



File Edit View Insert Cell Kernel Widgets Help

Python 3

You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the [Jupyter Notebook FAQ](#) course resource.

Classifier Visualization Playground

The purpose of this notebook is to let you visualize various classifiers' decision boundaries.

The data used in this notebook is based on the [UCI Mushroom Data Set](#) stored in `mushrooms.csv`.

In order to better visualize the decision boundaries, we'll perform Principal Component Analysis (PCA) on the data to reduce the dimensionality to 2 dimensions. Dimensionality reduction will be covered in a later module of this course.

Play around with different models and parameters to see how they affect the classifier's decision boundary and accuracy!

```
In [1]: %matplotlib notebook

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split

df = pd.read_csv('readonly/mushrooms.csv')
df2 = pd.get_dummies(df)

df3 = df2.sample(frac=0.08)

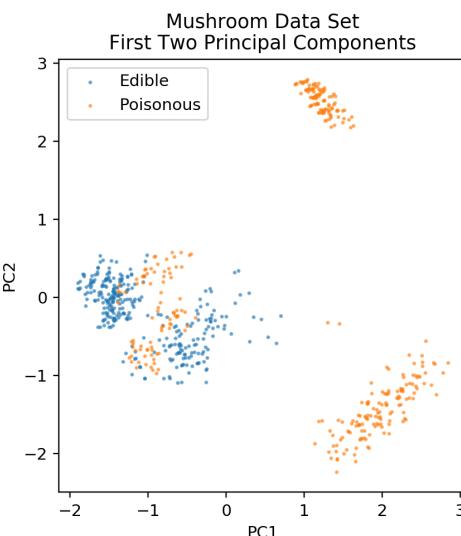
X = df3.iloc[:,2:]
y = df3.iloc[:,1]

pca = PCA(n_components=2).fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(pca, y, random_state=0)

plt.figure(dpi=120)
plt.scatter(pca[y.values==0,0], pca[y.values==0,1], alpha=0.5, label='Edible', s=2)
plt.scatter(pca[y.values==1,0], pca[y.values==1,1], alpha=0.5, label='Poisonous', s=2)
plt.legend()
plt.title('Mushroom Data Set\nFirst Two Principal Components')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.gca().set_aspect('equal')
```

Figure 1



Home Back Forward Stop Refresh

```
In [2]: def plot_mushroom_boundary(X, y, fitted_model):

    plt.figure(figsize=(9.8,5), dpi=100)

    for i, plot_type in enumerate(['Decision Boundary', 'Decision Probabilities']):
        plt.subplot(1,2,i+1)

        mesh_step_size = 0.01 # step size in the mesh
        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, mesh_step_size),
                             np.arange(y_min, y_max, mesh_step_size))
        if i == 0:
            Z = fitted_model.predict(np.c_[xx.ravel(), yy.ravel()])
        else:
            try:
                Z = fitted_model.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:,1]
            except:
                plt.text(0.4, 0.5, 'Probabilities Unavailable', horizontalalignment='center',
```

```

        verticalalignment='center', transform = plt.gca().transAxes, fontsize=12)
    plt.axis('off')
    break
Z = Z.reshape(xx.shape)
plt.scatter(X[y.values==0,0], X[y.values==0,1], alpha=0.4, label='Edible', s=5)
plt.scatter(X[y.values==1,0], X[y.values==1,1], alpha=0.4, label='Poisonous', s=5)
plt.imshow(Z, interpolation='nearest', cmap='RdYlBu_r', alpha=0.15,
           extent=(x_min, x_max, y_min, y_max), origin='lower')
plt.title(plot_type + '\n' +
          str(fitted_model).split('(')[0] + ' Test Accuracy: ' + str(np.round(fitted_model.score(X, y), 5)))
plt.gca().set_aspect('equal');

plt.tight_layout()
plt.subplots_adjust(top=0.9, bottom=0.08, wspace=0.02)

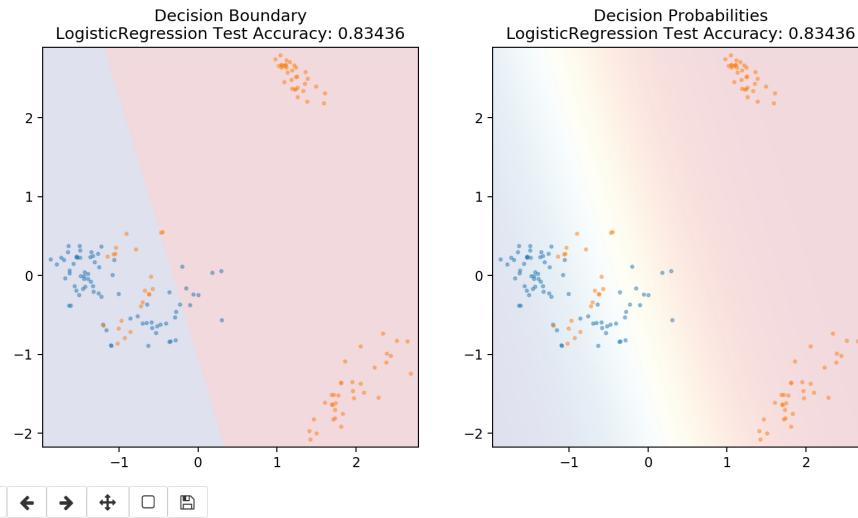
```

```
In [3]: from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)
```

