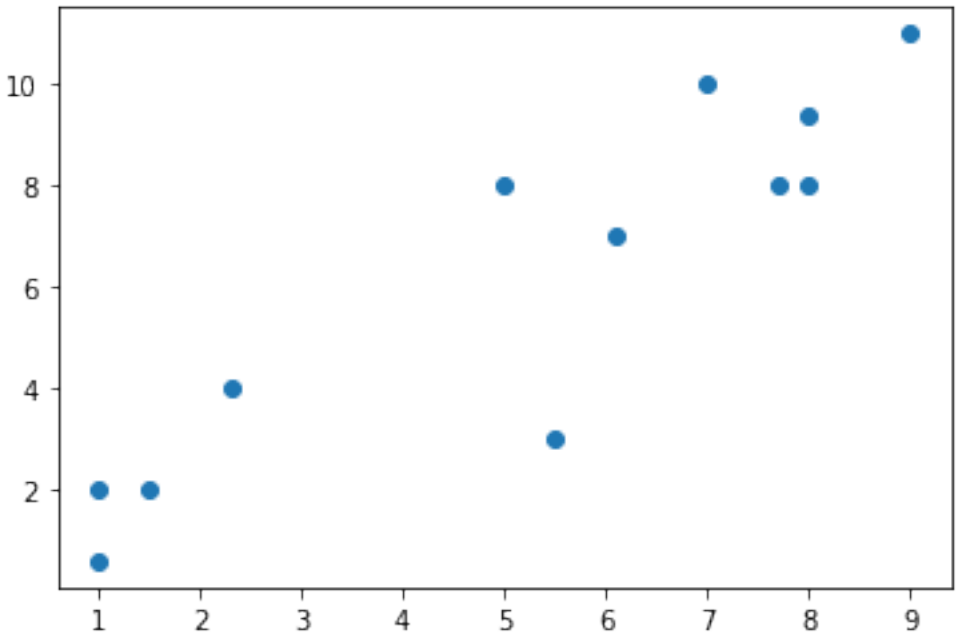
[7]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (0.0)
Requirement already satisfied: scikit-learn in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from sklearn) (1.0.1)
Requirement already satisfied: numpy>=1.14.6 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.19.5)
Requirement already satisfied: joblib>=0.11 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(3.0.0)
Requirement already satisfied: scipy>=1.1.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.7.3)
```

[8]:
```
from sklearn import svm
```

[9]:
```
# lineare daten

x = np.array([1,5,1.5,8,1,9,7,8,2.3,5.5,7.7,6.1])
y = np.array([2,8,2,8,0.6,11,10,9.4,4,3,8,7])
```

[10]:
```
plt.scatter(x,y)
plt.show()
```



[11]:
```
# Erzeugen einen 2D Array

training_points = np.vstack((x,y)).T
training_points
```

[11]:
```
array([[ 1. ,  2. ],
       [ 5. ,  8. ],
       [ 1.5,  2. ],
       [ 8. ,  8. ],
       [ 1. ,  0.6],
       [ 9. , 11. ],
       [ 7. , 10. ],
       [ 8. ,  9.4],
       [ 2.3,  4. ],
       [ 5.5,  3. ],
       [ 7.7,  8. ],
       [ 6.1,  7. ]])
```

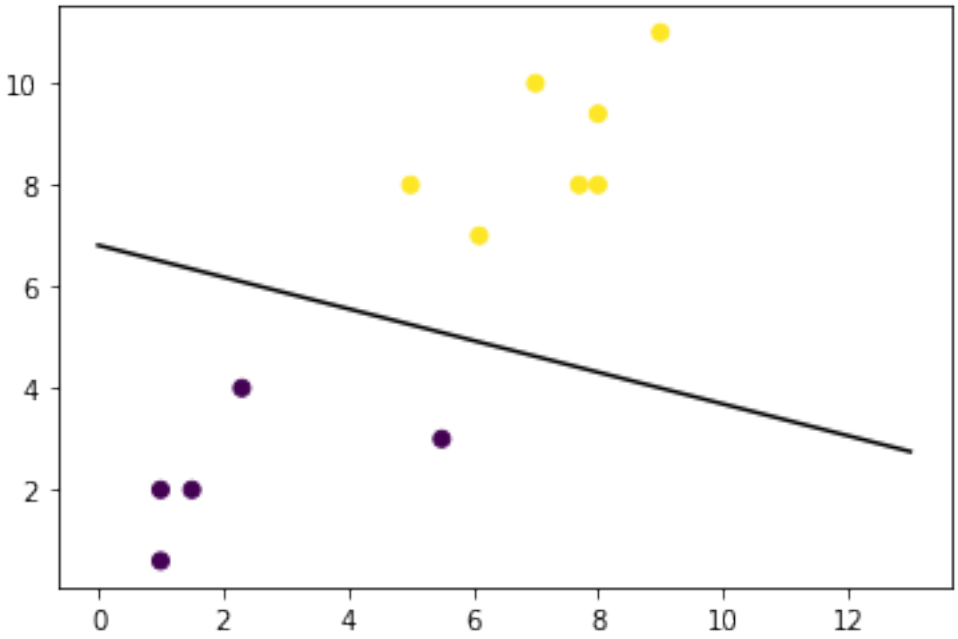
[12]:
```
# training labels

