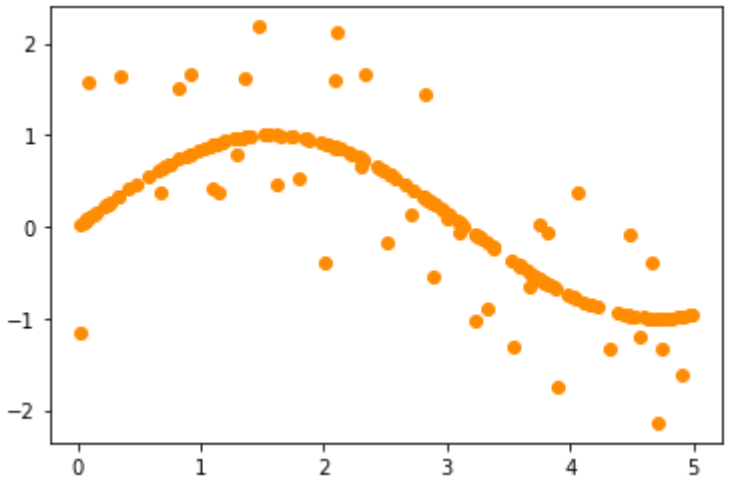
SVM para Regresión

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [11]:
```
X = np.sort(5*np.random.rand(200,1), axis=0)
Y = np.sin(X).ravel()
Y[::5] += 3*(0.5-np.random.rand(40)) # ::5 sginifica que de 5 en 5 va a crear un número aleat
```

In [12]:
```
plt.scatter(X,Y, color = "darkorange", label = "data")
```

Out[12]:
```
<matplotlib.collections.PathCollection at 0x278fa573ac0>
```



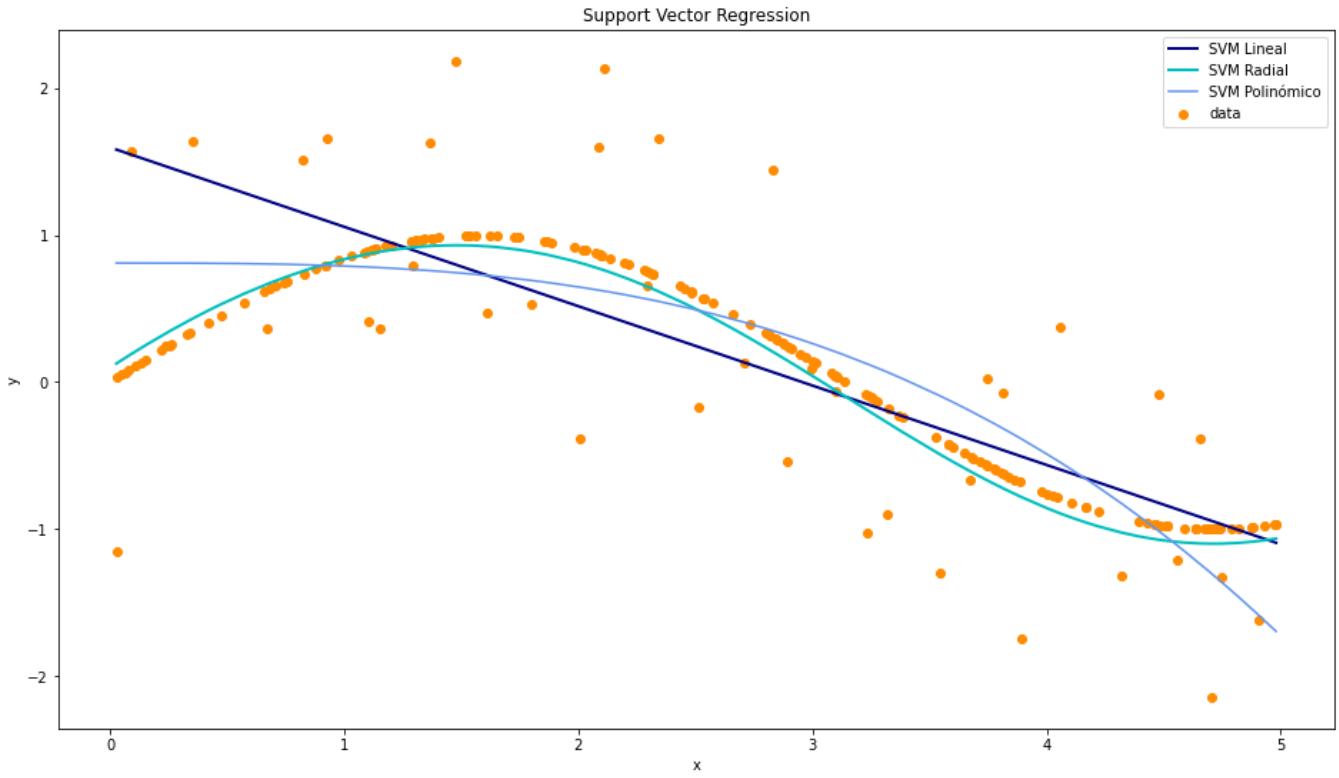
In [15]:
```
from sklearn.svm import SVR
```

In [16]:
```
C=1e3
svr_lin = SVR(kernel="linear", C=C)
svr_rbf = SVR(kernel="rbf", C=C, gamma = 0.1)
svr_pol = SVR(kernel="poly", C=C, degree=3)
```

Lo ideal sería hacer validación cruzada, obtener óptimamente los valores, etc. Pero simplemente vamos a ver cómo se emplean los SVR para cada tipo de kernel.

In [17]:
```
y_lin = svr_lin.fit(X,Y).predict(X)
y_rbf = svr_rbf.fit(X,Y).predict(X)
y_pol = svr_pol.fit(X,Y).predict(X)
```

In [18]:
```
lw = 2
plt.figure(figsize=(16,9))
plt.scatter(X,Y, color="darkorange", label = "data")
plt.plot(X, y_lin, color="navy", lw=lw, label="SVM Lineal")
plt.plot(X, y_rbf, color="c", lw=lw, label = "SVM Radial")
plt.plot(X, y_pol, color="cornflowerblue", label="SVM Polinómico")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Support Vector Regression")
plt.legend()
plt.show()
```


Support Vector Regression
SVM Lineal
SVM Radial
SVM Polinómico
data
x
y

In [ ]: