Einfaches Beispiel für Support Vector Regression (SVR)

Vorhersage einer simplen Regression (mit Daten aus Kapitel 5).

Wir nutzen die Datenpunkte eines simplen Beispiel (2-dimensional) einer Regression aus dem Kapitel "Regression". Dies erfogt in mehreren Schritten:

1. Datenerstellung: wir nehem die Datenpunkte aus Aufgabe (5.6)-Teil2
2. SVR-Modell: Ein linearer-Kernel wird verwendet, da es sich um eine lineare Regression handelt. Die Parameter C und gamma steuern die Anpassung des Modells.
3. Vergleich der Regressionsgeraden aus Aufgabe (5.6)-Teil2 mit der Geraden die durch SVR approximiert wurde

Autor: Dr. H. Völlinger, Dezember 2024

In [1]:
```
# Import and check needed libraries

import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

# to check the time of execution, import function time
import time

# check versions of libraries
print('numpy version is: {}'.format(np.__version__))
```

```
numpy version is: 1.18.1
```

Step1: Daten bereistellen und ausplotten

In diesem zweiten Schritt nehmen wir die Daten aus Übungsaufgabe (5.6)-Teil2. Diese Daten sind gegeben als 10 Datenpunkte in dem Bereich vom 3 nach 10 in der x-Achse und (19, 48) in der y-Achse. Wir plotten diese Daten aus.

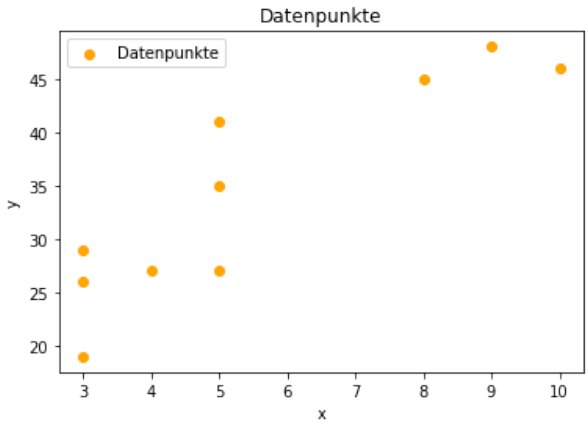
In [2]:

```python
# Daten erstellen
x = np.array([ 5, 4, 5, 3, 9, 8, 10, 5, 3, 3]).reshape((-1, 1))
y = np.array([41,27,35,26,48,45,46, 27,29,19])

# Daten ausdrucken
print ("This is how x and y look now:")
print("x=",x)
print("y=",y)

# Datenpunkte plotten
plt.scatter(x, y, color='orange', label='Datenpunkte')  # Streudiagramm
plt.xlabel('x')
plt.ylabel('y')
plt.title('Datenpunkte')
plt.legend()
plt.show()
```

```
This is how x and y look now:
x= [[ 5]
 [ 4]
 [ 5]
 [ 3]
 [ 9]
 [ 8]
 [10]
 [ 5]
 [ 3]
 [ 3]]
y= [41 27 35 26 48 45 46 27 29 19]
```



Step2: 3 SVR Modelle definieren (Typ =linear, RBF und Poly(Grad=3)) und anwenden

Setzen der SVR-Modell-Parameter: $C = 100, \gamma = auto oder 0.1$, $epsilon = 0.1$ ....

In [3]:

```python
# Fit regression model

svr_rbf = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
svr_lin = SVR(kernel="linear", C=100, gamma="auto")
svr_poly = SVR(kernel="poly", C=100, gamma="auto", degree=3, epsilon=0.1, coef0=1)

```

Ergebnisse visualisieren

Anzeige der SVR-Vorhersage und der Datenpunkte

In [4]:

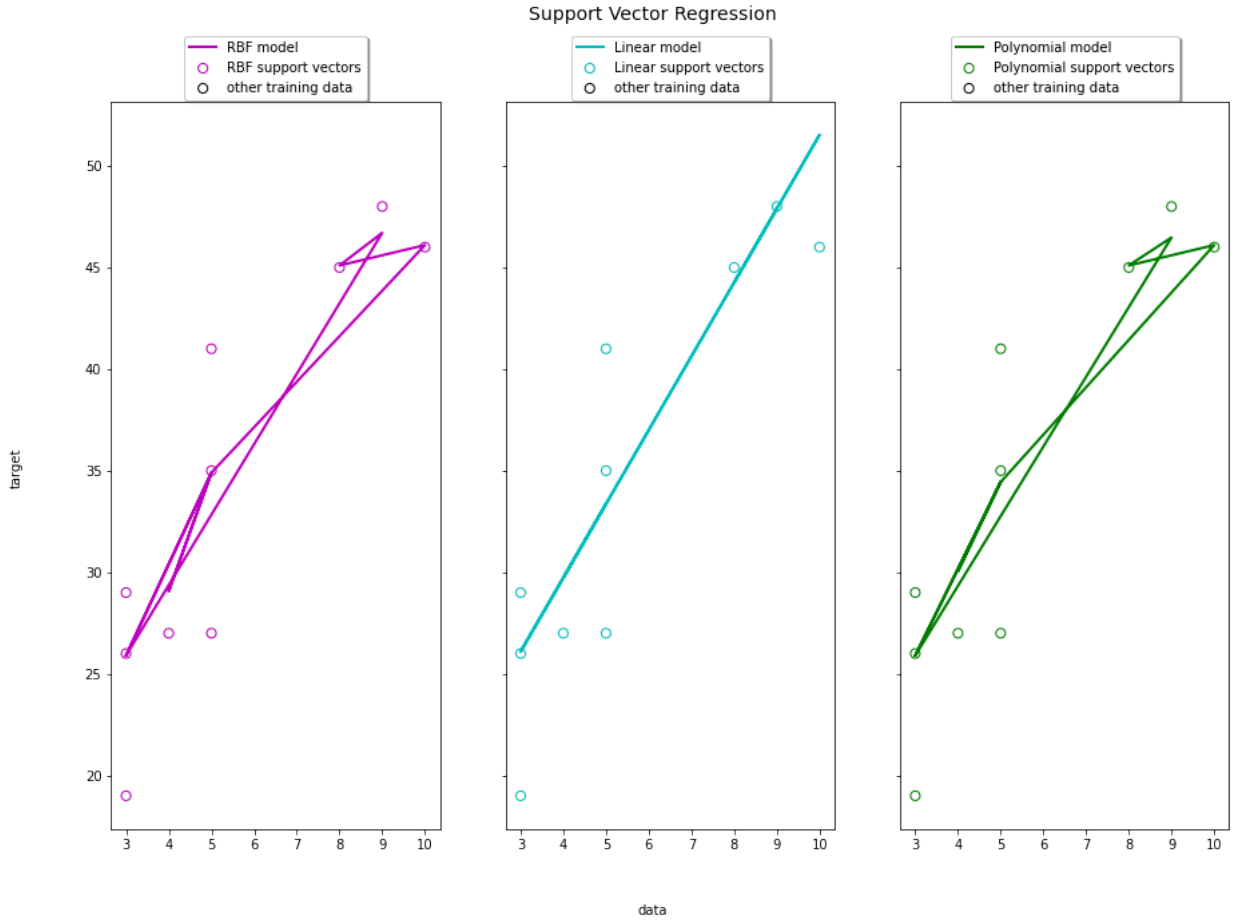
```
#Look at the results

lw = 2

svrs = [svr_rbf, svr_lin, svr_poly]
kernel_label = ["RBF", "Linear", "Polynomial"]
model_color = ["m", "c", "g"]

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 10), sharey=True)
for ix, svr in enumerate(svrs):
    axes[ix].plot(
        x,
        svr.fit(x, y).predict(x),
        color=model_color[ix],
        lw=lw,
        label="{} model".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        x[svr.support_],
        y[svr.support_],
        facecolor="none",
        edgecolor=model_color[ix],
        s=50,
        label="{} support vectors".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        x[np.setdiff1d(np.arange(len(x)), svr.support_)],
        y[np.setdiff1d(np.arange(len(x)), svr.support_)],
        facecolor="none",
        edgecolor="k",
        s=50,
        label="other training data",
    )
    axes[ix].legend(
        loc="upper center",
        bbox_to_anchor=(0.5, 1.1),
        ncol=1,
        fancybox=True,
        shadow=True,
    )

fig.text(0.5, 0.04, "data", ha="center", va="center")
fig.text(0.06, 0.5, "target", ha="center", va="center", rotation="vertical")
fig.suptitle("Support Vector Regression", fontsize=14)
plt.show()
```



Step3: Vergleich mit Ergebnissen der Übungsaufagbe (5.6)-Teil2

In [5]:
```
print('Lösungen bei Aufgabe (5.6)-Teil2 waren Steigung ~ 3,5 und Achsenabschnitt ~ 15')
print('Lösungen durch Abmessen der Datenpunkte bei linear: Steigung ~ 3,4 und Achsenabschnitt ~ 15,
print('Lösungen sind also bei beiden Verfahren durchaus vergleichbar!')
```

```
Lösungen bei Aufgabe (5.6)-Teil2 waren Steigung ~ 3,5 und Achsenabschnitt ~ 15
Lösungen durch Abmessen der Datenpunkte bei linear: Steigung ~ 3,4 und Achsenabschnitt ~ 15,2
Lösungen sind also bei beiden Verfahren durchaus vergleichbar!
```

In [6]:
```
# Print current date and time
print('******* Aktuelles Datum und Zeit*********')
print("Date & Time:",time.strftime("%d.%m.%Y  %H:%M:%S"))
# end of import test
print ("Ende Python-Programm ***SVR_LinRegression ***")
```

```
******* Aktuelles Datum und Zeit*********
Date & Time: 13.12.2024  21:58:09
Ende Python-Programm ***SVR_LinRegression ***
```