training_labels = [0,1,0,1,0,1,1,1,0,0,1,1]
```

[13]:
```
# definieren wir das Model

clf = svm.SVC(kernel='linear')
```

[14]:
```
# trainineren wir das Model

clf.fit(training_points, training_labels)
```

[14]:
```
SVC(kernel='linear')
```

[16]:
```
## nicht prüfungsrelevant

# graphische darstellung

w = clf.coef_[0]
a = -w[0]/w[1]

XX = np.linspace(0,13)
yy = a*XX-clf.intercept_[0]/w[1]

plt.plot(XX, yy, 'k-')

plt.scatter(training_points[:,0], training_points[:,1], c=training_labels)

plt.show()
```

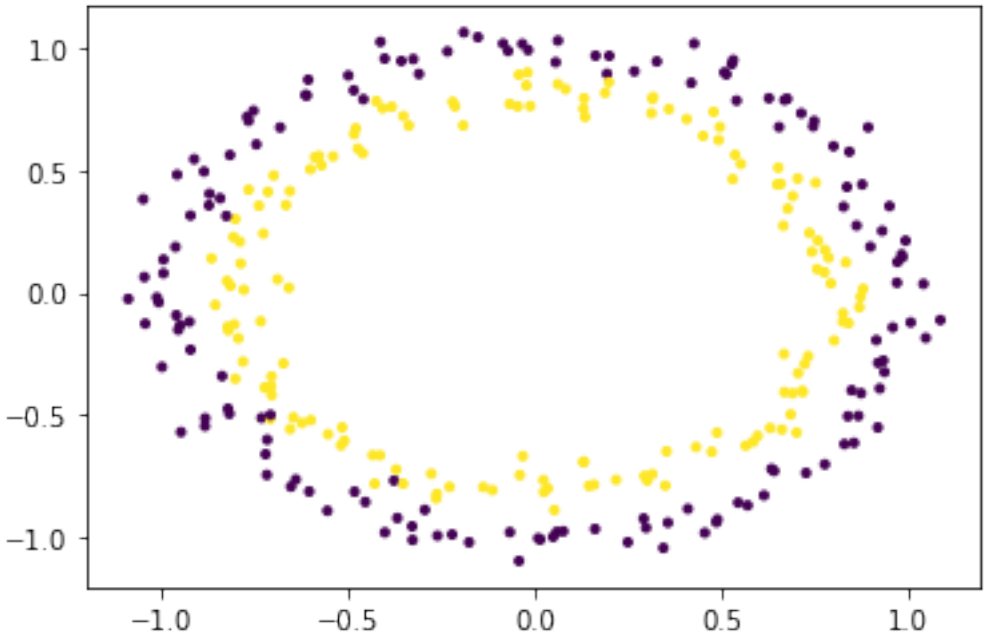


[17]:
```
# Beispiel 2. Gaussian Radial Basis Function SVM (RGB)
```

[18]:
```
from sklearn import datasets
```

[24]:
```
# non--linear data

