EECS 445: Machine Learning

Hands On 11: Info Theory and Decision Trees

Prove that the KL divergence $KL(p, q)$ for any distributions p, q is non negative.

Hint: Show that

$$\min_q KL(p, q) \geq 0$$

Prove that two random variables X and Y are independent if and only if $I(X, Y) = 0$.

Prove that

- $I(X, Y) = H(X) + H(Y) - H(X, Y)$
- $I(X, Y) = H(X) - H(X|Y)$

Prove that

$$I(X, Y) = H(X)$$

if Y is a determinisitc, one-to-one function of X

Decision trees vs linear models

Let's explore cases where decision trees perform better and worse than linear models.

Note: we're using the function 'plot_decision_regions' from the mlextend library (https://github.com/rasbt/mlxtend); you can install with pip, conda or just copy and paste the function into a cell from here (https://raw.githubusercontent.com/rasbt/mlxtend/master/mlxtend/plotting/decision_regions.py).

```
$ conda install -c rasbt mlxtend
```

In [1]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression, Perceptron
import numpy as np
import matplotlib.pyplot as plt

from mlxtend.evaluate import plot_decision_regions

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

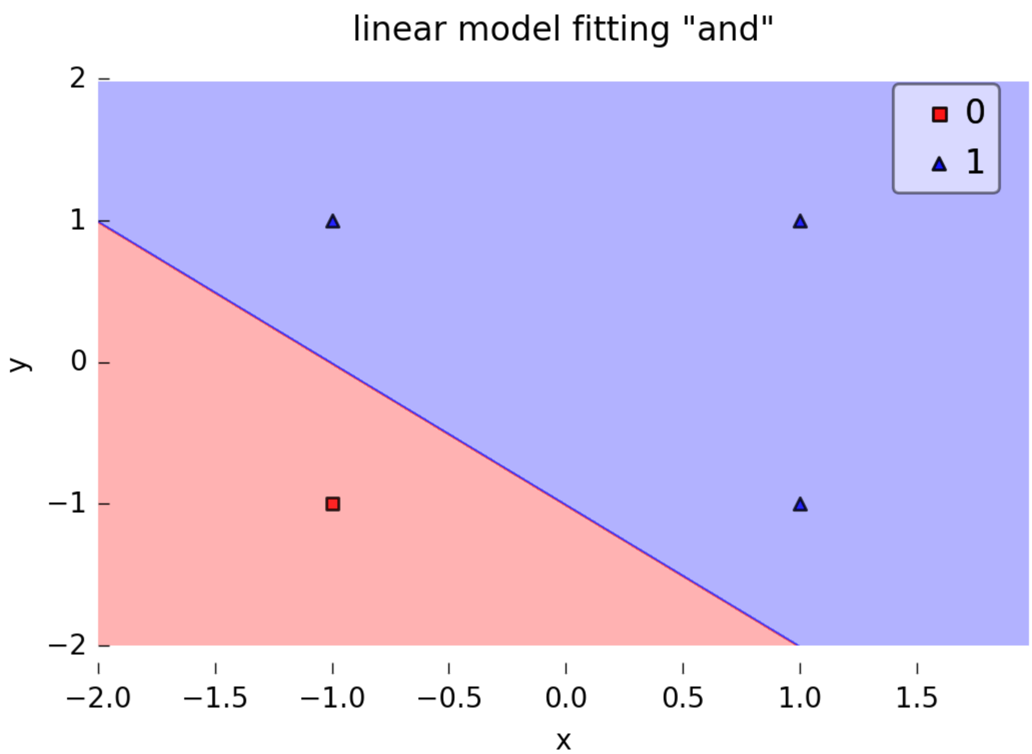
Trees can fit XOR

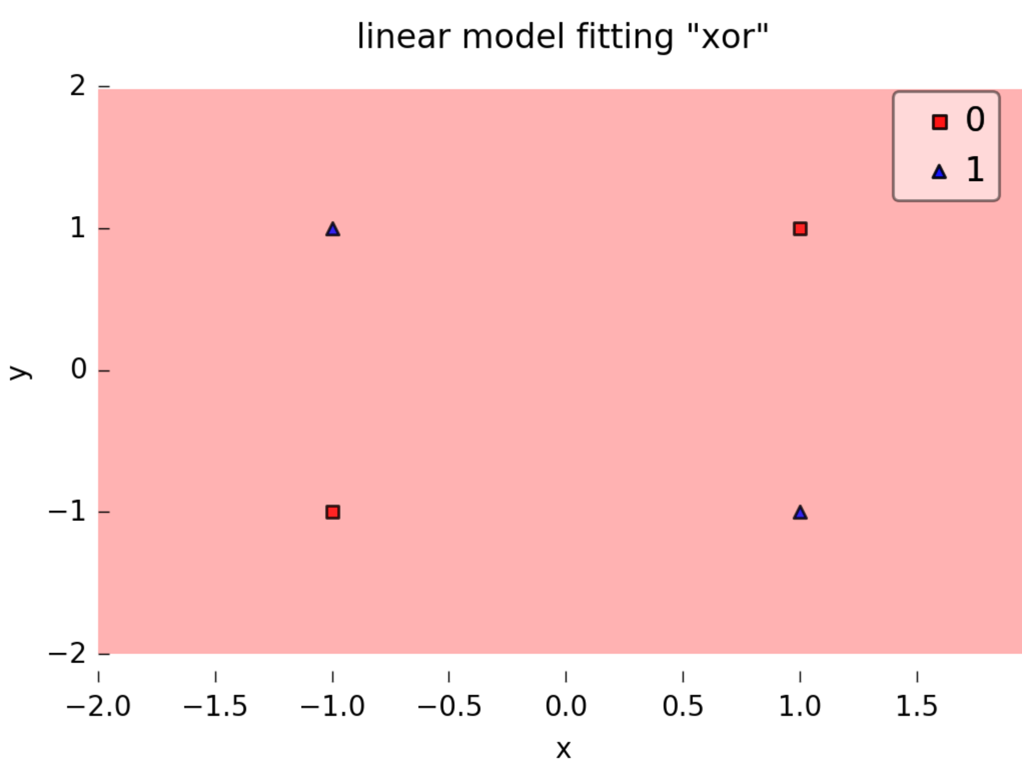
The classic minimal function that a linear model can't fit is XOR. Let's visualize how linear and tree models manage to fit AND and XOR.

In [2]:
```
X = np.array([
        [-1, -1],
        [-1, 1],
        [1, -1],
        [1, 1]
    ])

y_and = np.array([0, 1, 1, 1])
y_xor = np.array([0, 1, 1, 0])
```

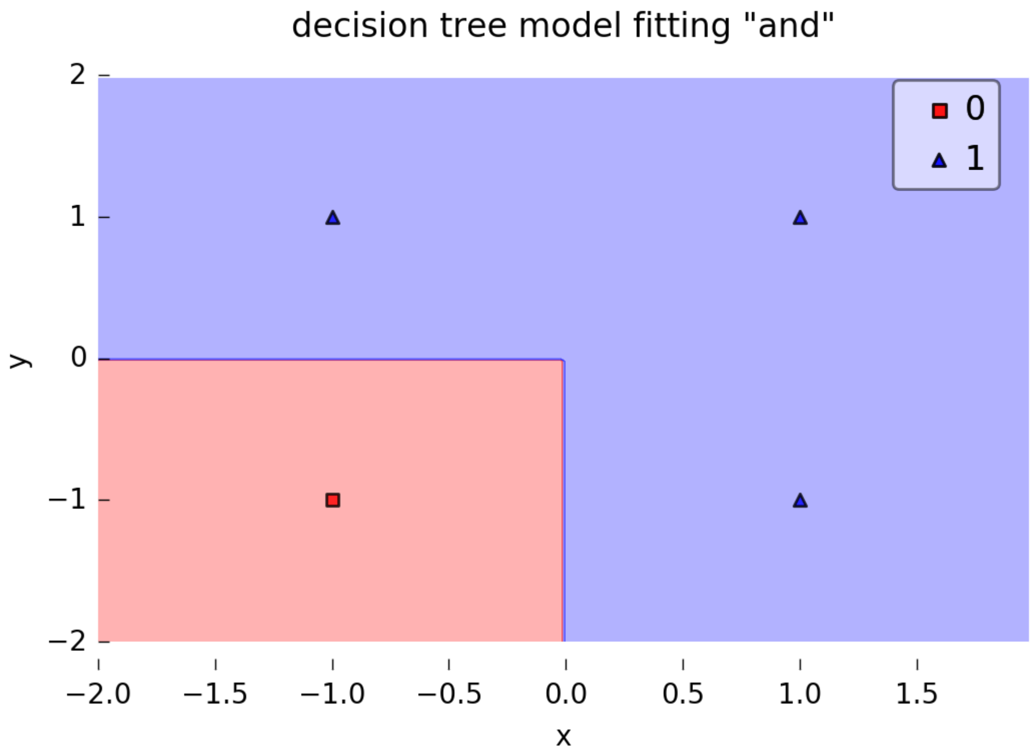
In [3]:

