Práctica 8: LDA

Descripción de campos importantes: Mushrooms

Campo	Descripción
class	edible(e) or poisonous(p)
cap-shape	bell(b), conical(c), convex(x), flat(f), knobbed(k), sunken(s)
cap-surface	fibrous(f), grooves(g), scaly(y), smooth(s)
cap-color	brown(n), buff(b), cinnamon(c), gray(g), green(r), pink(p), purple(u), red(e), white(w), yellow(y)
bruises	bruises(t), no bruises(f)
odor	almond(a), anise(l), creosote(c), fishy(y), foul(f), musty(m), none(n), pungent(p), spicy(s)
gill-attachment	attached(a), descending(d), free(f), notched(n)
gill-spacing	close(c), crowded(w), distant(d)
gill-size	broad(b), narrow(n)
gill-color	black(k), brown(n), buff(b), chocolate(h), gray(g), green(r), orange(o), pink(p), purple(u), red(e), white(w), yellow(y)
stalk-shape	enlarging(e), tapering(t)
stalk-root	bulbous(b), club(c), cup(u), equal(e), rhizomorphs(z), rooted(r), missing(?)
stalk-surface-above-ring	fibrous(f), scaly(y), silky(k), smooth(s)
stalk-surface-below-ring	fibrous(f), scaly(y), silky(k), smooth(s)
stalk-color-above-ring	brown(n), buff(b), cinnamon(c), gray(g), orange(o), pink(p), red(e), white(w), yellow(y)
stalk-color-below-ring	brown(n), buff(b), cinnamon(c), gray(g), orange(o), pink(p), red(e), white(w), yellow(y)
veil-type	partial(p), universal(u)
veil-color	brown(n), orange(o), white(w), yellow(y)
ring-number	none(n), one(o), two(t)
ring-type	cobwebby(c), evanescent(e), flaring(f), large(l), none(n), pendant(p), sheathing(s), zone(z)
spore-print-color	black(k), brown(n), buff(b), chocolate(h), green(r), orange(o), purple(u), white(w), yellow(y)
population	abundant(a), clustered(c), numerous(n), scattered(s), several(v), solitary(y)
habitat	grasses(g), leaves(l), meadows(m), paths(p), urban(u), waste(w), woods(d)

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import roc_curve, auc, confusion_matrix

%matplotlib inline
```

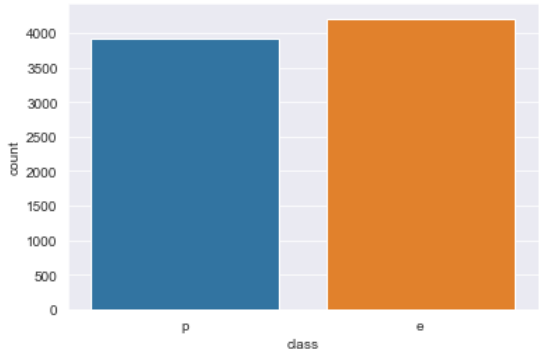
In [2]:
```
hongos = pd.read_csv("mushrooms.csv")
hongos.head()
```

Out[2]:

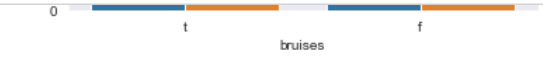
	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	po
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	

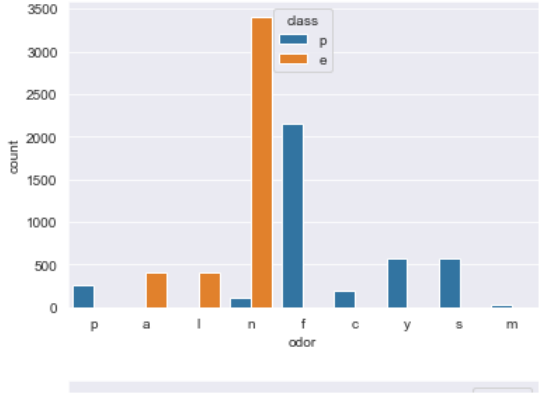
5 rows × 23 columns

In [3]:
```
sns.set_style('darkgrid')
x = hongos['class']
ax = sns.countplot(x=x, data=hongos)
```



In [4]:
```
def plot_data(hue, data):
    for i, col in enumerate(data.columns):
        plt.figure(i)
        ax = sns.countplot(x=data[col], hue=hue, data=data)
plot_data("class", hongos)
```





In [5]:
```
hongos.isnull().sum()
```

Out[5]:
```
class                     0
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gill-size                 0
gill-color                0
stalk-shape               0
stalk-root                0
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-below-ring    0
veil-type                 0
veil-color                0
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
dtype: int64
```

In [6]:
```
le = LabelEncoder()
hongos['class'] = le.fit_transform(hongos['class'])
hongos.head()
```

Out[6]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	po
0	1	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	
1	0	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	
2	0	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	
3	1	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	
4	0	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	

5 rows × 23 columns

In [7]:
```
codificada = pd.get_dummies(hongos)
codificada.head(3)
```

Out[7]:

	class	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	...	population_s	population_v	population_y	hab
0	1	0	0	0	0	0	1	0	0	1	...	1	0	0	
1	0	0	0	0	0	0	1	0	0	1	...	0	0	0	
2	0	1	0	0	0	0	0	0	0	1	...	0	0	0	

3 rows × 118 columns

In [8]:
```
y = hongos['class'].values.reshape(-1, 1)
X = codificada.drop(['class'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [9]:
```
from sklearn.linear_model import LogisticRegression
```

In [10]:
```
logistic_reg = LogisticRegression()

logistic_reg.fit(X_train, y_train.ravel())

y_prob = logistic_reg.predict_proba(X_test)[:,1]
y_pred = np.where(y_prob > 0.5, 1, 0)
```

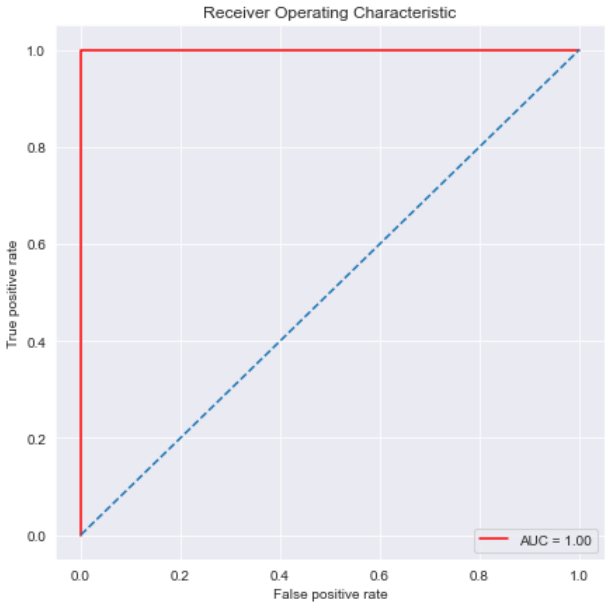
In [11]:
```
log_confusion_matrix = confusion_matrix(y_test, y_pred)
log_confusion_matrix
```

Out[11]:
```
array([[843,   0],
       [  0, 782]], dtype=int64)
```

In [12]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(false_positive_rate, true_positive_rate)
roc_auc
```

Out[12]: 1.0

In [13]:
```
def plot_roc(roc_auc):
    plt.figure(figsize=(7,7))
    plt.title('Receiver Operating Characteristic')
    plt.plot(false_positive_rate, true_positive_rate, color='red', label='AUC = %0.2f' % roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.axis('tight')
    plt.ylabel('True positive rate')
    plt.xlabel('False positive rate')
plot_roc(roc_auc)
```



LDA

In [14]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [15]:
```
lda = LinearDiscriminantAnalysis()

lda.fit(X_train, y_train.ravel())

y_prob_lda = lda.predict_proba(X_test)[:,1]
y_pred_lda = np.where(y_prob_lda > 0.5, 1, 0)
```

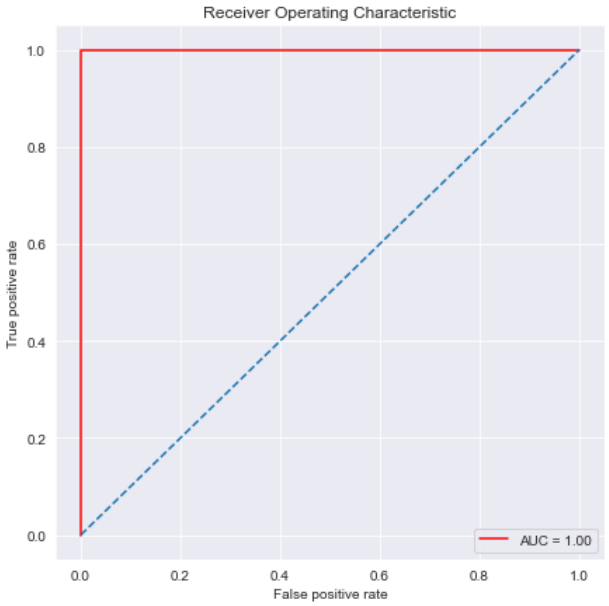
In [16]:
```
lda_confusion_matrix = confusion_matrix(y_test, y_pred_lda)
lda_confusion_matrix
```

Out[16]:
```
array([[843,   0],
       [  0, 782]], dtype=int64)
```

In [17]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_prob_lda)
roc_auc_lda = auc(false_positive_rate, true_positive_rate)
roc_auc_lda
```

Out[17]: 1.0

In [18]:
```
plot_roc(roc_auc_lda)
```



QDA

In [19]:
```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

In [20]:
```
qda = QuadraticDiscriminantAnalysis()

qda.fit(X_train, y_train.ravel())

y_prob_qda = qda.predict_proba(X_test)[:,1]
y_pred_qda = np.where(y_prob_qda > 0.5, 1, 0)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:715: UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
```

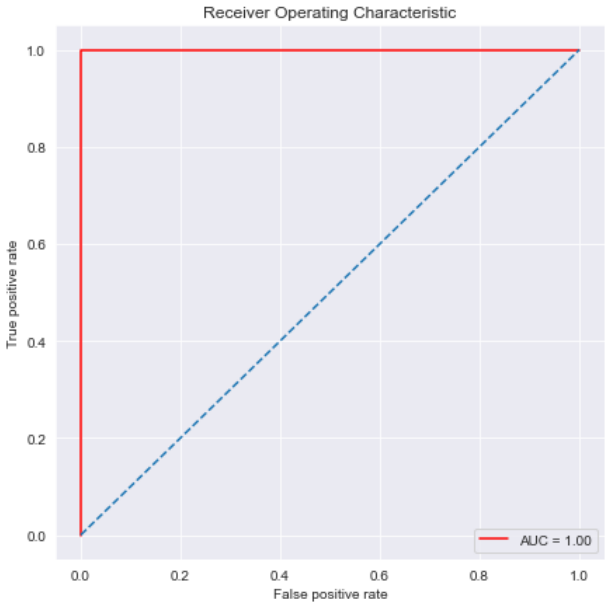
In [21]:
```
qda_confusion_matrix = confusion_matrix(y_test, y_pred_qda)
qda_confusion_matrix
```

Out[21]:
```
array([[843,   0],
       [  0, 782]], dtype=int64)
```

In [22]:
```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_prob_qda)
roc_auc_qda = auc(false_positive_rate, true_positive_rate)
roc_auc_qda
```

Out[22]: 1.0

In [23]:
```
plot_roc(roc_auc_qda)
```



In [ ]:
```

```