Παραγωγική Τεχνητή Νοημοσύνη: Generative AI

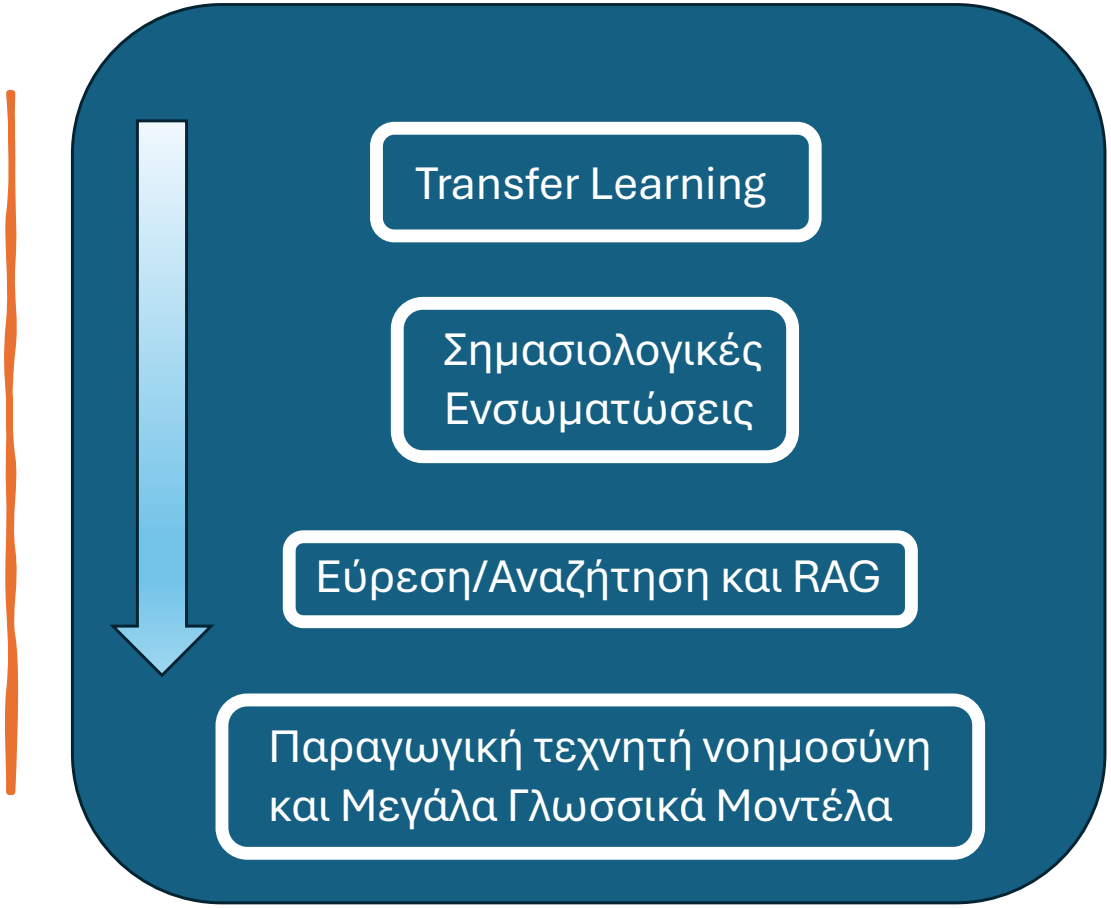
Κωνσταντίνος Καραμανής

The University of Texas at Austin & Archimedes/Athena RC

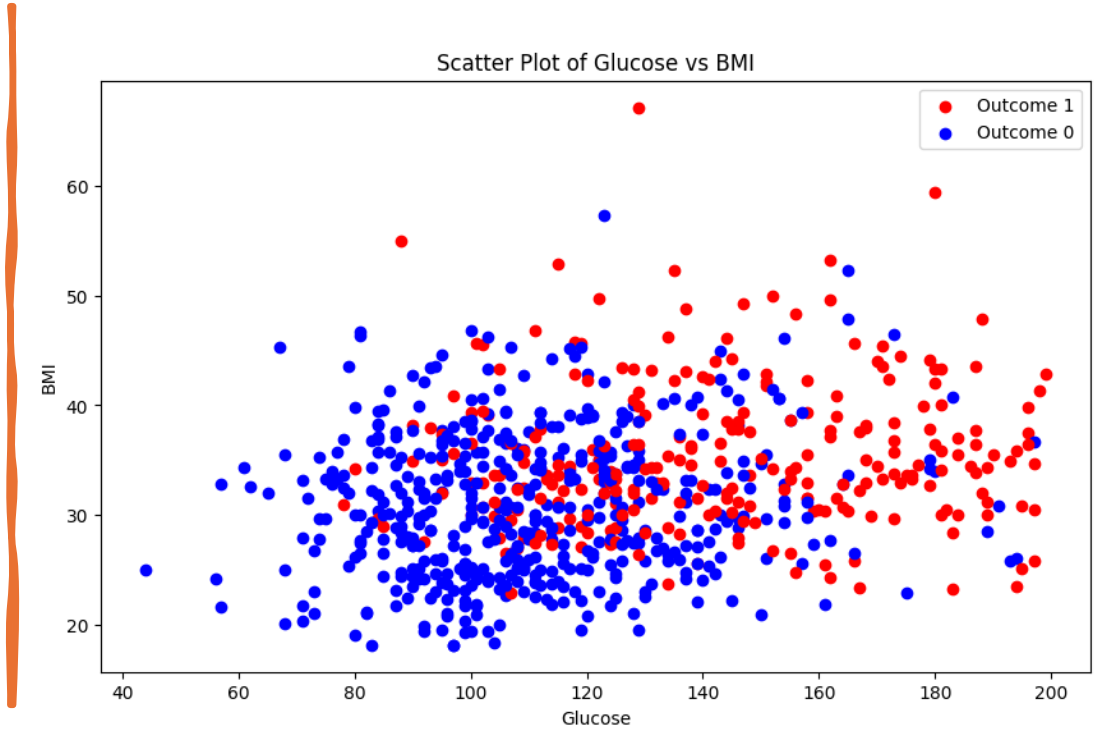
constantine@utexas.edu

https://caramanis.github.io/

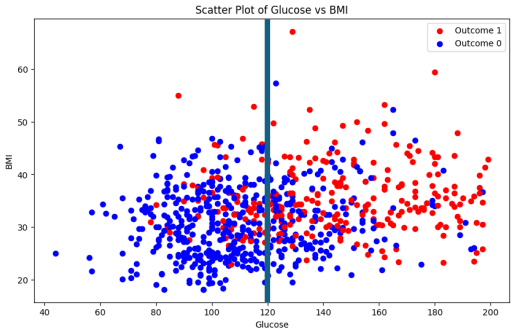
Η Γέφυρα προς την Παραγωγική Τεχνητή Νοημοσύνη



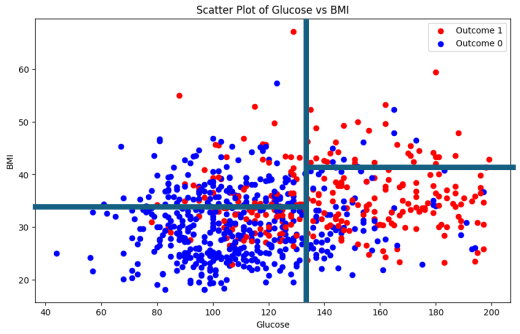
Λογιστική Παλινδρόμηση (Logistic Regression)



Λογιστική Παλινδρόμηση (Logistic Regression)

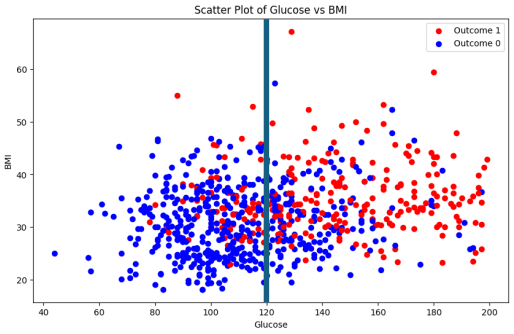


Δέντρο απόφασης βάθους 1

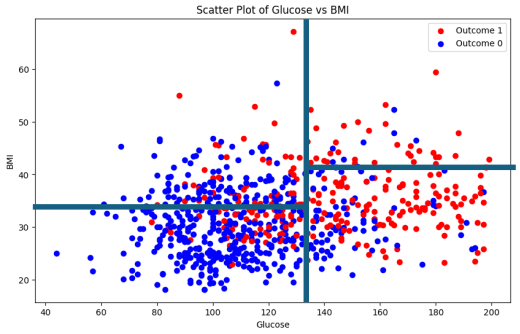


Δέντρο απόφασης βάθους 2

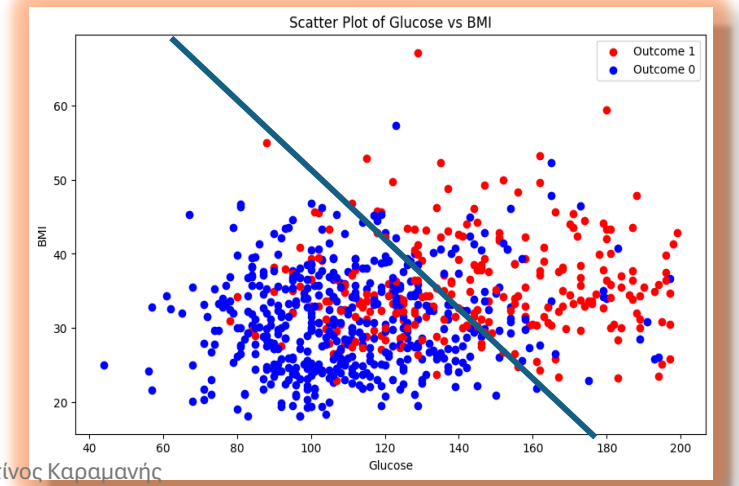
Λογιστική Παλινδρόμηση (Logistic Regression)



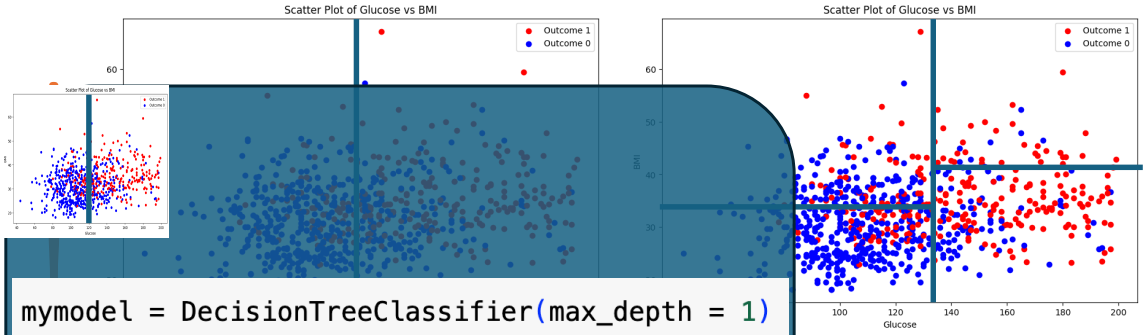
Δέντρο απόφασης βάθους 1



Δέντρο απόφασης βάθους 2

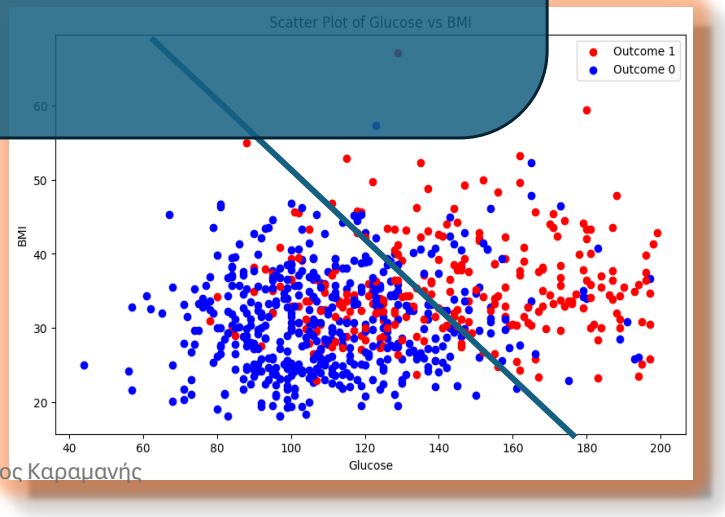


Λογιστική Παλινδρόμηση (Logistic Regression)

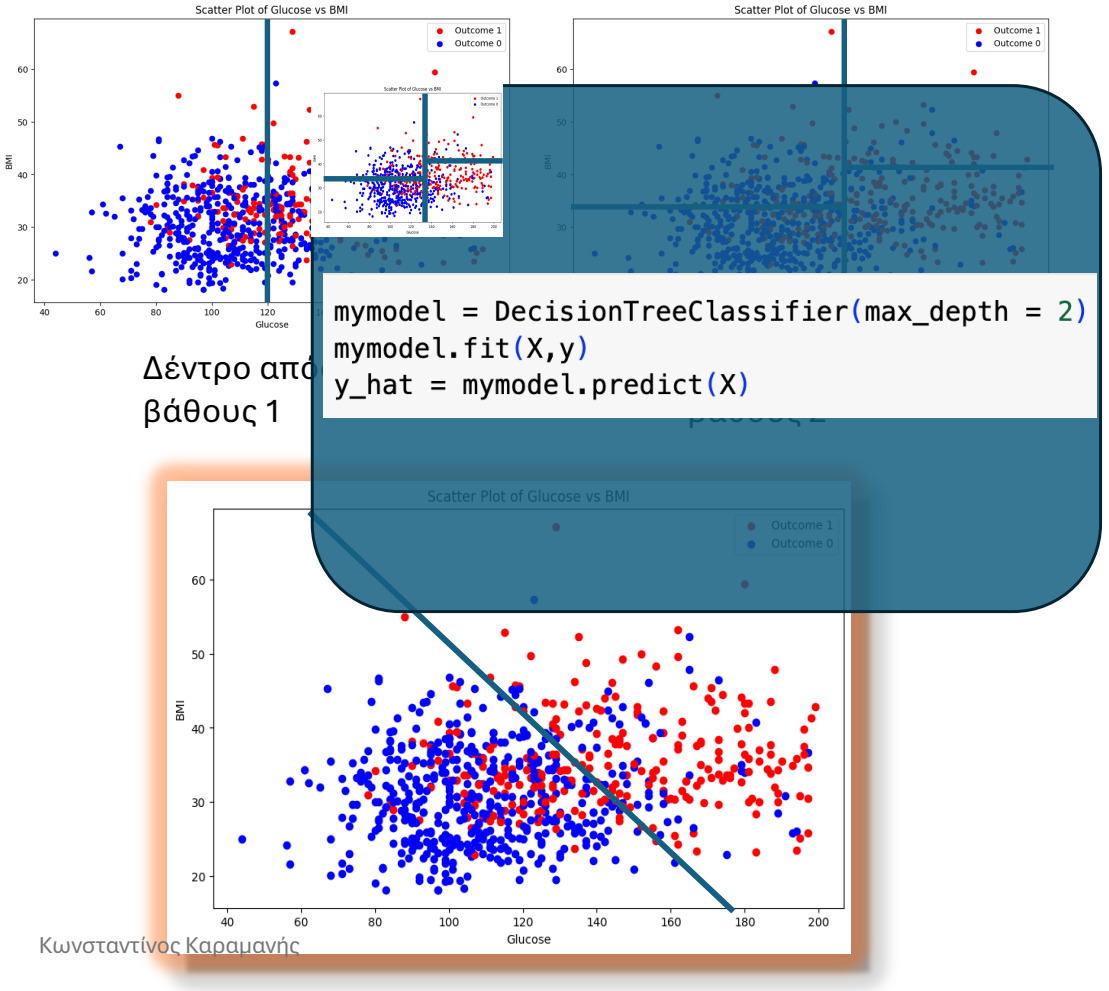



```
mymodel = DecisionTreeClassifier(max_depth = 1)
mymodel.fit(X,y)
y_hat = mymodel.predict(X)
```

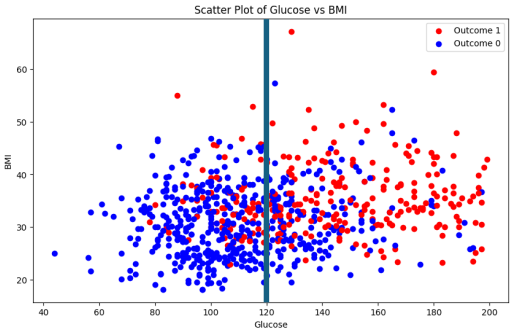
έντρο απόφασης
άθους 2

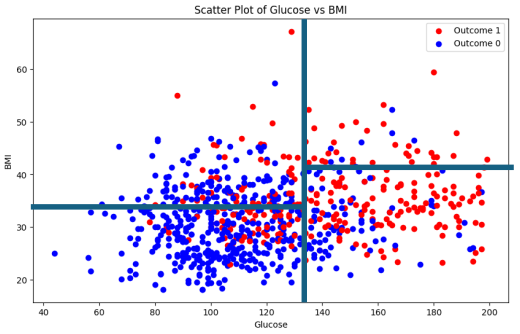


Λογιστική Παλινδρόμηση (Logistic Regression)



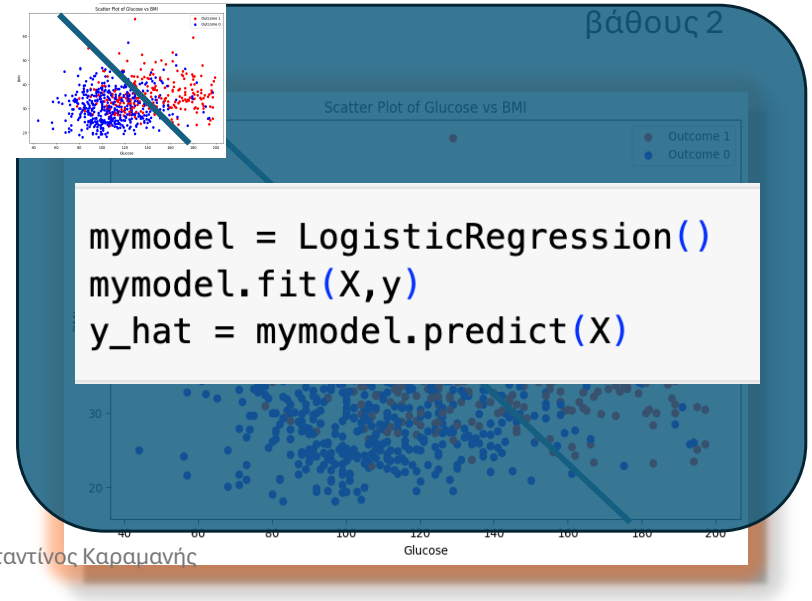
Λογιστική Παλινδρόμηση (Logistic Regression)





Δέντρο απόφασης

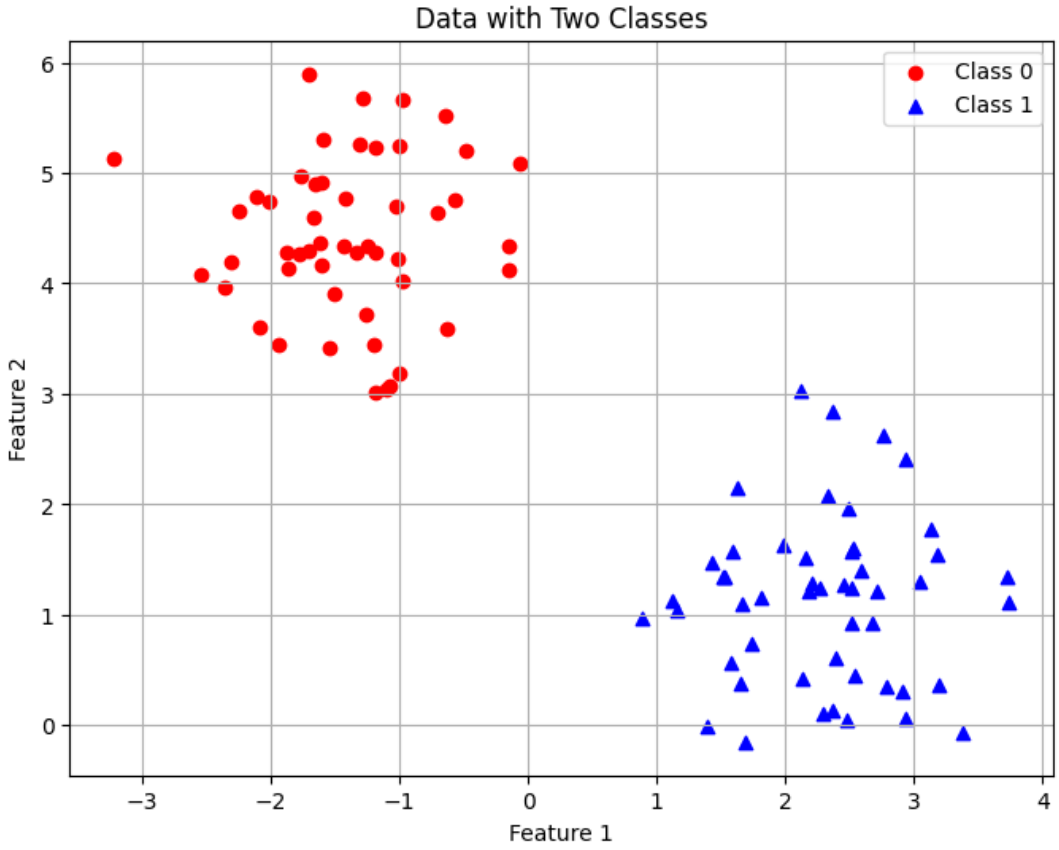
Δέντρο απόφασης βάθους 2




```
mymodel = LogisticRegression()
mymodel.fit(X,y)
y_hat = mymodel.predict(X)
```

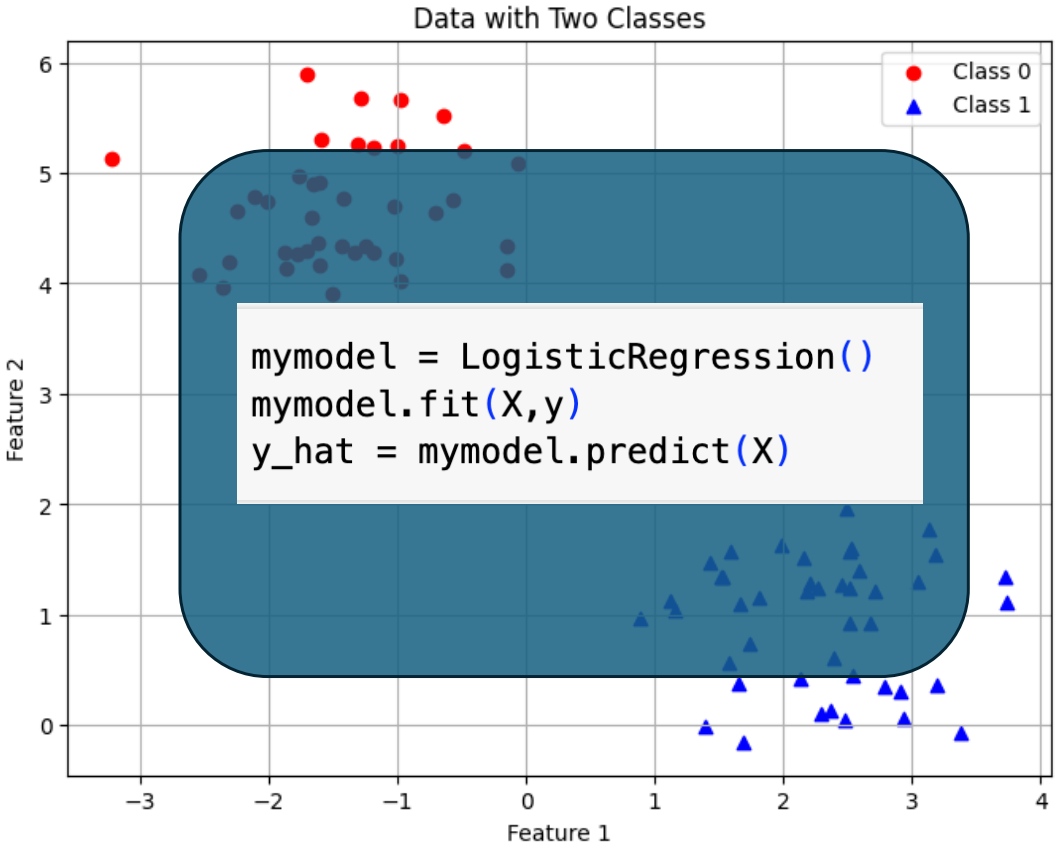
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



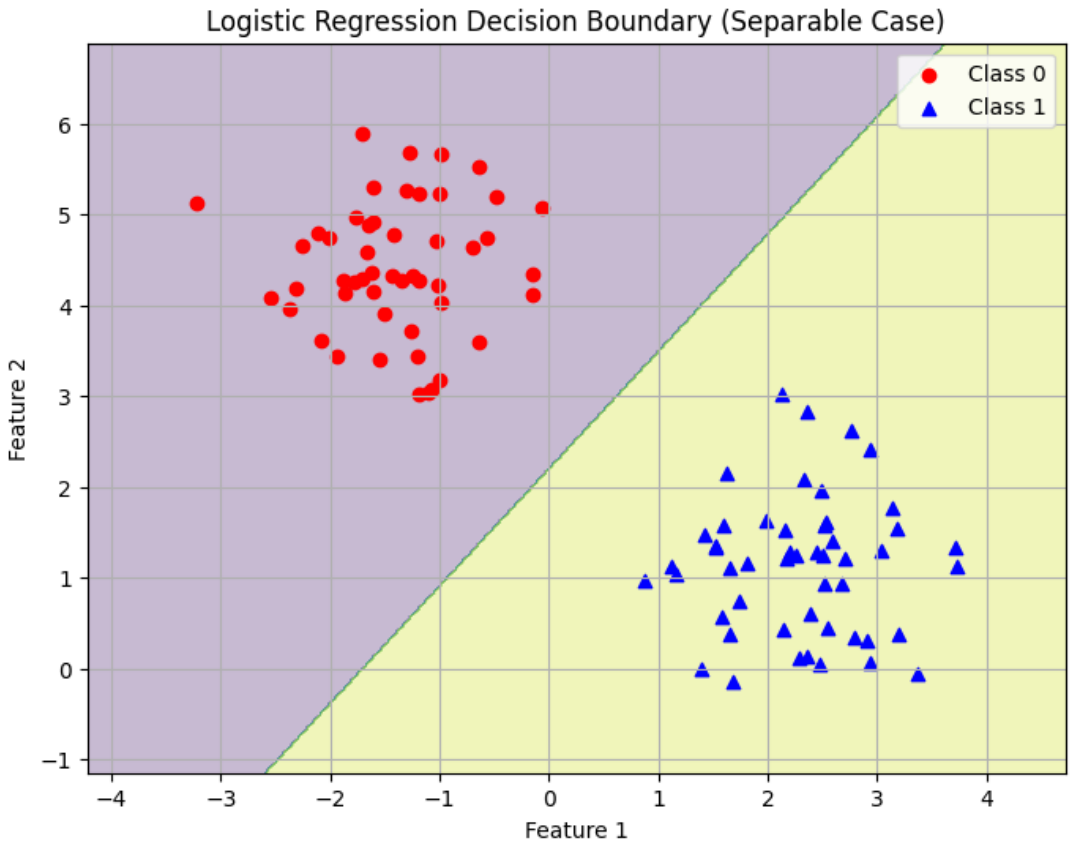
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



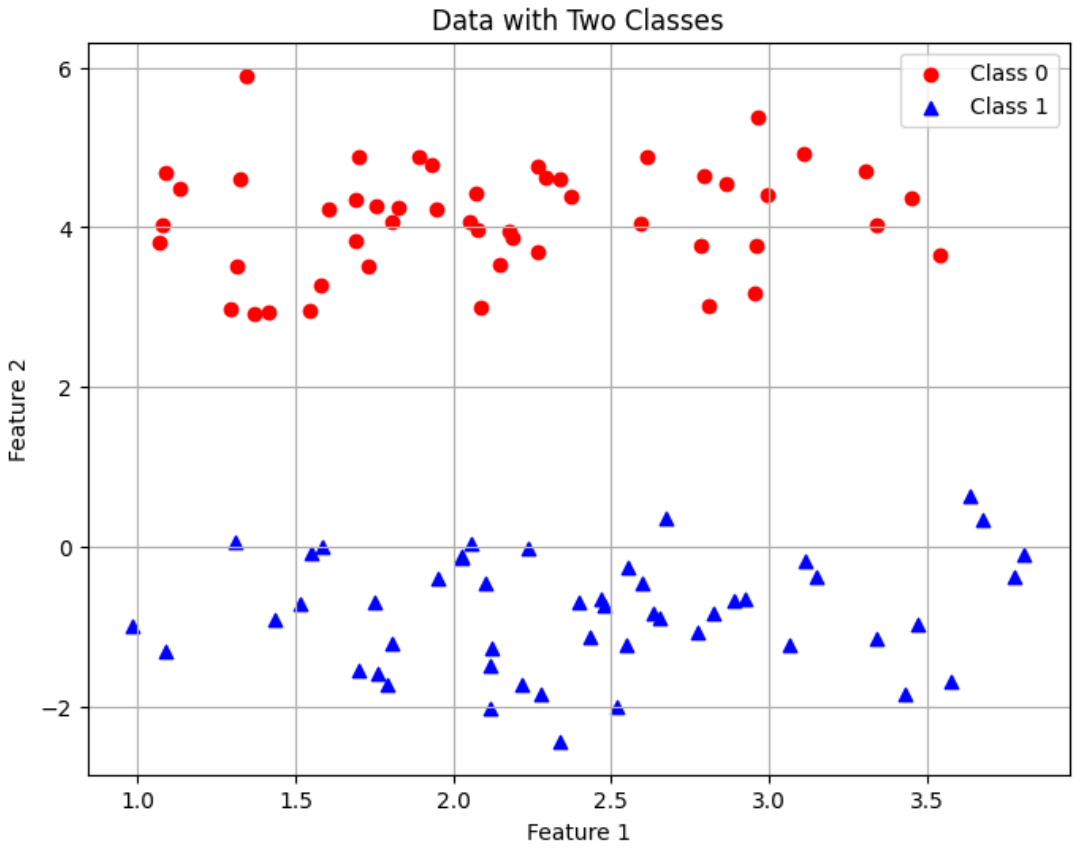
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



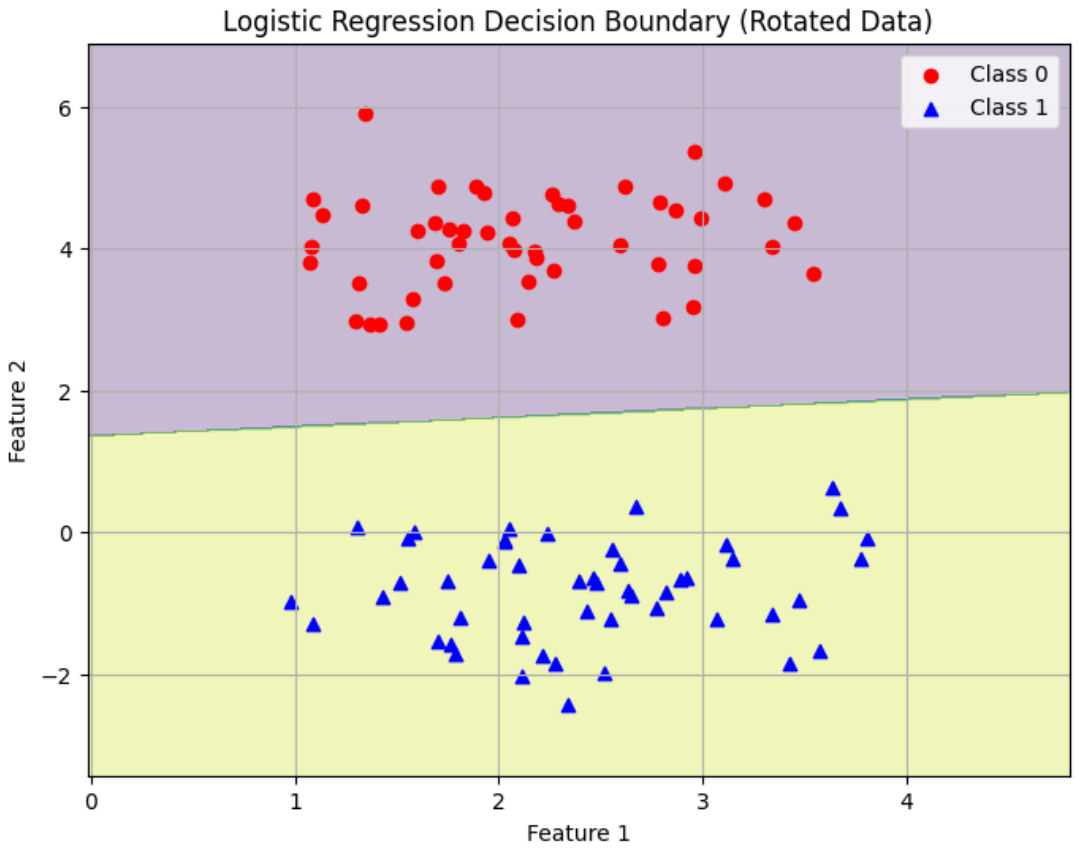
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



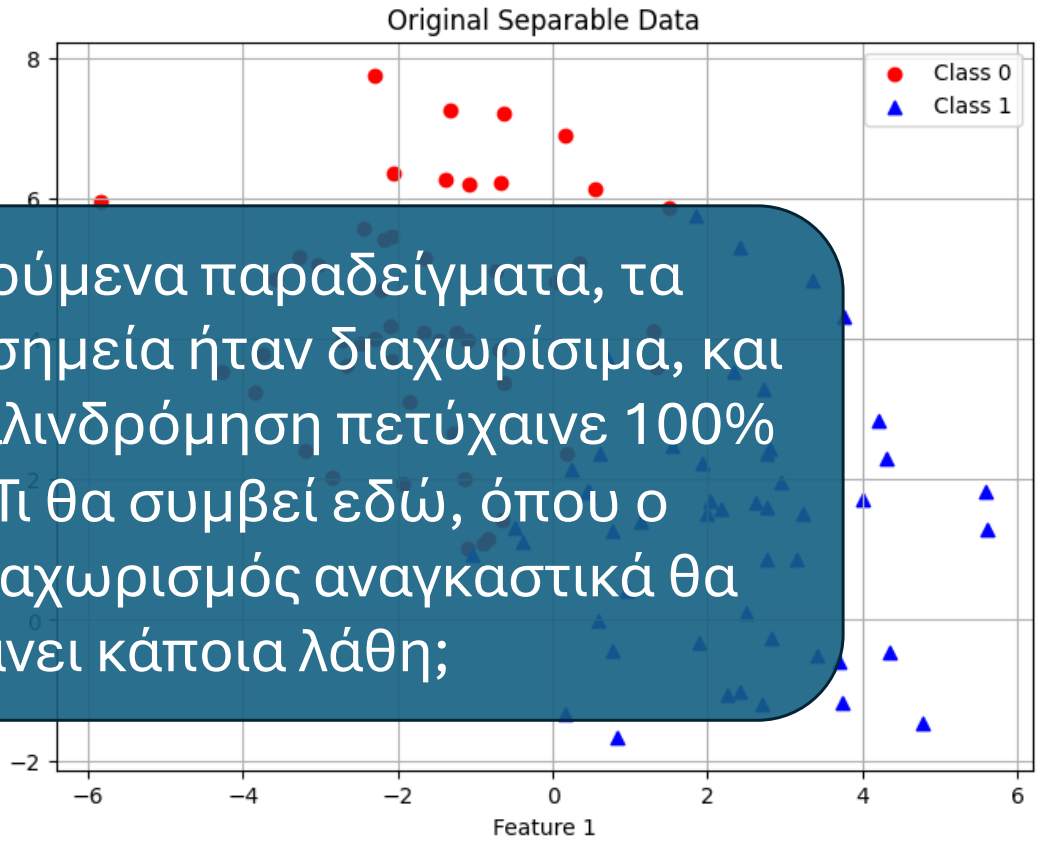
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



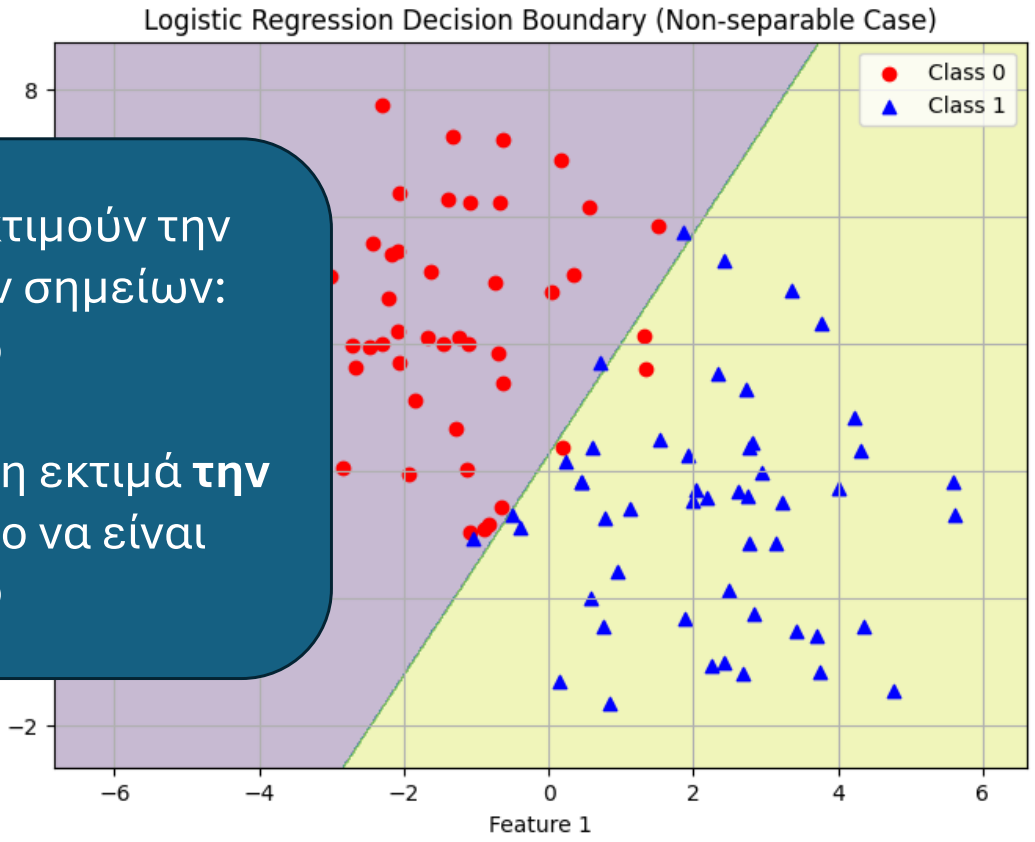
Στα προηγούμενα παραδείγματα, τα μπλε/κόκκινα σημεία ήταν διαχωρίσιμα, και η λογιστική παλινδρόμηση πετύχαινε 100% ακρίβεια. Τι θα συμβεί εδώ, όπου ο γραμμικός διαχωρισμός αναγκαστικά θα κάνει κάποια λάθη;

Λογιστική Παλινδρόμηση: Π

Τα δέντρα απόφασης εκτιμούν την ετικέτα (κατηγορία) των σημείων: μπλε/κόκκινο

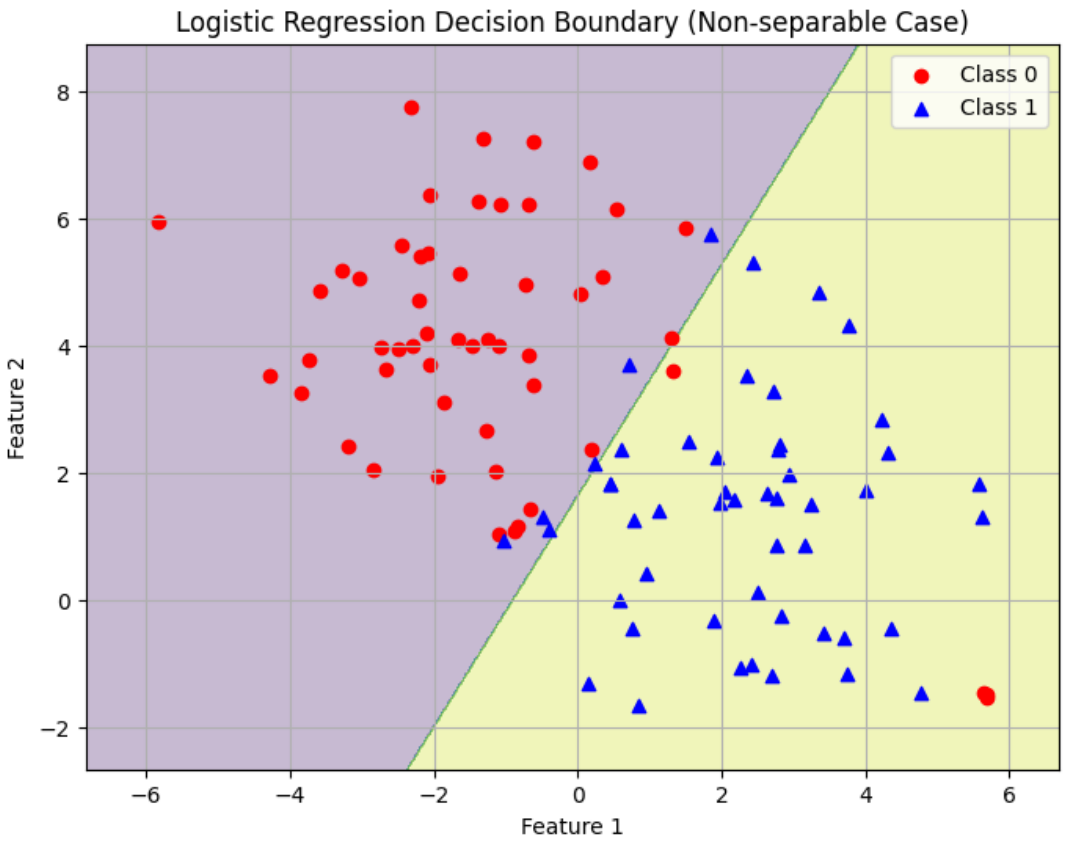
Η λογιστική παλινδρόμηση εκτιμά **την πιθανότητα** ένα σημείο να είναι μπλε/κόκκινο

- Δύ
  κλά
- Δύ
  εισ



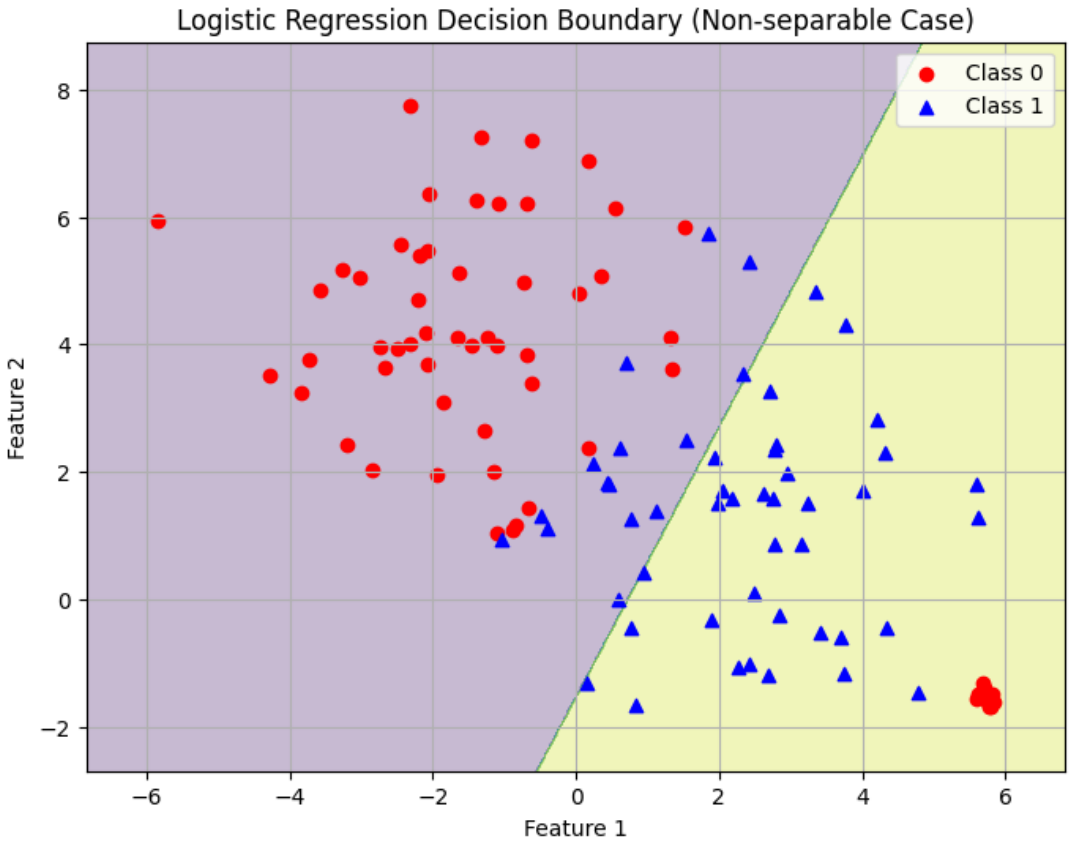
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



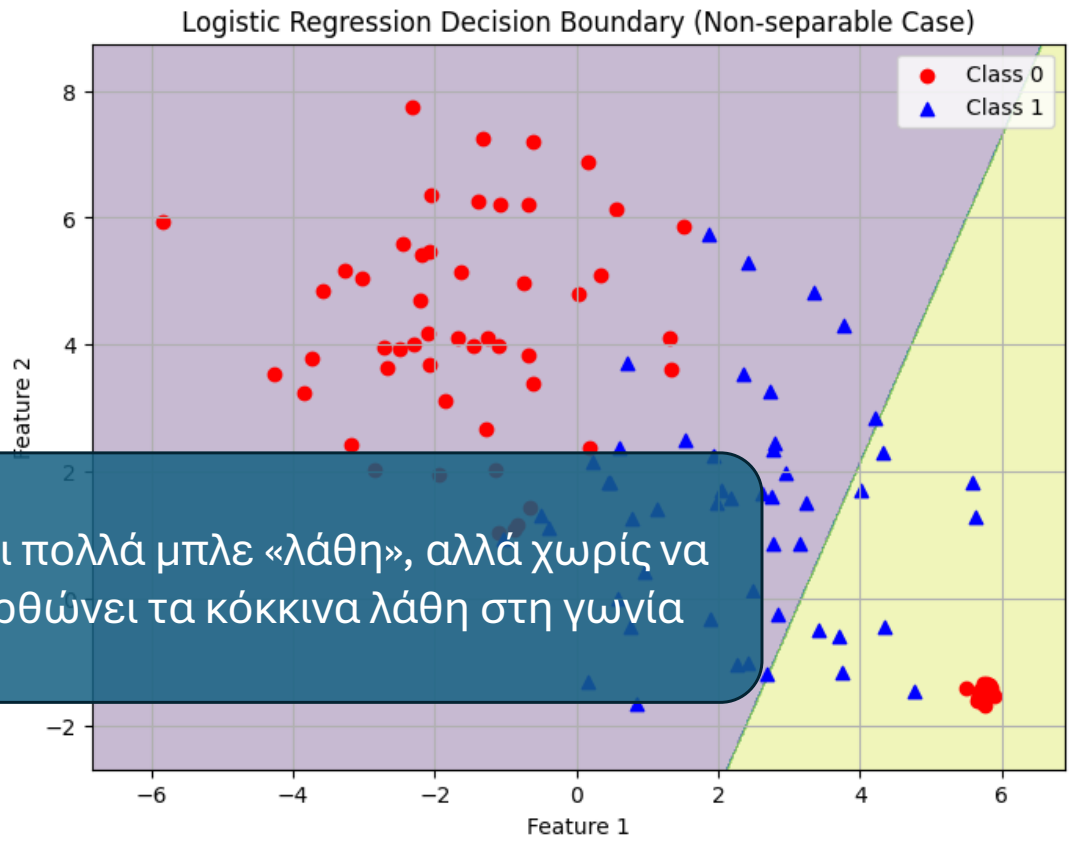
Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



Λογιστική Παλινδρόμηση: Παραδείγματα

- Δύο κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



Κάνει πολλά μπλε «λάθη», αλλά χωρίς να διορθώνει τα κόκκινα λάθη στη γωνία

Λογιστική Παλινδρόμηση

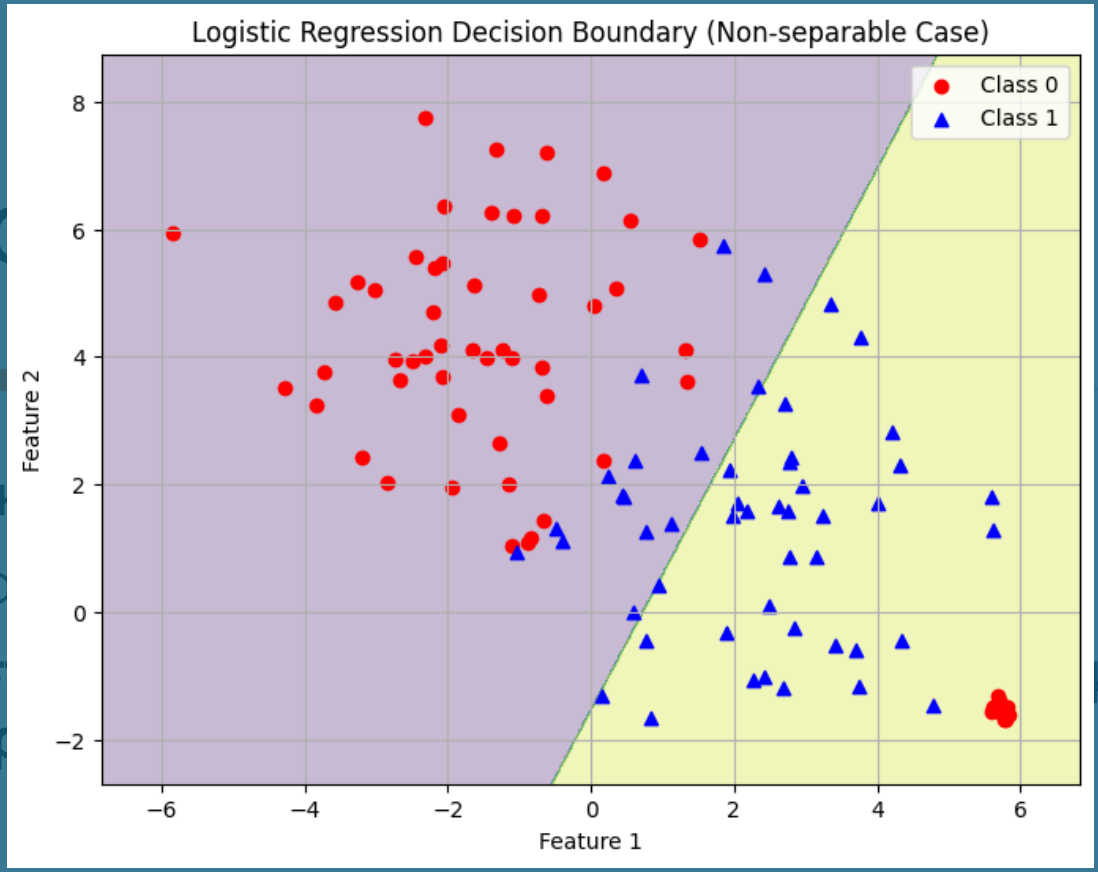
- Γραμμικός διαχωρισμός του χώρου
- Δεν ελαχιστοποιεί τα «λάθη»
- Ελαχιστοποιεί την «απώλεια» του γραμμικού διαχωρισμού που λέγεται Cross Entropy

Λογιστική Παλινδρόμηση

- Γραμμικός διαχωρισμός του χώρου
- Δεν ελαχιστοποιεί τα «λάθη»
- **Ελαχιστοποιεί την «απώλεια» του γραμμικού διαχωρισμού που λέγεται Cross Entropy**

Λογι

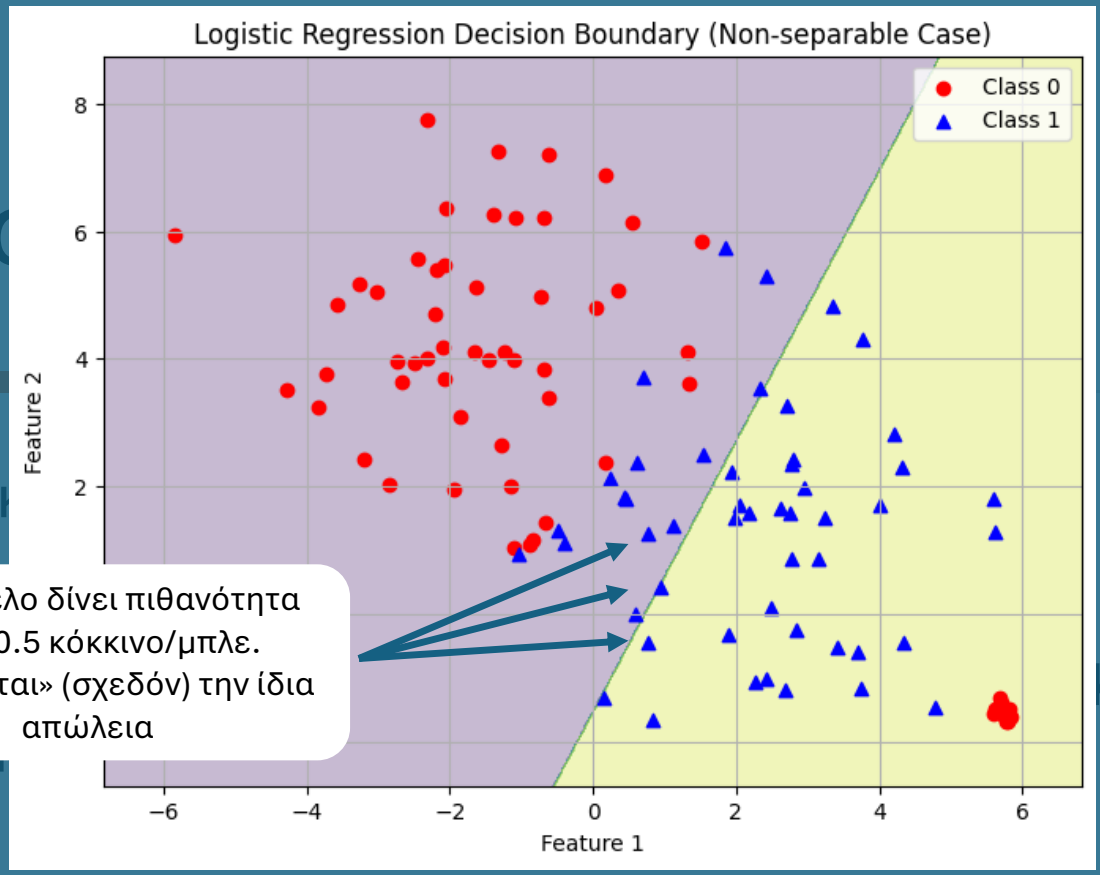
- Γραμμι
- Δεν ελο
- **Ελαχισ** **μικός διαχωρ**



Λογισ

- Γραμμι
- Δ
- Ι

μικός

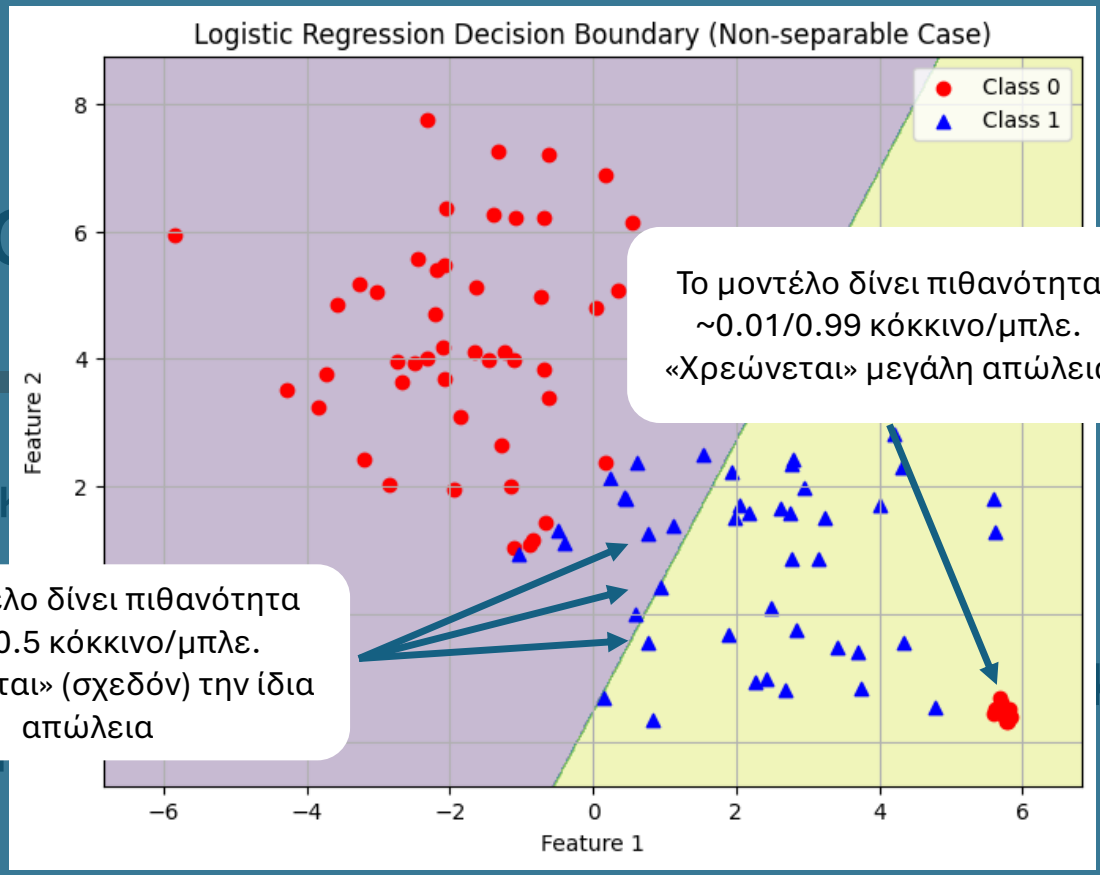


Το μοντέλο δίνει πιθανότητα ~0.5/0.5 κόκκινο/μπλε. «Χρεώνεται» (σχεδόν) την ίδια απώλεια

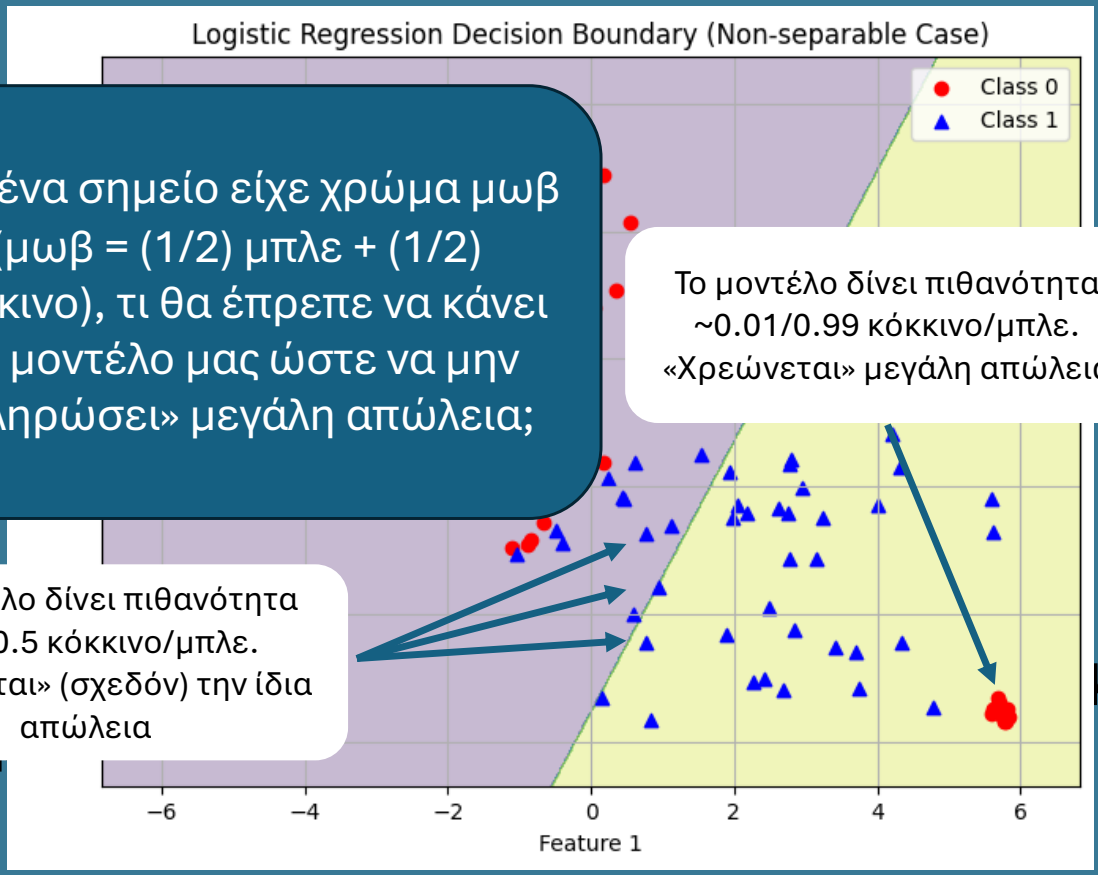
Λογι

- Γραμμι
- Δ
- Ι

μικός



Εάν ένα σημείο είχε χρώμα μωβ (μωβ = (1/2) μπλε + (1/2) κόκκινο), τι θα έπρεπε να κάνει το μοντέλο μας ώστε να μην «πληρώσει» μεγάλη απώλεια;



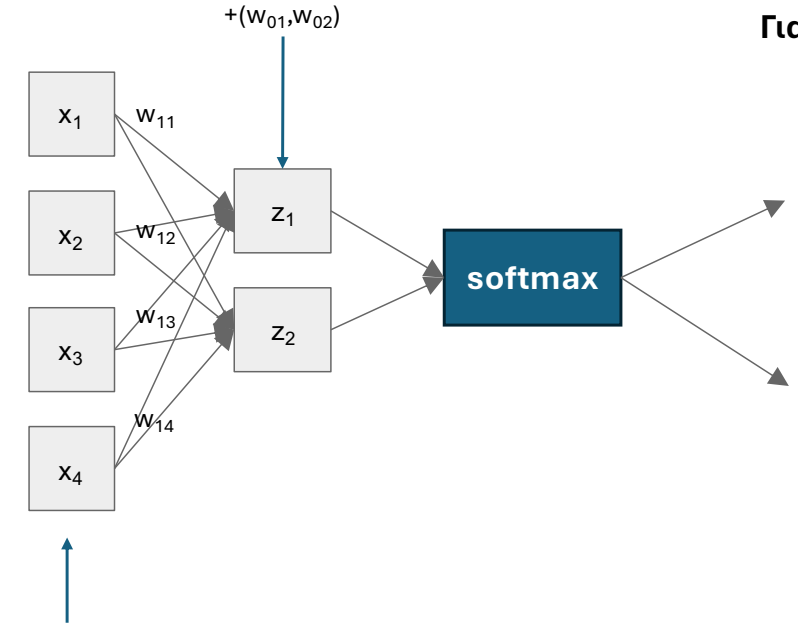
Λογιστική Παλινδρόμηση

- Γραμμικός διαχωρισμός του χώρου
- Δεν ελαχιστοποιεί τα «λάθη»
- **Ελαχιστοποιεί την «απώλεια» του γραμμικού διαχωρισμού**

Λογιστική Παλινδρόμηση

- Γραμμικός διαχωρισμός του χώρου
- Δεν ελαχιστοποιεί τα «λάθη»
- Ελαχιστοποιεί την «απώλεια» του γραμμικού διαχωρισμού
- **Είναι το πιο απλό νευρωνικό δίκτυο**

Λογ. Παλινδρόμηση: 1 γραμμικό επίπεδο



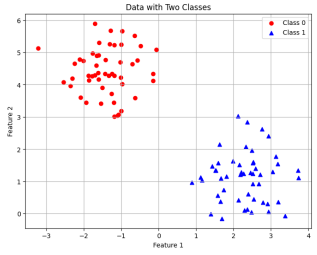
Για ταξινόμηση έχουμε 2 εξόδους (outputs)

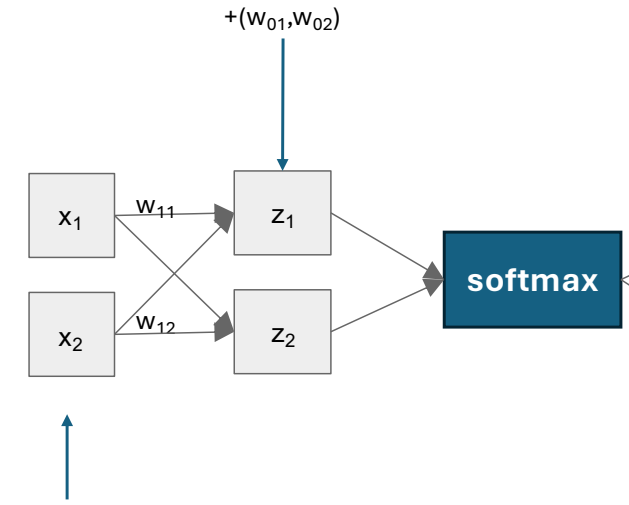
$$y_1 = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

$$y_2 = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}$$

$$0 \le y_1 \le 1, 0 \le y_2 \le 1$$
$$y_1 + y_2 = 1$$

Λογ. Παλινδρόμηση: 1 γραμμικό επίπεδο





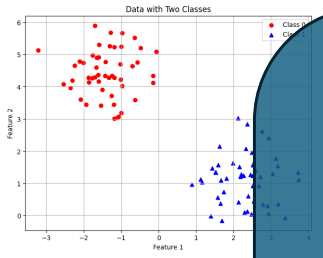
Για ταξινόμηση έχουμε 2 εξόδους (outputs)

$$y_1 = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

$$y_2 = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}$$

$$0 \le y_1 \le 1, 0 \le y_2 \le 1$$
$$y_1 + y_2 = 1$$

Λογ. Παλινδρόμηση: 1 γραμμικό επίπεδο



$+(c_1, c_2)$

Για ταξινόμηση έχουμε 2 εξόδους (outputs)

```python
class SimpleLogisticNN(nn.Module):
    def __init__(self):
        super(SimpleLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 2)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```

$\sigma(z_2)$

$\sigma(z_2)$

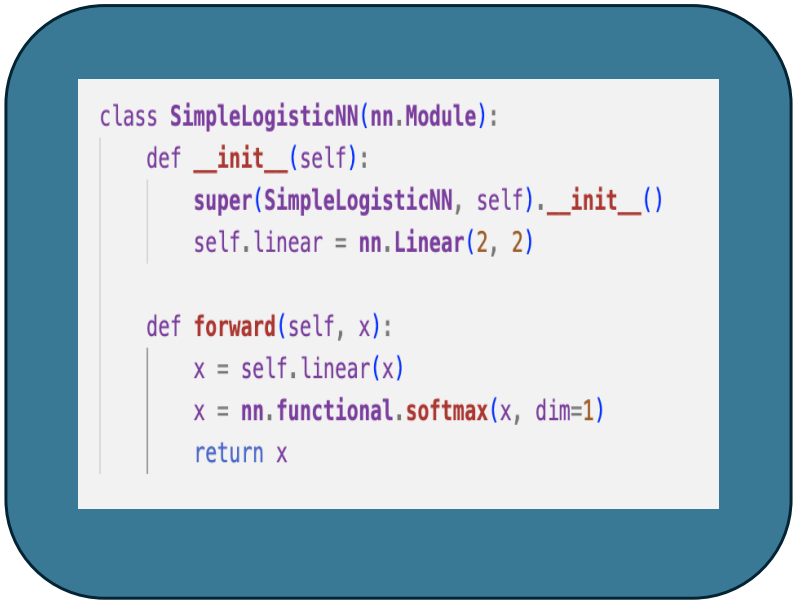
δεδομένα

$0 \leq y_1 \leq 1, 0 \leq y_2 \leq 1$

$y_1 + y_2 = 1$

Λογιστική Παλινδρόμηση: Τι κάνει;

- Ο κώδικάς μας εξηγεί ακριβώς τι κάνει η λογιστική παλινδρόμηση:



```
class SimpleLogisticNN(nn.Module):
    def __init__(self):
        super(SimpleLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 2)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```

Λογιστική Παλινδρόμηση: Τι κάνει;

- Ο κώδικάς μας εξηγεί ακριβώς τι κάνει η λογιστική παλινδρόμηση:

(x_1,x_2) → $z_1 = w_{11}x_1 + w_{12}x_2 + w_{10}$

(x_1,x_2) → $z_2 = w_{21}x_1 + w_{22}x_2 + w_{20}$

```
class SimpleLogisticNN(nn.Module):
    def __init__(self):
        super(SimpleLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 2)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```

Λογιστική Παλινδρόμηση: Τι κάνει;

- Ο κώδικάς μας εξηγεί ακριβώς τι κάνει η λογιστική παλινδρόμηση:

(x_1,x_2) → $z_1 = w_{11}x_1 + w_{12}x_2 + w_{10}$

(x_1,x_2) → $z_2 = w_{21}x_1 + w_{22}x_2 + w_{20}$

$$y_1 = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

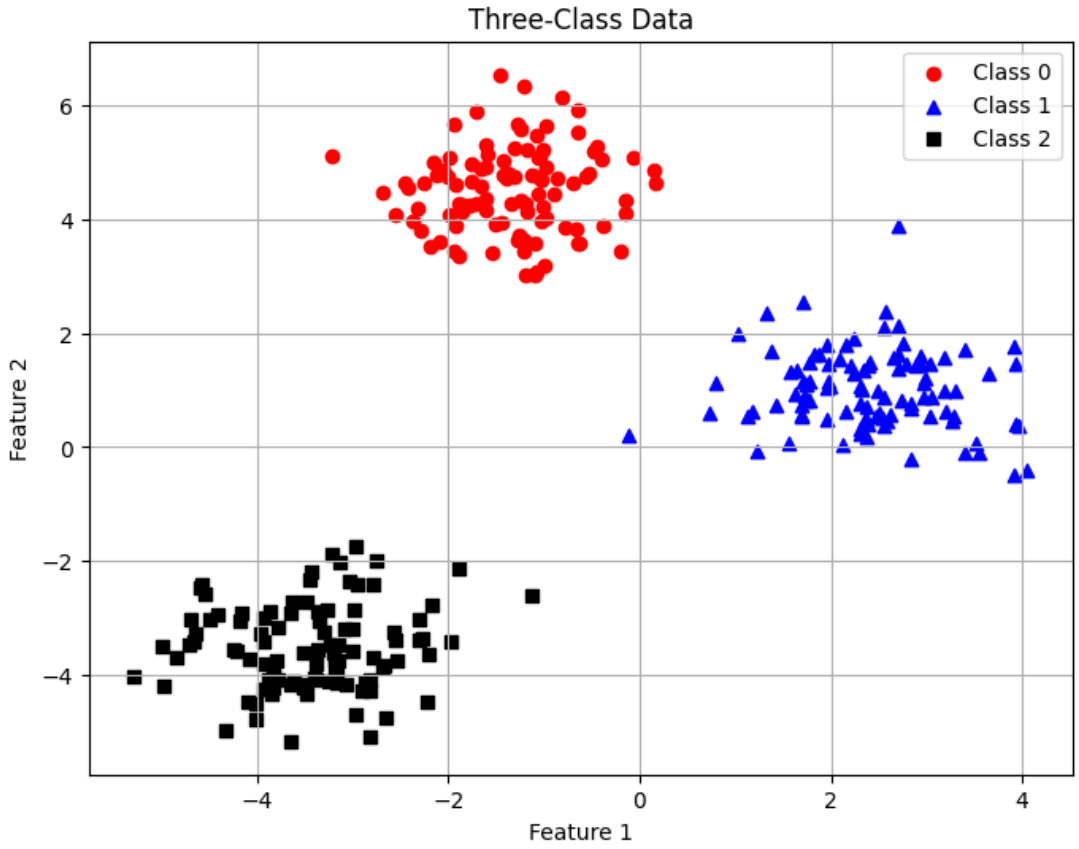
$$y_2 = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}$$

```
class SimpleLogisticNN(nn.Module):
    def __init__(self):
        super(SimpleLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 2)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```

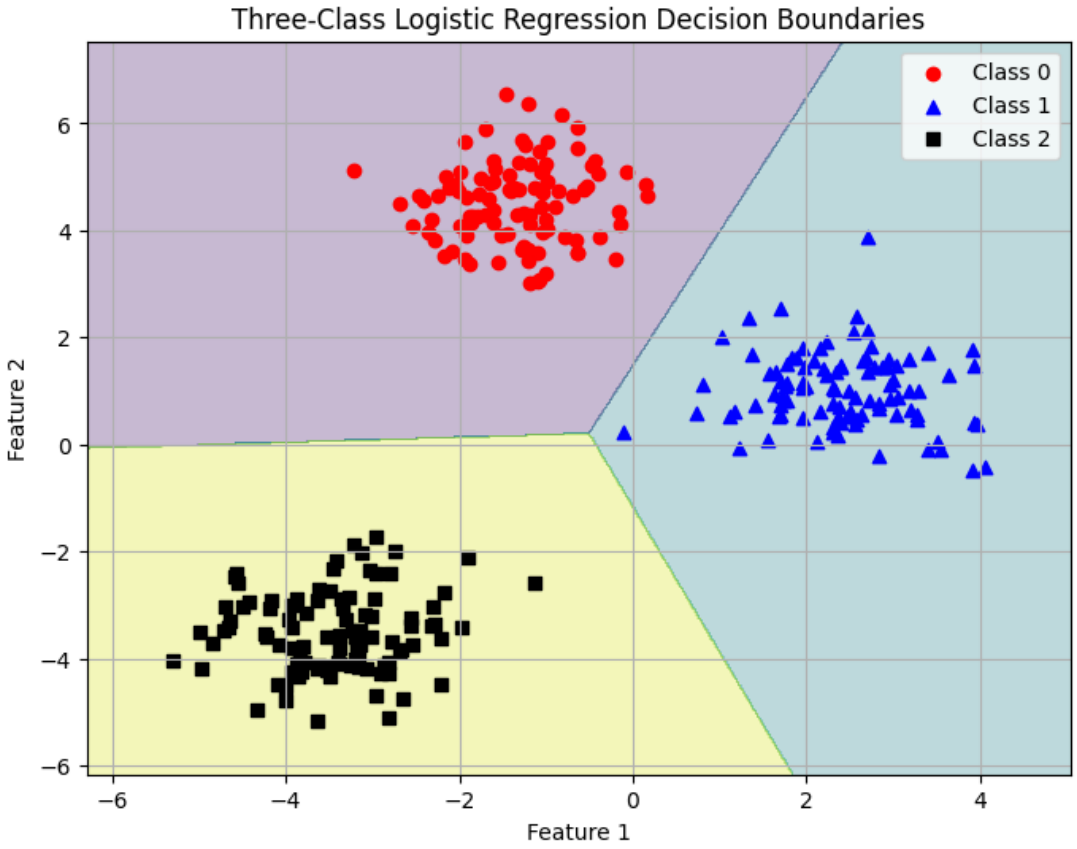
Λογιστική Παλινδρόμηση: Παραδείγματα

- **Τρεις** κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



Λογιστική Παλινδρόμηση: Παραδείγματα

- **Τρεις** κλάσεις/κατηγορίες
- Δύο δεδομένα εισόδου (features)



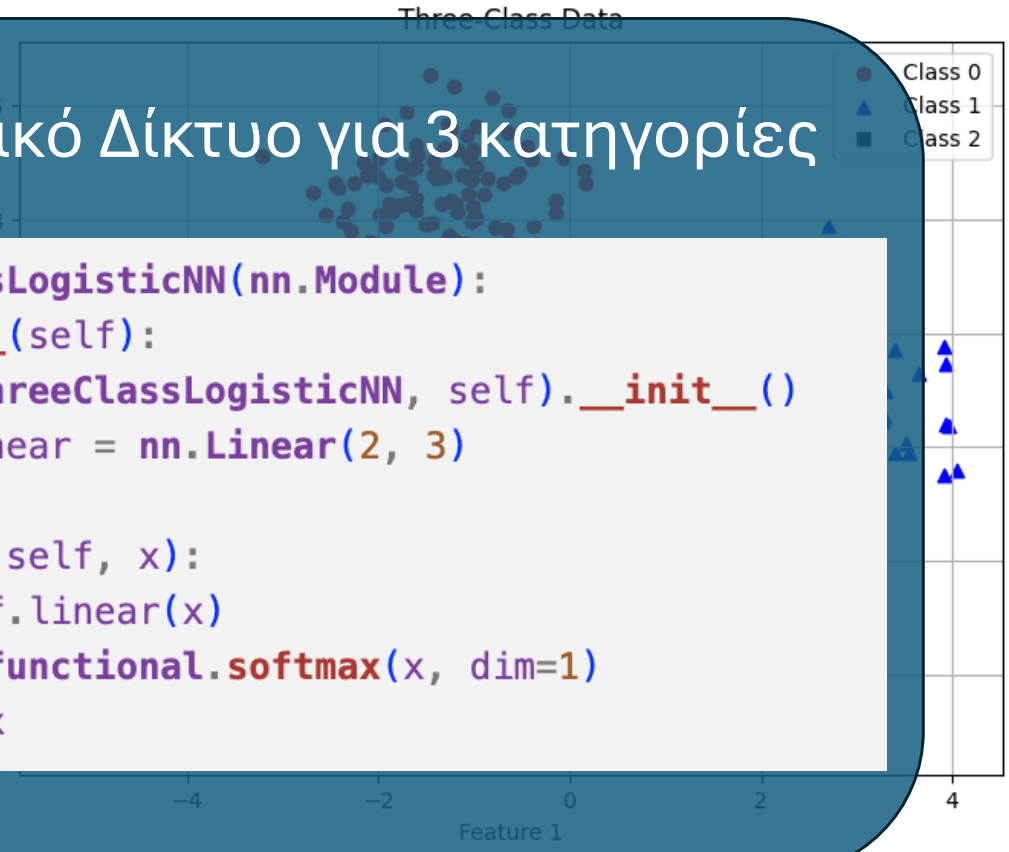
Λογιστική Παλινδρόμηση: Παραδείγματα

Νευρωνικό Δίκτυο για 3 κατηγορίες

- **Τρεις** κλάσεις/κατ
- Δύο δεδομέν εισόδου (fea

```
class ThreeClassLogisticNN(nn.Module):
    def __init__(self):
        super(ThreeClassLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 3)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```



Λογιστική Παλινδρόμηση: Τι κάνει;

- Ο κώδικάς μας εξηγεί ακριβώς τι κάνει η λογιστική παλινδρόμηση:

(x_1, x_2) →
$z_1 = w_{11}x_1 + w_{12}x_2 + w_{10}$
$z_2 = w_{21}x_1 + w_{22}x_2 + w_{20}$
$z_3 = w_{31}x_1 + w_{32}x_2 + w_{30}$

$$y_i = \frac{\exp(z_i)}{\exp(z_1) + \exp(z_2) + \exp(z_3)}$$

```
class ThreeClassLogisticNN(nn.Module):
    def __init__(self):
        super(ThreeClassLogisticNN, self).__init__()
        self.linear = nn.Linear(2, 3)

    def forward(self, x):
        x = self.linear(x)
        x = nn.functional.softmax(x, dim=1)
        return x
```

Λογιστική Παλινδρόμηση: Τι κάνει;

- Ο κώδικάς μας εξηγεί ακριβώς τι κάνει η λογιστική παλινδρόμηση:

Επόμενη Διάλεξη: γεωμετρική έννοια αυτής της απλής πράξης και η σχέση με τα embeddings

(x_1, x_2)

$$z_1 = w_{11}x_1 + w_{12}x_2 + w_{10}$$
$$z_2 = w_{21}x_1 + w_{22}x_2 + w_{20}$$
$$z_3 = w_{31}x_1 + w_{32}x_2 + w_{30}$$

$$y_i = \frac{\exp(z_i)}{\exp(z_1) + \exp(z_2) + \exp(z_3)}$$