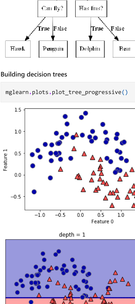
## Decision trees

In [56]:
```
import sys
sys.path
```

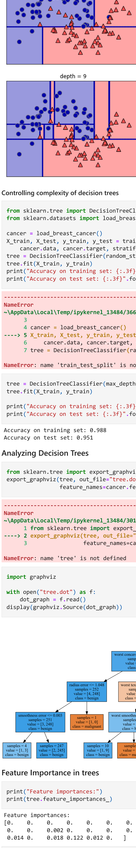
Out[56]:
```
['C:\\Users\\t3kci\\checkout\\introduction_to_ml_with_python',
 'C:\\Users\\t3kci\\anaconda3\\python37.zip',
 'C:\\Users\\t3kci\\anaconda3\\DLLs',
 'C:\\Users\\t3kci\\anaconda3\\lib',
 'C:\\Users\\t3kci\\anaconda3',
 '',
 'C:\\Users\\t3kci\\anaconda3\\lib\\site-packages',
 'c:\\users\\t3kci\\checkout\\scikit-learn',
 'C:\\Users\\t3kci\\checkout\\jupyter-book',
 'C:\\Users\\t3kci\\anaconda3\\lib\\site-packages\\win32',
 'C:\\Users\\t3kci\\anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\Users\\t3kci\\anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\Users\\t3kci\\anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\t3kci\\.ipython']
```

In [57]:
```
mglearn.plots.plot_animal_tree()
```




### Building decision trees

In [58]:
```
mglearn.plots.plot_tree_progressive()
```




### Controlling complexity of decision trees

In [7]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=42)
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13484/3664182047.py in <module>
      3
      4 cancer = load_breast_cancer()
----> 5 X_train, X_test, y_train, y_test = train_test_split(
      6     cancer.data, cancer.target, stratify=cancer.target, random_state=42)
      7 tree = DecisionTreeClassifier(random_state=0)

NameError: name 'train_test_split' is not defined
```

In [60]:
```
tree = DecisionTreeClassifier(max_depth=4, random_state=0)
tree.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

```
Accuracy on training set: 0.988
Accuracy on test set: 0.951
```

### Analyzing Decision Trees

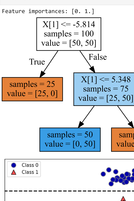
In [5]:
```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file="tree.dot", class_names=["malignant", "benign"],
                feature_names=cancer.feature_names, impurity=False, filled=True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13484/3012021028.py in <module>
      1 from sklearn.tree import export_graphviz
----> 2 export_graphviz(tree, out_file="tree.dot", class_names=["malignant", "benign"],
      3                 feature_names=cancer.feature_names, impurity=False, filled=True)

NameError: name 'tree' is not defined
```

In [62]:
```
import graphviz

with open("tree.dot") as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))
```




### Feature Importance in trees

In [63]:
```
print("Feature importances:")
print(tree.feature_importances_)

Feature importances:
[0.    0.    0.    0.    0.    0.    0.    0.    0.01  0.048
 0.    0.    0.002 0.    0.    0.    0.    0.    0.727 0.046 0.
 0.014 0.    0.018 0.122 0.012 0.   ]
```

In [64]:
```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1]
    plt.barh(np.arange(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), cancer.feature_names)
    plt.xlabel("Feature Importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)

plot_feature_importances_cancer(tree)
```




In [65]:
```
tree = mglearn.plots.plot_tree_not_monotone()
display(tree)

Feature importances: [0. 1.]
```




In [66]:
```
import os
ram_prices = pd.read_csv(os.path.join(mglearn.datasets.DATA_PATH, "ram_price.csv"))

plt.semilogy(ram_prices.date, ram_prices.price)
plt.xlabel("Year")
plt.ylabel("Price in $/Mbyte")

Text(0, 0.5, 'Price in $/Mbyte')
```

Out[66]:




In [67]:
```
from sklearn.tree import DecisionTreeRegressor
# use historical data to forecast prices after the year 2000
data_train = ram_prices[ram_prices.date < 2000]
data_test = ram_prices[ram_prices.date >= 2000]

# predict prices based on date
X_train = data_train.date[:, np.newaxis]
# we use a log-transform to get a simpler relationship of data to target
y_train = np.log(data_train.price)

tree = DecisionTreeRegressor(max_depth=3).fit(X_train, y_train)
linear_reg = LinearRegression().fit(X_train, y_train)

# predict on all data
X_all = ram_prices.date[:, np.newaxis]

pred_tree = tree.predict(X_all)
pred_lr = linear_reg.predict(X_all)

# undo log-transform
price_tree = np.exp(pred_tree)
price_lr = np.exp(pred_lr)
```

In [68]:
```
plt.semilogy(data_train.date, data_train.price, label="Training data")
plt.semilogy(data_test.date, data_test.price, label="Test data")
plt.semilogy(ram_prices.date, price_tree, label="Tree prediction")
plt.semilogy(ram_prices.date, price_lr, label="Linear prediction")
plt.legend()
```

Out[68]:
```
<matplotlib.legend.Legend at 0x1624b51f088>
```




In [ ]: