piecewise_classification

February 28, 2023

1 Piecewise classification with scikit-learn predictors

Piecewise regression is easier to understand but the concept can be extended to classification. That's what this notebook explores.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

[2]:
```
%matplotlib inline
```

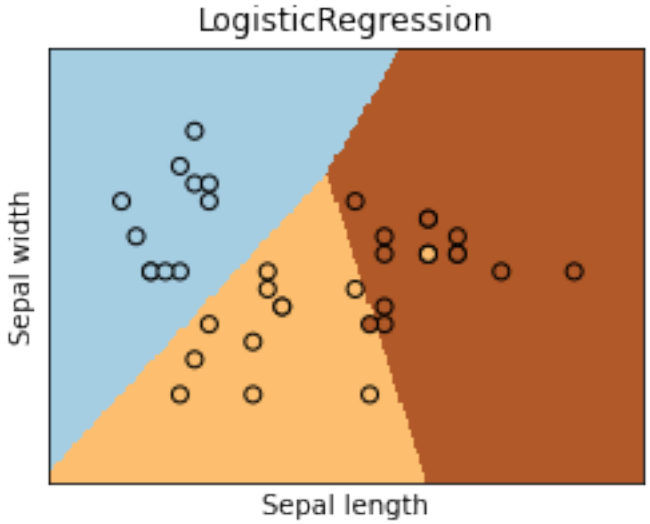
[3]:
```
import warnings
warnings.simplefilter("ignore")
```

1.1 Iris dataset and first logistic regression

[4]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, Y)
```

[5]:
```
import numpy
import matplotlib.pyplot as plt

def graph(X, Y, model):
    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
    h = .02  # step size in the mesh
    xx, yy = numpy.meshgrid(numpy.arange(x_min, x_max, h),
                            numpy.arange(y_min, y_max, h))
    Z = model.predict(numpy.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Put the result into a color plot
    fig, ax = plt.subplots(1, 1, figsize=(4, 3))
    ax.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

    # Plot also the training points
```

```
    ax.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
    ax.set_xlabel('Sepal length')
    ax.set_ylabel('Sepal width')

    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    return ax

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
ax = graph(X_test, y_test, logreg)
ax.set_title("LogisticRegression");
```



1.2 Piecewise classication

[6]:
```
from sklearn.dummy import DummyClassifier
from sklearn.preprocessing import KBinsDiscretizer
from mlinsights.mlmodel import PiecewiseClassifier

dummy = DummyClassifier(strategy='most_frequent')
piece4 = PiecewiseClassifier(KBinsDiscretizer(n_bins=2),
                             estimator=dummy, verbose=True)
piece4.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s finished
```

[6]:
```
PiecewiseClassifier(binner=KBinsDiscretizer(n_bins=2),
                    estimator=DummyClassifier(strategy='most_frequent'),
                    verbose=True)
```

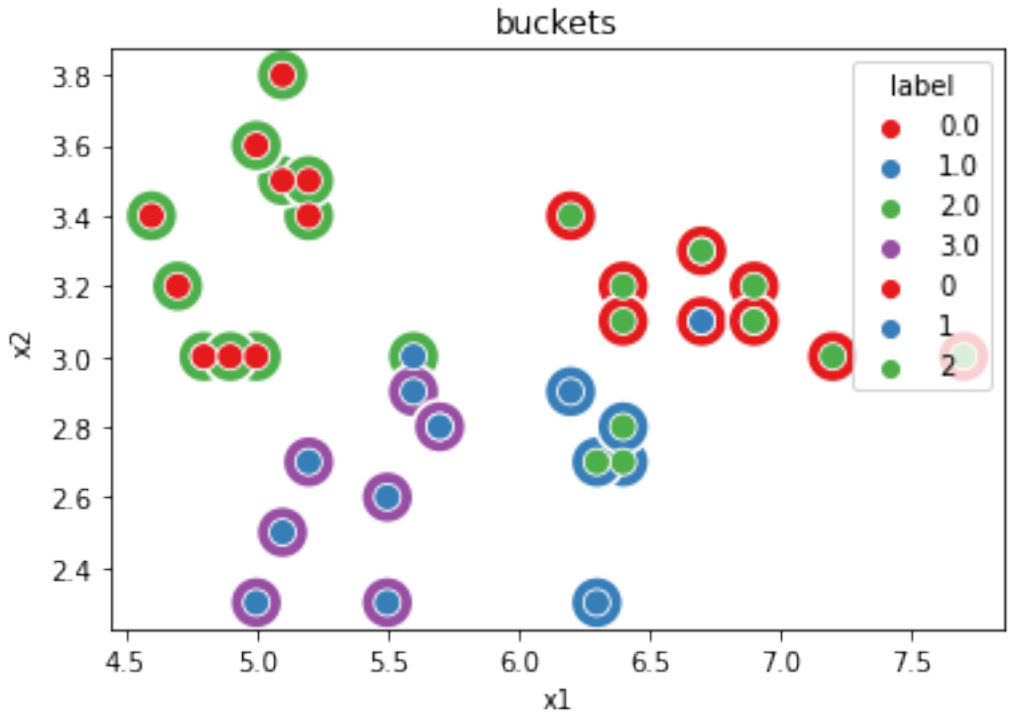
We look into the bucket given to each point.

[7]:
```
import pandas

bucket = piece4.transform_bins(X_test)
df = pandas.DataFrame(X_test, columns=("x1", "x2"))
df["bucket"] = bucket
df["label"] = y_test
df = df.set_index(bucket)
df.head(n=5)
```

[7]:

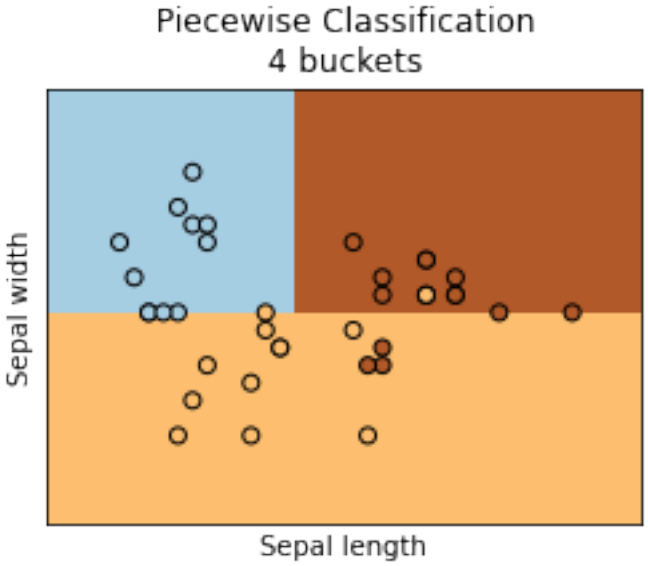
	x1	x2	bucket	label
0.0	6.2	3.4	0.0	2
0.0	6.7	3.1	0.0	1
2.0	5.1	3.8	2.0	0
2.0	4.8	3.0	2.0	0
3.0	5.5	2.3	3.0	1

[8]:
```
import seaborn
ax = seaborn.scatterplot("x1", "x2", "bucket", data=df, palette='Set1', s=400)
seaborn.scatterplot("x1", "x2", "label", data=df, palette='Set1', marker="o", ax=ax,␣
↪s=100)
ax.set_title("buckets");
```



We see there are four buckets. Two buckets only contains one label. The dummy classifier maps every bucket to the most frequent class in the bucket.

[9]:
```
ax = graph(X_test, y_test, piece4)
ax.set_title("Piecewise Classification\n4 buckets");
```



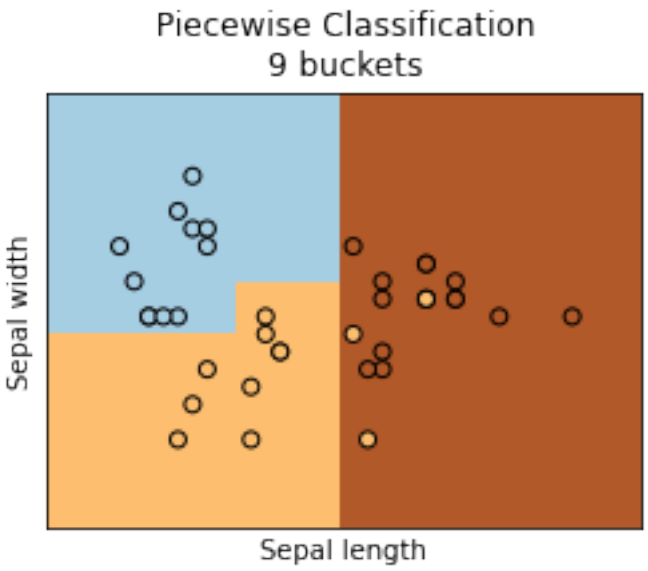
We can increase the number of buckets.

[10]:
```
dummy = DummyClassifier(strategy='most_frequent')
piece9 = PiecewiseClassifier(KBinsDiscretizer(n_bins=3),
                             estimator=dummy, verbose=True)
piece9.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s finished
```

[10]:
```
PiecewiseClassifier(binner=KBinsDiscretizer(n_bins=3),
                    estimator=DummyClassifier(strategy='most_frequent'),
                    verbose=True)
```

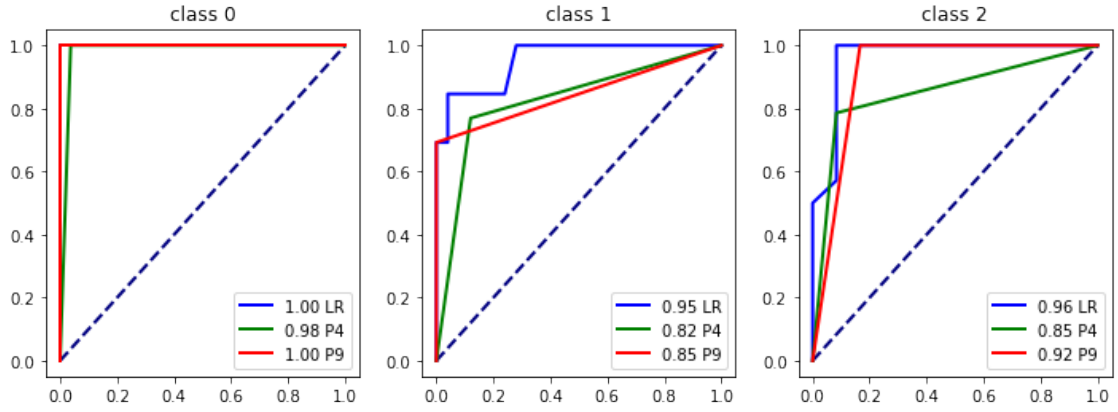
[11]:
```
ax = graph(X_test, y_test, piece9)
ax.set_title("Piecewise Classification\n9 buckets");
```



Let's compute the ROC curve.

[12]:
```
from sklearn.metrics import roc_curve, auc

def plot_roc_curve(models, X, y):
    if not isinstance(models, dict):
        return plot_roc_curve({models.__class__.__name__: models}, X, y)

    ax = None
    colors = 'bgrcmyk'
    for ic, (name, model) in enumerate(models.items()):
        fpr, tpr, roc_auc = dict(), dict(), dict()
        nb = len(model.classes_)
        y_score = model.predict_proba(X)
        for i in range(nb):
            c = model.classes_[i]
            fpr[i], tpr[i], _ = roc_curve(y_test == c, y_score[:, i])
            roc_auc[i] = auc(fpr[i], tpr[i])

        if ax is None:
            lw = 2
            _, ax = plt.subplots(1, nb, figsize=(4 * nb, 4))
            for i in range(nb):
                ax[i].plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
        plotname = "".join(c for c in name if "A" <= c <= "Z" or "0" <= c <= "9")
        for i in range(nb):
            ax[i].plot(fpr[i], tpr[i], color=colors[ic],
                       lw=lw, label='%0.2f %s' % (roc_auc[i], plotname))
            ax[i].set_title("class {}".format(model.classes_[i]))
    for k in range(ax.shape[0]):
        ax[k].legend()
    return ax
```

```
plot_roc_curve({'LR': logreg, 'P4': piece4, 'P9': piece9}, X_test, y_test);
```



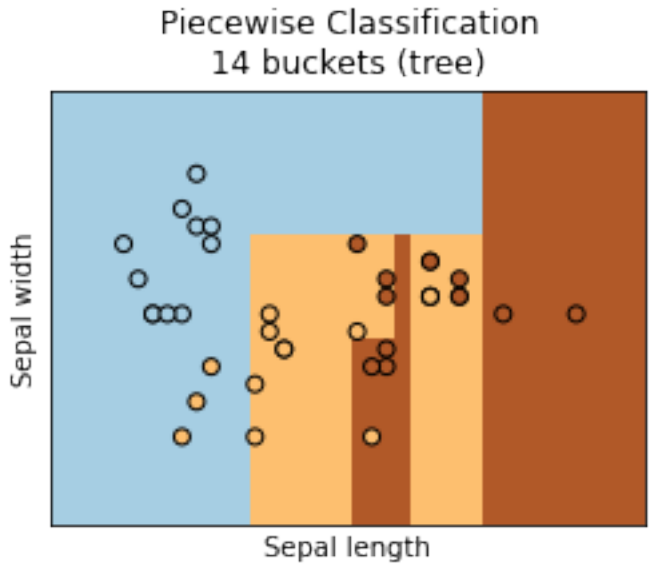
Let's use the decision tree to create buckets.

[13]:
```
dummy = DummyClassifier(strategy='most_frequent')
pieceT = PiecewiseClassifier("tree", estimator=dummy, verbose=True)
pieceT.fit(X_train, y_train)
```

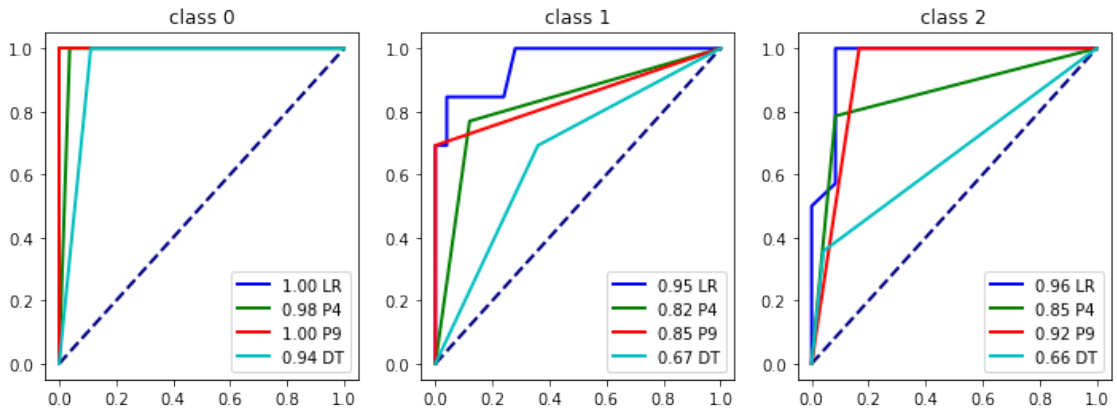
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    0.0s finished
```

[13]:
```
PiecewiseClassifier(binner=DecisionTreeClassifier(min_samples_leaf=5),
                    estimator=DummyClassifier(strategy='most_frequent'),
                    verbose=True)
```

[14]:
```
ax = graph(X_test, y_test, pieceT)
ax.set_title("Piecewise Classification\n%d buckets (tree)" % len(pieceT.estimators_));
```



[15]:
```
plot_roc_curve({'LR': logreg, 'P4': piece4, 'P9': piece9, "DT": pieceT},
               X_test, y_test);
```



[16]:

[17]: