linear_models_ex_04

June 25, 2024

1 Exercise M4.04

In the previous Module we tuned the hyperparameter `C` of the logistic regression without mentioning that it controls the regularization strength. Later, on the slides on **Intuitions on regularized linear models** we metioned that a small `C` provides a more regularized model, whereas a non-regularized model is obtained with an infinitely large value of `C`. Indeed, `C` behaves as the inverse of the `alpha` coefficient in the `Ridge` model.

In this exercise, we ask you to train a logistic regression classifier using different values of the parameter `C` to find its effects by yourself.

We start by loading the dataset. We only keep the Adelie and Chinstrap classes to keep the discussion simple.

Note

If you want a deeper overview regarding this dataset, you can refer to the Appendix - Datasets description section at the end of this MOOC.

[1]:
```
import pandas as pd

penguins = pd.read_csv("../datasets/penguins_classification.csv")
penguins = (
    penguins.set_index("Species").loc[["Adelie", "Chinstrap"]].reset_index()
)

culmen_columns = ["Culmen Length (mm)", "Culmen Depth (mm)"]
target_column = "Species"
```

[2]:
```
from sklearn.model_selection import train_test_split

penguins_train, penguins_test = train_test_split(
    penguins, random_state=0, test_size=0.4
)

data_train = penguins_train[culmen_columns]
data_test = penguins_test[culmen_columns]

target_train = penguins_train[target_column]
target_test = penguins_test[target_column]
```

We define a function to help us fit a given `model` and plot its decision boundary. We recall that by using a `DecisionBoundaryDisplay` with diverging colormap, `vmin=0` and `vmax=1`, we ensure that the 0.5 probability is mapped to the white color. Equivalently, the darker the color, the closer the predicted probability is to 0 or 1 and the more confident the classifier is in its predictions.

[3]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.inspection import DecisionBoundaryDisplay


def plot_decision_boundary(model):
    model.fit(data_train, target_train)
    accuracy = model.score(data_test, target_test)
    C = model.get_params()["logisticregression__C"]

    disp = DecisionBoundaryDisplay.from_estimator(
        model,
        data_train,
        response_method="predict_proba",
        plot_method="pcolormesh",
        cmap="RdBu_r",
        alpha=0.8,
        vmin=0.0,
        vmax=1.0,
    )
    DecisionBoundaryDisplay.from_estimator(
        model,
        data_train,
        response_method="predict_proba",
        plot_method="contour",
        linestyles="--",
        linewidths=1,
        alpha=0.8,
        levels=[0.5],
        ax=disp.ax_,
    )
    sns.scatterplot(
        data=penguins_train,
        x=culmen_columns[0],
        y=culmen_columns[1],
        hue=target_column,
        palette=["tab:blue", "tab:red"],
        ax=disp.ax_,
    )
    plt.legend(bbox_to_anchor=(1.05, 0.8), loc="upper left")
    plt.title(f"C: {C} \n Accuracy on the test set: {accuracy:.2f}")
```

Let’s now create our predictive model.

[4]:
```python
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

logistic_regression = make_pipeline(StandardScaler(), LogisticRegression())
```

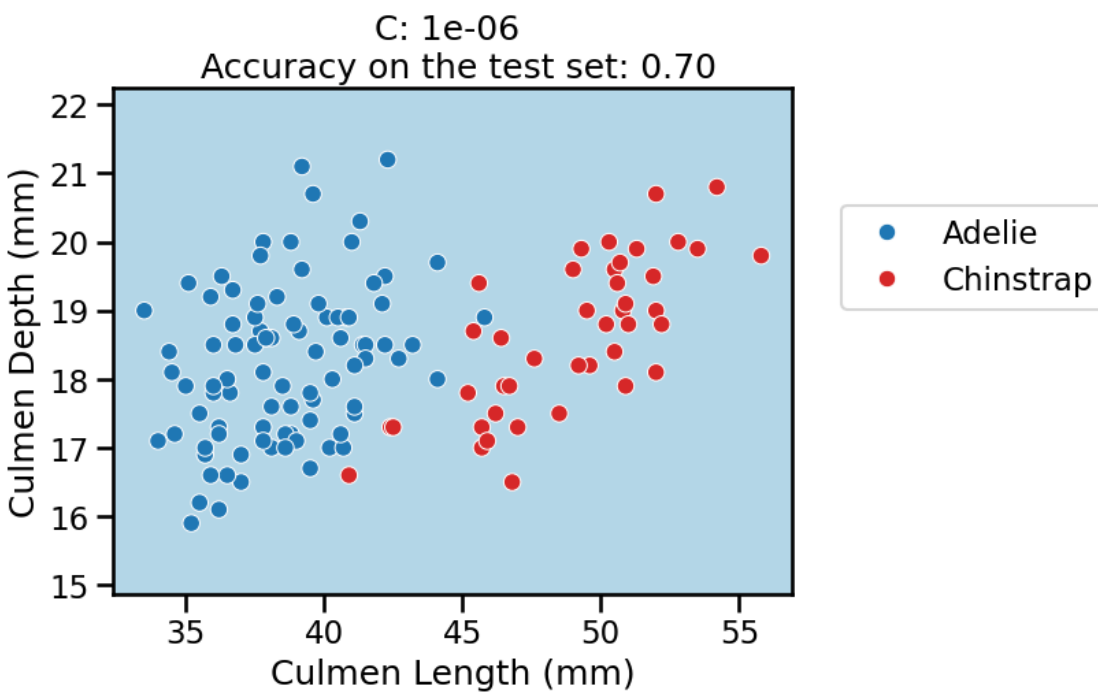
1.1 Influence of the parameter `C` on the decision boundary

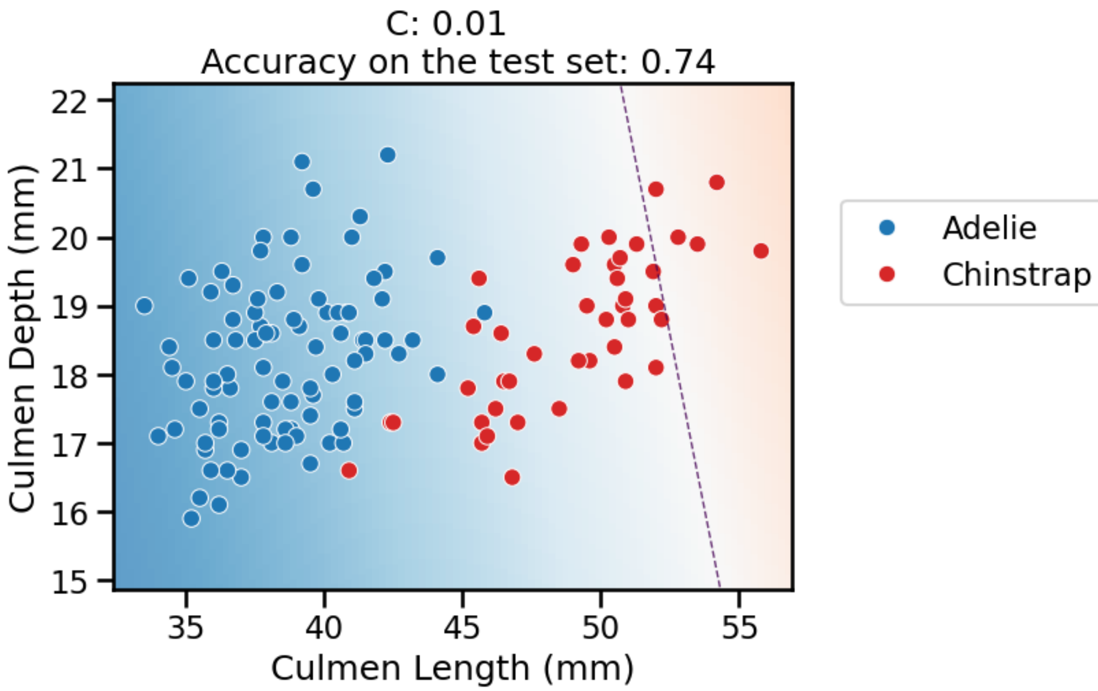
Given the following candidates for the `C` parameter and the `plot_decision_boundary` function, find out the impact of `C` on the classifier's decision boundary.

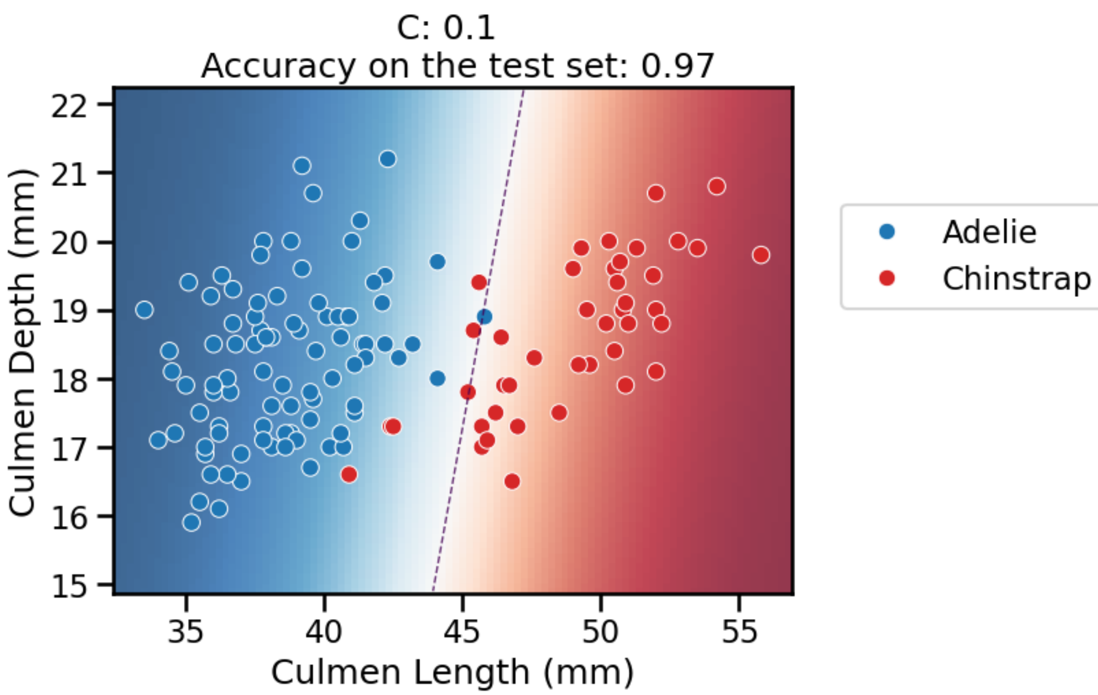
