In [18]:
```
!pip install numpy sklearn matplotlib
```

```
Requirement already satisfied: numpy in d:\anaconda\lib\site-packages (1.20.1)
Requirement already satisfied: sklearn in d:\anaconda\lib\site-packages (0.0)
Requirement already satisfied: matplotlib in d:\anaconda\lib\site-packages (3.
3.4)
Requirement already satisfied: python-dateutil>=2.1 in d:\anaconda\lib\site-pac
kages (from matplotlib) (2.8.1)
Requirement already satisfied: cycler>=0.10 in d:\anaconda\lib\site-packages (f
rom matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in d:\anaconda\lib\site-packages
(from matplotlib) (8.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\anaconda\lib\site-packag
es (from matplotlib) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in d:\a
naconda\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six in d:\anaconda\lib\site-packages (from cycle
r>=0.10->matplotlib) (1.15.0)
Requirement already satisfied: scikit-learn in d:\anaconda\lib\site-packages (f
rom sklearn) (0.24.1)
Requirement already satisfied: scipy>=0.19.1 in d:\anaconda\lib\site-packages
(from scikit-learn->sklearn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in d:\anaconda\lib\site-packages (f
rom scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda\lib\site-pac
kages (from scikit-learn->sklearn) (2.1.0)
```

Decision Tree Regression

In [151]:
```
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
```

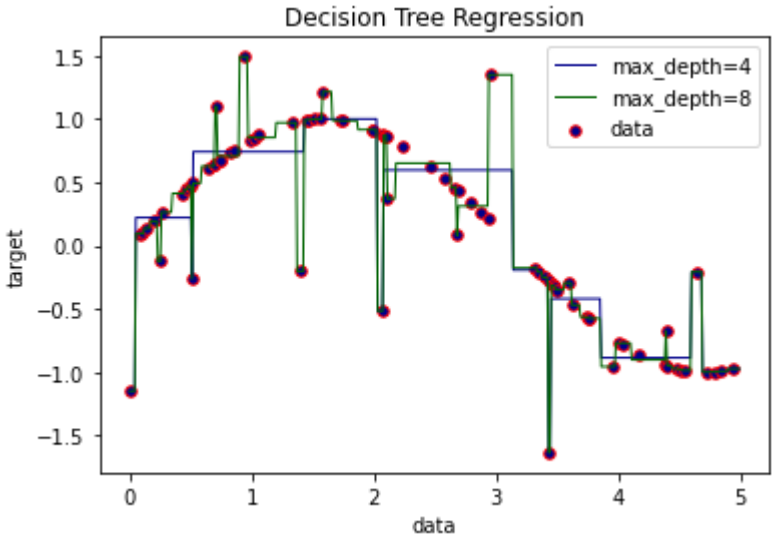
In [152]:
```
rng = np.random.RandomState(1)
x = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))
```

In [153]:
```
regr_1 = DecisionTreeRegressor(max_depth=4)
regr_2 = DecisionTreeRegressor(max_depth=8)
regr_1.fit(x, y)
regr_2.fit(x, y)
```

Out[153]:
```
DecisionTreeRegressor(max_depth=8)
```

In [154]:
```
x_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(x_test)
y_2 = regr_2.predict(x_test)
```

In [155]:
```
plt.figure()
plt.scatter(x, y, s=30, edgecolor="red", c="darkblue", label="data")
plt.plot(x_test, y_1, color="darkblue", label="max_depth=4", linewidth=1)
plt.plot(x_test, y_2, color="darkgreen", label="max_depth=8", linewidth=1)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```



Random Forest Regression

In [200]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
```

In [187]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

In [188]:
```
rf.fit(x_train, y_train)
```

Out[188]:
```
RandomForestRegressor()
```

In [189]:
```
rf.score(x_test, y_test)
```

Out[189]:
```
0.7637865252480194
```

In [190]:
```
rf.score(x_train, y_train)
```

Out[190]:
```
0.9467034906750891
```

In [214]:
```
from sklearn.model_selection import GridSearchCV
params = {'n_estimators':[50,70,90,110,140]}
grd_model = GridSearchCV(rf, params)
```

In [215]:
```
grd_model.fit(x_train, y_train)
```

Out[215]:
```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'n_estimators': [50, 70, 90, 110, 140]})
```

In [216]:
```
grd_model.best_score_
```

Out[216]:
```
0.5752068687109625
```

In [217]:
```
best_model = grd_model.best_estimator_
```

In [218]:
```
best_model.score(x_test, y_test)
```

Out[218]:
```
0.7824575115547285
```

In [213]:
```
best_model.score(x_train, y_train)
```

Out[213]:
```
0.943000122891788
```

SVM Regressor

In [221]:
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn import svm
```

In [226]:
```
x, y = make_classification(n_samples=15, random_state=5)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=5)
clf = svm.SVC(kernel='precomputed')
```

In [227]:
```
gram_train = np.dot(x_train, x_train.T)
clf.fit(gram_train, y_train)
```

Out[227]:
```
SVC(kernel='precomputed')
```

In [228]:
```
gram_test = np.dot(x_test, x_train.T)
clf.predict(gram_test)
```

Out[228]:
```
array([0, 0, 0, 1])
```