

# Classifier Visualization

September 18, 2020

---

*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.*

---

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

```
In [ ]: 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()])
```

```
            except:
```

```
                plt.text(0.4, 0.5, 'Probabilities Unavailable', horizontalalignment='center',
```

```
                        verticalalignment='center', transform = plt.gca().transData)
```

```
                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')
```

```
        plt.scatter(X[y.values==1,0], X[y.values==1,1], alpha=0.4, label='Poisonous')
```

```
        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(fitted_model.score(X_test, y_test)))
```

```
        plt.gca().set_aspect('equal');
```

```
    plt.tight_layout()
```

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

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

```
model = LogisticRegression()
```

```

model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: from sklearn.neighbors import KNeighborsClassifier

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

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: from sklearn.tree import DecisionTreeClassifier

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

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: from sklearn.tree import DecisionTreeClassifier

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

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: from sklearn.ensemble import RandomForestClassifier

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

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: from sklearn.svm import SVC

model = SVC(kernel='linear')
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)

In [ ]: 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)

In [ ]: 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)
```

```
In [ ]: from sklearn.naive_bayes import GaussianNB
```

```
model = GaussianNB()
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)
```

```
In [ ]: from sklearn.neural_network import MLPClassifier
```

```
model = MLPClassifier()
model.fit(X_train,y_train)

plot_mushroom_boundary(X_test, y_test, model)
```

```
In [ ]:
```