Regressione lineare con regolarizzazione L_2

Consideriamo il modello di **regressione lineare**, con un termime di regolarizzazione L_2

(Ridge regression)

$$E_{\text{aug}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \underset{1\times d}{\boldsymbol{\varphi}^{\top}(i)}\underset{d\times 1}{\boldsymbol{\theta}}\right)^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

$$= \frac{1}{N}\underset{N\times 1}{\|Y} - \underset{N\times d}{X}\underset{d\times 1}{\cdot\,\boldsymbol{\theta}}\|_2^2 + \lambda_{\text{reg}} \cdot \|\boldsymbol{\theta}\|_2^2$$

Si può dimostrare che la stima in forma chiusa dei parametri si ottiene come:

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}_{\text{reg}}} = \left(\underset{d\times d}{X^{\top}X} + \lambda_{\text{reg}} \cdot \underset{d\times d}{I_d}\right)^{-1}\underset{d\times N}{X^{\top}}\underset{N\times 1}{Y}$$

Ridge regression e gradient descent

Supponiamo di avere **solo 2 parametri** $\boldsymbol{\theta} = [\theta_0, \theta_1]^\top \in \mathbb{R}^{2\times 1}$ per semplicità

$$E_{\text{aug}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i)\right)^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

E' possibile implementare l'algoritmo del gradient descent come:

For {

$$\theta_0 = \theta_0 - \alpha \cdot 2\left[\frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i)\right) \cdot (-1) + \lambda_{\text{reg}} \cdot \theta_0\right]$$

$$\theta_1 = \theta_1 - \alpha \cdot 2\left[\frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i)\right) \cdot \left(-\varphi_1(i)\right) + \lambda_{\text{reg}} \cdot \theta_1\right]$$

}

Regressione logistica con regolarizzazione L_2

Consideriamo il modello di **regressione logistica**, con un termime di regolarizzazione L_2

$$E_{\text{aug}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \big(y(i) \cdot \ln \pi(i;\boldsymbol{\theta}) + \big(1 - y(i)\big) \cdot \ln[1 - \pi(i;\boldsymbol{\theta})] \big) + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1} \big(\theta_j\big)^2$$

$$\pi(i) \equiv \frac{1}{1 + e^{-\boldsymbol{\varphi}^{\mathsf{T}}(i)\boldsymbol{\theta}}} = P\big(y(i) = 1 | \boldsymbol{\varphi}(i)\big)$$

E' possibile implementare l'algoritmo del gradient descent come:

For {

$$\theta_0 = \theta_0 - \alpha \cdot \sum_{i=1}^{N} 1 \cdot \big(\pi(i) - y(i)\big) + 2\lambda_{\text{reg}} \cdot \theta_0$$

$$\vdots$$

$$\theta_{d-1} = \theta_{d-1} - \alpha \cdot \sum_{i=1}^{N} \varphi_{d-1}(i) \cdot \big(\pi(i) - y(i)\big) + 2\lambda_{\text{reg}} \cdot \theta_{d-1}$$

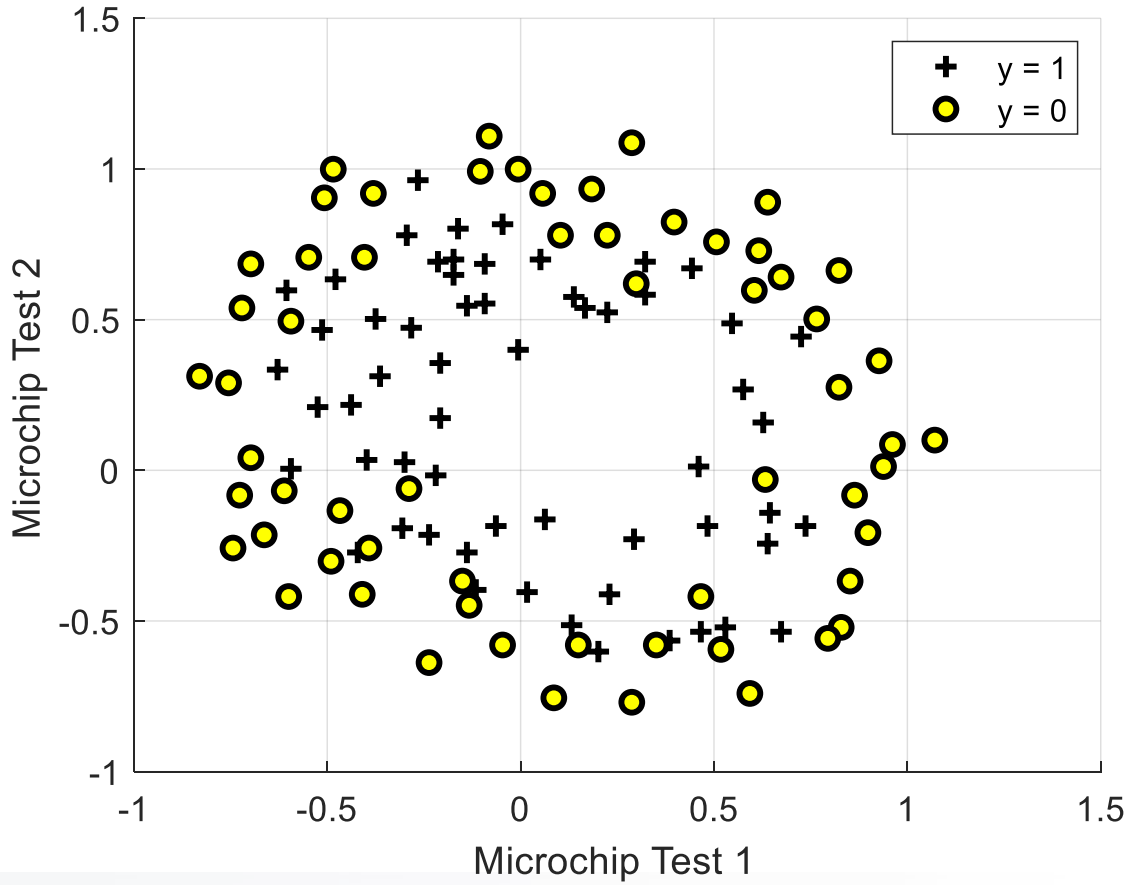
}

Esercizio: classificare microchips difettosi

Vogliamo rilevare se un microchip è **difettoso** in base ai risultati di due test di qualità alla fine della linea di produzione, tramite un modello di **regressione logistica**

- Ogni microchip è descritto dalle seguenti features
  - ✓ φ_1: Risultato del test 1
  - ✓ φ_2: Risultato del test 2

- Il dataset è costituito da $N = 118$ microchips

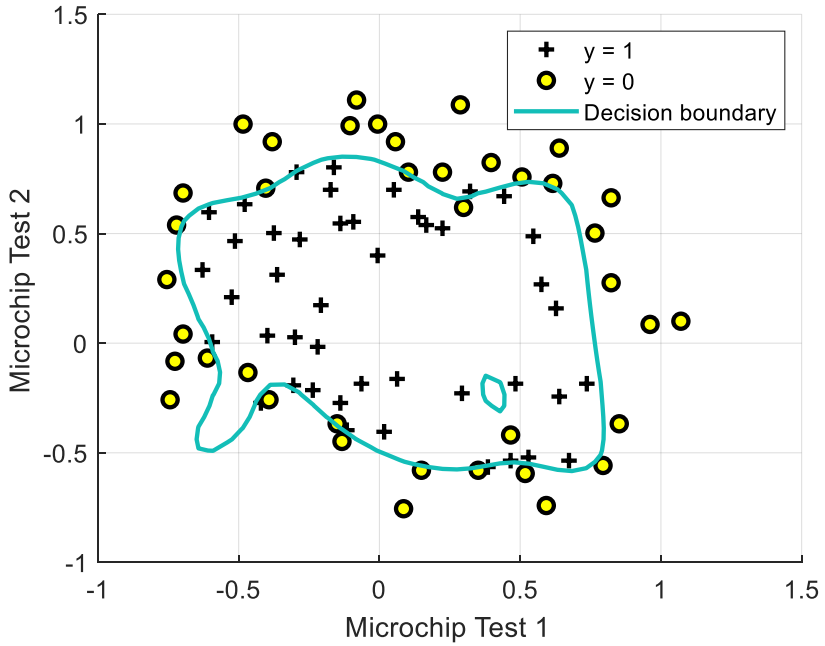


Esercizio: classificare microchips difettosi

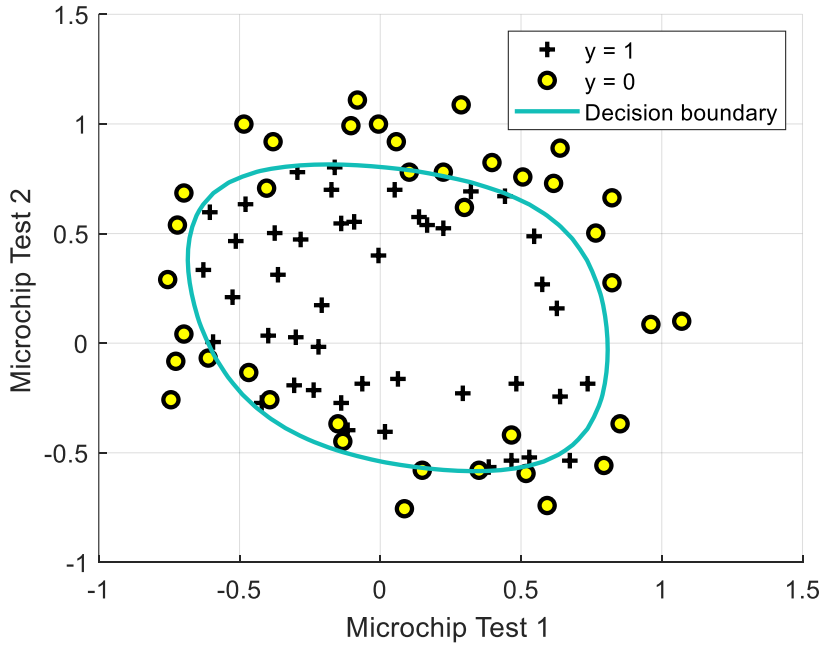
Per ottenere un **confine nonlineare** tramite il classificatore lineare, è possibile usare **features polinomiali**. Ad esempio, usando feature polinomiali di grado 2 otteniamo:

$$\varphi_1^2, \varphi_2^2, \dots, \varphi_{d-1}^2, \qquad \varphi_1\varphi_2, \dots, \varphi_1\varphi_{d-1}, \qquad \varphi_1\varphi_2^2, \dots, \varphi_1\varphi_{d-1}^2, \qquad \varphi_1^2\varphi_2, \dots, \varphi_1^2\varphi_{d-1}, \dots$$

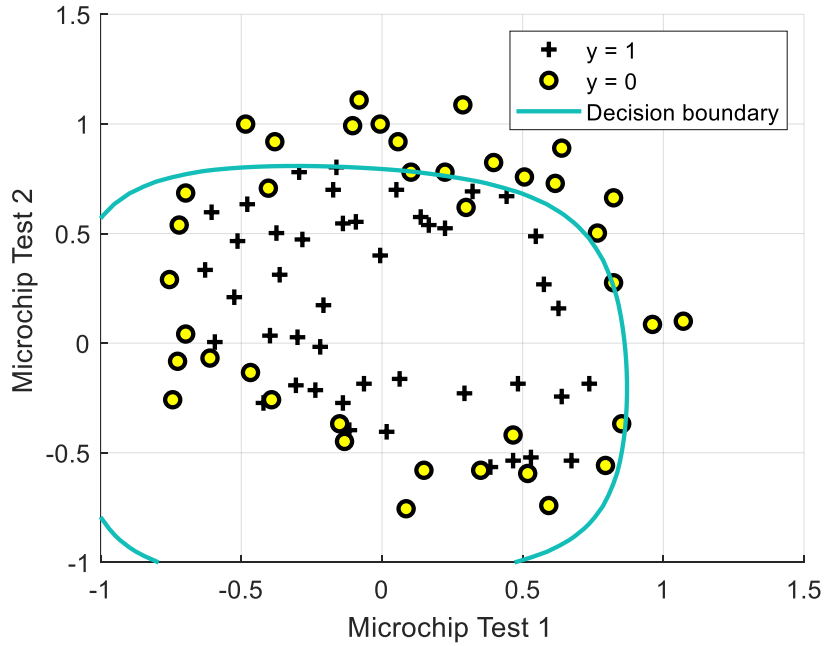
$\lambda_{\text{reg}} = 0$



$\lambda_{\text{reg}} = 0.1$

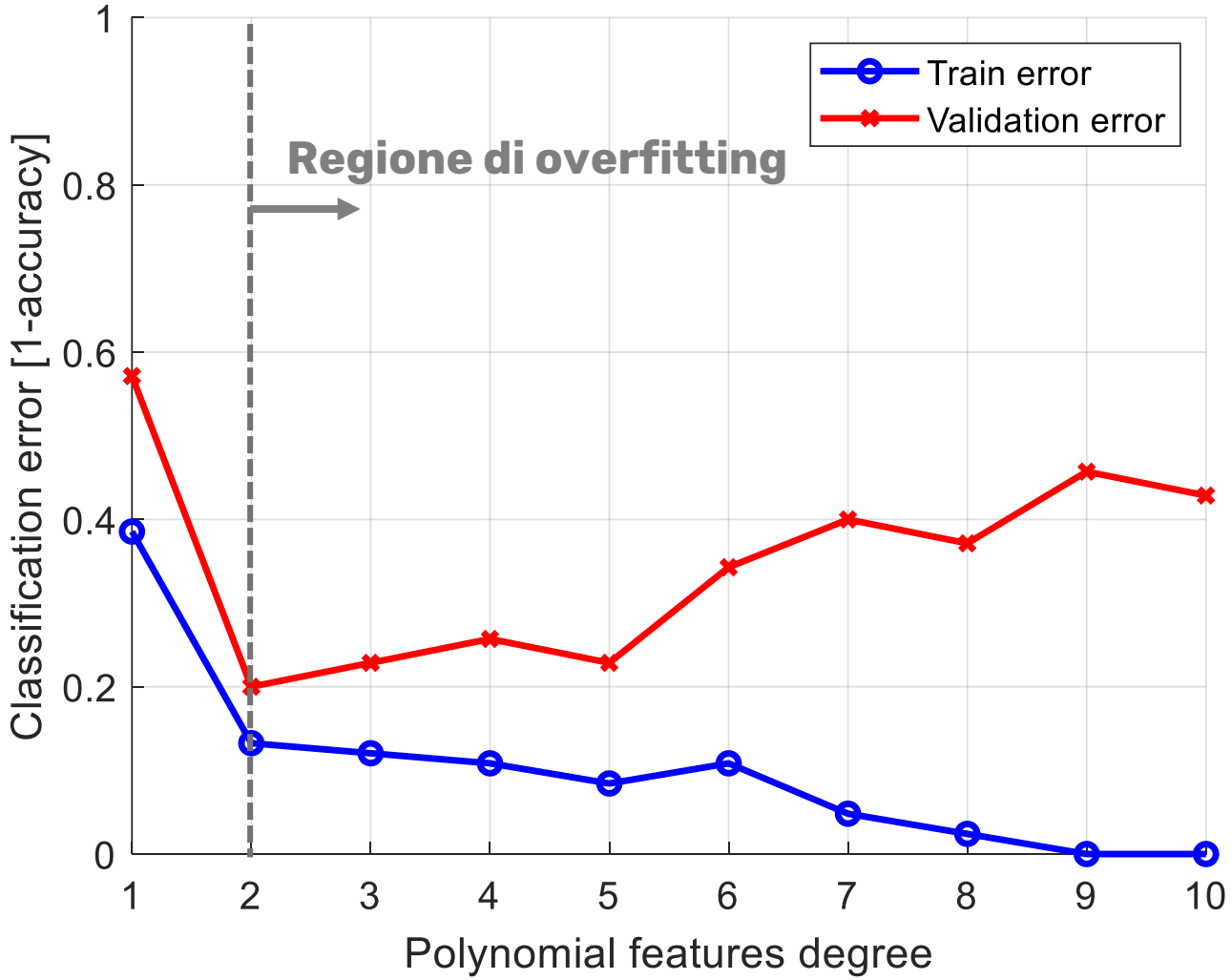


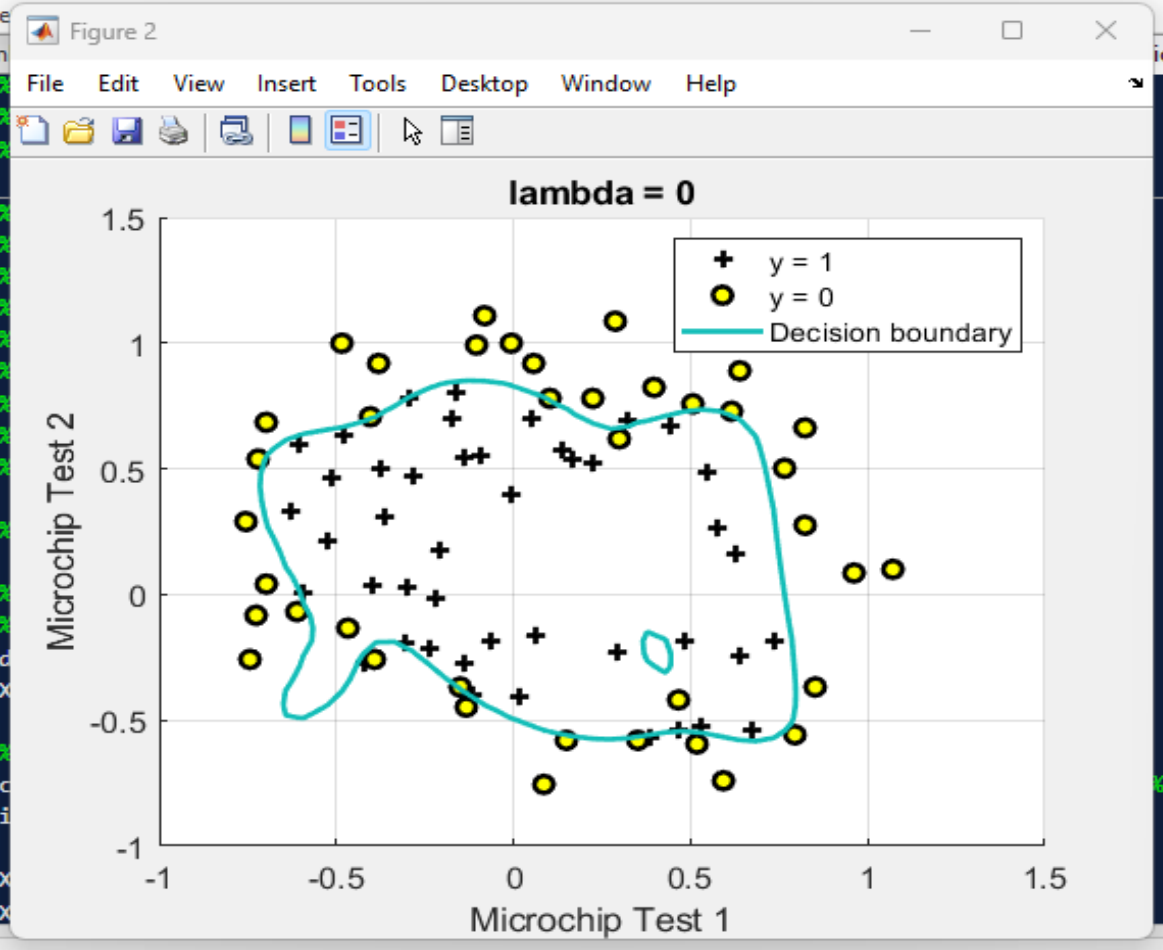
$\lambda_{\text{reg}} = 10$

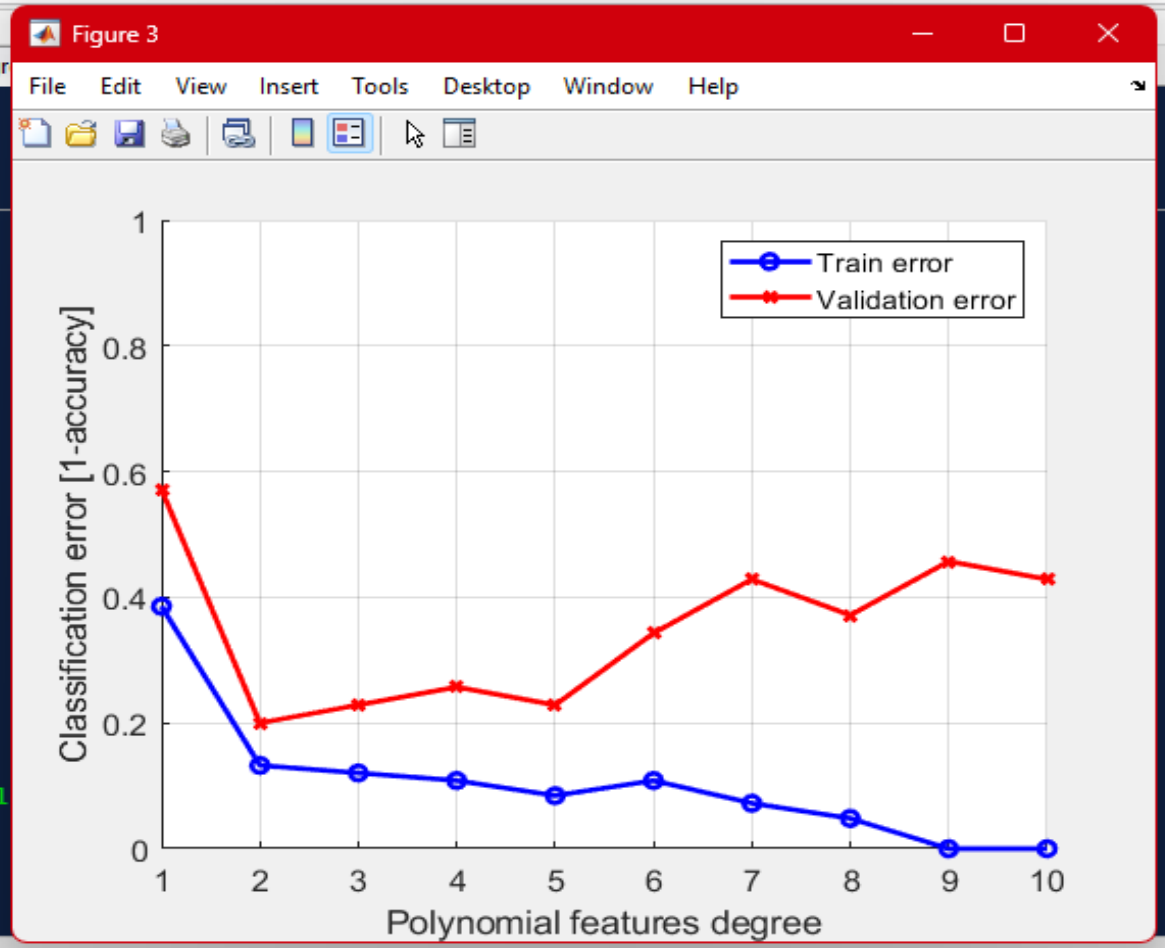


Esercizio: classificare microchips difettosi

È possible dividere i dati in training e validation set, in modo da **selezionare l'ordine ottimale** per le features polinomiali tramite **validazione**





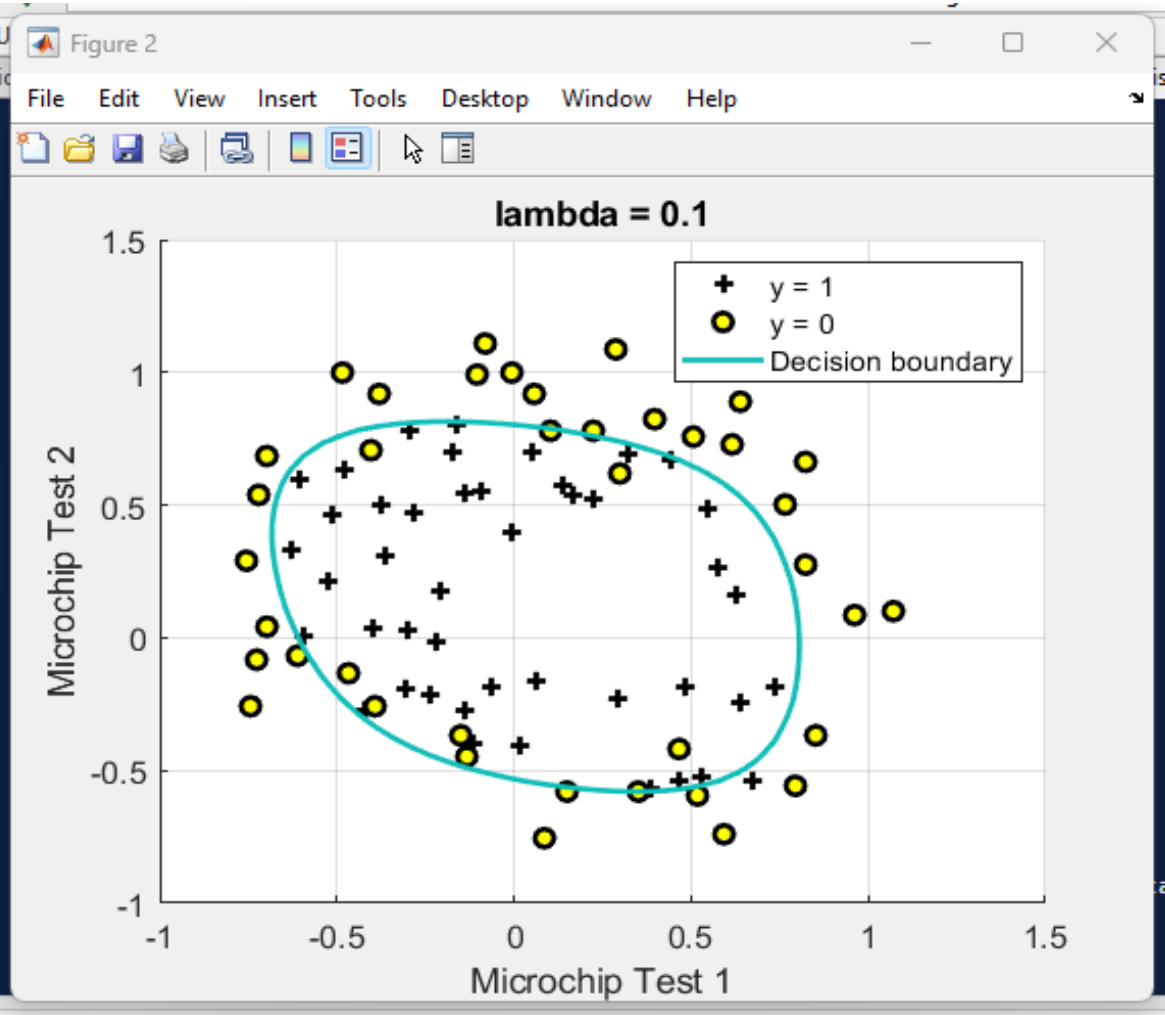


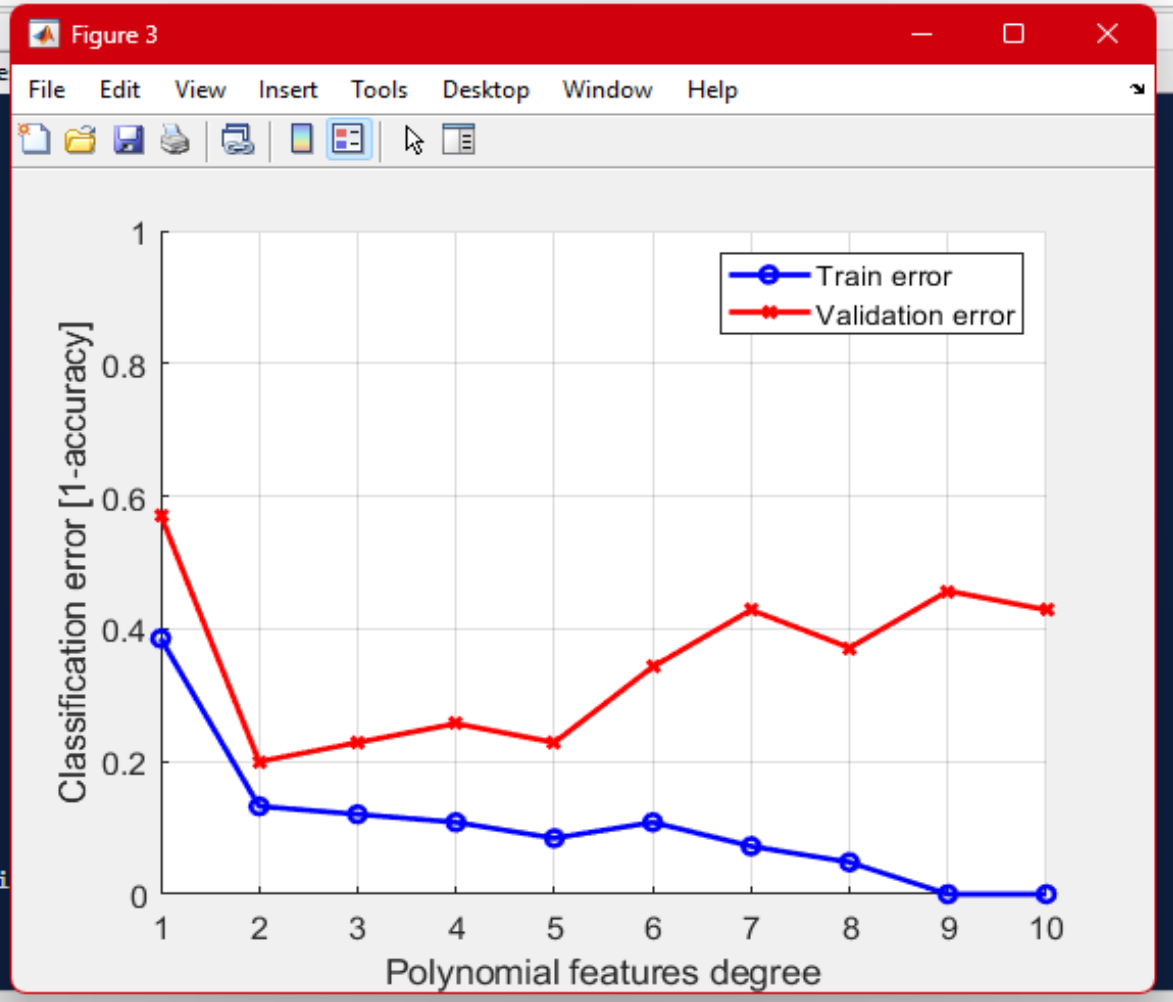

```
Cost at initial theta (zeros): 57.531216

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
Train Accuracy: 91.566265
Validation Accuracy: 71.428571
```





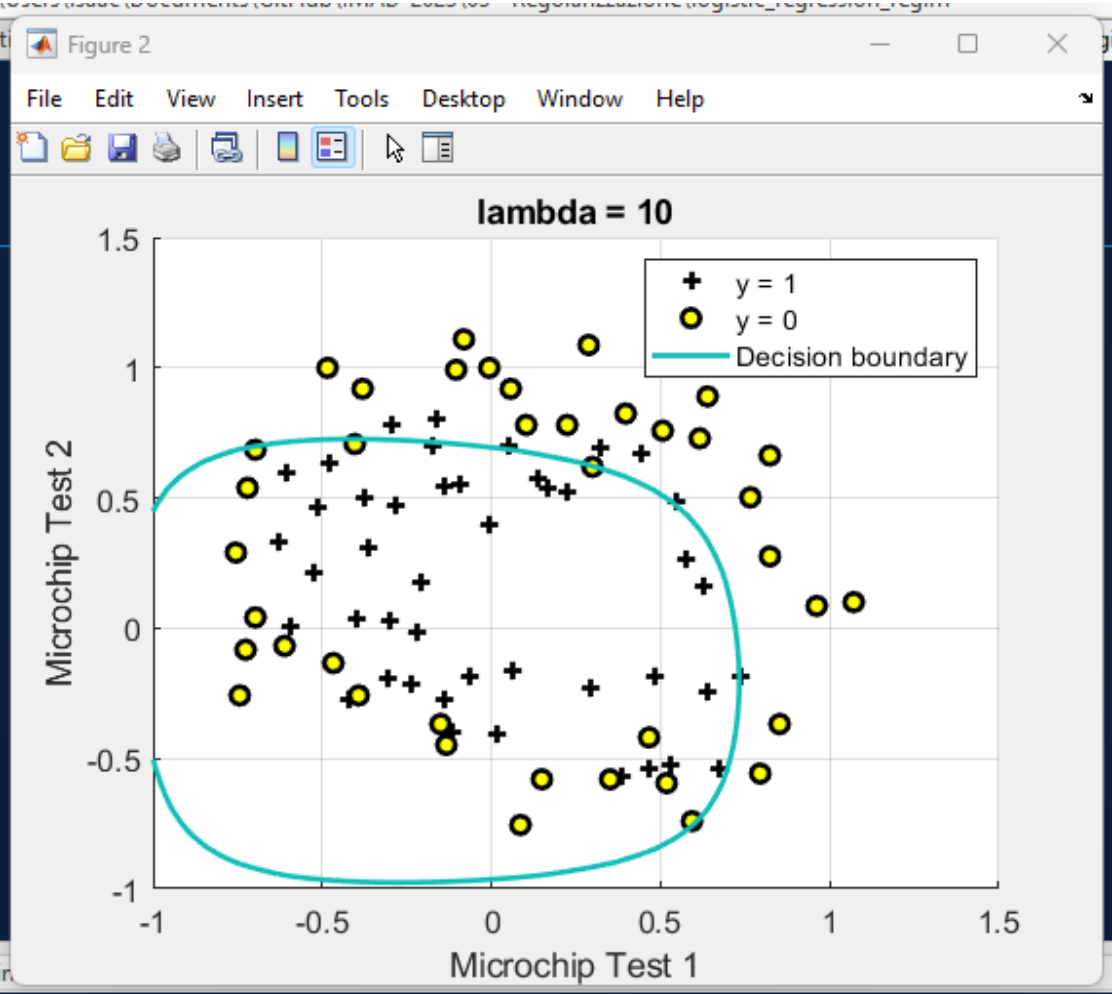
mmand Window

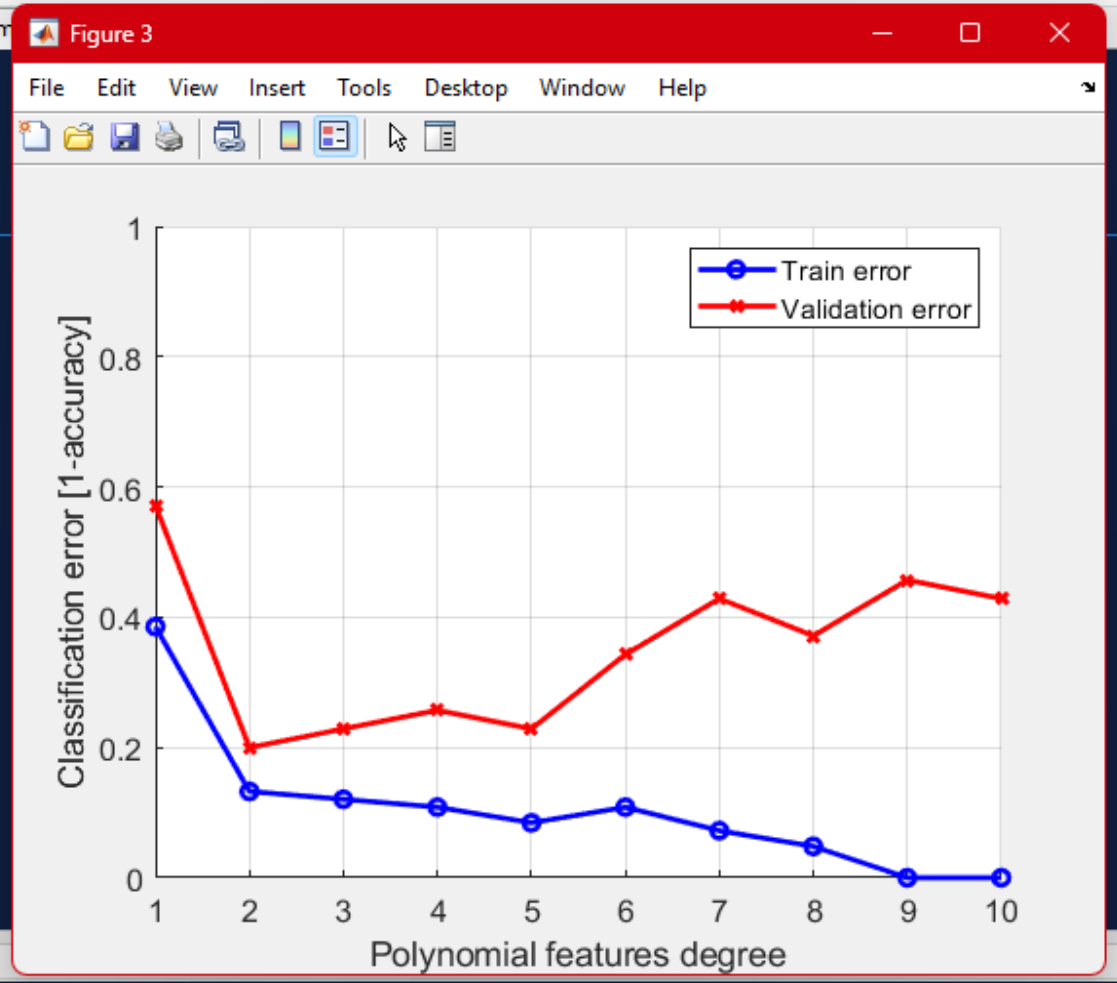
```
Cost at initial theta (zeros): 57.531216

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
Train Accuracy: 87.951807
Validation Accuracy: 77.142857
```






```
Cost at initial theta (zeros): 57.531216

Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

<stopping criteria details>
Train Accuracy: 68.674699
Validation Accuracy: 57.142857
```