El problema de Fisher resuelto con Gradient Boosting

September 7, 2018

0.1 El problema de Fisher resuelto con Gradient Boosting

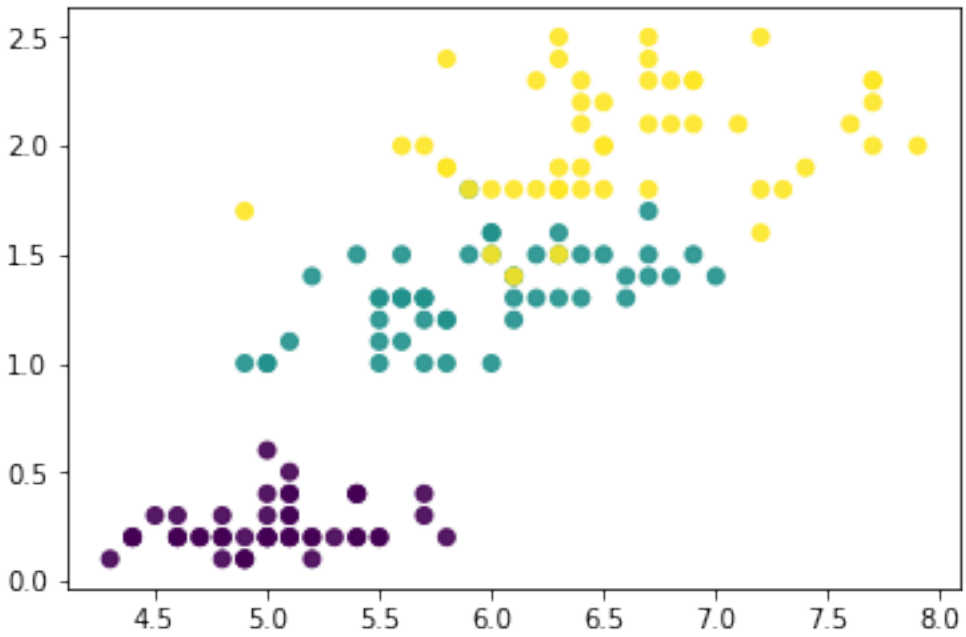
In [1]:
```
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np
from pandas import DataFrame
from sklearn import linear_model
from sklearn import ensemble

import seaborn as sns
%matplotlib inline
```

In [2]:
```
def plot_decision_boundaries(X, y, model_class, **model_params):
    """Function to plot the decision boundaries of a classification model.
    This uses just the first two columns of the data for fitting
    the model as we need to find the predicted value for every point in
    scatter plot.

    One possible improvement could be to use all columns fot fitting
    and using the first 2 columns and median of all other columns
    for predicting.

    Adopted from:
    http://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.h
    http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html
    """
    reduced_data = X[:, :2]
    model = model_class(**model_params)
    model.fit(reduced_data, y)

    # Step size of the mesh. Decrease to increase the quality of the VQ.
    h = .03     # point in the mesh [x_min, m_max]x[y_min, y_max].

    # Plot the decision boundary. For that, we will assign a color to each
    x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
    y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

```
    # Obtain labels for each point in mesh using the model.
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])

    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                         np.arange(y_min, y_max, 0.1))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.25)
    plt.scatter(X[:, 0], X[:, 1], c=y, alpha=0.9)

    return plt
```

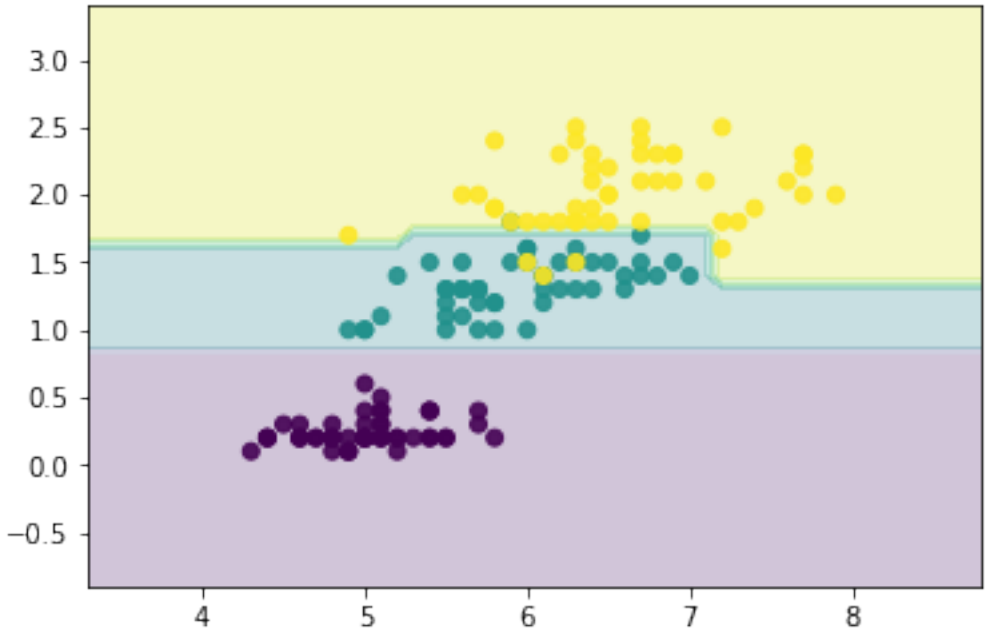
In [3]:
```
iris = datasets.load_iris()
X = iris.data[:, [0,3]]  # we only take two features.
y = iris.target
plt.scatter(X[:, 0], X[:, 1], c=y, alpha=0.9)
```

Out[3]: <matplotlib.collections.PathCollection at 0x11917c048>



In [4]:
```
plot_decision_boundaries(X,y,ensemble.GradientBoostingClassifier)
```

Out[4]: <module 'matplotlib.pyplot' from '/Users/rafa/anaconda3/lib/python3.6/site-packages/matp



In [5]: plot_decision_boundaries(X,y,linear_model.SGDClassifier)

Out[5]: <module 'matplotlib.pyplot' from '/Users/rafa/anaconda3/lib/python3.6/site-packages/matp