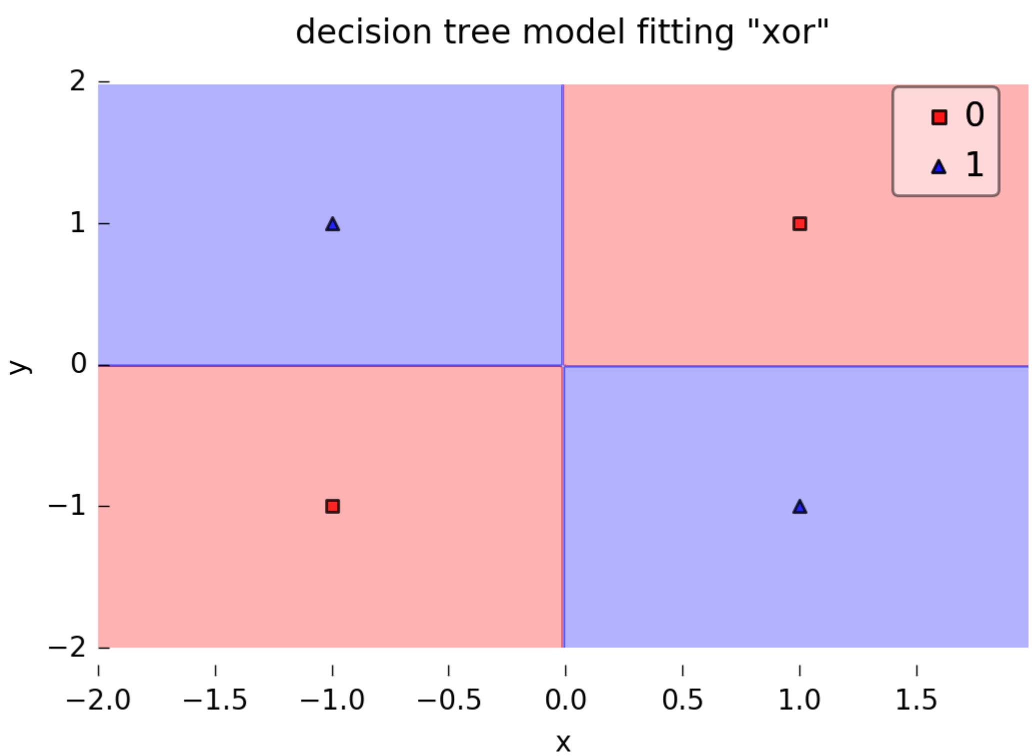
```
lr = LogisticRegression()

for label, y in [('and', y_and), ('xor', y_xor)]:
    lr.fit(X, y)
    plot_decision_regions(X, y, lr)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('linear model fitting "{}"'.format(label))
    plt.show()
```





In [4]:
```
tree = DecisionTreeClassifier(criterion='entropy', max_depth=2, random_s
tate=0)
tree.fit(X, y)

for label, y in [('and', y_and), ('xor', y_xor)]:
    tree.fit(X, y)
    plot_decision_regions(X, y, tree)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('decision tree model fitting "{}"'.format(label))
    plt.show()
```





When a linear models beat a decision tree

For this exercise, construct a dataset that a linear model can fit, but that a decision tree of depth 2 cannot.

In [5]:
```
from sklearn.metrics import accuracy_score

X_linear_wins = np.array([
        # place 2d samples here, each value between -1 and 1
    ])
y_linear_wins = np.array([
        # place class label 0, 1 for each 2d point here
])


# uncommment code below to test out whether your dataset is more accurat
ely predicted by a linear model
# than a tree of depth 3.

# for label, model in [('linear', lr), ('tree', tree)]:
#     model.fit(X_linear_wins, y_linear_wins)
#     plot_decision_regions(X_linear_wins, y_linear_wins, model)
#     plt.xlabel('x')
#     plt.ylabel('y')
#     title = "{}: accuracy {:.2f}".format(label, accuracy_score(y_linea
r_wins, model.predict(X_linear_wins)))
#     plt.title(title)
#     plt.show()
```

Depth matters

For this exercise, construct a dataset that cannot be 100% accurately classified with a tree of depth 2 but *can* be by a tree of depth 3.

In [6]:
```
X_needs_depth = np.array([
        # place 2d samples here, each value between -1 and 1
    ])

y_needs_depth = np.array([
        # place class label 0, 1 for each 2d point here
])


# uncomment the code below to compare

# tree_d2 = DecisionTreeClassifier(criterion='entropy', max_depth=2, ran
dom_state=0)
# tree_d4 = DecisionTreeClassifier(criterion='entropy', max_depth=4, ran
dom_state=0)

# tree_d2.fit(X_needs_depth, y_needs_depth)
# plot_decision_regions(X_needs_depth, y_needs_depth, tree_d2)
# plt.title("depth 2 fit: {:.2f}".format(accuracy_score(y_needs_depth, t
ree_d2.predict(X_needs_depth))))
# plt.show()

# tree_d4.fit(X_needs_depth, y_needs_depth)
# plot_decision_regions(X_needs_depth, y_needs_depth, tree_d4)
# plt.title("depth 4 fit: {:.2f}".format(accuracy_score(y_needs_depth, t
ree_d4.predict(X_needs_depth))))
# plt.show()
```