circle_one, circle_two = datasets.make_circles(n_samples=300, noise=0.05)
```

[36]:
```
# graphische darstellung

plt.scatter(circle_one[:,0], circle_one[:,1], c=circle_two, marker='.')
plt.show()
```



[26]:
```
# nicht linearen kernel RBF

nonlinear_clf=svm.SVC(kernel='rbf')
```

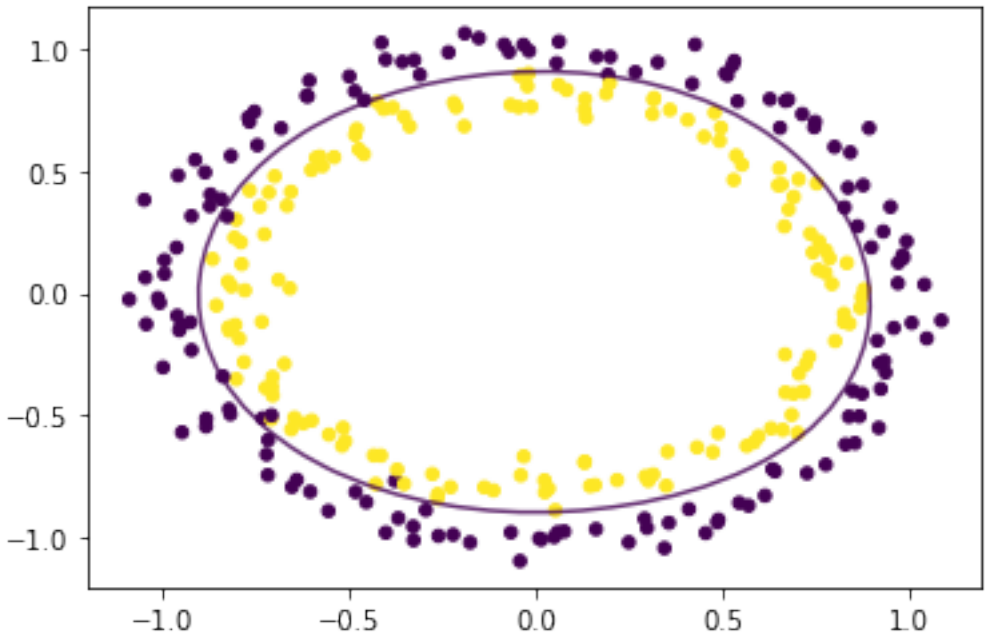
[27]:
```
# trainineren vom Modell
nonlinear_clf.fit(circle_one, circle_two)
```

[27]:
```
SVC()
```

[28]:
```
# nicht prüfungsrelevant

def plot_decision_boundary(model, ax=None):
    if ax is None:
        ax=plt.gca()
    xlim=ax.get_xlim()
    ylim=ax.get_ylim()

    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)

    Y,X =np.meshgrid(y,x)

    xy = np.vstack([X.ravel(), Y.ravel()]).T

    P = model.decision_function(xy).reshape(X.shape)
```

```
    ax.contour(X,Y,P,
               levels=[0], alpha=0.8, linestyles=['-'])
```

[31]:
```
plt.scatter(circle_one[:,0], circle_one[:,1], c=circle_two, s=20)
plot_decision_boundary(nonlinear_clf)
plt.show()
```



[37]:
```
# Beispiel Gesichtserkennung SVM
```

[39]:
```
from sklearn.datasets import fetch_lfw_people
```

[40]:
```
faces = fetch_lfw_people(min_faces_per_person=60)
```

[41]:
```
print(faces.target_names)
```

```
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony Blair']
```

[45]:
```
fig, ax = plt.subplots(3,5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],
            xlabel=faces.target_names[faces.target[i]])
```

[59]:
```
# pre processing mit PCA.
# damit wir die dimensionen reduzieren

from sklearn.svm import SVC
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.pipeline import make_pipeline

pca = RandomizedPCA(n_components=25, whiten=True, random_state=42)
svc = SVC(kernel='rbf', class_weight='balanced')
model = make_pipeline(pca,svc)
```

[60]:
```
# split der DAten in Train und Test

from sklearn.model_selection import train_test_split

Xtrain, Xtest, ytrain, ytest = train_test_split(faces.data, faces.target,␣
↪random_state=42)
```

[61]:
```
# nicht prüfungsrelevant

from sklearn.model_selection import GridSearchCV
param_grid ={'svc__C': [1,5,10,50],
             'svc__gamma':[0.0001, 0.0005,0.001, 0.005]}
grid = GridSearchCV(model, param_grid)
grid.fit(Xtrain, ytrain)
```

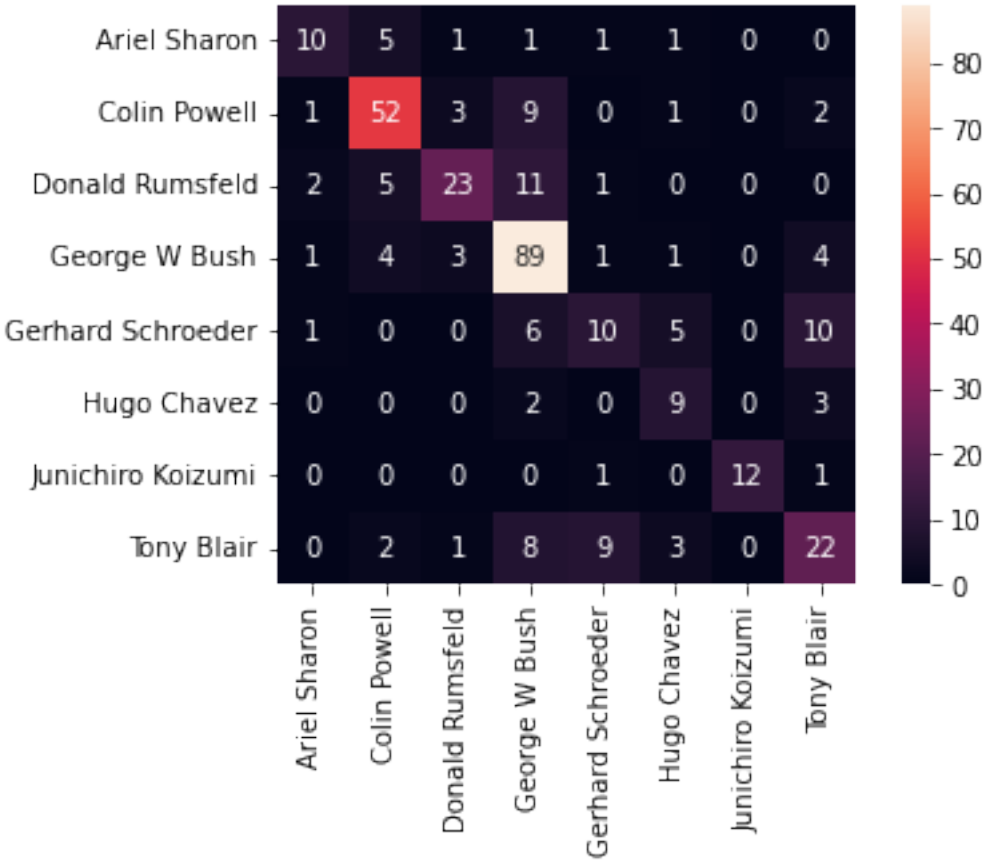
[61]:
```
GridSearchCV(estimator=Pipeline(steps=[('pca',
                                        PCA(n_components=25, random_state=42,
                                            whiten=True)),
                                       ('svc', SVC(class_weight='balanced'))]),
             param_grid={'svc__C': [1, 5, 10, 50],
                         'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]})
```

[62]:
```
model = grid.best_estimator_
yfit = model.predict(Xtest)
```

[63]:
```
from sklearn.metrics import confusion_matrix

mat = confusion_matrix(ytest, yfit)
```

[64]:
```
import seaborn as sns

sns.heatmap(mat.T, square=True, annot=True, xticklabels=faces.target_names,␣
↪yticklabels=faces.target_names)
```

[64]:
```
<AxesSubplot:>
```



[ ]: