Einfaches Beispiel für Support Vector Regression (SVR)

Vorhersage einer Quadratischen Funktion y = x².

Wir generieren Daten, die einer quadratischen Funktion y = x² folgen, fügen etwas Rauschen hinzu und verwenden SVR, um die Funktion zu approximieren. Dies erfogt in mehreren Schritten:

1. Datenerstellung: Wir generieren zufällige Daten für die quadratische Funktion y = x². Diese wird mit etwas Rauschen modelliert.
2. SVR-Modell: Ein RBF-Kernel wird verwendet, um die nichtlineare Beziehung zu approximieren. Die Parameter C, gamma und epsilon steuern die Anpassung des Modells.
3. Visualisierung: Der Plot zeigt die originalen Datenpunkte und die durch SVR approximierte Kurve.

Autor: Dr. H. Völlinger, November 2024

In [73]:
```
# 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
```

Daten erstellen, Rauschen einfügen; Datenpunkte plotten

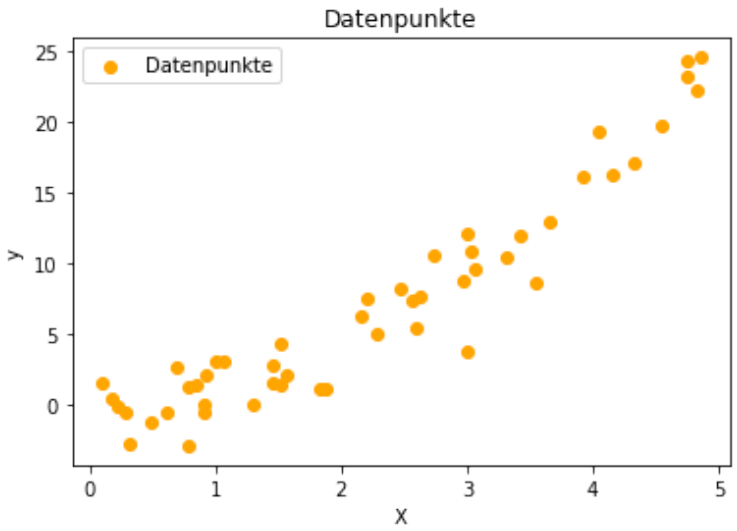
in mehreren Schritten:

1. Das Rauschen in dem oben angegebenen Python-Programm wird mit der Funktion np.random.randn() erzeugt.
2. np.random.randn(50): erzeugt 50 Stichproben aus einer Standardnormalverteilung, d.h. einer Gauß-Verteilung mit Mittelwert $\mu = 0$ und Standardabweichung $\sigma = 1$.
3. Das erzeugte Rauschen wird dann skaliert (mit * 2), um die Amplitude des Rauschens zu erhöhen, und zu den ursprünglichen Werten der quadratischen Funktion $y = x^2$ addiert: \textit{np.random.randn(50) * 2}\
4. Das erzeugte Rauschen kann sowohl positive als auch negative Werte haben, was den Daten natürliche Schwankungen hinzufügt.\

In [74]:

```python
# Daten erstellen
np.random.seed(42)
X = np.sort(5 * np.random.rand(50, 1), axis=0)  # Zufällige Werte zwischen 0 und 5
y = (X ** 2).ravel() + np.random.randn(50) * 2  # Quadratische Funktion mit Rausche

# Daten ausdrucken
print("Eingabewerte (X):")
print(X.flatten())  # Flacht die Werte ab für besseres Lesen
print("\nAusgabewerte (y):")
print(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()

```

```
Eingabewerte (X):
[0.10292247 0.17194261 0.23225206 0.29041806 0.32525796 0.48836057
 0.61019117 0.6974693  0.7799726  0.7800932  0.85262062 0.90912484
 0.91702255 0.92427228 0.99836891 1.06169555 1.29389991 1.4561457
 1.46072324 1.52121121 1.52306885 1.55855538 1.83180922 1.87270059
 2.15972509 2.20076247 2.28034992 2.47588455 2.57117219 2.60034011
 2.62378216 2.7335514  2.96207284 2.99329242 3.00557506 3.03772426
 3.05926447 3.31261142 3.42116513 3.54036289 3.65996971 3.92587981
 4.04198674 4.1622132  4.33088073 4.54660201 4.74442769 4.75357153
 4.82816017 4.84954926]

Ausgabewerte (y):
[ 1.48752620e+00  3.72300822e-01 -1.77355544e-01 -5.17864741e-01
 -2.85125124e+00 -1.20119237e+00 -5.48944273e-01  2.60070788e+00
  1.29559384e+00 -2.91753491e+00  1.37512986e+00  5.63434067e-02
 -5.12913645e-01  2.07763182e+00  3.05873953e+00  2.98975769e+00
 -4.25807452e-03  1.50193555e+00  2.79623925e+00  4.26517381e+00
  1.36139023e+00  2.05777692e+00  1.14285506e+00  1.11459427e+00
  6.28946412e+00  7.55583550e+00  5.05597552e+00  8.13707010e+00
  7.33419849e+00  5.47152916e+00  7.60702402e+00  1.05483764e+01
  8.70222346e+00  1.20890868e+01  3.79399123e+00  1.08715737e+01
  9.53319326e+00  1.03753797e+01  1.18878924e+01  8.55903156e+00
  1.29560345e+01  1.61267574e+01  1.92934449e+01  1.62874783e+01
  1.71395407e+01  1.96680758e+01  2.43403983e+01  2.32539445e+01
  2.22516102e+01  2.45446629e+01]
```



SVR Modell definieren (Typ =RBF) und anwenden

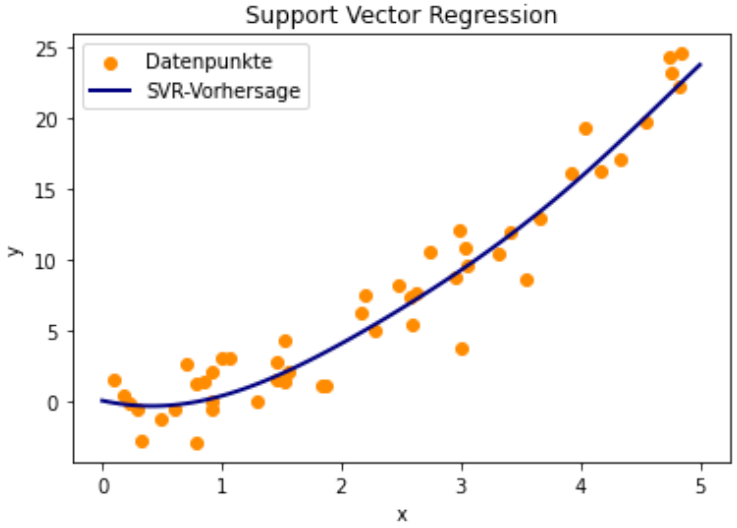
Setzen der 3 SVR-Modell-Parameter: $C = 100$, $\gamma = 0.1$ und $\epsilon = 0.1$.

In [75]:
```python
# SVR-Modell trainieren
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
svr_rbf.fit(X, y)

# Vorhersagen treffen
X_test = np.linspace(0, 5, 100).reshape(-1, 1)  # Werte zum Testen
y_pred = svr_rbf.predict(X_test)

```

Ergebnisse visualisieren

Anzeige der SVR-Vorhersage und der Datenpunkte

In [76]:
```python
# Ergebnisse visualisieren
plt.scatter(X, y, color='darkorange', label='Datenpunkte')
plt.plot(X_test, y_pred, color='navy', lw=2, label='SVR-Vorhersage')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Support Vector Regression')
plt.legend()
plt.show()
```



Suport-Vektoren anzeigen und zählen

Mit $\epsilon = 0.1$ und C=100 ist das Modell sehr empfindlich gegenüber kleinen Abweichungen, was dazu führt, dass fast alle Datenpunkte als Support-Vektoren betrachtet werden. \ Im obigem Fall sind 49 der 50 Datenpunkte auch schon Support-Vektoren.

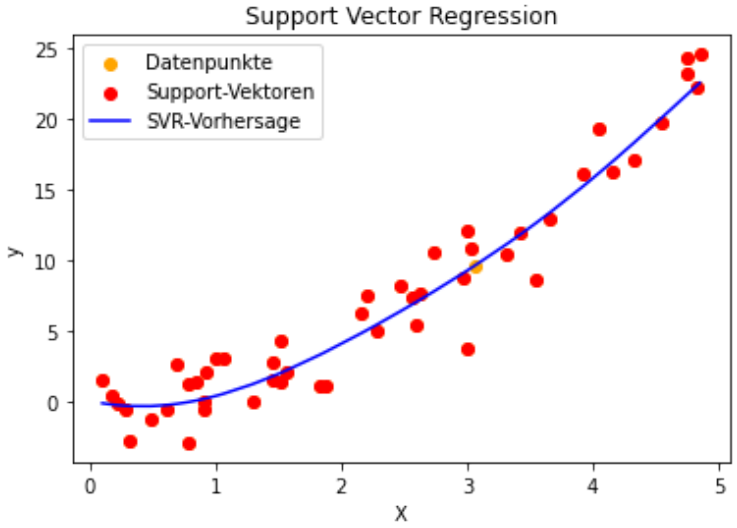
Durch Anpassen von ϵ und C kann das Verhalten des Modells reguliert werden.\

1. Enge Toleranz (ϵ=0.1): Ein größerer Toleranzbereich (z. B. ϵ=1.0) lässt mehr Punkte innerhalb der ϵ-Tubes fallen, wodurch sie keine Support-Vektoren mehr sind. Nimmt man als $\epsilon = 1.0$ so sind es lediglich 30 Datenpunkte.
2. Starker Regularisierungsterm (C=100): Der Parameter C kontrolliert, wie stark das Modell Fehler außerhalb des ϵ-Tubes bestraft wird. Ein kleinerer Wert von C (z. B. C=1) erlaubt dem Modell, größere Fehler zu tolerieren, und führt oft auch zu weniger Support-Vektoren. Konkret sind dies 47 Datenpunkte. Zu beachten ist jedoch, wenn man ϵ nicht ändert wird durch C=1 die Regression insgesamt sehr schlecht...

In [77]:

```
# Support-Vektoren anzeigen

# Anzahl der Support-Vektoren
print(f"Anzahl der Support-Vektoren: {len(svr_rbf.support_)}")

# Indizes der Support-Vektoren
print("Indizes der Support-Vektoren:")
print(svr_rbf.support_)

# print("Support-Vektoren:")
# print(svr_rbf.support_)

# Plot mit Support-Vektoren
plt.scatter(X, y, color='orange', label='Datenpunkte')  # Alle Datenpunkte
plt.scatter(X[svr_rbf.support_], y[svr_rbf.support_], color='red', label='Support-V
plt.plot(X, svr_rbf.predict(X), color='blue', label='SVR-Vorhersage')  # SVR-Funkti
plt.xlabel('X')
plt.ylabel('y')
plt.title('Support Vector Regression')
plt.legend()
plt.show()
```

```
Anzahl der Support-Vektoren: 49
Indizes der Support-Vektoren:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 37 38 39 40 41 42 43 44 45 46 47 48
 49]
```



In [78]:

```
# 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_SimExample ***")

```

```
******* Aktuelles Datum und Zeit*********
Date & Time: 27.11.2024  18:33:21
Ende Python-Programm ***SVR_SimExample ***
```