```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.metrics import accuracy_score


# Load the data
data = pd.read_csv('quiz2.csv')


# Filter out rows with HDS values other than 0 and 1
data = data[data['HDS'].isin([0, 1])]


data = data.reset_index(drop=True)
data
```

	age	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca	HDS
0	63	0	145	233	1	2	150	0	2.3	0	0
1	67	3	160	286	0	2	108	1	1.5	3	1
2	67	3	120	229	0	2	129	1	2.6	2	1
3	37	2	130	250	0	0	187	0	3.5	0	0
4	41	1	130	204	0	2	172	0	1.4	0	0
...	...	...	...	...	...	...	...	...	...	...	...
281	57	3	140	241	0	0	123	1	0.2	0	1
282	45	0	110	264	0	0	132	0	1.2	0	1
283	68	3	144	193	1	0	141	0	3.4	2	1
284	57	3	130	131	0	0	115	1	1.2	1	1
285	57	1	130	236	0	2	174	0	0.0	1	1

286 rows × 11 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
# (a) Kernel Density Estimation for "chol" based on "HDS" value
plt.figure(figsize=(10, 6))
sns.kdeplot(data[data['HDS'] == 0]['chol'], fill=True, label='HDS=0', kernel='gau')
sns.kdeplot(data[data['HDS'] == 1]['chol'], fill=True, label='HDS=1', kernel='gau')
plt.xlabel('Serum Cholesterol (chol)')
plt.ylabel('Density')
plt.title('Density Estimate of chol for HDS=0 and HDS=1')
plt.legend()
plt.show()
```

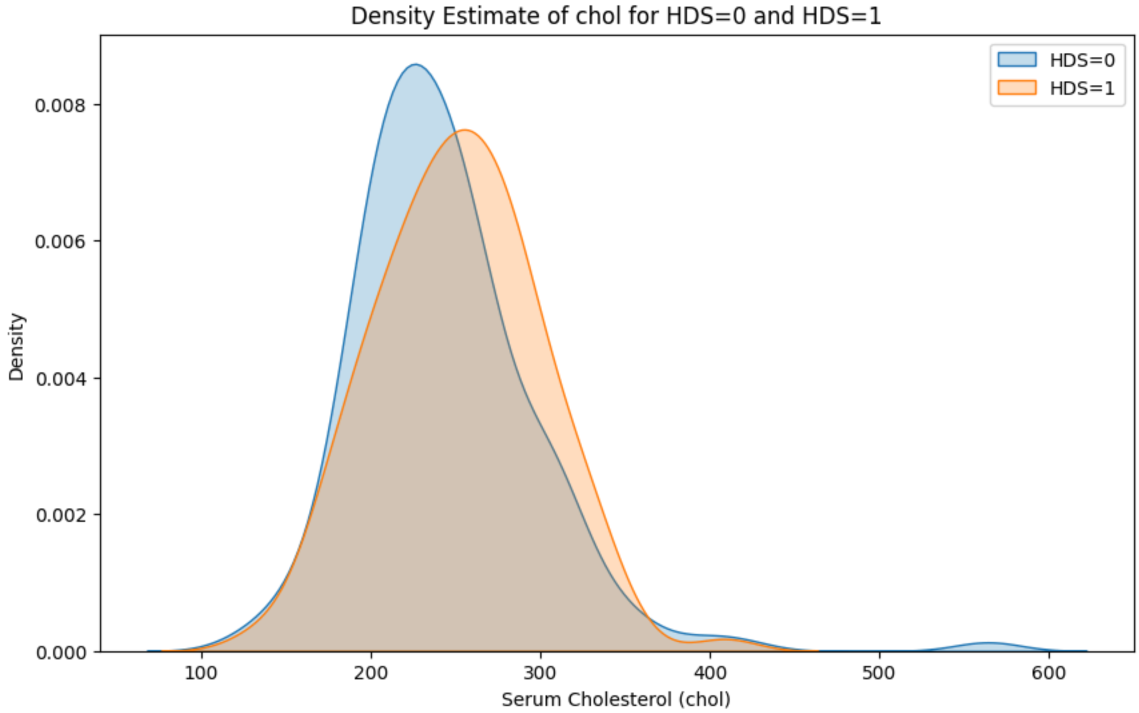
```
<ipython-input-31-c56dc5eb6067>:3: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[data['HDS'] == 0]['chol'], fill=True, label='HDS=0', kernel='gau')
<ipython-input-31-c56dc5eb6067>:4: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[data['HDS'] == 1]['chol'], fill=True, label='HDS=1', kernel='gau')
```




```
# (b) Calculate P(chol > 250) for HDS=1 using KDE
from scipy.stats import gaussian_kde

chol_hds_1 = data[data['HDS'] == 1]['chol']
kde = gaussian_kde(chol_hds_1, bw_method='scott')
p_chol_gt_250 = 1 - kde.integrate_box_1d(250, np.inf)
print(f"P(chol > 250 | HDS=1): {p_chol_gt_250}")
```

```
P(chol > 250 | HDS=1): 0.4900178468127713
```

```
# (c) Split data into training (90%) and test (10%) sets
split_index = int(0.9 * len(data))
train_data = data[:split_index]
test_data = data[split_index:]

X_train = train_data.drop('HDS', axis=1)
y_train = train_data['HDS']
X_test = test_data.drop('HDS', axis=1)
y_test = test_data['HDS']


# (c.i) Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
y_train_pred_lda = lda.predict(X_train)
y_test_pred_lda = lda.predict(X_test)

# (c.ii) Quadratic Discriminant Analysis
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
y_train_pred_qda = qda.predict(X_train)
y_test_pred_qda = qda.predict(X_test)


# (d) Calculate misclassification error rates for both models
train_error_lda = 1 - accuracy_score(y_train, y_train_pred_lda)
test_error_lda = 1 - accuracy_score(y_test, y_test_pred_lda)
train_error_qda = 1 - accuracy_score(y_train, y_train_pred_qda)
test_error_qda = 1 - accuracy_score(y_test, y_test_pred_qda)

print(f"Linear Discriminant Analysis - Training Error: {train_error_lda}, Test Error: {test_error_lda}")
```

```
print(f"Quadratic Discriminant Analysis - Training Error: {train_error_qda}, Test Error: {test_error_qda}")
```

```
Linear Discriminant Analysis - Training Error: 0.1712062256809338, Test Error: 0.27586206896551724
Quadratic Discriminant Analysis - Training Error: 0.1712062256809338, Test Error: 0.24137931034482762
```