In [24]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

In [5]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winec
data = pd.read_csv(url, sep=';')
data
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcol
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	
...	...	...	...	...	...	...	...	...	...	...	
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	1
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	1
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	1

4898 rows × 12 columns

In [6]:
```
# Load the winequality-white dataset
# This example assumes that the dataset has already been preprocessed
# and is available as a pandas dataframe called "data"

# Define the features and target variables
X = data.drop('quality', axis=1)
y = data['quality']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [8]:
```
# Create the logistic regression model
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict the target variable for the test data
y_pred = model.predict(X_test)

# Estimate the class probabilities for the test data
probas = model.predict_proba(X_test)
```

```
# Print the accuracy score for the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```

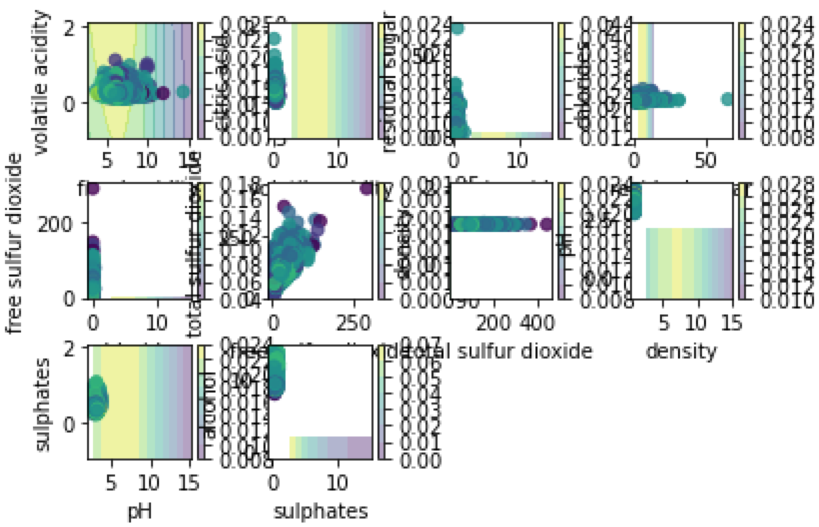
```
Accuracy:  0.45918367346938777
```

```
C:\Users\maham\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [17]:

```
# Create a meshgrid for all 11 features
x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

for i in range(2, 12):
    plt.subplot(3, 4, i-1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    X_plot = np.zeros((xx.shape[0] * xx.shape[1], 11))
    X_plot[:, 0] = xx.ravel()
    X_plot[:, 1] = yy.ravel()

    for j in range(2, 12):
        if i != j:
            X_plot[:, j-1] = X[X.columns[j-1]].mean()

    Z = model.predict_proba(X_plot)[:, 1]
    Z = Z.reshape(xx.shape)

    # Plot the decision boundaries as a contour plot
    plt.contourf(xx, yy, Z, alpha=0.4)
    plt.colorbar()

    # Plot the data points on top of the decision boundaries
    plt.scatter(X.iloc[:, i-2], X.iloc[:, i-1], c=y, alpha=0.8)
    plt.xlabel(X.columns[i-2])
    plt.ylabel(X.columns[i-1])

plt.show()
```

```
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
C:\Users\maham\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X doe
s not have valid feature names, but LogisticRegression was fitted with feature nam
es
  warnings.warn(
```



In [ ]:

In [ ]:

In [ ]: