Autor: Martín de las Heras

Práctica 4

Siguiendo el código del Cap. 4 (Fig. 4-23) de Gerón (2019) generar dicha figura y la misma cambiando la variable **X** a *petal length*, *sepal width* y *sepal length*. Generar también los correspondientes gráficos equivalentes a Fig. 4-24.

En primer lugar, importamos las librerías que utilizaremos a lo largo de la práctica:

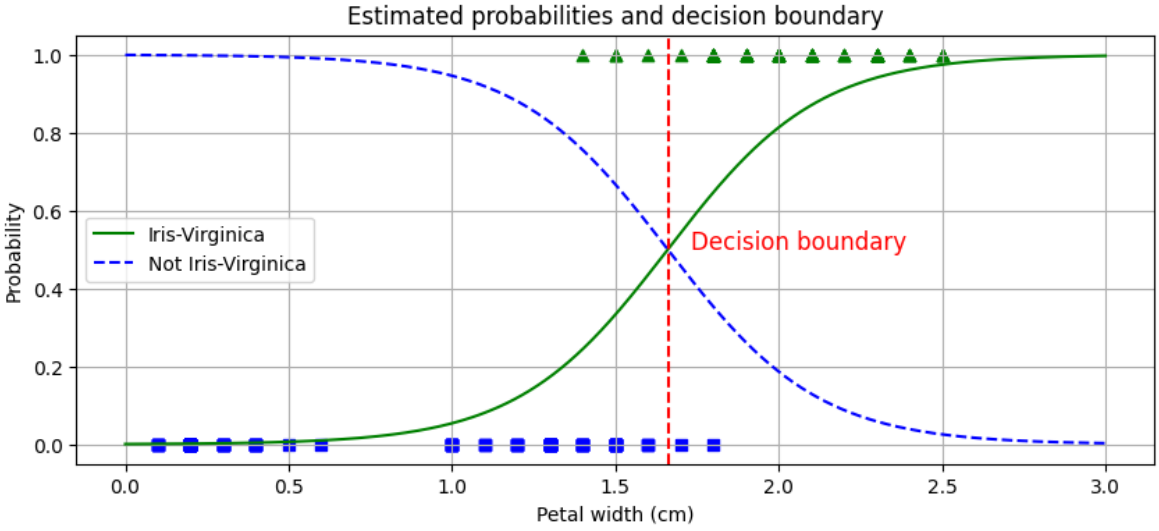
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay
```

Cargamos los datos del dataset, así como las etiquetas de los valores que son *Iris Virginica*.

In [ ]:
```
iris = datasets.load_iris()
petal_data = iris["data"]  # Petal length
y = (iris["target"] == 2).astype(int)  # 1 if Iris-Virginica, else 0
```

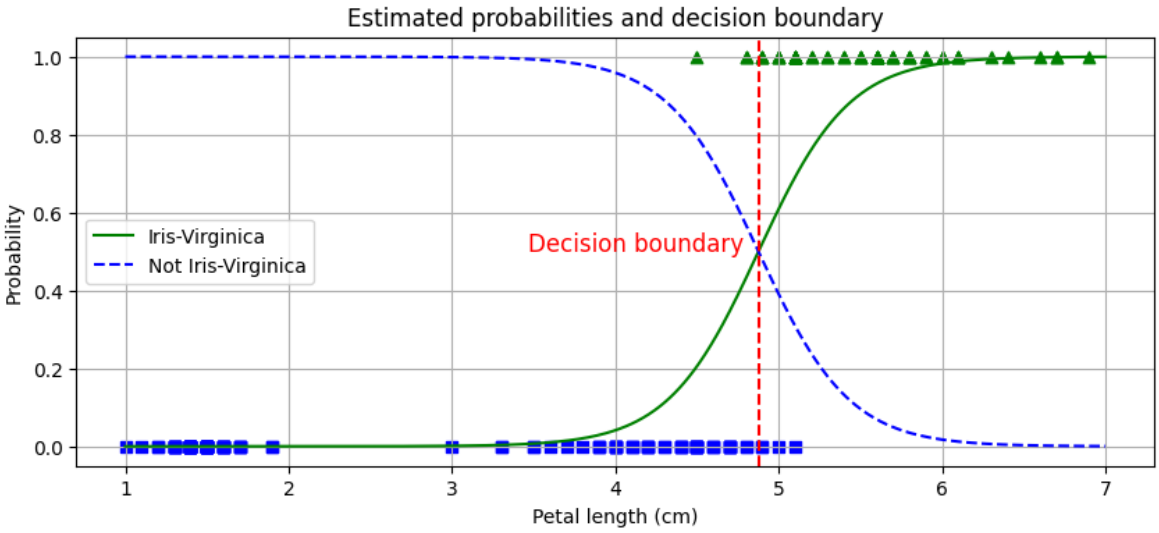
A continuación, graficamos la regresión logística para el *petal width* para *Iris Virginica*.

In [ ]:
```
X = petal_data[:,3:]  # Petal width
log_reg = LogisticRegression()
log_reg.fit(X, y)

X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)

plt.figure(figsize=(10, 4))
plt.grid()
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
plt.xlabel("Petal width (cm)")
plt.ylabel("Probability")
plt.legend(loc="center left")
plt.title("Estimated probabilities and decision boundary")

decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]
plt.axvline(x=decision_boundary, color='red', linestyle='--')
plt.text(decision_boundary+0.4, 0.5, "Decision boundary", fontsize=12, color="re

plt.scatter(X[y==1],y[y==1], marker='^', c='g')
plt.scatter(X[y==0],y[y==0], marker='s', c='b')

plt.show()
```



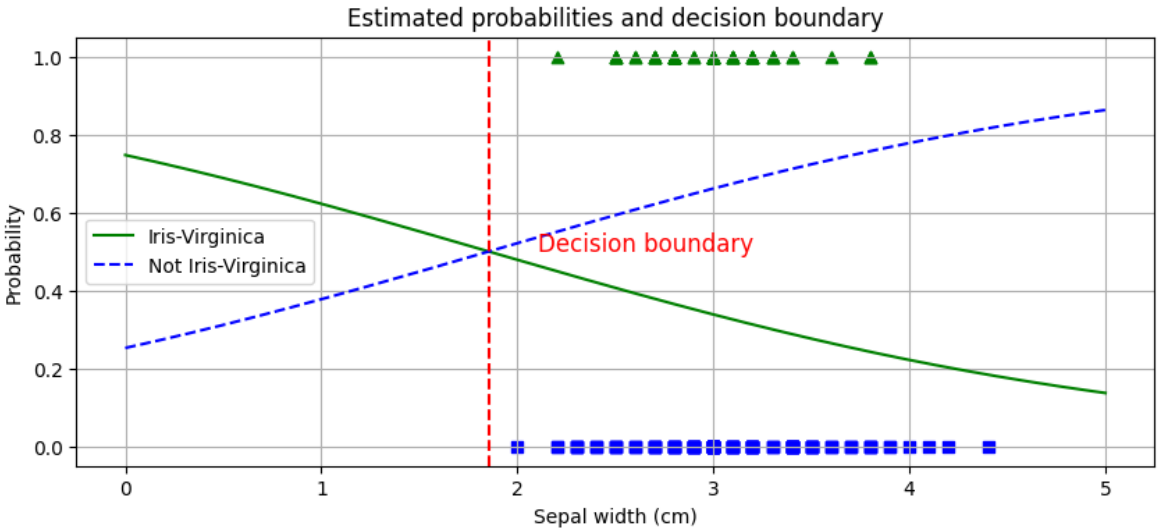
Repetimos para *petal length*:

In [ ]:
```
X = petal_data[:,2:3]
log_reg = LogisticRegression()
log_reg.fit(X, y)

X_new = np.linspace(1, 7, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)

plt.figure(figsize=(10, 4))
plt.grid()
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
plt.xlabel("Petal length (cm)")
plt.ylabel("Probability")
plt.legend(loc="center left")
plt.title("Estimated probabilities and decision boundary")

decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]
plt.axvline(x=decision_boundary, color='red', linestyle='--')
plt.text(decision_boundary-0.75, 0.5, "Decision boundary", fontsize=12, color="r

plt.scatter(X[y==1],y[y==1], marker='^', c='g')
plt.scatter(X[y==0],y[y==0], marker='s', c='b')

plt.show()
```



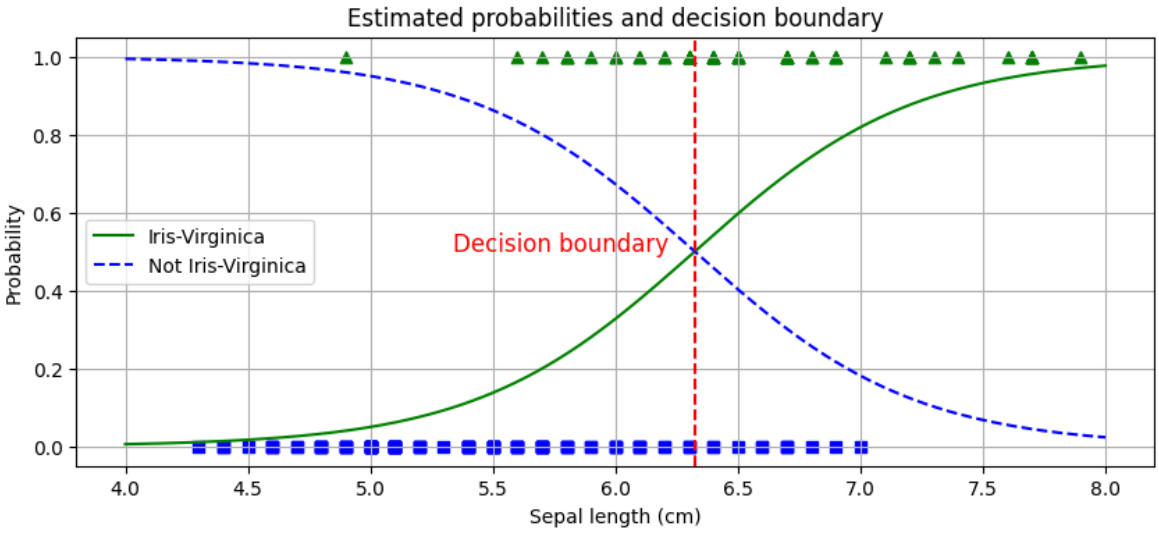
Para *sepal width*:

In [ ]:
```
X = petal_data[:,1:2]
log_reg = LogisticRegression()
log_reg.fit(X, y)

X_new = np.linspace(0, 5, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)

plt.figure(figsize=(10, 4))
plt.grid()
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
plt.xlabel("Sepal width (cm)")
plt.ylabel("Probability")
plt.legend(loc="center left")
plt.title("Estimated probabilities and decision boundary")

decision_boundary = X_new[y_proba[:, 1] < 0.5][0]
plt.axvline(x=decision_boundary, color='red', linestyle='--')
plt.text(decision_boundary+0.8, 0.5, "Decision boundary", fontsize=12, color="re

plt.scatter(X[y==1],y[y==1], marker='^', c='g')
plt.scatter(X[y==0],y[y==0], marker='s', c='b')

plt.show()
```



Y por último para *sepal length*:

In [ ]:
```
X = petal_data[:,0:1]
log_reg = LogisticRegression()
log_reg.fit(X, y)

X_new = np.linspace(4, 8, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)

plt.figure(figsize=(10, 4))
plt.grid()
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
plt.xlabel("Sepal length (cm)")
plt.ylabel("Probability")
plt.legend(loc="center left")
plt.title("Estimated probabilities and decision boundary")

decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]
plt.axvline(x=decision_boundary, color='red', linestyle='--')
plt.text(decision_boundary-0.55, 0.5, "Decision boundary", fontsize=12, color="r

plt.scatter(X[y==1],y[y==1], marker='^', c='g')
plt.scatter(X[y==0],y[y==0], marker='s', c='b')

plt.show()
```

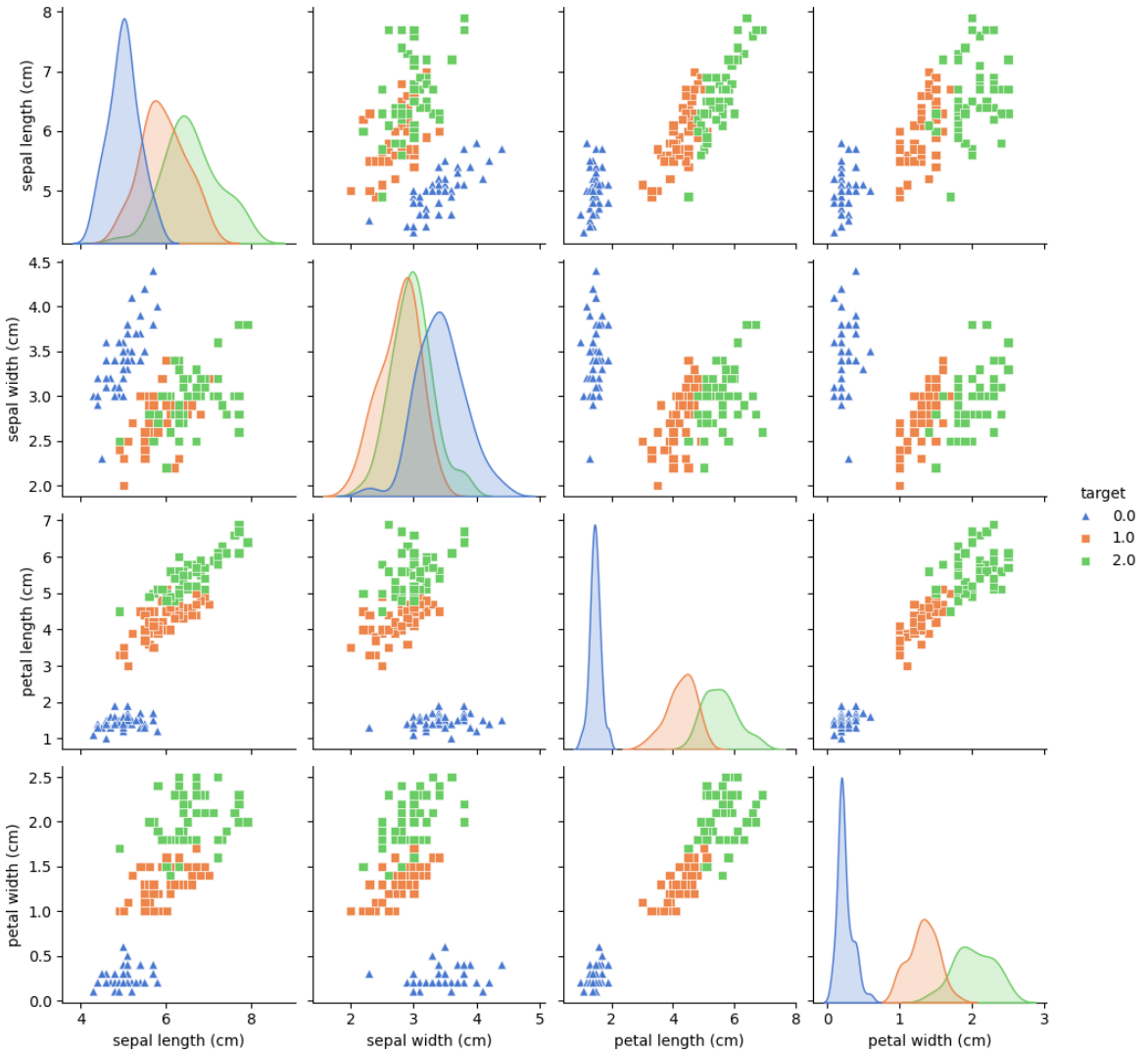


La línea que hemos llamado *Decision boundary* no es otra que el valor a partir del cual la regresión logística pasa de clasificar de un tipo de Iris al otro, en este caso puesto a 0.5 de probabilidad, pero esto se podría cambiar con facilidad en el código.

Vemos que lo consigue con relativa facilidad para todos menos para *sepal width*, pero también vemos que en el caso de *sepal width*, las muestras que tenemos tienen casi los mismos valores, por lo que sería muy difícil discernir entre las dos clases.

A continuación, graficamos las relaciones entre todas las categorías que tenemos de *Iris*, de esa manera podemos distinguir los grupos que se forman.

In [ ]:
```
import seaborn as sns
import pandas as pd
df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                     columns= iris['feature_names'] + ['target'])
sns.pairplot(df.iloc[:,:],hue='target', palette='muted', markers=['^','s','s']);
```

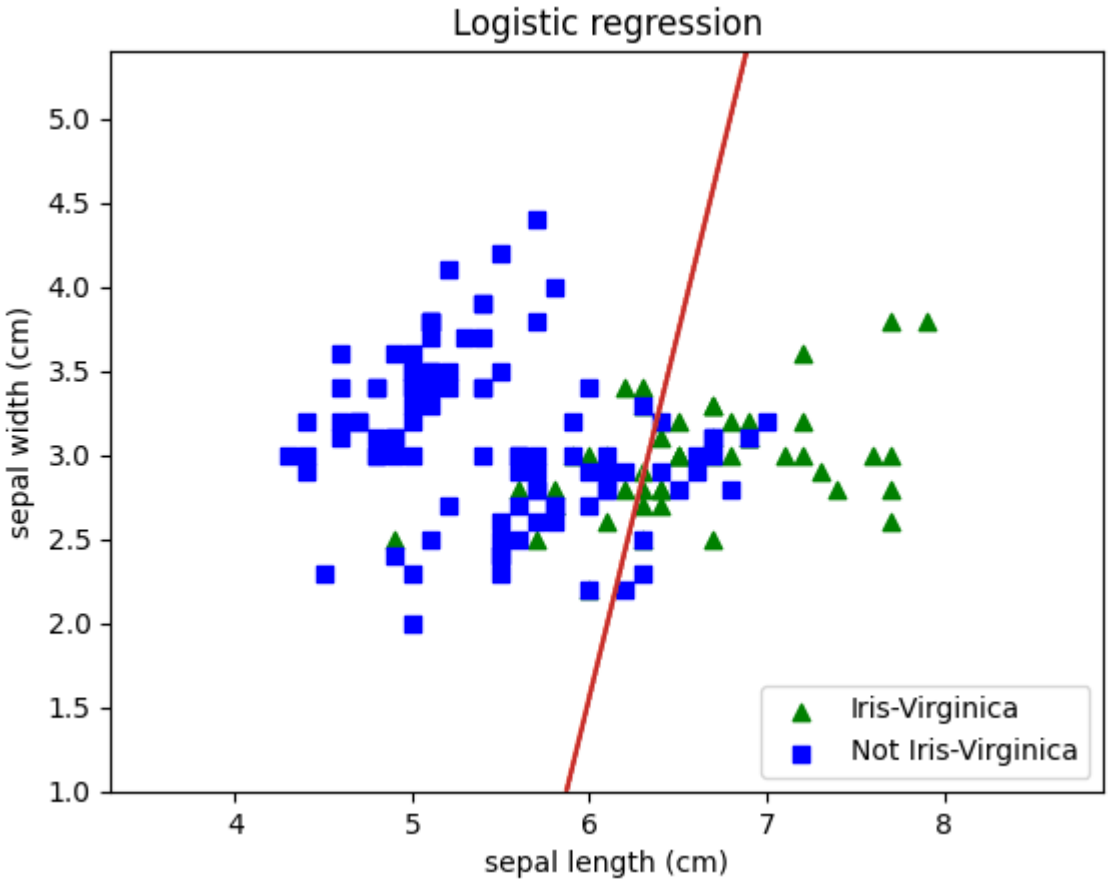


Vemos que, al igual que en la regresión logística, los valores de *sepal width* son los más solapados, por lo que será el atributo desde el cual es más difícil diferenciar entre las clases.

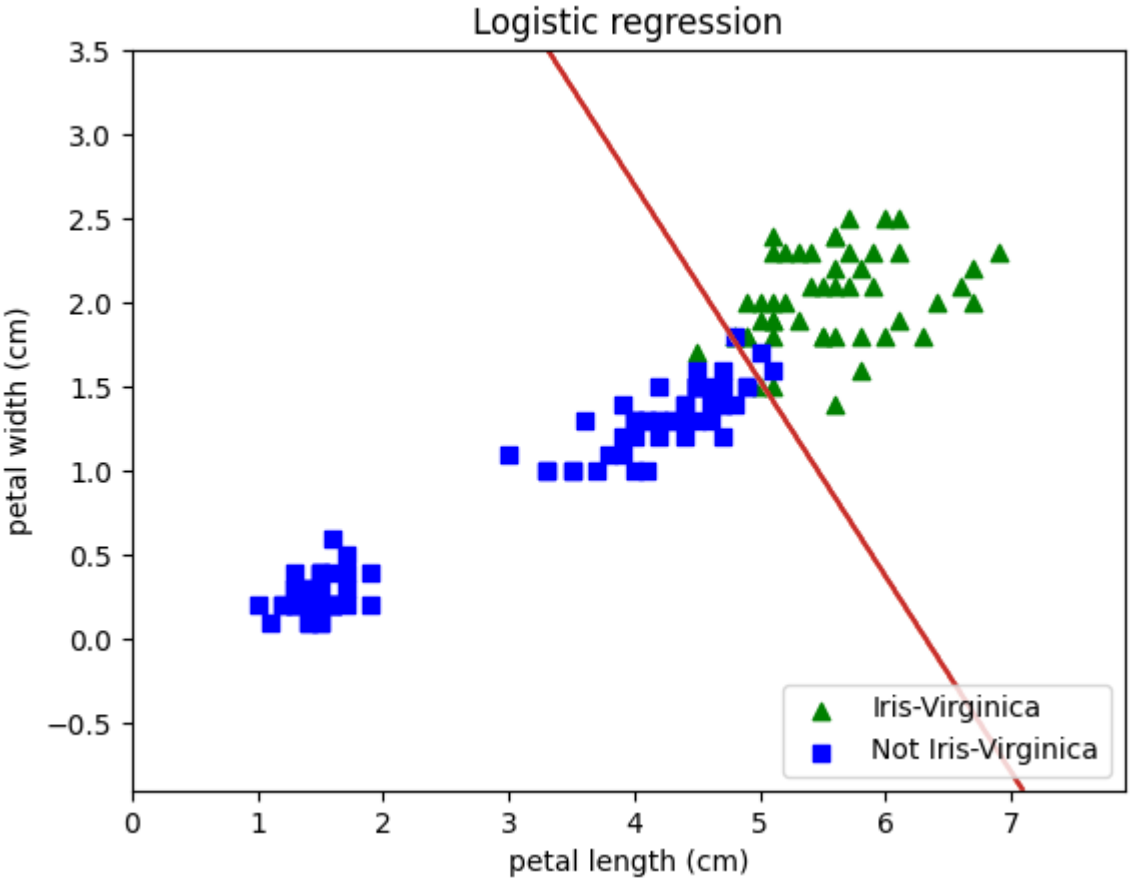
Para cualquiera de estas parejas de atributos, podemos realizar una regresión lineal que nos separe las dos entre *Iris virginica* y las demás de la siguiente manera:

In [ ]:
```
X = iris.data[:, :2]
log_reg = LogisticRegression()
log_reg.fit(X, y)

DecisionBoundaryDisplay.from_estimator(
    log_reg,
    X,
    grid_resolution=1000,
    cmap='Reds',
    response_method="predict",
    plot_method="contour",
    xlabel="sepal length (cm)",
    ylabel="sepal width (cm)",
    alpha=0.5,
)

# Plot also the training points
plt.scatter(df['sepal length (cm)'][df['target']==2.0], df['sepal width (cm)'][d
```

```
plt.scatter(df['sepal length (cm)'][df['target']!=2.0], df['sepal width (cm)'][d
plt.legend(loc="lower right")
plt.title("Logistic regression");
```



In [ ]:
```
X = iris.data[:, 2:]
log_reg = LogisticRegression()
log_reg.fit(X, y)

DecisionBoundaryDisplay.from_estimator(
    log_reg,
    X,
    grid_resolution=1000,
    cmap='Reds',
    response_method="predict",
    plot_method="contour",
    xlabel="petal length (cm)",
    ylabel="petal width (cm)",
    alpha=0.5,
)

# Plot also the training points
plt.scatter(df['petal length (cm)'][df['target']==2.0], df['petal width (cm)'][d
plt.scatter(df['petal length (cm)'][df['target']!=2.0], df['petal width (cm)'][d
plt.legend(loc="lower right")
plt.title("Logistic regression");
```



Esto se puede extender a cualquier pareja de atributos, obteniendo los distintos *decission boundaries*. De la misma manera que en el apartado anterior, esto está señalado por la línea roja para los valores en los que supera el 50%.

Como hemos visto, la regresión logística es capaz de separar y clasificar los tipos de Iris en función de los atributos que le especifiquemos de manera muy fácil. En las gráficas agrupadas que hemos visto antes, podemos imaginar dónde irían las distintas *decission boundaries*, para luego con el código que hemos usado, sustituir los valores que queremos utilizar para la regresión, y hacer la clasificación.

De la misma manera, esto se puede hacer para distinguir entre los tres tipos de Iris, no solamente entre los que son *Iris-Virginica* y los que no.