- How does the value of `C` impact the confidence on the predictions?
- How does it impact the underfit/overfit trade-off?
- How does it impact the position and orientation of the decision boundary?

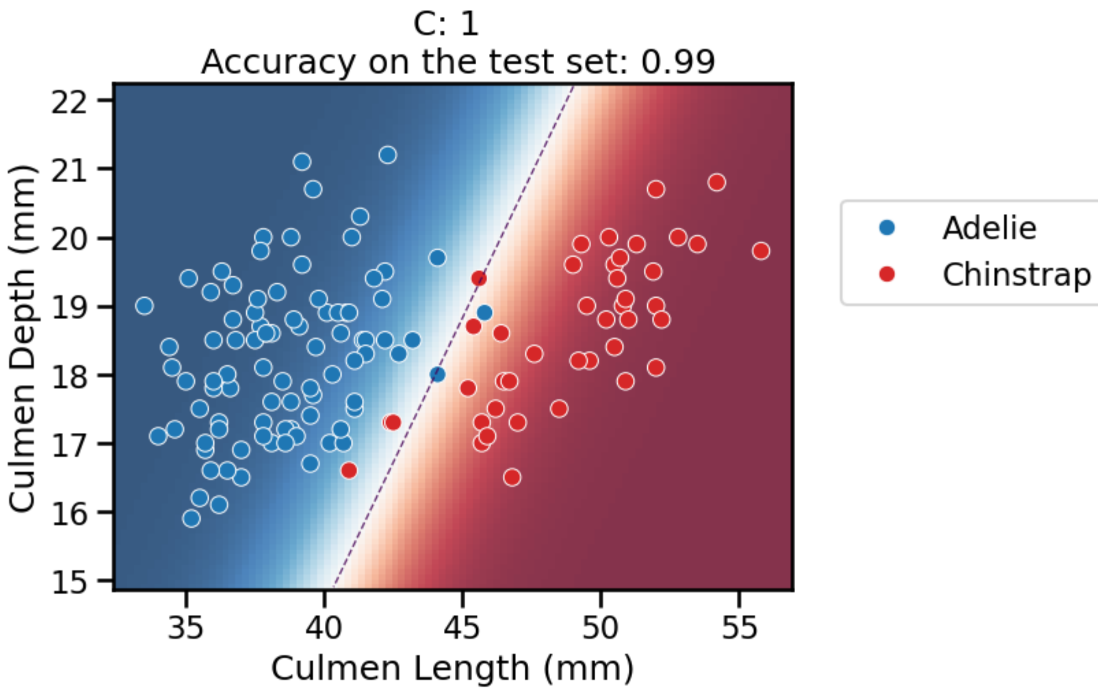
Try to give an interpretation on the reason for such behavior.

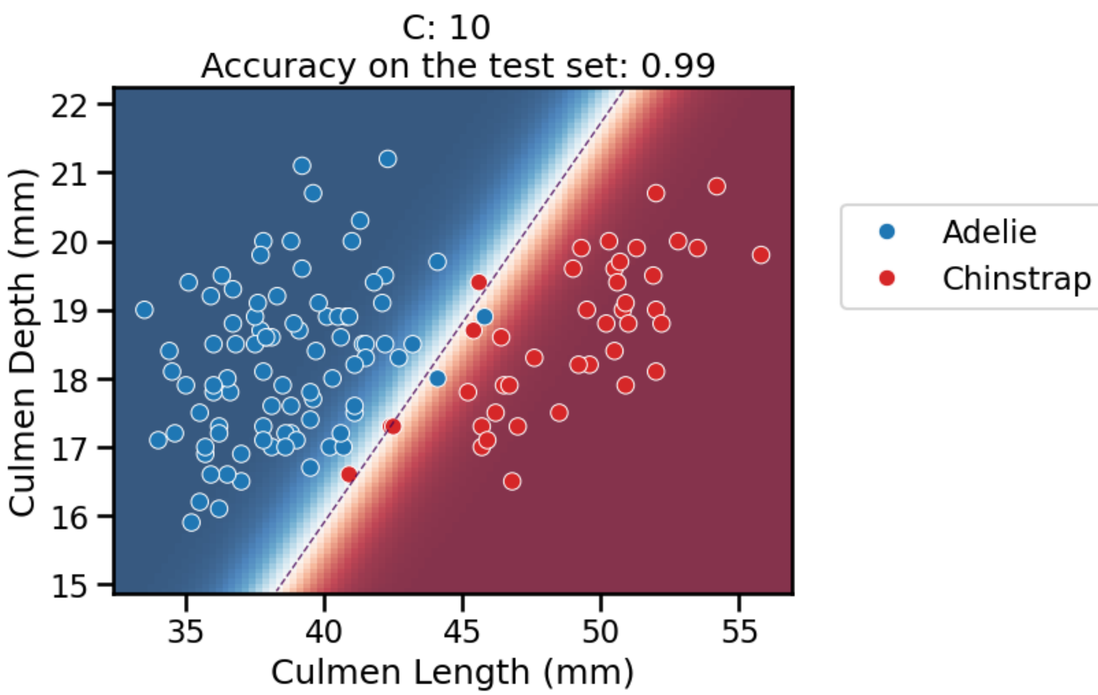
[9]:
```python
Cs = [1e-6, 0.01, 0.1, 1, 10, 100, 1e6]

for C in Cs:
    logistic_regression.set_params(logisticregression__C=C)
    plot_decision_boundary(logistic_regression)
```

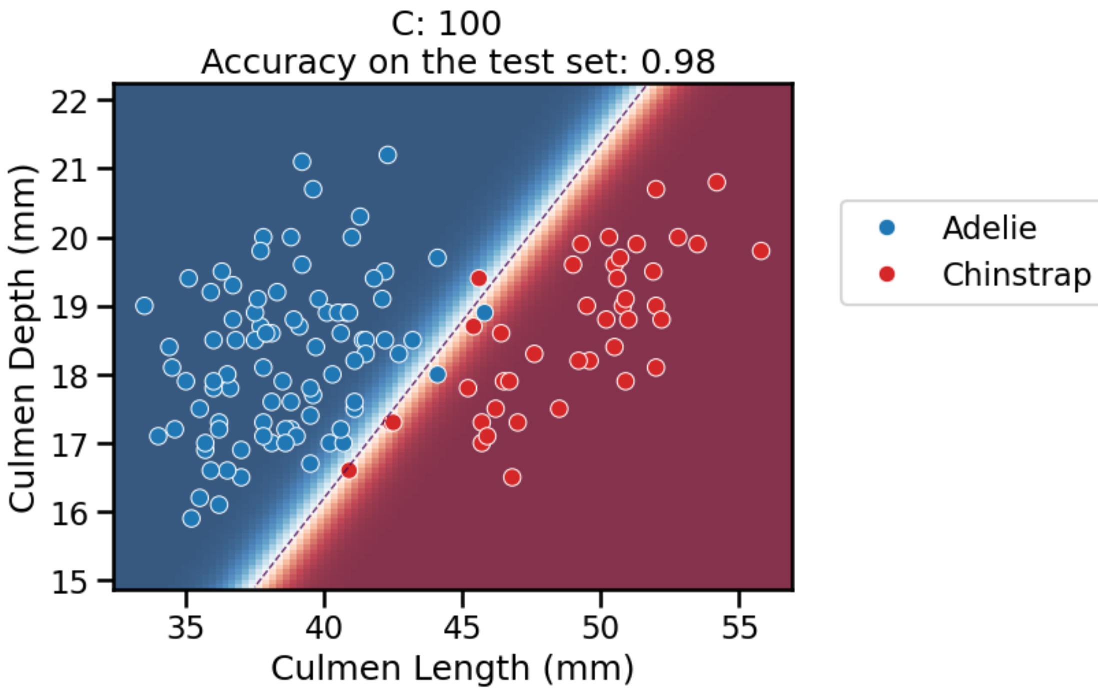


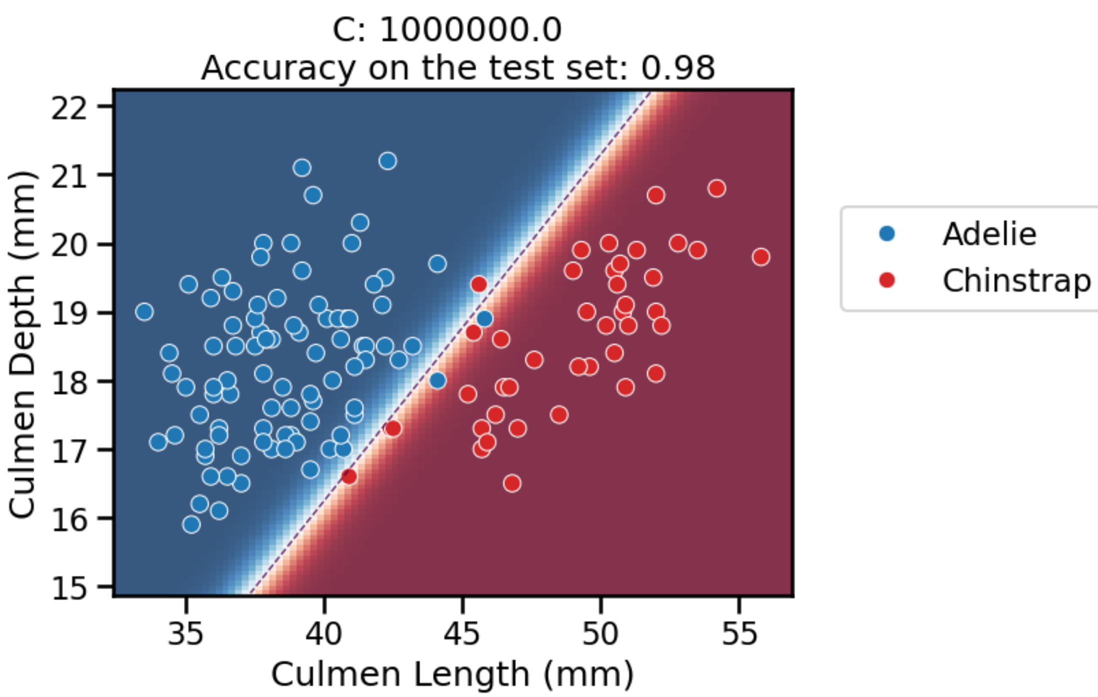

C: 0.01
Accuracy on the test set: 0.74
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 0.1
Accuracy on the test set: 0.97
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 1
Accuracy on the test set: 0.99
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 10
Accuracy on the test set: 0.99
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap

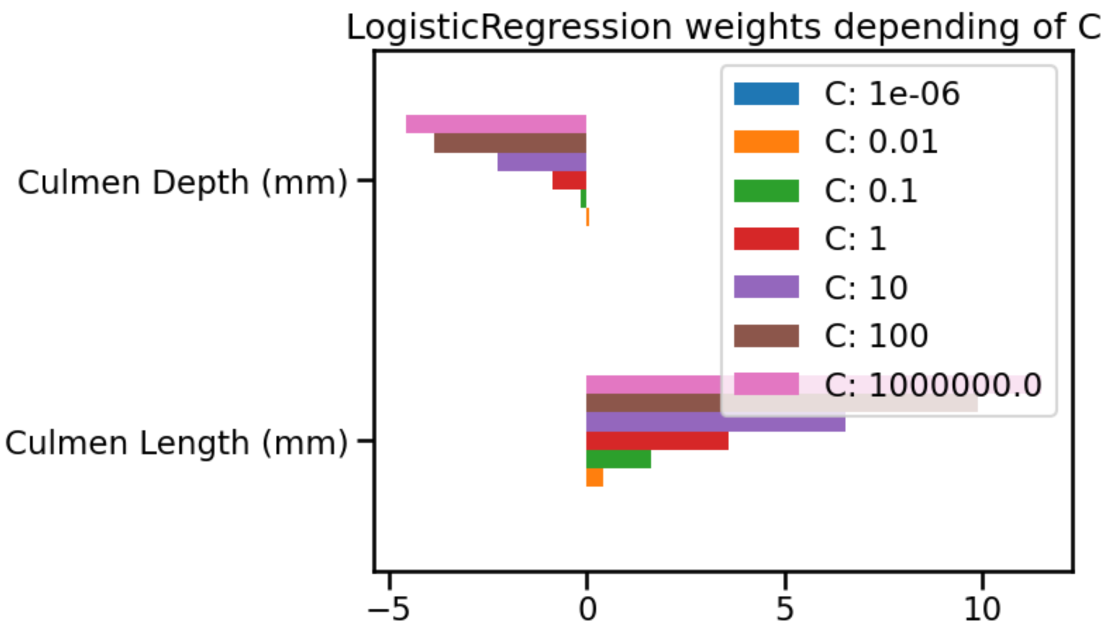

C: 100
Accuracy on the test set: 0.98
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 1000000.0
Accuracy on the test set: 0.98
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap

1.2 Impact of the regularization on the weights

Look at the impact of the `C` hyperparameter on the magnitude of the weights. **Hint**: You can access pipeline steps by name or position. Then you can query the attributes of that step such as `coef_`.

[16]:
```
weights = []
for C in Cs:
    logistic_regression.set_params(logisticregression__C=C)
    logistic_regression.fit(data_train, target_train)
    coefs= logistic_regression[-1].coef_[0]
    weights.append(pd.Series(coefs, index=culmen_columns))
weights = pd.concat(weights, axis=1, keys=[f"C: {C}" for C in Cs])
weights.plot.barh()
_ = plt.title("LogisticRegression weights depending of C")
```



1.3 Impact of the regularization on with non-linear feature engineering

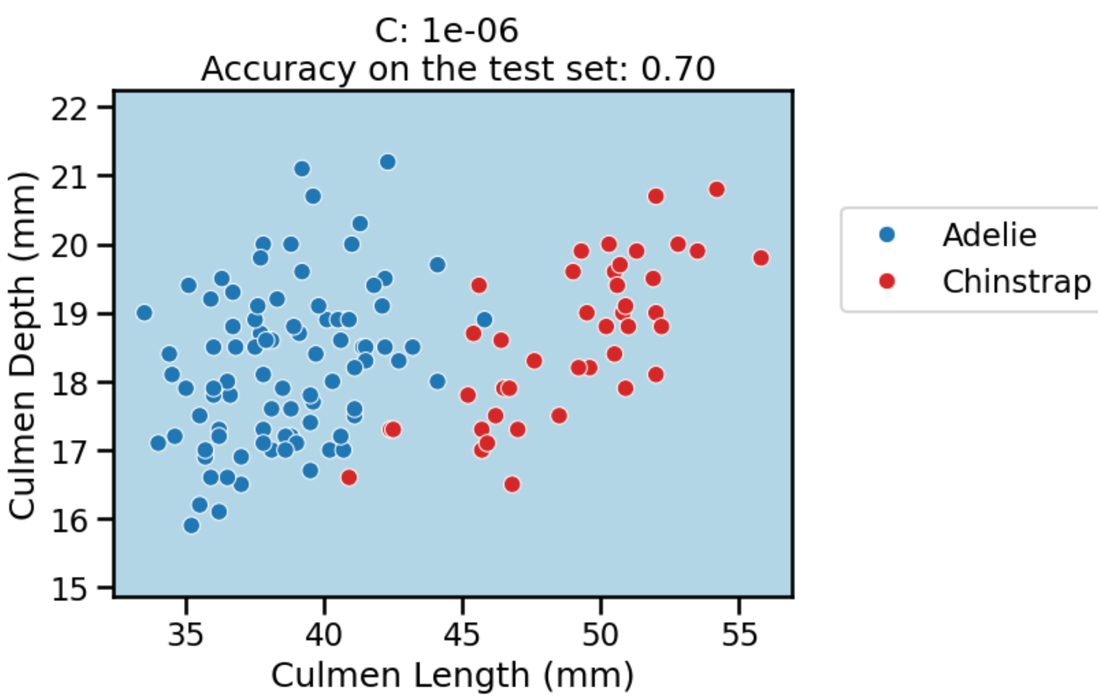
Use the `plot_decision_boundary` function to repeat the experiment using a non-linear feature engineering pipeline. For such purpose, insert `Nystroem(kernel="rbf", gamma=1, n_components=100)` between the `StandardScaler` and the `LogisticRegression` steps.

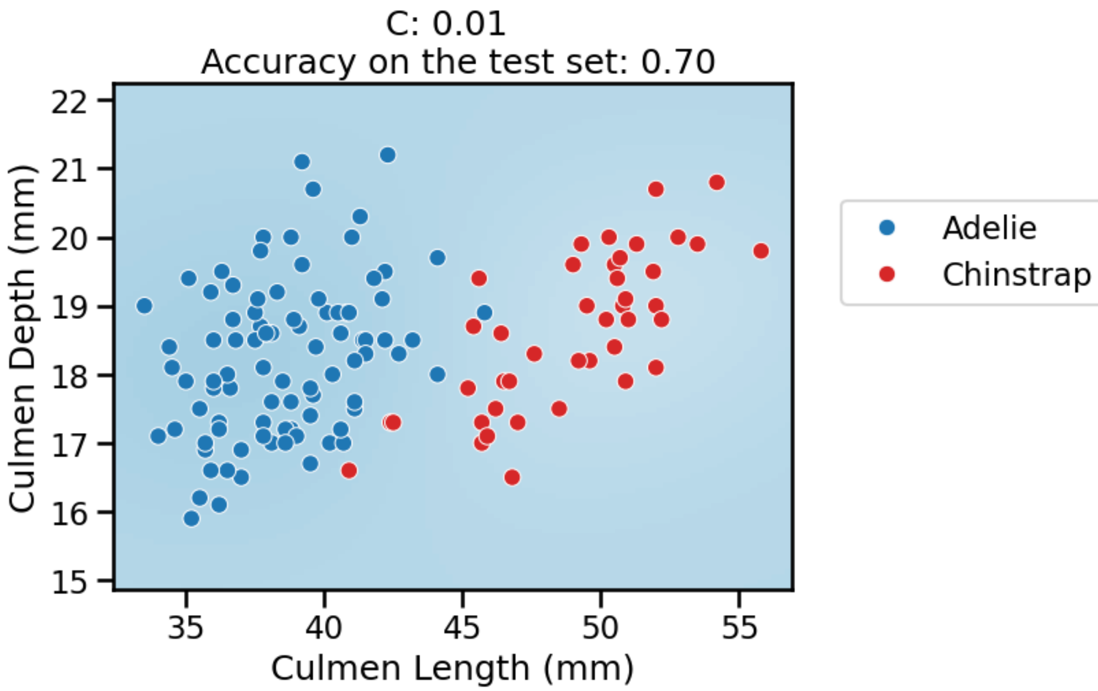
- Does the value of `C` still impact the position of the decision boundary and the confidence of the model?
- What can you say about the impact of `C` on the underfitting vs overfitting trade-off?

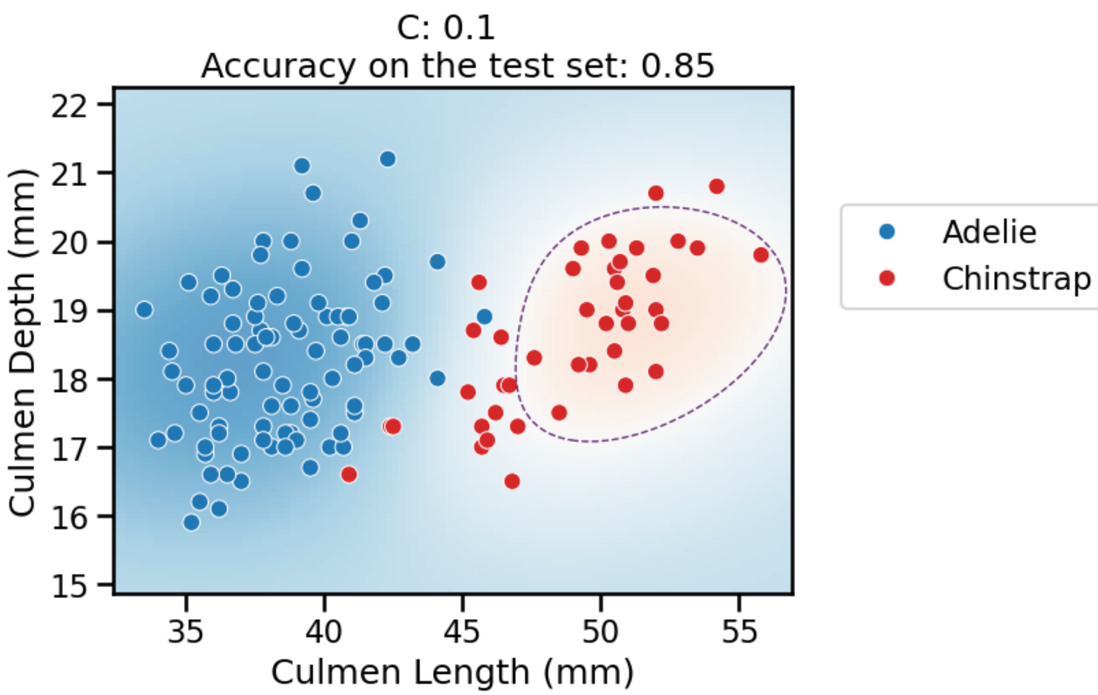
[18]:
```
from sklearn.kernel_approximation import Nystroem
```

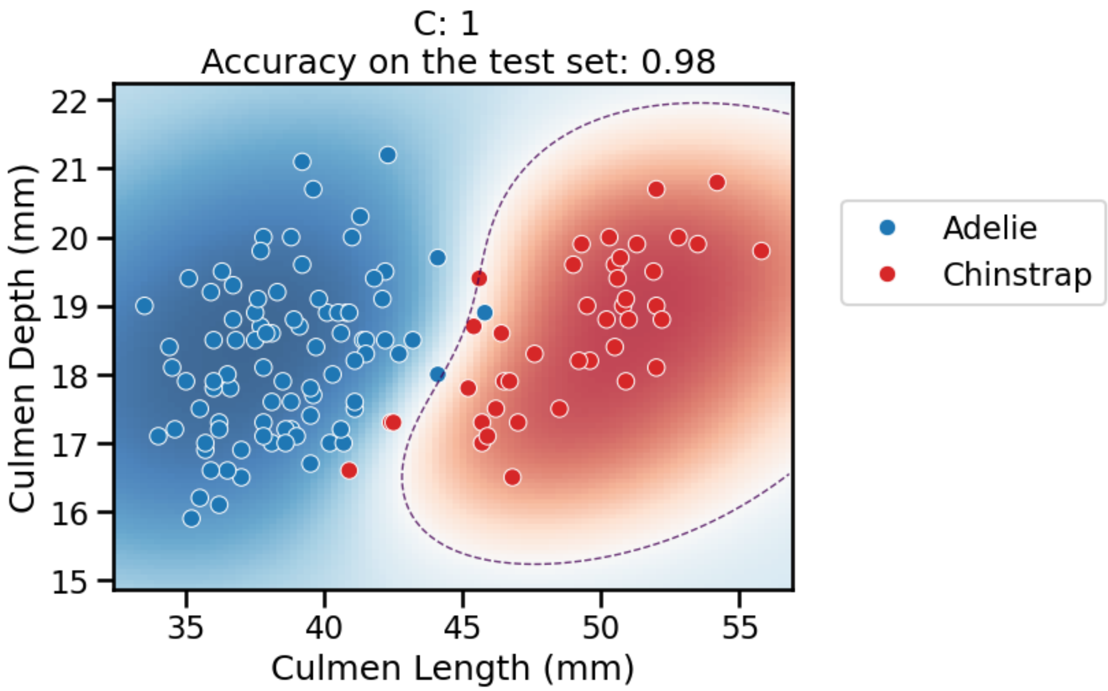
```python
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

