

Iris Data - Precision / Recall

```
In [1]: from sklearn import svm, datasets
        from sklearn.model_selection import train_test_split
        import numpy as np

        iris = datasets.load_iris()
        X = iris.data
        y = iris.target

        # Add noisy features
        random_state = np.random.RandomState(0)
        n_samples, n_features = X.shape
        X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

        # Limit to the two first classes, and split into training and test
        X_train, X_test, y_train, y_test = train_test_split(X[y < 2], y[y < 2],
                                                            test_size=.5,
                                                            random_state=random_state)

        # Create a simple classifier
        classifier = svm.LinearSVC(random_state=random_state)
        classifier.fit(X_train, y_train)
        y_score = classifier.decision_function(X_test)

        from sklearn.metrics import average_precision_score
        average_precision = average_precision_score(y_test, y_score)

        print('Average precision-recall score: {0:0.2f}'.format(
            average_precision))
```

Average precision-recall score: 0.88

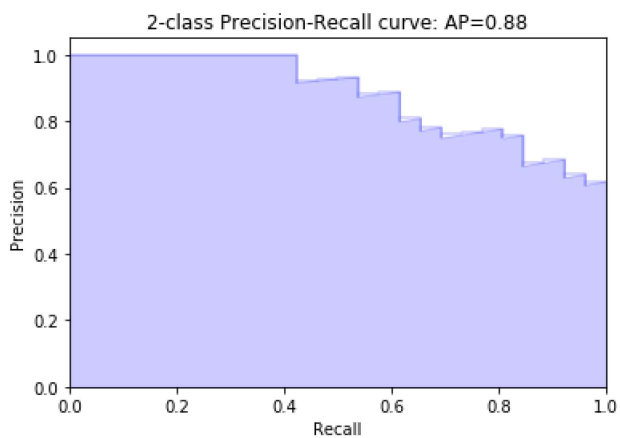
```
In [3]: from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
from inspect import signature

precision, recall, _ = precision_recall_curve(y_test, y_score)

plt.step(recall, precision, color='b', alpha=0.2, where='post')
plt.fill_between(recall, precision, alpha=0.2, color='b')

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(
    average_precision))
```

Out[3]: Text(0.5,1,'2-class Precision-Recall curve: AP=0.88')

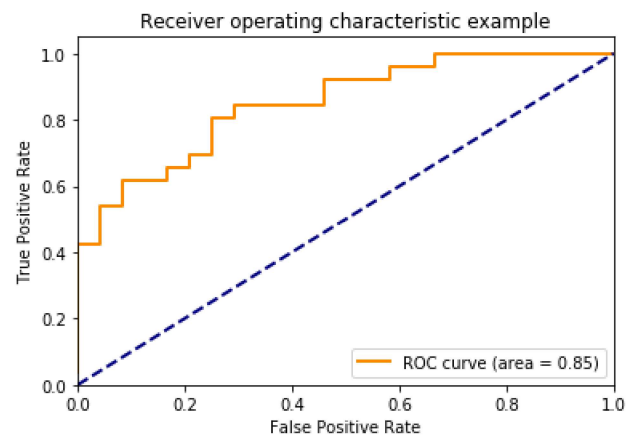


Iris Data - ROC curve

```
In [4]: from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



k-NN Classifier - Precision / Recall

```
In [5]: # precision-recall curve and f1
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
from matplotlib import pyplot

# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[1,1], random_state=1)

# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)

# fit a model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(trainX, trainy)

# predict probabilities
probs = model.predict_proba(testX)

# keep probabilities for the positive outcome only
probs = probs[:, 1]

# predict class values
yhat = model.predict(testX)

# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(testy, probs)

# calculate F1 score
f1 = f1_score(testy, yhat)

# calculate precision-recall AUC
auc = auc(recall, precision)

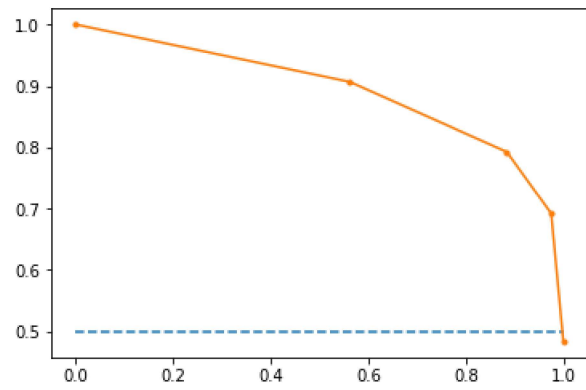
# calculate average precision score
ap = average_precision_score(testy, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))

# plot no skill
pyplot.plot([0, 1], [0.5, 0.5], linestyle='--')

# plot the precision-recall curve for the model
pyplot.plot(recall, precision, marker='.')

# show the plot
pyplot.show()
```

f1=0.836 auc=0.892 ap=0.840



k-NN Classifier - ROC Curve

```
In [6]: # roc curve and auc
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[1,1], random_state=1)

# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)

# fit a model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(trainX, trainy)

# predict probabilities
probs = model.predict_proba(testX)

# keep probabilities for the positive outcome only
probs = probs[:, 1]

# calculate AUC
auc = roc_auc_score(testy, probs)
print('AUC: %.3f' % auc)

# calculate roc curve
fpr, tpr, thresholds = roc_curve(testy, probs)

# plot no skill
pyplot.plot([0, 1], [0, 1], linestyle='--')

# plot the roc curve for the model
pyplot.plot(fpr, tpr, marker='.')

# show the plot
pyplot.show()
```

AUC: 0.895