Figure 2

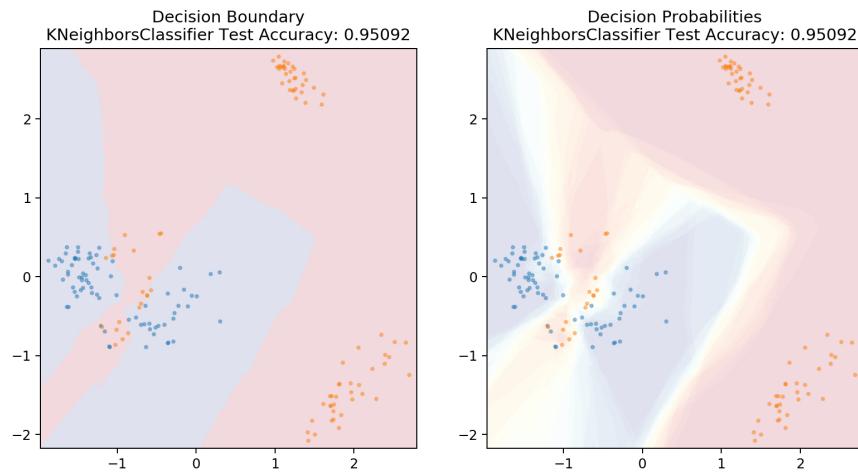


```
In [4]: from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=20)
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)
```

Figure 3

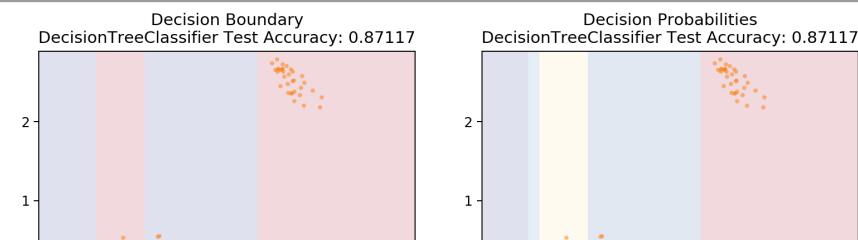


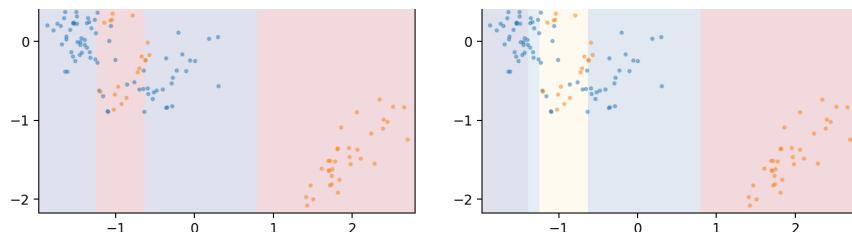
```
In [5]: from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=3)
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)
```

Figure 4

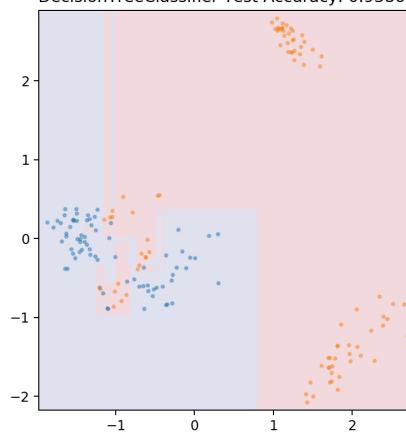




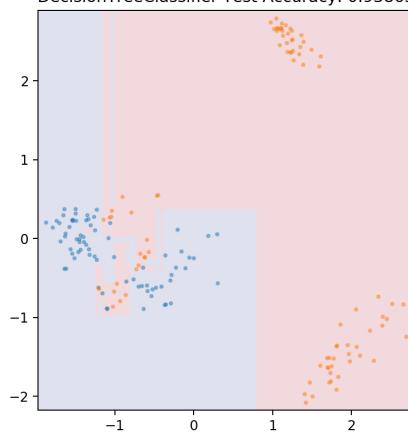
```
In [6]: from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

Figure 5

Decision Boundary
DecisionTreeClassifier Test Accuracy: 0.93865



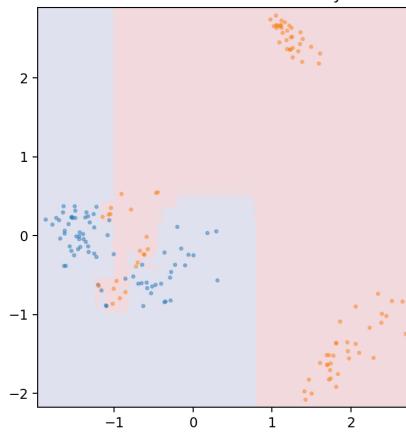
Decision Probabilities
DecisionTreeClassifier Test Accuracy: 0.93865



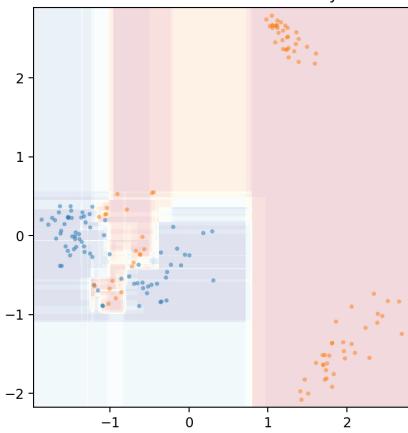
```
In [7]: from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

Figure 6

Decision Boundary
RandomForestClassifier Test Accuracy: 0.94479



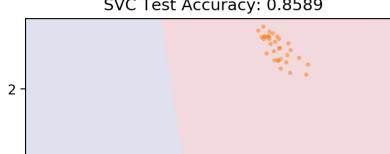
Decision Probabilities
RandomForestClassifier Test Accuracy: 0.94479

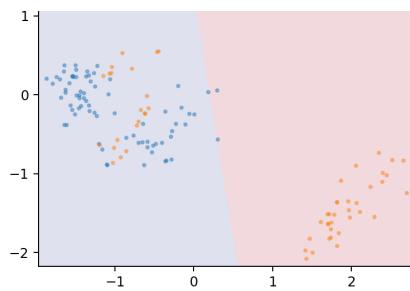


```
In [8]: from sklearn.svm import SVC
model = SVC(kernel='linear')
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

Figure 7

Decision Boundary
SVC Test Accuracy: 0.8589

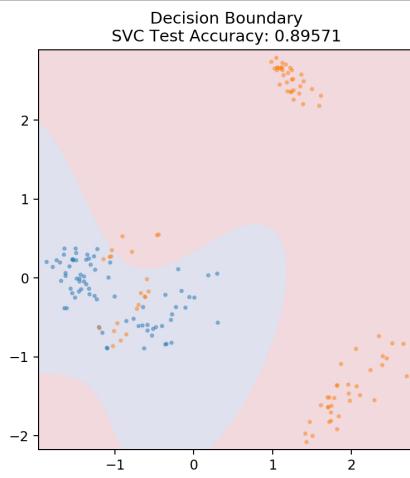




Probabilities Unavailable

```
In [9]: from sklearn.svm import SVC
model = SVC(kernel='rbf', C=1)
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

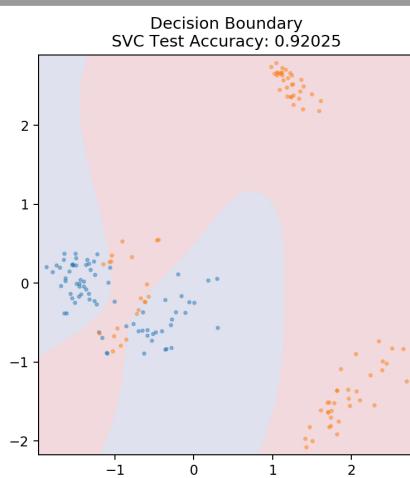
Figure 8



Probabilities Unavailable

```
In [10]: from sklearn.svm import SVC
model = SVC(kernel='rbf', C=10)
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

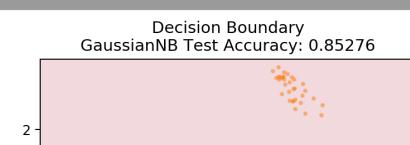
Figure 9



Probabilities Unavailable

```
In [11]: from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train,y_train)
plot_mushroom_boundary(X_test, y_test, model)
```

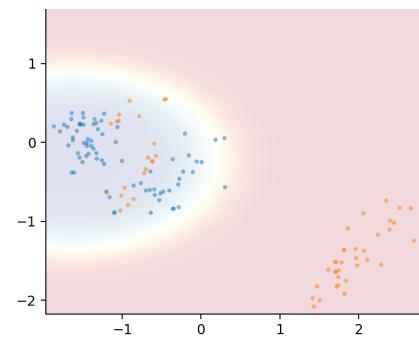
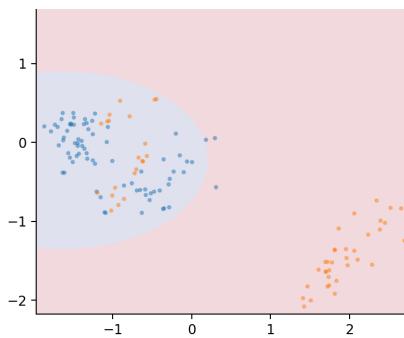
Figure 10



Decision Probabilities
GaussianNB Test Accuracy: 0.85276



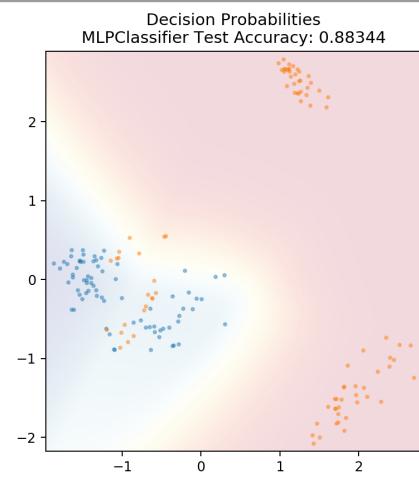
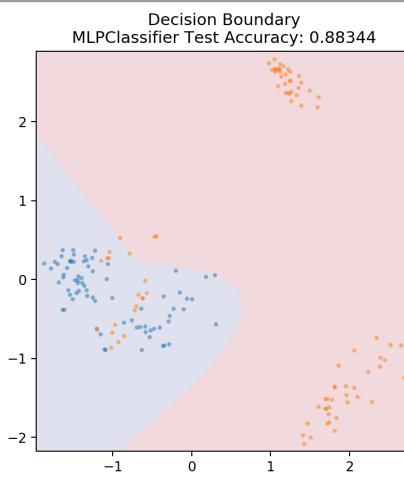
x=-0.46416 y=1.95579 [1]



x=-1.42725 y=2.51663 [1]

```
In [12]: from sklearn.neural_network import MLPClassifier  
model = MLPClassifier()  
model.fit(X_train,y_train)  
plot_mushroom_boundary(X_test, y_test, model)  
/opt/conda/lib/python3.6/site-packages/sklearn/neural_network/multilayer_perceptron.py:563: ConvergenceWarning: Stochastic Optimizer: Maximum iterations reached and the optimization hasn't converged yet.  
% (), ConvergenceWarning)
```

Figure 11



```
In [ ]:
```