model = make_pipeline(StandardScaler(), Nystroem(), LogisticRegression())

Cs = [1e-6, 0.01, 0.1, 1, 10, 100, 1e6]

for C in Cs:
    model.set_params(logisticregression__C=C)
    plot_decision_boundary(model)
```

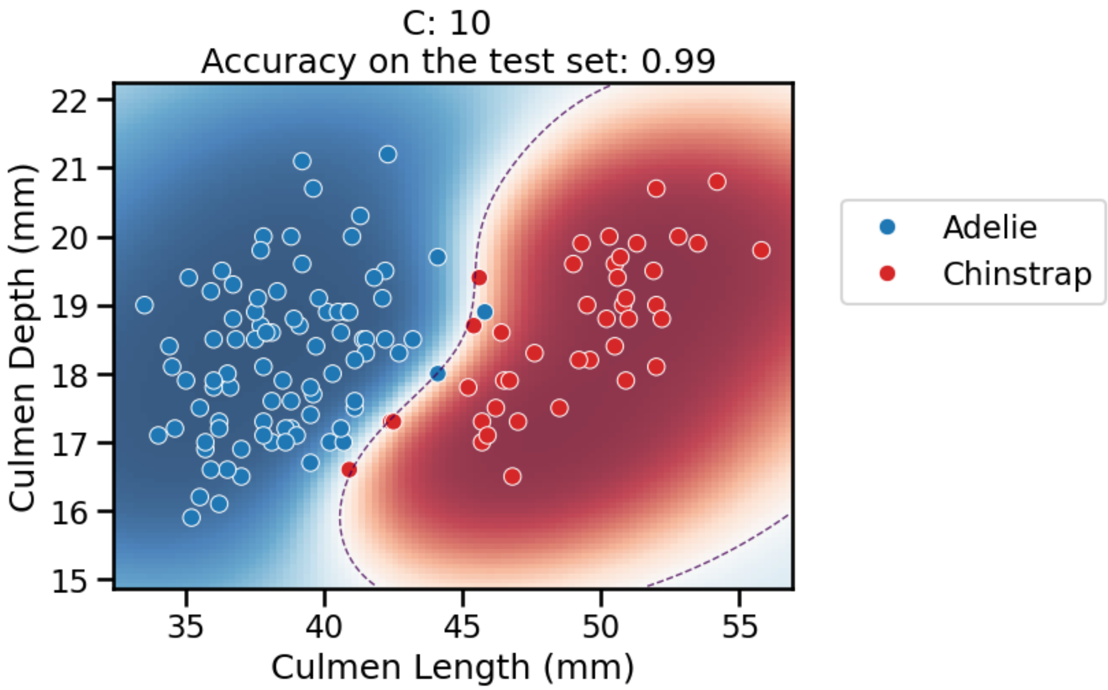
```
/opt/conda/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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(
```

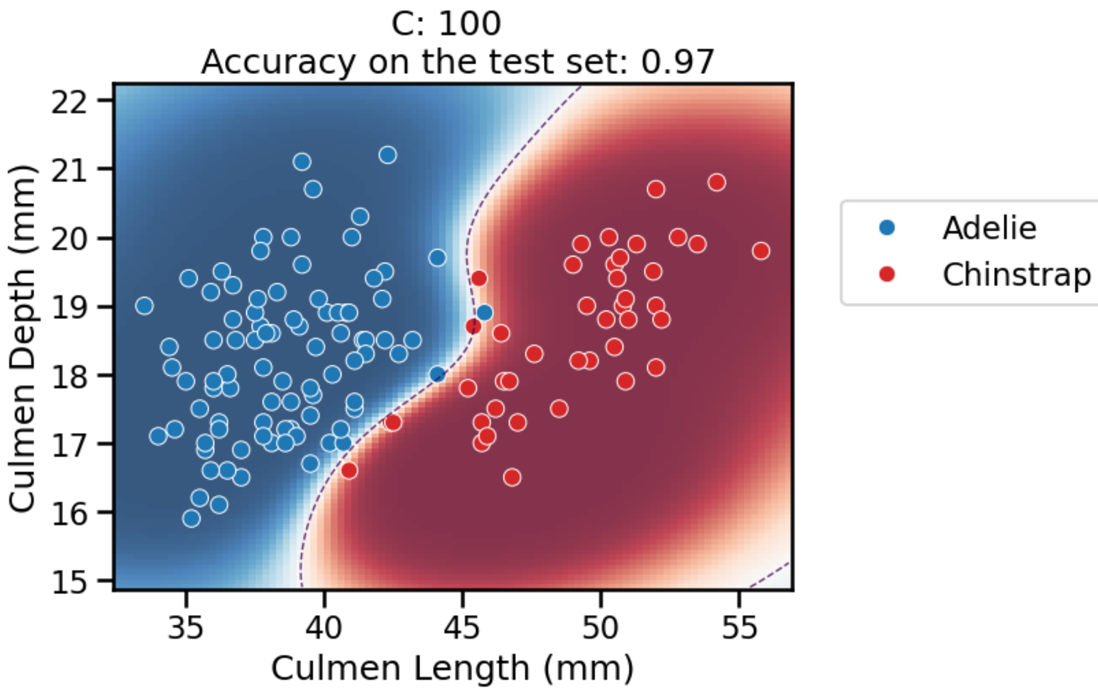


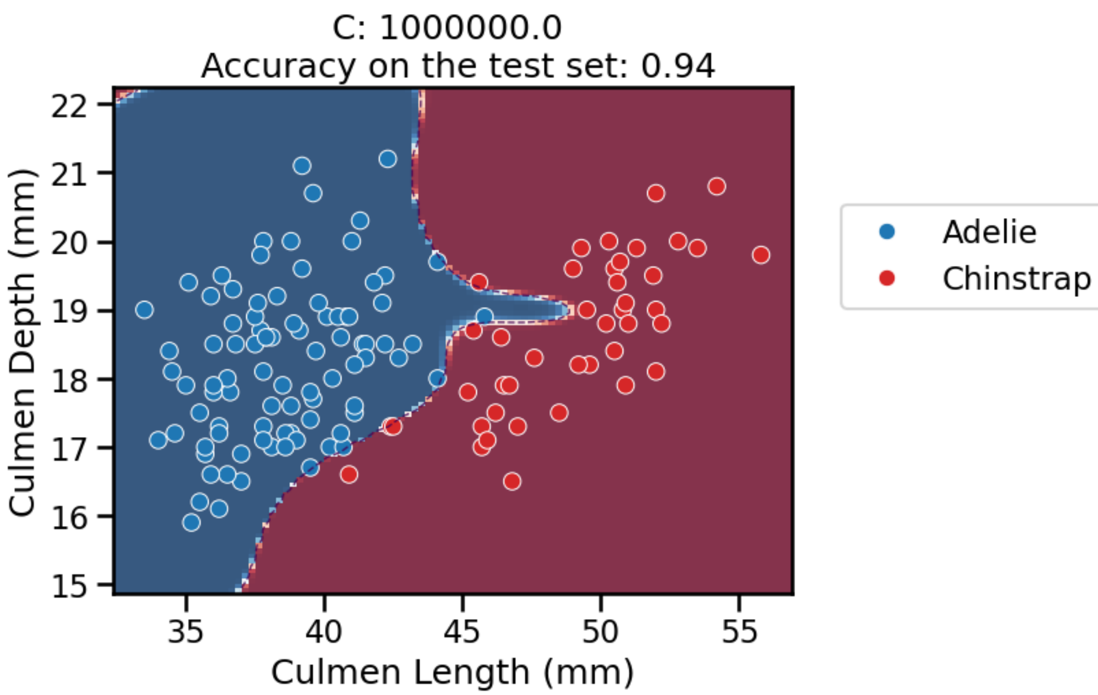

C: 0.01
Accuracy on the test set: 0.70
Culmen Depth (mm)
22
21
20
19
18
17
16
15
35
40
45
50
55
Culmen Length (mm)
Adelie
Chinstrap


C: 0.1
Accuracy on the test set: 0.85
Culmen Depth (mm)
22
21
20
19
18
17
16
15
35
40
45
50
55
Culmen Length (mm)
Adelie
Chinstrap


C: 1
Accuracy on the test set: 0.98
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 10
Accuracy on the test set: 0.99
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 100
Accuracy on the test set: 0.97
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap


C: 1000000.0
Accuracy on the test set: 0.94
Culmen Depth (mm)
Culmen Length (mm)
Adelie
Chinstrap

[ ]: